***MATRIX – END MATRIX functions***

SPSS macros by Kirill Orlov

kior@akado.ru, ttnphns@gmail.com

<https://www.spsstools.net/en/KO-spssmacros>

All rights reserved

*Функции для MATRIX – END MATRIX.* Большое собрание полезных статистических, математических, переструктурирующих и иных функций для матричного сеанса в SPSS. Продвинутым пользователям в помощь анализу данных и написанию статистических алгоритмов.

Что нового май 2025

* !KO\_levenshtein новая функция
* !KO\_lcs новая функция

Что нового янв 2025

* !KO\_apriori новая функция

Что нового ноя 2024

* !KO\_robustadp новая функция
* !KO\_robustlts новая функция
* !KO\_trimmean новая функция
* !KO\_mestim новая функция
* !KO\_cronalpha новая функция
* !KO\_mcdomega новая функция

**Некоторые функции, которые вам могли бы понадобиться часто:**

* [среднее](#_ВЕКТОР_СРЕДНИХ_АРИФМЕТИЧЕСКИХ); [дисперсия](#_ВЕКТОР_ДИСПЕРСИЙ); [описательные статистики по группам](#_РАЗНЫЕ_ОДНОПЕРЕМЕННЫЕ_СТАТИСТИКИ,); [медиана](#_ВЕКТОР_МЕДИАН); [процентили](#_ПРОЦЕНТИЛИ);
* [частоты в переменной](#_ФИКТИВНЫЕ_ПЕРЕМЕННЫЕ_И); [частотная кросстабуляция](#_ДВУМЕРНАЯ_ЧАСТОТНАЯ_ПЕРЕКРЕСТНАЯ); [многомерная частотная агрегация](#_ФИКТИВНЫЕ_ПЕРЕМЕННЫЕ_ВЗАИМОДЕЙСТВИЯ); [частоты в категориальном наборе множественного ответа](#_ЭЛЕМЕНТАРНЫЕ_ПЕРЕМЕННЫЕ_И);
* [частоты заданных значений](#_ЧАСТОТЫ_ЗАДАННЫХ_ЗНАЧЕНИЙ); [кросстаб для заданных значений](#_ДВУМЕРНАЯ_K_x); [счет элементов, равных любому из заданных](#_СЧЕТ_ЗНАЧЕНИЙ_В);
* [ковариации](#_КОВАРИАЦИОННАЯ_МАТРИЦА); [корреляции](#_КОРРЕЛЯЦИОННАЯ_МАТРИЦА); [ковариации/корреляции по группам](#_КОВАРИАЦИОННАЯ/КОРРЕЛЯЦИОННАЯ_МАТРИ);
* [центрация](#_ЦЕНТРАЦИЯ_СТОЛБЦОВ_ДАННЫХ); [стандартизация](#_Z-СТАНДАРТИЗАЦИЯ_СТОЛБЦОВ_ДАННЫХ); [центрация по группам](#_ЦЕНТРАЦИЯ_СТОЛБЦОВ_ДАННЫХ,); [стандартизация по группам](#_Z-СТАНДАРТИЗАЦИЯ_СТОЛБЦОВ_ДАННЫХ,);
* [случайные значения из нормального распределения](#_СЛУЧАЙНЫЕ_ЗНАЧЕНИЯ_ИЗ); [случайно выбрать n элементов](#_СЛУЧАЙНО_ВЫБРАТЬ_N);
* [перекодировать данные](#_ПЕРЕКОДИРОВКА_ЗНАЧЕНИЙ_(ТОЧНОЕ);
* [выписать позиции ненулевых элементов](#_ПОЗИЦИИ_НЕНУЛЕВЫХ_ЭЛЕМЕНТОВ); [позиция первой встречи значения](#_ПОЗИЦИЯ_ПЕРВОЙ_ВСТРЕЧИ);
* [сортировать элементы](#_СОРТИРОВКА_ВЕКТОРА); [сортировать ряды](#_ПРОСТАЯ_СОРТИРОВКА_РЯДОВ); [расщепить данные по группам](#_РАЗДЕЛЕНИЕ_МАТРИЦЫ_ГОРИЗОНТАЛЬНО);

[ОБЩИЕ СВЕДЕНИЯ](#_Toc197355173)

[МАТЕМАТИЧЕСКИЕ ФУНКЦИИ](#_Toc197355174)

* [НАКОПЛЕННЫЕ СУММЫ В СТОЛБЦАХ [!KO\_ccum]](#_Toc197355175)
* [ПРОИЗВЕДЕНИЕ ЭЛЕМЕНТОВ МАТРИЦЫ [!KO\_prod]](#_Toc197355176)
* [СУММА НАДДИАГОНАЛЬНЫХ ИЛИ ПОДДИАГОНАЛЬНЫХ ЭЛЕМЕНТОВ МАТРИЦЫ [!KO\_trsum]](#_Toc197355177)
* [ПРОВЕРКА ТРЕУГОЛЬНОГО НЕРАВЕНСТВА В МАТРИЦЕ РАЗЛИЧИЙ [!KO\_trineq]](#_Toc197355178)
* [НИЖНЯЯ НЕПОЛНАЯ ГАММА-ФУНКЦИЯ [!KO\_ligamma]](#_Toc197355179)
* [QR-РАЗЛОЖЕНИЕ МЕТОДОМ ОТРАЖЕНИЙ ХАУСХОЛДЕРА [!KO\_qrdc]](#_Toc197355180)
* [ХЕССЕНБЕРГОВА МАТРИЦА / ТРИДИАГОНАЛЬНАЯ МАТРИЦА [!KO\_hess]](#_Toc197355181)
* [СОБСТВЕННОЧИСЛОВОЕ РАЗЛОЖЕНИЕ 2x2 МАТРИЦЫ [!KO\_eig2x2]](#_Toc197355182)
* [СОБСТВЕННОЧИСЛОВОЕ РАЗЛОЖЕНИЕ 2x2 МАТРИЦЫ (ВАРИАНТ ШУРА) [!KO\_schur2x2]](#_Toc197355183)
* [СОБСТВЕННОЧИСЛОВОЕ РАЗЛОЖЕНИЕ (ВАРИАНТ ШУРА) [!KO\_qreig]](#_Toc197355184)
* [ДОМИНАНТНОЕ СОБСТВЕННОЕ ЧИСЛО И ЕГО СОБСТВЕННЫЙ ВЕКТОР [!KO\_poweig]](#_Toc197355185)

[СТАТИСТИЧЕСКИЕ ОПИСАТЕЛЬНЫЕ ФУНКЦИИ](#_Toc197355186)

* [ВЕКТОР СРЕДНИХ АРИФМЕТИЧЕСКИХ [!KO\_mean]](#_Toc197355187)
* [ВЕКТОР ДИСПЕРСИЙ [!KO\_variance]](#_Toc197355188)
* [ВЕКТОР ДИСПЕРСИЙ (DF=N) [!KO\_variance2]](#_Toc197355189)
* [ВЕКТОР АСИММЕТРИЙ [!KO\_skewness]](#_Toc197355190)
* [ВЕКТОР ЭКСЦЕССОВ [!KO\_kurtosis]](#_Toc197355191)
* [ВЕКТОР МЕДИАН [!KO\_median]](#_Toc197355192)
* [ПРОЦЕНТИЛИ [!KO\_ptile]](#_Toc197355193)
* [МИНИМУМ, МАКСИМУМ, СРЕДНЯЯ И ДИСПЕРСИЯ, ПО ГРУППАМ [!KO\_gdescr]](#_Toc197355194)
* [РАЗНЫЕ ОДНОПЕРЕМЕННЫЕ СТАТИСТИКИ, ПО ГРУППАМ [!KO\_aggr]](#_Toc197355195)
* [ЗНАЧЕНИЕ, БЛИЖАЙШЕЕ К ЗАДАННОМУ СНИЗУ ИЛИ СВЕРХУ [!KO\_closest]](#_Toc197355196)
* [КОВАРИАЦИОННАЯ МАТРИЦА [!KO\_cov]](#_Toc197355197)
* [КОВАРИАЦИОННАЯ МАТРИЦА (DF=N) [!KO\_cov2]](#_Toc197355198)
* [КОРРЕЛЯЦИОННАЯ МАТРИЦА [!KO\_corr]](#_Toc197355199)
* [КОСИНУСНАЯ МАТРИЦА [!KO\_cosine]](#_Toc197355200)
* [МАТРИЦА КОЭФФИЦИЕНТОВ ТОЖДЕСТВЕННОСТИ [!KO\_idc]](#_Toc197355201)
* [МАТРИЦА ОТНОШЕНИЙ ПОДОБИЯ [!KO\_simr]](#_Toc197355202)
* [КОВАРИАЦИЯ, КОРРЕЛЯЦИЯ, КОСИНУС, КОЭФФИЦИЕНТ ТОЖДЕСТВЕННОСТИ, ОТНОШЕНИЕ ПОДОБИЯ [!KO\_biv]](#_Toc197355203)
* [ПРЯМОУГОЛЬНАЯ МАТРИЦА КОВАРИАЦИЙ/КОРРЕЛЯЦИЙ/КОСИНУСОВ [!KO\_rect]](#_Toc197355204)
* [КОВАРИАЦИОННАЯ/КОРРЕЛЯЦИОННАЯ МАТРИЦА, ПО ГРУППАМ И УСРЕДНЕННАЯ [!KO\_gcov]](#_Toc197355205)
* [МЕЖГРУППОВАЯ И ОБЪЕДИНЕННАЯ ВНУТРИГРУППОВАЯ МАТРИЦЫ РАССЕЯНИЯ [!KO\_bwscat]](#_Toc197355206)
* [МАТРИЦА ОБРАЗОВ ИЛИ АНТИОБРАЗОВ [!KO\_image]](#_Toc197355207)
* [МАТРИЦА ВЛИЯНИЯ (HAT-МАТРИЦА) [!KO\_hat]](#_Toc197355208)
* [МАТРИЦЫ СОЧАСТОТ ДЛЯ ДВОИЧНЫХ ДАННЫХ [!KO\_bincnt]](#_Toc197355209)
* [МАТРИЦА КВАДРАТНЫХ ЕВКЛИДОВЫХ РАССТОЯНИЙ [!KO\_seuclid]](#_Toc197355210)
* [ПРЯМОУГОЛЬНАЯ МАТРИЦА КВАДРАТНЫХ ЕВКЛИДОВЫХ РАССТОЯНИЙ [!KO\_seuclidr]](#_Toc197355211)
* [МАТРИЦА СТЕПЕННЫХ ВЗВЕШЕННЫХ РАССТОЯНИЙ МИНКОВСКОГО [!KO\_pwmink]](#_Toc197355212)
* [ПРЯМОУГОЛЬНАЯ МАТРИЦА СТЕПЕННЫХ ВЗВЕШЕННЫХ РАССТОЯНИЙ МИНКОВСКОГО [!KO\_pwminkr]](#_Toc197355213)
* [МАТРИЦА КВАДРАТНЫХ РАССТОЯНИЙ МАХАЛАНОБИСА [!KO\_smahal]](#_Toc197355214)
* [ВЕКТОР КВАДРАТНЫХ РАССТОЯНИЙ МАХАЛАНОБИСА ДО ЦЕНТРОИДА [!KO\_smahalc]](#_Toc197355215)
* [МАТРИЦА КВАДРАТНЫХ БАТЛЕРОВЫХ РАССТОЯНИЙ [!KO\_sbutler]](#_Toc197355216)
* [ЕСТЬ ЛИ В МАТРИЦЕ ОДИНАКОВЫЕ ЗНАЧЕНИЯ](#_Toc197355217)
* [СПИСОК УНИКАЛЬНЫХ ЗНАЧЕНИЙ В ВЕКТОРЕ И ПОЗИЦИЙ ИХ ПЕРВОГО ПОЯВЛЕНИЯ [!KO\_unique]](#_Toc197355218)
* [ЧАСТОТЫ ЗАДАННЫХ ЗНАЧЕНИЙ В СТОЛБЦАХ [!KO\_freqval]](#_Toc197355219)
* [ДВУМЕРНАЯ ЧАСТОТНАЯ ПЕРЕКРЕСТНАЯ ТАБЛИЦА ДЛЯ ЗАДАННЫХ ЗНАЧЕНИЙ [!KO\_ctabval]](#_Toc197355220)
* [ФИКТИВНЫЕ ПЕРЕМЕННЫЕ И ЧАСТОТЫ ЗНАЧЕНИЙ В СТОЛБЦЕ [!KO\_freq]](#_Toc197355221)
* [ДВУМЕРНАЯ ЧАСТОТНАЯ ПЕРЕКРЕСТНАЯ ТАБЛИЦА [!KO\_crosstab]](#_Toc197355222)
* [ФИКТИВНЫЕ ПЕРЕМЕННЫЕ ВЗАИМОДЕЙСТВИЯ И МНОГОМЕРНАЯ ЧАСТОТНАЯ АГРЕГАЦИЯ [!KO\_aggrtab]](#_Toc197355223)
* [ФИКТИВНЫЕ ПЕРЕМЕННЫЕ ВЗАИМОДЕЙСТВИЯ И МНОГОМЕРНАЯ ЧАСТОТНАЯ АГРЕГАЦИЯ (ДИХОТОМИЧЕСКИЕ ДАННЫЕ) [!KO\_baggrtab]](#_Toc197355224)
* [ЭЛЕМЕНТАРНЫЕ ПЕРЕМЕННЫЕ И ЧАСТОТЫ ЗНАЧЕНИЙ В КАТЕГОРИАЛЬНОМ НАБОРЕ МНОЖЕСТВЕННОГО ОТВЕТА [!KO\_mrfreq]](#_Toc197355225)
* [ПРОЦЕНТ ПРАВИЛЬНОЙ КЛАССИФИКАЦИИ [!KO\_classres]](#_Toc197355226)
* [ЧАСТОТЫ КОНКОРДАНТНЫХ, ДИСКОРДАНТНЫХ И СВЯЗАННЫХ ПАР [!KO\_concdisc]](#_Toc197355227)
* [ЧАСТОТЫ КОНКОРДАНТНЫХ, ДИСКОРДАНТНЫХ И СВЯЗАННЫХ ПАР (ИЗ ТАБЛИЦЫ) [!KO\_concdisct]](#_Toc197355228)
* [МАТРИЦА ПУТАНОСТИ СОЧЛЕНСТВА ДЛЯ ДВУХ РАЗБИЕНИЙ [!KO\_ccm]](#_Toc197355229)
* [СЧЕТ ЗНАЧЕНИЙ В РЯДАХ (ТОЧНОЕ СОВПАДЕНИЕ) [!KO\_count1]](#_Toc197355230)
* [СЧЕТ ЗНАЧЕНИЙ В РЯДАХ (ПОПАДАНИЕ В ДИАПАЗОН) [!KO\_count2]](#_Toc197355231)
* [ХИ-КВАДРАТ (ПИРСОНА И ОТНОШЕНИЯ ПРАВДОПОДОБИЯ) ДВУМЕРНОЙ ТАБЛИЦЫ СОПРЯЖЕННОСТИ [!KO\_chitab]](#_Toc197355232)
* [ДОЛИ И ОСТАТКИ В ДВУМЕРНОЙ ТАБЛИЦЕ СОПРЯЖЕННОСТИ [!KO\_cells]](#_Toc197355233)
* [МАТРИЦА КВАДРАТНЫХ ХИ-КВАДРАТ-РАССТОЯНИЙ В ТАБЛИЦЕ СОПРЯЖЕННОСТИ [!KO\_schitab]](#_Toc197355234)
* [МАТРИЦА ХИ-КВАДРАТОВ МЕЖДУ РЯДЯМИ ЧАСТОТНЫХ ДАННЫХ ПОПАРНО [!KO\_schi2c]](#_Toc197355235)
* [РАССТОЯНИЯ ДО ЦЕНТРОИДОВ ГРУПП [!KO\_dtoc]](#_Toc197355236)
* [РАССТОЯНИЯ МЕЖДУ ЦЕНТРОИДАМИ ГРУПП](#_Toc197355237)
* [РАССТОЯНИЯ ДО ЦЕНТРОИДОВ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАССТОЯНИЙ) [!KO\_dtocfrd]](#_Toc197355238)
* [РАССТОЯНИЯ МЕЖДУ ЦЕНТРОИДАМИ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАССТОЯНИЙ) [!KO\_dbwcfrd]](#_Toc197355239)
* [УСРЕДНЕННЫЕ РАССТОЯНИЯ ДО ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_datofrd]](#_Toc197355240)
* [УСРЕДНЕННЫЕ РАССТОЯНИЯ МЕЖДУ ГРУППАМИ (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dabwfrd]](#_Toc197355241)
* [РАССТОЯНИЯ ДО ДАЛЬНИХ СОСЕДЕЙ ИЗ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dtoffrd]](#_Toc197355242)
* [РАССТОЯНИЯ МЕЖДУ ДАЛЬНИМИ СОСЕДЯМИ ИЗ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dbwffrd]](#_Toc197355243)
* [РАССТОЯНИЯ ДО БЛИЖНИХ СОСЕДЕЙ ИЗ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dtonfrd]](#_Toc197355244)
* [РАССТОЯНИЯ МЕЖДУ БЛИЖНИМИ СОСЕДЯМИ ИЗ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dbwnfrd]](#_Toc197355245)
* [РАССТОЯНИЯ ДО МЕДОИДОВ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dtomfrd]](#_Toc197355246)
* [ВНУТРИГРУППОВЫЕ СУММЫ КВАДРАТОВ ОТКЛОНЕНИЙ (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАССТОЯНИЙ) [!KO\_sswfrd]](#_Toc197355247)
* [K БЛИЖНИХ СОСЕДЕЙ (ВЫПИСКА), ВЕРСИЯ "RANDOM" [!KO\_knnr]](#_Toc197355248)
* [K БЛИЖНИХ СОСЕДЕЙ (ВЫПИСКА), ВЕРСИЯ "PLUS" [!KO\_knnp]](#_Toc197355249)
* [ВНУТРИКЛАССОВЫЙ КОЭФФИЦИЕНТ КОРРЕЛЯЦИИ (ОДНОФАКТОРНЫЙ) [!KO\_iccow]](#_Toc197355250)
* [ВНУТРИКЛАССОВЫЙ КОЭФФИЦИЕНТ КОРРЕЛЯЦИИ (ОДНОФАКТОРНЫЙ, СБАЛАНСИРОВАННЫЕ КЛАССЫ) [!KO\_iccowb]](#_Toc197355251)
* [КОЭФФИЦИЕНТ ДИАГОНАЛЬНОСТИ МАТРИЦЫ [!KO\_diagns]](#_Toc197355252)
* [РОБАСТНЫЕ К ВЫБРОСАМ СРЕДНЯЯ И ДИСПЕРСИЯ (ADP) [!KO\_robustadp]](#_Toc197355253)
* [РОБАСТНЫЕ К ВЫБРОСАМ СРЕДНЯЯ И ДИСПЕРСИЯ (LTS) [!KO\_robustlts]](#_Toc197355254)
* [УСЕЧЕННАЯ СРЕДНЯЯ [!KO\_trimmean]](#_Toc197355255)
* [M-ОЦЕНИВАТЕЛИ ПОЛОЖЕНИЯ [!KO\_mestim]](#_Toc197355256)
* [АЛЬФА КРОНБАХА [!KO\_cronalpha]](#_Toc197355257)
* [ОМЕГА МАКДОНАЛЬДА [!KO\_mcdomega]](#_Toc197355258)

[СТАТИСТИЧЕСКИЕ ПРЕОБРАЗОВАТЕЛЬНЫЕ ФУНКЦИИ](#_Toc197355259)

* [ЦЕНТРАЦИЯ СТОЛБЦОВ ДАННЫХ [!KO\_center]](#_Toc197355260)
* [Z-СТАНДАРТИЗАЦИЯ СТОЛБЦОВ ДАННЫХ [!KO\_zscore]](#_Toc197355261)
* [Z-СТАНДАРТИЗАЦИЯ СТОЛБЦОВ ДАННЫХ (DF=N) [!KO\_zscore2]](#_Toc197355262)
* [МАСШТАБИРОВАНИЕ СТОЛБЦОВ ДАННЫХ К SS=1 [!KO\_scale]](#_Toc197355263)
* [ЦЕНТРАЦИЯ СТОЛБЦОВ ДАННЫХ, ПО ГРУППАМ [!KO\_gcenter]](#_Toc197355264)
* [Z-СТАНДАРТИЗАЦИЯ СТОЛБЦОВ ДАННЫХ, ПО ГРУППАМ [!KO\_gzscore]](#_Toc197355265)
* [МАСШТАБИРОВАНИЕ СТОЛБЦОВ ДАННЫХ, ПО ГРУППАМ [!KO\_gscale]](#_Toc197355266)
* [ЛИНЕЙНОЕ ПЕРЕШКАЛИРОВАНИЕ [!KO\_rescale]](#_Toc197355267)
* [ДВУВХОДОВАЯ ЦЕНТРАЦИЯ](#_Toc197355268)
* [КОНВЕРТАЦИЯ КОВАРИАЦИОННОЙ МАТРИЦЫ В КОРРЕЛЯЦИОННУЮ [!KO\_covcorr]](#_Toc197355269)
* [КОНВЕРТАЦИЯ КОРРЕЛЯЦИОННОЙ МАТРИЦЫ В КОВАРИАЦИОННУЮ [!KO\_corrcov]](#_Toc197355270)
* [ДВОЙНАЯ ЦЕНТРАЦИЯ МАТРИЦЫ КВАДРАТНЫХ РАССТОЯНИЙ [!KO\_dcenter]](#_Toc197355271)
* [КОНВЕРТАЦИЯ МАТРИЦЫ УГЛОВЫХ СХОДСТВ В КВАДРАТНЫЕ РАССТОЯНИЯ ПО ТЕОРЕМЕ КОСИНУСОВ [!KO\_sdcosth]](#_Toc197355272)
* [ПЕРЕКЛЮЧЕНИЕ МАТРИЦЫ СКАЛЯРНЫХ ПРОИЗВЕДЕНИЙ НА НОВУЮ ДИАГОНАЛЬ (СОХРАНЯЯ УГОЛ) [!KO\_swdiag1]](#_Toc197355273)
* [ПЕРЕКЛЮЧЕНИЕ МАТРИЦЫ СКАЛЯРНЫХ ПРОИЗВЕДЕНИЙ НА НОВУЮ ДИАГОНАЛЬ (СОХРАНЯЯ РАССТОЯНИЕ) [!KO\_swdiag2]](#_Toc197355274)
* [ПЕРЕКЛЮЧЕНИЕ MSCP-МАТРИЦЫ НА НОВЫЙ ЦЕНТРОИД ДАННЫХ [!KO\_swcentr]](#_Toc197355275)
* [ФИКТИВНЫЕ (DUMMY) ПЕРЕМЕННЫЕ](#_Toc197355276)
* [НАБОР ПЕРЕМЕННЫХ МНОЖЕСТВЕННОГО ОТВЕТА](#_Toc197355277)
* [РАНЖИРОВАНИЕ ЗНАЧЕНИЙ](#_Toc197355278)
* [РИДИТЫ ПОРЯДКОВЫХ ГРАДАЦИЙ [!KO\_ridit]](#_Toc197355279)
* [ЭКСПОНЕНЦИАЛЬНОЕ НОРМИРОВАНИЕ (SOFTMAX-ФУНКЦИЯ) [!KO\_nef]](#_Toc197355280)
* [ЛОГ-ОТНОШЕНЧЕСКАЯ ЦЕНТРАЦИЯ (CENTERED LOGRATIO TRANSFORM) [!KO\_clr]](#_Toc197355281)
* [ПРЕОБРАЗОВАНИЕ БОКСА-КОКСА [!KO\_boxcox]](#_Toc197355282)
* [МАТРИЦА ОТБЕЛИВАНИЯ (СФЕРИЗАЦИИ) [!KO\_white]](#_Toc197355283)
* [ПРЕОБРАЗОВАТЬ ПЕРЕМЕННЫЕ ТОЧНО К ЗАДАННЫМ КОВАРИАЦИЯМ [!KO\_tocov]](#_Toc197355284)
* [ПРЕОБРАЗОВАТЬ ПЕРЕМЕННУЮ ТОЧНО К ЗАДАННЫМ КОВАРИАЦИЯМ С ДРУГИМИ ПЕРЕМЕННЫМИ [!KO\_ytocov]](#_Toc197355285)
* [НОРМАЛЬНОЕ ОБЛАКО В РАВНОМЕРНЫЙ ШАР [!KO\_unifball]](#_Toc197355286)

[СТАТИСТИЧЕСКИЕ АНАЛИТИЧЕСКИЕ ФУНКЦИИ](#_Toc197355287)

* [ЛИНЕЙНАЯ РЕГРЕССИЯ [!KO\_regress]](#_Toc197355288)
* [ГЛАВНЫЕ КОМПОНЕНТЫ [!KO\_pcomp]](#_Toc197355289)
* [ГЛАВНЫЕ КОМПОНЕНТЫ (ДЛЯ N<P) [!KO\_pcomp2]](#_Toc197355290)
* [ГЛАВНЫЕ КОМПОНЕНТЫ (ВХОДЯЩИЕ - МАТРИЦА) [!KO\_pca]](#_Toc197355291)
* [ОРТОГОНАЛЬНОЕ ПРОКРУСТОВО ВРАЩЕНИЕ [!KO\_procr]](#_Toc197355292)
* [ЛИНЕЙНЫЕ ДИСКРИМИНАНТЫ [!KO\_discrim]](#_Toc197355293)
* [ГАУССОВ КЛАССИФИКАТОР [!KO\_gaclass]](#_Toc197355294)
* [ГАУССОВ КЛАССИФИКАТОР (С ОПЦИЕЙ "СКОЛЬЗЯЩИЙ КОНТРОЛЬ-1") [!KO\_gaclass2]](#_Toc197355295)
* [КАНОНИЧЕСКИЕ КОРРЕЛЯЦИИ [!KO\_cancorr]](#_Toc197355296)
* [ГЛАВНЫЕ КООРДИНАТЫ [!KO\_pcoord]](#_Toc197355297)
* [БИПЛОТ [!KO\_biplot]](#_Toc197355298)
* [ПРОСТОЙ АНАЛИЗ СООТВЕТСТВИЙ [!KO\_corresp]](#_Toc197355299)
* [ФАКТОРЫ МЕТОДОМ ГЛАВНЫХ ОСЕЙ [!KO\_paf]](#_Toc197355300)
* [ОРТОГОНАЛЬНЫЕ АНАЛИТИЧЕСКИЕ ВРАЩЕНИЯ [!KO\_ortrot]](#_Toc197355301)
* [КОСОУГОЛЬНОЕ ВРАЩЕНИЕ ПРОМАКС И ПРОМАЙ [!KO\_promax]](#_Toc197355302)
* [КОЭФФИЦИЕНТЫ ФАКТОРНЫХ БАЛЛОВ [!KO\_fsc]](#_Toc197355303)
* [КЛАСТЕРИЗАЦИЯ МЕТОДОМ K-СРЕДНИХ [!KO\_kmeans]](#_Toc197355304)
* [БЛОК-ДИАГОНАЛИЗАЦИЯ МЕТОДОМ VAT/IVAT [!KO\_vat]](#_Toc197355305)
* [K-БЛИЖНИХ СОСЕДЕЙ (С ЗАВИСИМОЙ ПЕРЕМЕННОЙ) [!KO\_knnpred]](#_Toc197355306)
* [НАИВНЫЙ БАЙЕСОВ КЛАССИФИКАТОР [!KO\_nbclass]](#_Toc197355307)

[ФУНКЦИИ СЛУЧАЙНЫХ ДАННЫХ](#_Toc197355308)

* [СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ СТАНДАРТНОГО РАВНОМЕРНОРГО РАСПРЕДЕЛЕНИЯ](#_Toc197355309)
* [СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ СТАНДАРТНОГО НОРМАЛЬНОГО РАСПРЕДЕЛЕНИЯ [!KO\_normal]](#_Toc197355310)
* [СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ НОРМАЛЬНОГО РАСПРЕДЕЛЕНИЯ С ЗАДАННЫМИ ПАРАМЕТРАМИ [!KO\_rvnorm]](#_Toc197355311)
* [СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ РАВНОМЕРНОГО РАСПРЕДЕЛЕНИЯ С ЗАДАННЫМИ ПАРАМЕТРАМИ [!KO\_rvunif]](#_Toc197355312)
* [СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ ХИ-КВАДРАТ РАСПРЕДЕЛЕНИЯ [!KO\_chisq]](#_Toc197355313)
* [СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ БИНОМИАЛЬНОГО РАСПРЕДЕЛЕНИЯ [!KO\_binom]](#_Toc197355314)
* [СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ КАТЕГОРИАЛЬНОГО РАСПРЕДЕЛЕНИЯ [!KO\_categ]](#_Toc197355315)
* [ВЫБОРКА ИЗ КАТЕГОРИАЛЬНОГО РАСПРЕДЕЛЕНИЯ БЕЗ ВОЗВРАЩЕНИЯ [!KO\_catwor]](#_Toc197355316)
* [СЛУЧАЙНО ВЫБРАТЬ N ЭЛЕМЕНТОВ ВЕКТОРА [!KO\_sample]](#_Toc197355317)
* [СЛУЧАЙНАЯ МАТРИЦА ИЗ РАСПРЕДЕЛЕНИЯ УИШАРТА [!KO\_wishart]](#_Toc197355318)
* [СЛУЧАЙНЫЕ ДАННЫЕ ИЗ НОРМАЛЬНОЙ ПОПУЛЯЦИИ С ЗАДАННЫМИ КОВАРИАЦИЯМИ [!KO\_mvnorm]](#_Toc197355319)
* [СОЗДАТЬ СЛУЧАЙНЫЕ ДАННЫЕ ТОЧНО С ЗАДАННЫМИ КОВАРИАЦИЯМИ](#_Toc197355320)

[ФУНКЦИИ ПЕРЕКОДИРОВКИ/ЗАМЕНЫ](#_Toc197355321)

* [ПЕРЕКОДИРОВКА ЗНАЧЕНИЙ (ТОЧНОЕ СОВПАДЕНИЕ) [!KO\_recode1]](#_Toc197355322)
* [ПЕРЕКОДИРОВКА ЗНАЧЕНИЙ (ПОПАДАНИЕ В ДИАПАЗОН) [!KO\_recode2]](#_Toc197355323)
* [РАЗНЫЕ ОДНОПЕРЕМЕННЫЕ СТАТИСТИКИ, ПО ГРУППАМ: ЗАМЕНА ИМИ ИСХОДНЫХ ЗНАЧЕНИЙ [!KO\_aggrv]](#_Toc197355324)

[ФУНКЦИИ ПОИСКА/ПОМЕТКИ](#_Toc197355325)

* [ПОЗИЦИИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ В ВЕКТОРЕ [!KO\_indices]](#_Toc197355326)
* [ПОЗИЦИИ НЕНУЛЕВЫХ И НУЛЕВЫХ ЭЛЕМЕНТОВ В ВЕКТОРЕ [!KO\_indices2]](#_Toc197355327)
* [ПОЗИЦИИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ В РЯДАХ ИЛИ В СТОЛБЦАХ МАТРИЦЫ](#_Toc197355328)
* [ПОЗИЦИИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ В МАТРИЦЕ (И САМИ ЭЛЕМЕНТЫ) [!KO\_indicesm]](#_Toc197355329)
* [ПОЗИЦИЯ ПЕРВОЙ ВСТРЕЧИ ЗНАЧЕНИЯ В РЯДУ/РЯДАХ [!KO\_indx]](#_Toc197355330)
* [ПОЗИЦИЯ ПОСЛЕДНЕЙ ВСТРЕЧИ ЗНАЧЕНИЯ В РЯДУ/РЯДАХ [!KO\_rindx]](#_Toc197355331)
* [ПОЗИЦИЯ СЛУЧАЙНОЙ ВСТРЕЧИ ЗНАЧЕНИЯ В РЯДУ/РЯДАХ [!KO\_randindx]](#_Toc197355332)
* [ПОЗИЦИЯ (ПАРА ИНДЕКСОВ) ОДНОГО ЗНАЧЕНИЯ В МАТРИЦЕ [!KO\_ij]](#_Toc197355333)
* [ПОМЕТКА В РЯДУ И СТОЛБЦЕ НЕ БОЛЕЕ ОДНОГО ЭЛЕМЕНТА, РАВНОГО ЗНАЧЕНИЮ [!KO\_prime]](#_Toc197355334)
* [ИНДИКАЦИЯ НАБЛЮДЕНИЙ С ЗАДАННЫМ ПРОФИЛЕМ ЗНАЧЕНИЙ [!KO\_indic]](#_Toc197355335)
* [ПОМЕТКА ПРЯМЫХ ЦЕПОЧЕК [!KO\_runs]](#_Toc197355336)
* [ПОМЕТКА ПРЯМЫХ ЦЕПОЧЕК (ДРУГОЙ АЛГОРИТМ) [!KO\_runs2]](#_Toc197355337)
* [ПОМЕТКА КОСЫХ ЦЕПОЧЕК [!KO\_slant]](#_Toc197355338)
* [ПОМЕТКА КОСЫХ ЦЕПОЧЕК (ДРУГОЙ АЛГОРИТМ) [!KO\_slant2]](#_Toc197355339)
* [ПОИСК В ГЛУБИНУ В ОДНОДОЛЬНОМ ГРАФЕ [!KO\_dfs]](#_Toc197355340)
* [ПОИСК В ШИРИНУ В ОДНОДОЛЬНОМ ГРАФЕ [!KO\_bfs]](#_Toc197355341)
* [АЛГОРИТМ МАКСИМАЛЬНЫХ КЛИК БРОНА-КЕРБОША (PIVOT-ВЕРСИЯ) [!KO\_bronkerb]](#_Toc197355342)
* [АЛГОРИТМ МАКСИМАЛЬНЫХ КЛИК БРОНА-КЕРБОША (БАЗОВАЯ ВЕРСИЯ) [!KO\_bronkerb2]](#_Toc197355343)
* [ВЫПУКЛАЯ ОБОЛОЧКА: 2D АЛГОРИТМ ДЖАРВИСА [!KO\_chjarv]](#_Toc197355344)
* [ВЫПУКЛАЯ ОБОЛОЧКА: 2D АЛГОРИТМ ГРЭМА [!KO\_chgrah]](#_Toc197355345)
* [ВЕЛИЧИНЫ ОТСТОЯНИЙ ЭЛЕМЕНТОВ ОТ ДИАГОНАЛИ МАТРИЦЫ [!KO\_diagoff]](#_Toc197355346)
* [ВЕЛИЧИНЫ ОТСТОЯНИЙ ЭЛЕМЕНТОВ ОТ ПОЛОСЫ ДИАГОНАЛЕЙ МАТРИЦЫ [!KO\_diagboff]](#_Toc197355347)

[ФУНКЦИИ ПЕРЕСТРУКТУРИРОВАНИЯ](#_Toc197355348)

* [СЧИТКА ВСТРАИВАЕМЫХ ДАННЫХ [!KO\_read]](#_Toc197355349)
* [СЧИТКА ВСТРАИВАЕМЫХ ДАННЫХ (БЕЗ РАСКАВЫЧКИ) [!KO\_read2]](#_Toc197355350)
* [РАЗРЕЖЕННУЮ МАТРИЦУ В СПИСОК НЕНУЛЕВЫХ ЗНАЧЕНИЙ [!KO\_nzlist]](#_Toc197355351)
* [СПИСОК В РАЗРЕЖЕННУЮ МАТРИЦУ [!KO\_nzunlist]](#_Toc197355352)
* [УТРАМБОВКА ПО ГОРИЗОНТАЛИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ ИЛИ ИХ ИНДЕКСОВ [!KO\_ram]](#_Toc197355353)
* [УТРАМБОВКА ПО ВЕРТИКАЛИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ ИЛИ ИХ ИНДЕКСОВ [!KO\_vram]](#_Toc197355354)
* [РАЗМНОЖЕНИЕ РЯДОВ [!KO\_propag]](#_Toc197355355)
* [ВЫБОР РЯДОВ (ФИЛЬТРАЦИЯ НАБЛЮДЕНИЙ)](#_Toc197355356)
* [ПРИБАВЛЕНИЕ КОНСТАНТЫ К ЭЛЕМЕНТАМ ТРЕУГОЛЬНИКА МАТРИЦЫ [!KO\_tradd]](#_Toc197355357)
* [УМНОЖЕНИЕ НА КОНСТАНТУ ЭЛЕМЕНТОВ ТРЕУГОЛЬНИКА МАТРИЦЫ [!KO\_trmult]](#_Toc197355358)
* [ЭКСПОНЕНЦИИРОВАНИЕ КОНСТАНТОЙ ЭЛЕМЕНТОВ ТРЕУГОЛЬНИКА МАТРИЦЫ [!KO\_trexp]](#_Toc197355359)
* [РАЗВЕРТКА ТРЕУГОЛЬНИКОВ КВАДРАТНОЙ МАТРИЦЫ В ВЕКТОРЫ [!KO\_unftri]](#_Toc197355360)
* [СВЕРТКА ДВУХ ВЕКТОРОВ В ТРЕУГОЛЬНИКИ КВАДРАТНОЙ МАТРИЦЫ [!KO\_foltri]](#_Toc197355361)
* [СИММЕТРИЗАЦИЯ КВАДРАТНОЙ МАТРИЦЫ: ЗАМЕНА ОДНОГО ТРЕУГОЛЬНИКА ВТОРЫМ [!KO\_symtri1]](#_Toc197355362)
* [СИММЕТРИЗАЦИЯ КВАДРАТНОЙ МАТРИЦЫ: ЗАМЕНА БОЛЬШЕГО ЭЛЕМЕНТА МЕНЬШИМ ИЛИ МЕНЬШЕГО БОЛЬШИМ [!KO\_symtri2]](#_Toc197355363)
* [СОРТИРОВКА ВЕКТОРА [!KO\_sort]](#_Toc197355364)
* [ПРОСТАЯ СОРТИРОВКА РЯДОВ В МАТРИЦЕ [!KO\_rsort]](#_Toc197355365)
* [СОРТИРОВКА РЯДОВ В МАТРИЦЕ ПО ЗНАЧЕНИЯМ ВНЕШНЕГО СТОЛБЦА [!KO\_rsortc]](#_Toc197355366)
* [ИЕРАРХИЧЕСКАЯ СОРТИРОВКА РЯДОВ В МАТРИЦЕ [!KO\_hiesort]](#_Toc197355367)
* [СОРТИРОВАТЬ В СЛУЧАЙНОМ ПОРЯДКЕ](#_Toc197355368)
* [РАСТАСКИВАНИЕ МАТРИЦЫ НА ДВЕ [!KO\_split]](#_Toc197355369)
* [СМЕШЕНИЕ ДВУХ МАТРИЦ [!KO\_merge]](#_Toc197355370)
* [РАЗДЕЛЕНИЕ МАТРИЦЫ ГОРИЗОНТАЛЬНО НА НЕСКОЛЬКО, ПО ГРУППАМ [!KO\_msplit]](#_Toc197355371)
* [ПЕРЕСТРУКТУРИРОВАНИЕ "ПЕРЕМЕННЫЕ В НАБЛЮДЕНИЯ" [!KO\_vartocas]](#_Toc197355372)
* [ПЕРЕСТРУКТУРИРОВАНИЕ "НАБЛЮДЕНИЯ В ПЕРЕМЕННЫЕ" [!KO\_castovar]](#_Toc197355373)
* [СМЕЩЕНИЕ РЯДОВ ПО ГОРИЗОНТАЛИ [!KO\_shift]](#_Toc197355374)
* [БЛОК-ВРЕЗКА [!KO\_blockic]](#_Toc197355375)

[КОМБИНАТОРНЫЕ ФУНКЦИИ И МНОЖЕСТВА](#_Toc197355376)

* [ДВУМЕСТНЫЕ ОПЕРАЦИИ С МНОЖЕСТВАМИ [!KO\_setdo]](#_Toc197355377)
* [ДЕКАРТОВО ПРОИЗВЕДЕНИЕ ДВУХ МНОЖЕСТВ [!KO\_cartp]](#_Toc197355378)
* [ДЕКАРТОВО ПРОИЗВЕДЕНИЕ N МНОЖЕСТВ [!KO\_ncartp]](#_Toc197355379)
* [СОЧЕТАНИЯ ПО K ЭЛЕМЕНТОВ [!KO\_combk]](#_Toc197355380)
* [ВСЕ СОЧЕТАНИЯ ПО K=2,3... ЭЛЕМЕНТОВ [!KO\_allcomb]](#_Toc197355381)
* [СОЧЕТАНИЯ ПО K ЭЛЕМЕНТОВ ИЗ РАЗНЫХ МНОЖЕСТВ [!KO\_dscombk]](#_Toc197355382)
* [ВСЕ СОЧЕТАНИЯ ПО K=2,3... ЭЛЕМЕНТОВ ИЗ РАЗНЫХ МНОЖЕСТВ [!KO\_dsallcomb]](#_Toc197355383)
* [ГОРИЗОНТАЛЬНЫЕ АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ В СОЧЕТАНИЯХ ПЕРЕМЕННЫХ [!KO\_comboper]](#_Toc197355384)
* [КОЛИЧЕСТВА НА БАЗЕ ГОРИЗОНТАЛЬНЫХ СУММ ИЛИ СРЕДНИХ В СОЧЕТАНИЯХ ПЕРЕМЕННЫХ [!KO\_turflike]](#_Toc197355385)
* [АЛГОРИТМ APRIORI [!KO\_apriori]](#_Toc197355386)

[КОМБИНАТОРНЫЕ ОПТИМИЗАЦИОННЫЕ ФУНКЦИИ](#_Toc197355387)

* [СПАРИВАНИЕ ВЕНГЕРСКИМ АЛГОРИТМОМ [!KO\_hungar]](#_Toc197355388)
* [СПАРИВАНИЕ АЛГОРИТМОМ ХОПКРОФТА-КАРПА [!KO\_hopckarp]](#_Toc197355389)
* [ПРОСТОЕ ЖАДНОЕ СПАРИВАНИЕ [!KO\_greedm]](#_Toc197355390)
* [ПРОСТОЕ ЖАДНОЕ СПАРИВАНИЕ (ДИСКРЕТНЫЕ ДАННЫЕ) [!KO\_greedm2]](#_Toc197355391)
* [АЛГОРИТМ ФЛОЙДА-УОРШАЛЛА: КРАТЧАЙШИЕ ПУТИ / ЛЕГЧАЙШИЕ ПРОХОДЫ [!KO\_flowar]](#_Toc197355392)
* [АЛГОРИТМ ФЛОЙДА-УОРШАЛЛА (СИММЕТРИЧНАЯ МАТРИЦА) [!KO\_sflowar]](#_Toc197355393)
* [АЛГОРИТМ ДЕЙКСТРЫ: КРАТЧАЙШИЙ ПУТЬ / ЛЕГЧАЙШИЙ ПРОХОД [!KO\_dijkstra]](#_Toc197355394)
* [АЛГОРИТМ МИНИМАЛЬНОГО ОСТОВНОГО ДЕРЕВА ПРИМА [!KO\_prim]](#_Toc197355395)

[ПОСЛЕДОВАТЕЛЬНОСТИ](#_Toc197355396)

* [РАССТОЯНИЕ ЛЕВЕНШТЕЙНА (АЛГОРИТМ ВАГНЕРА-ФИШЕРА) [!KO\_levenshtein]](#_Toc197355397)
* [ДЛИННЕЙШАЯ ОБЩАЯ ПОДПОСЛЕДОВАТЕЛЬНОСТЬ [!KO\_lcs]](#_Toc197355398)

# ОБЩИЕ СВЕДЕНИЯ

Матричный сеанс в SPSS Statistics – это серия особых команд, стоящих между командами MATRIX и END MATRIX (описание команд матричного языка см. в *SPSS Statistics Command Syntax Reference* – справке по синтаксису SPSS). В матричном сеансе вы оперируете данными как целыми матрицами, что позволяет быстро и эффективно выполнять преобразования, которые трудоемко программировать обычным (внематричными) синтаксисом. Матричный сеанс удобен для программирования собственных статистических и прочих алгоритмов. К сожалению, возможности матричного языка в SPSS долго не обновлялись; удобных функций явно недостаточно. Поэтому я написал ряд функций, которых до сих пор нет в матричном языке SPSS, но в которых вижу острую нужду. Эти функции написаны на самом матричном языке SPSS и оформлены в виде макросов. Следовательно, эти функции в общем не так быстры, как если бы они были написаны на низкоуровневом языке. Тем не менее, до тех пор пока IBM SPSS не обновила свой матричный язык радикально, я рекомендую эти функции пользователям.

***Правила***

1) Аргументы функции разделяются символом %, а не запятой.

2) Функции пускаются как команды, первым словом в строке – а не как «справа в выражении», и имя возвращаемого результата надо писать среди аргументов функции, в качестве последнего из них. Т.е. правильно: **!функция(аргумент % результат)**; неправильно: **результат = !функция(аргумент).** Некоторые функции выдают более одного результата.

3) Любой аргумент, кроме результата, может быть выражением, не только именем или значением. Например, **!функция(1.5\*имя(2,4)+3 % результат)** - допустимо. Выражение однако не может содержать функции, описанные в этом документе.

4) Имя результата может совпадать с именем любого из аргументов. Т.е. можно, например: **!функция(имя1 % имя2 % имя1).** В этом случае результат будет *имя1*, а то, что прежде носило это имя, пропадет.

5) Не используйте в вашем синтаксисе в MATRIX - END MATRIX имена, начинающиеся с символа @, поскольку этот символ зарезервирован здешними функциями, и его использование вами 1-м символом чревато потерей данных.

6) Функции *не проверяют и не выдают* особых сообщений насчет соответствия ваших входящих данных требованиям данной функции (так сделано в интересах быстродействия). Поэтому внимательно ознакомливайтесь с требованиями конкретных функций, чтобы избегать ошибок.

7) Функции можно ставить внутрь LOOP - END LOOP и DO IF - END IF.

8) Для отключения функции в синтаксисе используйте не \*, а /\*, например:

**/\*!func(arg%result).**

или

**/\*!func(arg%result).\*/.**

9) Эти матричные функции рассчитаны на числовые входящие данные. Некоторые из функций, не связанные с числовым анализом, *могут* работать с текстовыми данными (например, подсчет значений, перекодировка, переструктурирование), однако не гарантированно, т.к. код писался в расчете на числа. Пробуйте и проверяйте текстовые данные сами.

10) Многие из этих функций требуют заранее, до первой в сессии команды MATRIX, установить предельное число циклов на достаточно большое значение, чтобы справляться с потенциально большими данными. Я рекомендую делать заранее предустановку на большое число циклов вообще всегда, когда вы собираетесь работать со здешними функциями - чтобы не беспокоиться об этом дальше. Например:

**set mxloops 100000000.** /\*или 1E8

**matrix.**

**etc.**

# 

11) Дисперсия/ковариация и корреляция, когда вычисляются внутри функций, не всегда используют одну и ту же вычислительную формулу. Например, /\*!KO\_variance \*/ или /\*!KO\_corr\*/ применяют одну формулу (дающую более вычислительно устойчивый результат), тогда как другие функции могут калькулировать по другой формуле (дающей менее вычислительно устойчивый результат).

# МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

### НАКОПЛЕННЫЕ СУММЫ В СТОЛБЦАХ [!KO\_ccum]

\*/\*!KO\_ccum(mat%dir%name)\*/\*.

\*Version 1.

\*Берет матрицу или столбец MAT с числом рядов не меньше 2 и выдает матрицу NAME того же размера;

\*элементы в каждом столбце NAME есть накопленные суммы в этом столбце при направлении суммирования

\*сверху вниз (если DIR положительно) либо снизу вверх (если DIR неположительно).

\*DIR - скаляр.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ПРОИЗВЕДЕНИЕ ЭЛЕМЕНТОВ МАТРИЦЫ [!KO\_prod]

\*/\*!KO\_prod(mat%name)\*/\*.

\*Version 2.

\*Перемножает элементы MAT.

\*Выдает скаляр NAME.

ПРИМЕР. Факториал 6.

matrix.

!KO\_prod({1:6}%fact).

print fact.

end matrix.

### СУММА НАДДИАГОНАЛЬНЫХ ИЛИ ПОДДИАГОНАЛЬНЫХ ЭЛЕМЕНТОВ МАТРИЦЫ [!KO\_trsum]

\*/\*!KO\_trsum(mat%tr%name)\*/\*.

\*Version 1.

\*Суммирует элементы верхнего или нижнего треугольника MAT.

\*Выдает скаляр NAME.

\*Матрица MAT не обязана быть квадратной, но должна быть размером не меньше 2x2.

\*TR - скаляр; если положительный, суммируются значения над главной диагональю;

\*если неположительный, суммируются значения под главной диагональю.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ПРОВЕРКА ТРЕУГОЛЬНОГО НЕРАВЕНСТВА В МАТРИЦЕ РАЗЛИЧИЙ [!KO\_trineq]

\*/\*!KO\_trineq(dis%ineq%stop%name1%name2)\*/\*.

\*Version 1.

\*Берет квадратную матрицу расстояний (различий) DIS и проверяет, есть ли в ней расстояния,

\*нарушающие "аксиому треугольного неравенства". Если такие есть, то расстояния не являются метрическими.

\*Функция проверяет только наддиагональные элементы DIS; если вы хотите проверить

\*поддиагональные элементы DIS - транспонируйте матрицу.

\*DIS - квадратная с неотрицательными значениями, размером не менее 3.

\*INEQ - цифра 0 или 1 (не имя и не выражение; цифру можно опционально взять в кавычки или апострофы).

\*Если 0, проверяется нестрогое треугольное неравенство: Dab <= Dac+Dbc (где a b c - объекты, т.е.

\*ряды/столбцы матрицы). Если 1, проверяется строгое треугольное неравенство: Dab < Dac+Dbc (Dab = Dac+Dbc

\*не засчитывается как нарушение).

\*STOP - цифра 0 или 1 (не имя и не выражение; цифру можно опционально взять в кавычки или апострофы).

\*Если 0, проверяются все наддиагональные элементы: функция работает до конца. Если 1, функция обрывается

\*после встречи первого же нарушения треугольного неравенства, если есть.

\*Результаты:

\*NAME1 - вектор-ряд длиной 2: первое значение - число троек (сочетаний элементов по три), проверенных

\*над диагональю матрицы; второе значение - число троек из них с нарушенным треугольным

\*неравенством. (При STOP=1 это значение всегда =1.)

\*NAME2 - вектор-ряд длиной 4; он содержит максимальную найденную величину нарушения треугольного

\*неравенства (1-е значение вектора) и номера трех объектов (рядов/столбцов матрицы), составляющих

\*эту тройку. Если нарушений найдено не было, NAME2 состоит из -1.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1000000.

matrix.

compute dis= {0, 12, 10, 17;

12, 0, 22, 8;

10, 22, 0, 13;

17, 8, 13, 0}.

print dis.

!KO\_trineq(dis%'1'%'0'%name1%name2).

print name1 /title 'Number of triplets of (above-diagonal) elements checked'+

' and number of defective triplets'.

print name2 /title 'The size of maximal defect encountered and the indices of'+

' corresponding three objects'.

end matrix.

### НИЖНЯЯ НЕПОЛНАЯ ГАММА-ФУНКЦИЯ [!KO\_ligamma]

\*/\*!KO\_ligamma(arg%ubound%name)\*/\*.

\*Version 1.

\*Выдает значение нижней неполной гамма функции для аргумента ARG и верхнего предела

\*интегрирования UBOUND.

\*ARG - положительное число или числа: скаляр, вектор или матрица.

\*UBOUND - неотрицательное число или числа: матрица размером как ARG.

\*Выдает матрицу NAME размером как ARG.

\*Для этой функции требуется предварительная установка предела числу циклов на 100 (или больше)

\*командой SET MXLOOPS.

ПРИМЕР.

set mxloops 100.

matrix.

!KO\_ligamma(0.5%12.6%result).

print result.

end matrix.

# 

### QR-РАЗЛОЖЕНИЕ МЕТОДОМ ОТРАЖЕНИЙ ХАУСХОЛДЕРА [!KO\_qrdc]

\*/\*!KO\_qrdc(mat%name1%name2)\*/\*.

\*Version 1.

\*Делает QR-разложение матрицы MAT на верхнетреугольную матрицу R (NAME2) и ортонормированную

\*матрицу Q (NAME1) методом отражений Хаусхолдера. Q\*R=MAT.

\*MAT - матрица с числом столбцов не меньше, чем число рядов.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Встроенная функция GSCH (ортонормирование Грама-Шмидта) - тоже QR-разложение, но использующее другой

\*метод и дающее иной результат.

### ХЕССЕНБЕРГОВА МАТРИЦА / ТРИДИАГОНАЛЬНАЯ МАТРИЦА [!KO\_hess]

\*/\*!KO\_hess(mat%name1%name2)\*/\*.

\*Version 1.

\*Превращает квадратную матрицу MAT (размером не меньше 3) в верхнюю матрицу Хессенберга NAME2.

\*Эта матрица верхнетреугольная с заполненной также верхней поддиагональю, и ее собственные

\*числа те же, что у MAT. Выдается также ортонормированная матрица NAME1, такая,

\*что NAME1\*MAT\*t(NAME1)=NAME2 и t(NAME1)\*NAME2\*NAME1=MAT.

\*Если MAT симметричная матрица, то NAME2 получается тридиагональной.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### СОБСТВЕННОЧИСЛОВОЕ РАЗЛОЖЕНИЕ 2x2 МАТРИЦЫ [!KO\_eig2x2]

\*/\*!KO\_eig2x2(mat%name1%name2)\*/\*.

\*Version 1.

\*Выдает (вещественные) собственные числа (NAME2) и собственные векторы (столбцы NAME1) квадратной

\*матрицы MAT размером 2x2. MAT может быть симметричной или асимметричной; в последнем случае

\*собственные векторы могут быть неортогональны (используйте ф-цию /\*!KO\_schur2x2\*/ для получения

\*ортогональных собственных векторов).

\*Восстановление матрицы: MAT = NAME1\*mdiag(NAME2)\*inv(NAME1).

\*Если собственные числа MAT не вещественные, NAME1 и NAME2 выдаются как нулевые скаляры.

ПРИМЕР.

matrix.

compute A= {4,-2;-1,3.6}.

print A.

!KO\_eig2x2(A%eivec%eival).

print eival. /\*eigenvalues L

print eivec. /\*eigenvectors V

print sscp(eivec). /\*V are not orthogonal if A is asymmetric

print ( eivec\*mdiag(eival)\*inv(eivec) ). /\*A restored

end matrix.

### СОБСТВЕННОЧИСЛОВОЕ РАЗЛОЖЕНИЕ 2x2 МАТРИЦЫ (ВАРИАНТ ШУРА) [!KO\_schur2x2]

\*/\*!KO\_schur2x2(mat%name1%name2)\*/\*.

\*Version 1.

\*Выдает (вещественные) собственные числа и собственные векторы квадратной матрицы MAT размером 2x2.

\*Собственные числа находятся на диагонали верхнетреугольной матрицы NAME2, а собственные векторы

\*это столбцы NAME1. Это ортогональные собственные векторы и NAME1 это унитарная матрица.

\*Восстановление матрицы: MAT = NAME1\*NAME2\*t(NAME1).

\*Если MAT симметрична, то NAME2 диагональная матрица.

\*Если собственные числа MAT не вещественные, NAME1 и NAME2 выдаются как нулевые скаляры.

\*Для шуровского разложения матрицы большего размера используйте /\*!KO\_qreig\*/.

### СОБСТВЕННОЧИСЛОВОЕ РАЗЛОЖЕНИЕ (ВАРИАНТ ШУРА) [!KO\_qreig]

\*/\*!KO\_qreig(mat%needvec%conv%name1%name2%name3)\*/\*.

\*Version 1.

\*Делает собственночисловое разложение квадратной матрицы MAT (симметричной или асимметричной).

\*Используемый алгоритм - Implicit QR iterations on recursively splitted Hessenberg matrix with

\*double (Francis) shifts. Функция выдает верхнетреугольную матрицу NAME2 (с собственными числами на

\*диагонали, матрица Шура) и унитарную матрицу NAME1, столбцы которой – взаимоортогональные

\*собственные векторы (вернее, эти ортогональные векторы называются векторами Шура).

\*Порядок собственных векторов отвечает порядку собственных чисел, но собственные числа не обязательно

\*сортированы по величине. Если MAT симметричная, то NAME2 диагональна.

\*Восстановление матрицы: MAT = NAME1\*NAME2\*t(NAME1).

\*Аргумент NEEDVEC - цифра (не имя и не выражение) 1 или 0 (цифру можно взять в кавычки или апострофы).

\*Если "1", то функция выдаст собственные числа NAME2 и собственные векторы NAME1. Если "0", то

\*функция выдаст только собственные числа NAME2, не тратя время на получение собственных векторов

\*(NAME1 выйдет как скаляр 1).

\*Аргумент CONV - скаляр; это задаваемая точность схождения при вычислении собственных чисел;

\*рекомендуемое значение: от 1E-8 до 1E-12.

\*NAME3 - этот скаляр на выходе есть максимальное потребовавшееся число итераций для вычисления одного

\*собственного числа.

\*Несхождение/неудача. Асимметричные матрицы MAT иногда имеют комплексные (невещественные) собственные

\*числа, связанные в пары. Если некий поддиагональный элемент в NAME2 оказался ненулевой, а NAME3 не

\*более 100 (предельное число итераций в функции), то налицо пара комплексных собственных чисел.

\*Вещественные части этих двух значений находятся над и справа от несошедшегося к нулю поддиагонального

\*элемента, однако они не обязательно вычислены точно. Если поддиагональный элемент в NAME2 оказался

\*ненулевой и NAME3=101 (т.е. исчерпаны все итерации), то налицо неудача алгоритма - это сравнительно

\*редкий случай, связанный с особой дефектностью данной MAT.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если MAT размером 2x2, функция /\*!KO\_schur2x2\*/ делает то же разложение, что и эта функция.

\*Если MAT симметрична, встроенная функция CALL EIGEN делает то же разложение, что и эта функция.

ПРИМЕР. Разложение случайной матрицы с заданными собственными числами.

set mxloops 10000.

matrix.

compute L= (uniform(10,1)-.3)\*2. /\*Some random values to use as eigenvalues

print L.

compute s= uniform(10,10)-.3.

compute A= s\*mdiag(L)\*inv(s). /\*Random matrix A with eigenvalues L

print A.

!KO\_qreig(A%1%1E-10%eivec%schur%mxit).

print mxit.

print schur. /\*Schur matrix T: normally upper triangular matrix with eigenvalues

print diag(schur). /\*occupying its diagonal

print eivec. /\*Columns are corresponding eigenvectors U

print sscp(eivec). /\*which are orthogonal

print ( eivec\*schur\*t(eivec) ). /\*Reconstruction of A: U\*T\*t(U)

end matrix.

ПРИМЕР. Матрица с комплексными собственными числами.

set mxloops 10000.

matrix.

compute A=

{7, 3, 4, -11, -9, -2;

-6, 4, -5, 7, 1, 12;

-1, -9, 2, 2, 9, 1;

-8, 0, -1, 5, 0, 8;

-4, 3, -5, 7, 2, 10;

6, 1, 4, -11, -7, -1}. /\*Matrix with two real eigenvalues (3, 4) and

/\*two complex conjugate paired eigenvalues (1±2i, 5±6i)

print A.

!KO\_qreig(A%0%1E-10%eivec%schur%mxit).

print mxit.

print schur. /\*Because of complex eigenvalues, lower triangle of the Schur matrix

/\*is not completely empty; still, out of 6 eigenvalues both real and

/\*one pair of complex ones converged to their true values

end matrix.

### ДОМИНАНТНОЕ СОБСТВЕННОЕ ЧИСЛО И ЕГО СОБСТВЕННЫЙ ВЕКТОР [!KO\_poweig]

\*/\*!KO\_poweig(mat%inivec%conv%name1%name2%name3)\*/\*.

\*Version 1.

\*Берет квадратную n x n матрицу MAT и выдает ее доминантное собственное число NAME2

\*(т.е. собственное число наибольшей абсолютной величины) и связанный с ним собственный

\*вектор NAME1. Используется степенной итеративный метод (power iteration method).

\*Алгоритм сходится, если доминантное собственное число существует в единственном экземпляре,

\*и сходится тем скорее, чем больше оно доминирует над прочими собственными числами (т.е. велика

\*"спектральная щель"). По теореме Перрона-Фробениуса, всякая положительная квадратная

\*матрица и некоторые (т.н. примитивные) неотрицательные квадратные матрицы имеют доминантное

\*собственное число строго одного экземпляра (а все элементы его собственного вектора - одного знака).

\*Поэтому в случаях этих матриц схождение будет всегда.

\*INIVEC - начальный для итераций собственный вектор, столбец длиной n. Обычно это случайные числа

\*из равномерного распределения. Если у вас есть более точная исходная оценка собственного вектора,

\*укажите ее в качестве INIVEC.

\*CONV - параметры итеративного схождения: вектор из двух значений, например {1E-9,100}. Первое

\*это порог изменения собственного вектора - неотрицательное число; если наибольшее изменение в нем

\*будет меньше порога, вектор считается стабилизировавшимся и итерации прекращаются.

\*Второе это максимально позволенное число итераций - задайте целое положительное число.

\*NAME3 - отчет о схождении, вектор из двух чисел, который можно сравнить с аргументом CONV:

\*первое число это максимальное изменение в собственном векторе, случившееся на последней

\*проделанной итерации и которое, ожидается, должно быть меньше порога в CONV. Второе число - число

\*проделанных итераций; если это число больше на 1 чем заданное в CONV, значит, было проделано

\*максимальное число итераций, но заказанная точность схождения так и не была достигнута.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

# СТАТИСТИЧЕСКИЕ ОПИСАТЕЛЬНЫЕ ФУНКЦИИ

### ВЕКТОР СРЕДНИХ АРИФМЕТИЧЕСКИХ [!KO\_mean]

\*/\*!KO\_mean(data%name)\*/\*.

\*Version 1.

\*Вычисляет средние в столбцах DATA.

\*Выдает вектор-ряд NAME.

### ВЕКТОР ДИСПЕРСИЙ [!KO\_variance]

\*/\*!KO\_variance(data%name)\*/\*.

\*Version 1.

\*Вычисляет дисперсии в столбцах DATA.

\*Выдает вектор-ряд NAME.

### ВЕКТОР ДИСПЕРСИЙ (DF=N) [!KO\_variance2]

\*/\*!KO\_variance2(data%name)\*/\*.

\*Version 1.

\*Вычисляет смещенные (посчитанные на "df=n", а не "df=n-1") дисперсии в столбцах DATA.

\*Выдает вектор-ряд NAME.

### ВЕКТОР АСИММЕТРИЙ [!KO\_skewness]

\*/\*!KO\_skewness(data%name)\*/\*.

\*Version 1.

\*Вычисляет скошенности в столбцах DATA.

\*Выдает вектор-ряд NAME.

### ВЕКТОР ЭКСЦЕССОВ [!KO\_kurtosis]

\*/\*!KO\_kurtosis(data%name)\*/\*.

\*Version 1.

\*Вычисляет эксцессы (хвостности) в столбцах DATA. В DATA должно быть не менее трех рядов.

\*Выдает вектор-ряд NAME.

### ВЕКТОР МЕДИАН [!KO\_median]

\*/\*!KO\_median(data%name)\*/\*.

\*Version 1.

\*Вычисляет медианы в столбцах DATA.

\*Выдает вектор-ряд NAME.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 100000.

matrix.

get vars /variables= v1 to v3.

!KO\_median(vars%medians).

print medians.

end matrix.

### ПРОЦЕНТИЛИ [!KO\_ptile]

\*/\*!KO\_ptile(data%sort%p%method%name)\*/\*.

\*Version 1.

\*Вычисляет заказанные процентили в столбцах DATA.

\*Выдает матрицу NAME с рядами, отвечающими процентильным уровням.

\*SORT - цифра (не имя и не выражение) 1 или 0 (опционально в кавычках или апострофах). Используйте 0

\*лишь в том особом случае, когда значения в каждом столбце DATA уже сортированы по возрастанию; эта

\*опция позволяет не тратить время на сортировку.

\*P - скаляр или вектор (ряд или столбец), содержащий проценты (процентные уровни, процентильные ранги),

\*для которых вам нужно определить значения процентией в переменных (каждом столбце DATA).

\*Значения в P должны быть больше 0 и меньше 100. Они не обязаны следовать по возрастающей.

\*METHOD - метод вычисления процентилей, заглавное ключевое слово (опционально в кавычках или

\*апострофах):

\*"WAVER" - Weighted Average Definition 2

\*"HAVER" - Weighted Average Definition 1

\*"HAVER2" - Weighted Average Definition 1

\*"EMPIR" - Empirical Distribution Function

\*"AEMPIR" - Empirical Distribution Function with Averaging

\*"ROUND" - Observation closest aka Rounded

\*Это те же методы, что находятся в SPSS-процедуре EXAMINE (Explore). HAVER и HAVER2 это одно и то же,

\*только HAVER2 не вычисляет "слишком высокие" для входящих данных процентили (заменяя их в NAME на

\*условное число -999), так же поступают SPSS-процедуры Explore и Frequencies для дефолтного

\*метода "HAVERAGE". HAVER вместо "-999" ставит максимальное в переменной значение.

\*Наиболее популярные и рекомендуемые методы - HAVER/HAVER2 и AEMPIR. 50-й процентиль этих методов

\*совпадает с медианой ф-ции /\*!KO\_median\*/.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 100000.

matrix.

get vars /variables= v1 v2 v3.

compute plevels= {5,25,50,90}. /\*Request percentiles for

!KO\_ptile(vars%1%plevels%AEMPIR%ptiles). /\*5-th, 25-th, 50-th, 90-th %

print {t(plevels),ptiles} /clabel= '%-level' 'v1' 'v2' 'v3'.

end matrix.

### МИНИМУМ, МАКСИМУМ, СРЕДНЯЯ И ДИСПЕРСИЯ, ПО ГРУППАМ [!KO\_gdescr]

\*/\*!KO\_gdescr(col%bin%name1%name2%name3%name4)\*/\*.

\*Version 2.

\*Берет столбец COL (одна анализируемая переменная) и одну или более двоичных (1 vs 0)

\*переменных, являющихся столбцами матрицы BIN с числом рядов как в COL.

\*Каждый столбец BIN означает группу, и значение 1 означает, что наблюдение (ряд) принадлежит ей.

\*Группы могут пересекаться или непересекаться составом наблюдений (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей, в этом случае группой

\*выступит весь массив, все ряды: вычислится статистика "в целом".

\*Функция выдает 4 столбца по-группных однопеременных статистик, с рядами, соответствующими группам, столбцам BIN:

\*наименьшее значение (NAME1), наибольшее значение (NAME2), средняя арифметическая (NAME3),

\*дисперсия (на "df=n-1", NAME4). NAME4 не будет вычислено (с сообщением об ошибке деления на 0),

\*если какой-л столбец BIN содержит менее двух единиц.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

ПРИМЕР. Базовая статитстика для количественной переменной, по группам.

matrix.

get var /variable= v1.

get group /variable= gr. /\*Grouping variable

!KO\_freq(group%1%dummy%freq%codes).

!KO\_gdescr(var%dummy%min%max%mean%variance).

print {codes,freq,min,max,mean,variance}

/clabels= 'group' 'n' 'min' 'max' 'mean' 'variance'.

end matrix.

### РАЗНЫЕ ОДНОПЕРЕМЕННЫЕ СТАТИСТИКИ, ПО ГРУППАМ [!KO\_aggr]

\*/\*!KO\_aggr(data%bin%stat%name)\*/\*.

\*Version 1.

\*Вычисляет требуемую описательную статистику в столбцах DATA, в целом и/или по группам

\*наблюдений (рядов).

\*Эта функция напоминает внематричную команду AGGREGATE.

\*Берет данные DATA (матрица с любым числом столбов, переменных) и одну или более двоичных (1 vs 0)

\*переменных, являющихся столбцами матрицы BIN с числом рядов как в DATA.

\*Каждый столбец BIN означает группу, и значение 1 означает, что наблюдение (ряд) принадлежит ей.

\*Группы могут пересекаться или непересекаться составом наблюдений (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей, в этом случае группой

\*выступит весь массив, все ряды: вычислится статистика "в целом".

\*Результат NAME - матрица со столбцами, отвечающими столбцам DATA, т.е. переменным, и рядами,

\*отвечающими столбцам BIN, т.е. группам. NAME содержит значения вычисленной статистики для каждой

\*переменной в каждой группе.

\*STAT - требуемая статистика. Это ключевое слово заглавными буквами. Кавычки или апострофы вокруг

\*слова - опциональны. Следующие статистики возможны:

\*"SUM" - сумма

\*"MEAN" - средняя арифметическая

\*"SSDEV" - сумма квадратов отклонений

\*"VARIANCE" - дисперсия [сумма в каждом столбце BIN должна быть >1]

\*"VARIANCE2" - дисперсия, посчитанная на "df=n", а не "df=n-1"

\*"SEMEAN" - ст. ошибка средней арифметической [сумма в каждом столбце BIN должна быть >1]

\*"MIN" - наименьшее значение

\*"MAX" - наибольшее значение

\*"FIRST" - первое значение

\*"LAST" - последнее значение

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

\*Если хотите заместить вычисленной статистикой значения в DATA, используйте функцию /\*!KO\_aggrv\*/.

ПРИМЕР. Средняя в целом и по группам.

matrix.

get vars /variables= v1 v2 /names= names.

get grvar /variable= group. /\*Categorical grouping variable

!KO\_freq(grvar%1%dummy%freq%codes). /\*Create dummy variables

compute dummy= {make(nrow(dummy),1,1),dummy}. /\*Add constant 1 as the 1st column

!KO\_aggr(vars%dummy%'MEAN'%table). /\*Compute means

compute names= {'Group',names}.

print {{9999;codes},table} /format= f8.4 /cnames= names

/title= 'Means: Groups X Variables table (Group=9999 is Total sample):'.

end matrix.

ПРИМЕР. Подсчет числа значений, удовлетворяющих некоему условию, в группах.

matrix.

get vars /variables= v1 v2 /names= names. /\*Get some variable(s)

get grvar /variable= group. /\*Categorical grouping variable

!KO\_freq(grvar%1%dummy%freq%codes). /\*Create dummy variables

print codes /title 'Group codes'.

compute flag= vars>.2 and vars<.9. /\*Compute binary matrix which flags (by 1s)

/\*values in VARS which are >.2 but <.9 (some condition)

!KO\_aggr(flag%dummy%'SUM'%table). /\*Compute sum of flagged elements

print table /cnames= names

/title 'Count, by groups, of values which are >.2 but <.9:'.

end matrix.

### ЗНАЧЕНИЕ, БЛИЖАЙШЕЕ К ЗАДАННОМУ СНИЗУ ИЛИ СВЕРХУ [!KO\_closest]

\*/\*!KO\_closest(data%ineq%val%name)\*/\*.

\*Version 1.

\*В каждом столбце данных DATA выясняет наблюдаемое значение, ближайшее к значению VAL по величине;

\*а условие INEQ оговаривает, должно оно быть меньше или больше, чем VAL.

\*INEQ это оператор неравенства, ключевое слово (можно взять в кавычки или апострофы)

\*на выбор: >, >=, <, <= (или буквами: GT, GE, LT, LE). Например, <= означает выяснить наибольшее

\*наблюдаемое значение, которое <=VAL, т.е. прилегает к нему "снизу" и либо меньше, либо равно ему.

\*VAL - скаляр или вектор-ряд длиной как число столбцов DATA. Вектор означает задание отдельного числа

\*для каждого столбца DATA.

\*Результат - ряд NAME, содержащий искомые значения. Значение 1E+300 либо 1E-300 в NAME говорит о том,

\*что в данном столбце DATA не нашлось значений, удовлетворяющих условию INEQ VAL.

ПРИМЕР.

matrix.

compute data= rnd((uniform(8,5)-.3)\*15). /\*Some random data, 5 variables

print data /format f8.

!KO\_mean(data%mean).

print mean /format f8.3 /title 'Column means'.

!KO\_closest(data%GT%mean%name).

print name /format f8 /title 'Smallest values that are Greater Than the means'.

!KO\_closest(data%LE%0%name).

print name /format f8 /title 'Largest values that are Lesser or Equal to 0'.

end matrix.

### КОВАРИАЦИОННАЯ МАТРИЦА [!KO\_cov]

\*/\*!KO\_cov(data%name)\*/\*.

\*Version 1.

\*Вычисляет ковариации между столбцами DATA.

\*Выдает квадратную симметрическую матрицу NAME.

ПРИМЕР.

matrix.

get vars /variables= v1 v2 v3 /names= names.

!KO\_cov(vars%cov).

print cov /rnames= names /cnames= names.

end matrix.

### КОВАРИАЦИОННАЯ МАТРИЦА (DF=N) [!KO\_cov2]

\*/\*!KO\_cov2(data%name)\*/\*.

\*Version 1.

\*Вычисляет ковариации между столбцами DATA. Значения посчитаны на "df=n" (а не "df=n-1").

\*Выдает квадратную симметрическую матрицу NAME.

### КОРРЕЛЯЦИОННАЯ МАТРИЦА [!KO\_corr]

\*/\*!KO\_corr(data%name)\*/\*.

\*Version 1.

\*Вычисляет корреляции Пирсона между столбцами DATA.

\*Выдает квадратную симметрическую матрицу NAME.

### КОСИНУСНАЯ МАТРИЦА [!KO\_cosine]

\*/\*!KO\_cosine(data%name)\*/\*.

\*Version 1.

\*Вычисляет косинусы-сходства (коэффициент пропорциональности, или конгруентности Такера)

\*между столбцами DATA.

\*Выдает квадратную симметрическую матрицу NAME.

### МАТРИЦА КОЭФФИЦИЕНТОВ ТОЖДЕСТВЕННОСТИ [!KO\_idc]

\*/\*!KO\_idc(data%name)\*/\*.

\*Version 1.

\*Вычисляет сходства-коэффициенты тождественности (идентичности) [Zegers, ten Berge, 1985]

\*между столбцами DATA.

\*Выдает квадратную симметрическую матрицу NAME.

### МАТРИЦА ОТНОШЕНИЙ ПОДОБИЯ [!KO\_simr]

\*/\*!KO\_simr(data%name)\*/\*.

\*Version 1.

\*Вычисляет отношения подобия (similarity ratio), известные также как сходства Kohonen,

\*между столбцами DATA.

\*Выдает квадратную симметрическую матрицу NAME.

### КОВАРИАЦИЯ, КОРРЕЛЯЦИЯ, КОСИНУС, КОЭФФИЦИЕНТ ТОЖДЕСТВЕННОСТИ, ОТНОШЕНИЕ ПОДОБИЯ [!KO\_biv]

\*/\*!KO\_biv(col1%col2%name)\*/\*.

\*Version 2.

\*Для двух векторов-столбцов COL1 и COL2 (одинаковой длины) вычисляет ряд из пяти статистик сходства:

\*коэффициент ковариации (1-й элемент NAME), коэффициент корреляции Пирсона (2-й элемент), коэффициент

\*пропорциональности (косинус, 3-й элемент), коэффициент тождественности (4-й элемент),

\*отношение подобия (5-й элемент).

### ПРЯМОУГОЛЬНАЯ МАТРИЦА КОВАРИАЦИЙ/КОРРЕЛЯЦИЙ/КОСИНУСОВ [!KO\_rect]

\*/\*!KO\_rect(vars1%vars2%type%name)\*/\*.

\*Version 1.

\*Вычисляет заказанный коэффициент ассоциации между столбцами VARS1 и столбцами VARS2.

\*Эти две матрицы должны иметь одинаковое число рядов (ряды это наблюдения, а столбцы это переменные).

\*Выдает прямоугольную матрицу NAME размером число\_столбцов\_VARS1 x число\_столбцов\_VARS2.

\*TYPE - заказанная мера связи, заглавное кл. слово (опционально можно в кавычках или апострофах):

\*"COV" - коэффициент ковариации (на "df=n-1")

\*"COV2" - коэффициент ковариации (на "df=n")

\*"CORR" - коэффициент корреляции Пирсона

\*"COSINE" - косинус-сходство (коэффициент конгруэнтности Такера).

### КОВАРИАЦИОННАЯ/КОРРЕЛЯЦИОННАЯ МАТРИЦА, ПО ГРУППАМ И УСРЕДНЕННАЯ [!KO\_gcov]

\*/\*!KO\_gcov(data%bin%type%out%name)\*/\*.

\*Version 2.

\*Вычисляет рассеянную (scatter), ковариационную или корреляционную матрицу переменных,

\*по группам наблюдений, а также усредненную (объединенную, pooled).

\*Берет данные DATA (матрица с p столбцами-переменными, p>=1) и одну или более двоичных (1 vs 0)

\*переменных, являющихся столбцами матрицы BIN с числом рядов как в DATA.

\*Каждый столбец BIN означает группу, и значение 1 означает, что наблюдение (ряд) принадлежит ей.

\*Группы могут пересекаться или непересекаться составом наблюдений (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей, в этом случае группой

\*выступит весь массив, все ряды: вычислится матрица для всей выборки в целом.

\*Аргументы TYPE и OUT это ключевое слово заглавными буквами. Кавычки или апострофы вокруг

\*слова - опциональны.

\*TYPE - требуемый тип матрицы. Следующие типы возможны:

\*"SCAT" - матрица рассеяния

\*"COV" - ковариационная матрица [внимание: сумма во всяком столбце BIN должна быть >1]

\*"COV2" - ковариационная матрица, посчитанная на "df=n", а не "df=n-1"

\*"CORR" - корреляционная матрица [внимание: сумма во всяком столбце BIN должна быть >1].

\*OUT - этот аргумент определяет состав выдаваемого результата NAME:

\*"BYGR" - матрицы, своя для каждой группы, столбца BIN; матрицы соединены в стопку одна под другой,

\*их порядок отвечает порядку столбцов BIN

\*"POOL" - объединенная (pooled) матрица групп

\*"BOTH" - и то и другое (pooled матрица в стопке будет верхней).

\*Что есть объединенная матрица в зависимости от типа матрицы:

\*- объединенная матрица рассеяния это суммированная матрица рассеяния групп;

\*- объединенная ковариационная матрица это взвешенно усредненная ковариационная матрица групп,

\*где весом выступает число степеней свободы ("n-1" при COV и "n" при COV2, где n численность в группе);

\*- объединенная корреляционная матрица получается из объединенной ковариационной матрицы

\*(или из объединенной матрицы рассеяния), а не усреднением корреляционных матриц групп.

\*При вычислении объединенной матрицы любого типа функция за совокупную численность выборки принимает

\*сумму численностей в группах, т е число единиц в BIN, - а не число рядов BIN (ведь в BIN некоторые

\*ряды могут быть пусты или могут входить более чем в одну группу).

\*Поскольку, если DATA имеет лишь один столбец, COV (или COV2) матрица обращается в скаляр, дисперсию,

\*то вы можете применять данную функцию и для вычисления усредненных дисперсий групп.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

ПРИМЕР.

set mxloops 1E9.

matrix.

get vars /variables= v1 to v5.

get group /var= gr. /\*Categorical grouping variable

!KO\_freq(group%1%dummy%freq%codes). /\*Create dummy variables

!KO\_gcov(vars%dummy%COV%BOTH%m).

print m(1:ncol(m),:) /title 'Pooled within-group covariance matrix'.

print m((ncol(m)+1):nrow(m),:)

/title 'Within-group covariance matrices, stacked one under another'.

print codes /title 'The group codes'.

end matrix.

### МЕЖГРУППОВАЯ И ОБЪЕДИНЕННАЯ ВНУТРИГРУППОВАЯ МАТРИЦЫ РАССЕЯНИЯ [!KO\_bwscat]

\*/\*!KO\_bwscat(data%dummy%name1%name2)\*/\*.

\*Version 1.

\*Берет данные DATA и группы DUMMY и выдает межгрупповую матрицу рассеяния (between-group

\*scatter matrix) NAME1 и объединенную (суммативную) внутригрупповую матрицу рассеяния

\*(pooled within-group scatter matrix) NAME2.

\*(SSbetween = trace(NAME1) и SSwithin = trace(NAME1).)

\*DATA - данные с любым числом столбцов (переменных).

\*NAME1 (внимание!) зависит от того, центрованы или нет

\*были столбцы DATA, но NAME2 не зависит (за исключением вычислительной устойчивости, ради лучше

\*при центрованных данных).

\*DUMMY - двоичные фиктивные переменные, помечающие группы (каждая переменная соответствует группе,

\*значение 1 = наблюдение принадлежит ей, 0 = не принадлежит ей).

\*Всякий столбец DUMMY должен содержать по меньшей мере одну единицу.

\*Группы должны быть непересекающиеся составом: сумма в рядах DUMMY не может превосходить единицу.

\*Если в DUMMY есть ряды, полные нулей (наблюдения вне групп), то NAME2 не будет корректна.

\*Выдаваемые матрицы складываются в sscp-матрицу переменных всего массива: NAME1+NAME2=sscp(DATA).

\*Фиктивные переменные DUMMY вы можете создать из категориальной группирующей переменной

\*с помощью функции /\*!KO\_freq\*/.

\*Если вам нужна объединенная внутригрупповая матрица рассеяния от групп, пересекающихся составом,

\*используйте ф-цию /\*!KO\_gcov\*/.

ПРИМЕР. Некоторые главные статистики однофакторного MANOVA.

matrix.

get vars /variables= v1 to v5.

get group /var= gr. /\*Categorical grouping variable

!KO\_freq(group%0%dummy%freq%codes). /\*Create dummies

!KO\_center(vars%vars). /\*Center the data

!KO\_bwscat(vars%dummy%b%w).

print (det(w)/det(b+w)) /title "Wilks' lambda".

print trace(inv(w)\*b) /title "Hotelling's trace".

print trace(b\*(inv(b+w))) /title "Pillai's trace".

end matrix.

### МАТРИЦА ОБРАЗОВ ИЛИ АНТИОБРАЗОВ [!KO\_image]

\*/\*!KO\_image(cov%type%name)\*/\*.

\*Version 1.

\*Берет ковариационную или корреляционную матрицу COV и выдает как NAME ковариационную

\*матрицу образов (если TYPE положительный скаляр) или антиобразов (если TYPE неположительный скаляр).

\*COV должна быть квадратной симметрической невырожденной, в норме - положительно определенной.

\*TYPE - скаляр.

\*На диагонали матрицы образов находятся "образы", равные Rsq\*Ssq, где Rsq это квадрат множественной

\*корреляции для данной переменной как зависимой от остальных, а Ssq диагональный элемент COV.

\*Внедиагональные элементы матрицы антиобразов это (с обратным знаком) коэффициенты частной ковариации.

\*Если обратить знак внедиагональных элементов матрицы антиобразов и сложить ее с матрицей образов,

\*восстановится исходная матрица COV.

\*Эту функцию можно использовать для любой матрицы sscp-типа: корреляционной, ковариационной, косинусной,

\*сырой sscp.

### МАТРИЦА ВЛИЯНИЯ (HAT-МАТРИЦА) [!KO\_hat]

\*/\*!KO\_hat(data%name)\*/\*.

\*Version 1.

\*Берет матрицу данных DATA, которая должна быть с неколлинеарными столбцами.

\*При необходимости прежде центруйте или стандартизуйте столбцы DATA.

\*Выдает матрицу влияния (hat-матрицу, проекционную матрицу) NAME.

\*Значения на ее диагонали известны как рычаги (leverages).

\*Если DATA (имеющая n рядов) имеет центрованные столбцы, то NAME\*(n-1) есть двойной

\*центрат (см. ф-цию /\*!KO\_dcenter\*/) матрицы квадратных расстояний Махаланобиса между рядами DATA,

\*а leverages\*(n-1) есть квадратные расстояния Махаланобиса до центроида DATA.

### МАТРИЦЫ СОЧАСТОТ ДЛЯ ДВОИЧНЫХ ДАННЫХ [!KO\_bincnt]

\*/\*!KO\_bincnt(data%name1%name2)\*/\*.

\*Version 1.

\*Берет матрицу двоичных (значения 0 и 1) данных DATA и выдает две квадратные матрицы с сочастотами

\*между столбцами DATA:

\*NAME1 - симметричная матрица, содержащая в ячейке ij число рядов DATA, в которых столбец i

\*содержит 1 и столбец j содержит 1. Эти частоты известны как частоты A (признак есть в обоих столбцах).

\*NAME2 - асимметричная матрица, содержащия в ячейке ij число рядов DATA, в которых столбец i

\*содержит 0 и столбец j содержит 1. Эти частоты известны как частоты B (признак есть в одном и

\*отсутствует в другом столбце).

\*Частоты C есть транспонат B.

\*Частоты D (признака нет в обоих столбцах) = число\_рядов\_DATA-A-B-C.

\*Частоты A, B, C, D полезны для вычисления разнообразных мер близости, придуманных для двоичных данных.

ПРИМЕР. Вычисление между столбцами 1) сходства Жаккара и 2) различия паттернов.

matrix.

compute data= rnd(uniform(10,5)).

print data.

!KO\_bincnt(data%a%b).

compute c= t(b).

compute d= nrow(data)-a-b-c.

compute jaccard= a/(a+b+c).

print jaccard /title 'Jaccard similarity matrix '.

compute pattern= (b&\*c)/(a+b+c+d)&\*\*2.

print pattern /title 'Pattern difference matrix '.

end matrix.

### МАТРИЦА КВАДРАТНЫХ ЕВКЛИДОВЫХ РАССТОЯНИЙ [!KO\_seuclid]

\*/\*!KO\_seuclid(data%name)\*/\*.

\*Version 2.

\*Вычисляет квадратные евклидовы расстояния между рядами DATA.

\*Выдает квадратную симметрическую матрицу NAME.

\*Если число столбцов DATA >1, функция вычисляет расстояния через скалярные произведения;

\*это более быстрый, но калькуляторно потенциально менее точный способ, чем "прямой" способ

\*через возведение разниц в квадрат (если вам нужен "прямой способ", используте более

\*общую функцию /\*!KO\_pwmink\*/).

ПРИМЕР.

matrix.

get data /variables= v1 v2 v3.

!KO\_seuclid(data%dist).

save dist /outfile= \*.

end matrix.

### ПРЯМОУГОЛЬНАЯ МАТРИЦА КВАДРАТНЫХ ЕВКЛИДОВЫХ РАССТОЯНИЙ [!KO\_seuclidr]

\*/\*!KO\_seuclidr(cases1%cases2%name)\*/\*.

\*Version 2.

\*Вычисляет квадратные евклидовы расстояния между рядами CASES1 и рядами CASES2.

\*Эти две матрицы должны иметь одинаковое число столбцов (столбцы это измерения данных).

\*Выдает прямоугольную матрицу NAME размером число\_рядов\_CASES1 x число\_рядов\_CASES2.

\*Если число столбцов данных >1, функция вычисляет расстояния через скалярные произведения;

\*это более быстрый, но калькуляторно потенциально менее точный способ, чем "прямой" способ

\*через возведение разниц в квадрат.

### МАТРИЦА СТЕПЕННЫХ ВЗВЕШЕННЫХ РАССТОЯНИЙ МИНКОВСКОГО [!KO\_pwmink]

\*/\*!KO\_pwmink(data%power%weights%name)\*/\*.

\*Version 1.

\*Вычисляет расстояния Минковского (возведенные в степень POWER) между рядами DATA.

\*Выдает квадратную симметрическую матрицу NAME.

\*POWER - скаляр (типично >=1); это степень для расстояния.

\*WEIGHTS - вектор (ряд или столбец) длиной как число столбцов DATA, содержащий положительные числа;

\*это веса для столбцов DATA. Только соотношение величин весов важно, т.к. функция нормирует их.

\*Если все веса одинаковы, получится невзвешенное расстояние.

\*Расстояние между рядами a и b = Sum(w\_i\*|a\_i-b\_i|^p), где a\_i, b\_i значения в столбце i,

\*p степень, w\_i вес столбца i, Sum сумма по всем столбцам.

\*Если POWER=2, функция выдаст (взвешенные) квадратные евклидовы расстояния.

\*Если POWER=1, функция выдаст (взвешенные) манхэттенские расстояния.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Взвешенное Манхэттенское расстояние.

set mxloops 10000.

matrix.

get data /variables= v1 v2 v3.

!KO\_pwmink(data%1%{1,2,1}%dist).

save dist /outfile= \*.

end matrix.

### ПРЯМОУГОЛЬНАЯ МАТРИЦА СТЕПЕННЫХ ВЗВЕШЕННЫХ РАССТОЯНИЙ МИНКОВСКОГО [!KO\_pwminkr]

\*/\*!KO\_pwminkr(cases1%cases2%power%weights%name)\*/\*.

\*Version 1.

\*Вычисляет расстояния Минковского (возведенные в степень POWER) между рядами CASES1 и рядами CASES2.

\*Эти две матрицы должны иметь одинаковое число столбцов (столбцы это измерения данных).

\*Выдает прямоугольную матрицу NAME размером число\_рядов\_CASES1 x число\_рядов\_CASES2.

\*Аргументы POWER и WEIGHTS - то же, что в функции /\*!KO\_pwmink\*/.

\*Если POWER=2, функция выдаст (взвешенные) квадратные евклидовы расстояния.

\*Если POWER=1, функция выдаст (взвешенные) манхэттенские расстояния.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### МАТРИЦА КВАДРАТНЫХ РАССТОЯНИЙ МАХАЛАНОБИСА [!KO\_smahal]

\*/\*!KO\_smahal(data%cov%name)\*/\*.

\*Version 1.

\*Вычисляет квадратные расстояния Махаланобиса между рядами DATA (матрица n x p, p>=1).

\*Выдает квадратную симметрическую матрицу NAME.

\*Аргумент COV - либо квадратная p x p симметричная невырожденная матрица, являющаяся

\*пользовательской ковариационной матрицей между переменными, либо скаляр 0 - тогда матрица COV

\*будет принята равной наблюдаемой матрице ковариаций между p переменными DATA. Тогда DATA должна

\*иметь больше рядов, чем столбцов, и столбцы должны быть неколлинеарны.

\*Формула расстояния: d^2 = (x-y)'inv(COV)(x-y).

\*Если ковариационная матрица - единичная матрица, то результат - квадратные евклидовы расстояния.

\*Если вводимая COV это корреляционная матрица, получатся квадратные махаланобисовы расстояния

\*без масштабирования.

\*Если вы имеете пользовательскую inv(COV) вместо COV, то можете использовать /\*!KO\_sbutler\*/

\*для получения махаланобисовых расстояний.

### ВЕКТОР КВАДРАТНЫХ РАССТОЯНИЙ МАХАЛАНОБИСА ДО ЦЕНТРОИДА [!KO\_smahalc]

\*/\*!KO\_smahalc(data%mean%cov%name)\*/\*.

\*Version 2.

\*Вычисляет квадратные расстояния Махаланобиса между рядами DATA и центроидом.

\*Выдает вектор-столбец NAME с числом рядов как у DATA.

\*DATA должна иметь p (p>=1) столбцов и любое число рядов. Это координаты точек в p-мерном

\*пространстве.

\*Если вы хотите получить расстояния до центроида самого наблюдаемого облака DATA, укажите

\*аргумент COV как нулевой скаляр. DATA тогда должна иметь минимум два ряда и неколлинеарные

\*столбцы. Аргумент MEAN будет проигнорирован.

\*Если вы хотите получить расстояния точек DATA до центроида некоторого другого облака в том же

\*пространстве, необходимо тогда указать вектор средних MEAN и ковариационную матрицу COV

\*того облака: MEAN должна иметь p столбцов и 1 ряд. Это координаты центра того облака. COV должна

\*быть p X p, симметричной и невырожденной. Это ковариации в том облаке.

\*Если вы хотите получить расстояние между двумя группами из одной популяции, то введите ряд

\*DATA = центроид одной группы, ряд MEAN = центроид другой группы, COV = усредненная

\*внутригрупповая ковариационная матрица (последнюю вы можете получить ф-цией /\*!KO\_gcov\*/).

\*Если COV - единичная матрица, то результат - квадратные евклидовы расстояния от точек DATA до

\*центроида MEAN.

\*Если COV - неграмова (имеет отрицательные собственные числа), некоторые расстояния выйдут

\*отрицательными.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Расстояния Махаланобиса от точек облака до центроида этого же облака.

matrix.

get data /variables= v1 v2 v3.

!KO\_smahalc(data%0%0%dist).

print sqrt(dist).

end matrix.

### МАТРИЦА КВАДРАТНЫХ БАТЛЕРОВЫХ РАССТОЯНИЙ [!KO\_sbutler]

\*/\*!KO\_sbutler(data%cos%name)\*/\*.

\*Version 1.

\*Вычисляет квадратные расстояния Батлера между рядами DATA (матрица n x p, p>=1).

\*Выдает квадратную симметрическую матрицу NAME.

\*"Батлеровы" расстояния - это евклидовы расстояния не в декартовой (ортогональной), а в

\*косоугольной системе, где косинусы углов между осями равны корреляциям между переменными или

\*заданы пользователем; входящие же данные принимаются за косые (контравариантные) координаты

\*в этой системе. Формула: d^2 = (x-y)'COS(x-y).

\*Аргумент COS - либо квадратная p x p симметричная матрица косинусов углов между осями

\*(это задание углов пользователем), либо скаляр 0 - тогда матрица косинусов углов COS будет

\*принята равной наблюдаемой матрице корреляций между p переменными DATA.

\*Если матрица косинусов углов - единичная матрица, то результат - квадратные евклидовы расстояния.

\*Если в качестве COS ввести обращенную матрицу ковариаций, то результат - квадратные махаланобисовы

\*расстояния, а если обращенную матрицу корреляций - то немасштабированные квадратные махаланобисовы

\*расстояния.

### ЕСТЬ ЛИ В МАТРИЦЕ ОДИНАКОВЫЕ ЗНАЧЕНИЯ

Используйте для этого выражение any(grade(mat)<>rnkorder(mat)), которое выдает 1, если в MAT есть дубликаты, и выдает 0, если, все значения разные.

### СПИСОК УНИКАЛЬНЫХ ЗНАЧЕНИЙ В ВЕКТОРЕ И ПОЗИЦИЙ ИХ ПЕРВОГО ПОЯВЛЕНИЯ [!KO\_unique]

\*/\*!KO\_unique(vec%ignore%name1%name2)\*/\*.

\*Version 3.

\*Берет вектор (столбец или ряд) VEC и выдает вектор-ряд NAME1, являющийся списком

\*уникальных значений VEC; значения идут в порядке их первого появления в VEC.

\*Позиции первого появления выдаются как соответствующий ряд NAME2.

\*Аргумент IGNORE - цифра (не имя и не выражение, опционально в кавычках или апострофах): если 1,

\*значение 0 в VEC игнорируются (считается невалидным); если 0, то нуль считается валидным значением.

\*Результат NAME2 в виде скаляра 0 в случае IGNORE=1 означает, что VEC состоял из нулей.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

compute var= {4;0;8;8;0;-1;8;6;3}.

print var.

!KO\_unique(var%0%vals%pos).

print {vals;pos}.

end matrix.

ПРИМЕР. Позиции последнего (а не первого) появления значений.

matrix.

compute var= {4;0;8;8;0;-1;8;6;3}.

print var.

!KO\_unique(var(nrow(var):1)%0%vals%pos).

print {vals;nrow(var)-pos+1}.

end matrix.

### ЧАСТОТЫ ЗАДАННЫХ ЗНАЧЕНИЙ В СТОЛБЦАХ [!KO\_freqval]

\*/\*!KO\_freqval(data%vals%name)\*/\*.

\*Version 1.

\*Берет матрицу DATA, состоящую из одного или более столбцов, и список значений VALS (скаляр или вектор),

\*и выдает частоту каждого из этих значений в каждом столбце DATA. Результат NAME это матрица этих частот;

\*в ней ряды соответствуют значениям VALS и столбцы соответствуют столбцам DATA.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ДВУМЕРНАЯ ЧАСТОТНАЯ ПЕРЕКРЕСТНАЯ ТАБЛИЦА ДЛЯ ЗАДАННЫХ ЗНАЧЕНИЙ [!KO\_ctabval]

\*/\*!KO\_ctabval(rcol%ccol%rvals%cvals%name)\*/\*.

\*Version 1.

\*Берет векторы-столбцы RCOL и CCOL, которые должны быть одинаковой длины, и списки значений: RVALS (для RCOL)

\*и CVALS (для CCOL). RVALS и CVALS это векторы (или скаляры), их длина может быть разной.

\*Выдает частотную кросс-таблицу NAME размером r x c, с рядами, задаваемыми r значениями RVALS и столбцами,

\*задаваемыми c значениями CVALS.

\*Частоты в таблице - число наблюдений с соответствующим сочетанием значений между RCOL и CCOL.

ПРИМЕР.

matrix.

compute x= {1;2;2;0;0;0;1;1;2;1;0;2;2;2;2;0;0;1;0;0;0;1;1;0;1;1;2;1;2;2;3;2;2;3;9;2}.

compute y= {0;1;0;1;1;1;1;1;2;1;1;1;1;1;3;2;1;2;2;2;9;2;1;2;2;2;2;2;1;2;2;2;2;3;2;2}.

print {x,y}.

!KO\_ctabval(x%y%{0,1,3,9}%{1,2}%tab).

print tab /title 'X by Y frequency crosstabulation for the requested lists'

+' of values: (0,1,3,9) by (1,2).'.

end matrix.

### ФИКТИВНЫЕ ПЕРЕМЕННЫЕ И ЧАСТОТЫ ЗНАЧЕНИЙ В СТОЛБЦЕ [!KO\_freq]

\*/\*!KO\_freq(col%sort%name1%name2%name3)\*/\*.

\*Version 2.

\*Берет вектор-столбец COL, который должен не быть константой, и выдает:

\*матрицу NAME1 фиктивных (dummy) переменных, по одному столбцу на каждое уникальное значение COL;

\*вектор-столбец NAME2 частот уникальных значений COL (это суммы в столбцах NAME1);

\*вектор-столбец NAME3 - список уникальных значений COL.

\*Аргумент SORT (скаляр) - сортировать ли результаты по возрастанию значений: если положительный,

\*то столбцы в NAME1 и ряды в NAME2 и NAME3 будут сортированы в возрастающем порядке уникальных значений COL;

\*если же аргумент неположительный, то порядок рядов и столбцов в результатах определяется очередностью

\*встречи уникальных значений COL (при просмотре COL сверху вниз).

\*Эта функция может требовать от компьютера много оперативной памяти при массивных данных.

ПРИМЕР.

matrix.

compute var= {4;0;8;8;0;-1;8;6;3}.

print var.

!KO\_freq(var%1%dummy%freq%vals).

print {vals,freq} /clabels= 'Value' 'Freq'.

print dummy.

end matrix.

### ДВУМЕРНАЯ ЧАСТОТНАЯ ПЕРЕКРЕСТНАЯ ТАБЛИЦА [!KO\_crosstab]

\*/\*!KO\_crosstab(rcol%ccol%sort%name1%name2%name3)\*/\*.

\*Version 2.

\*Берет векторы-столбцы RCOL и CCOL, которые должны не быть константами и должны быть одинаковой длины.

\*Выдает частотную кросс-таблицу NAME1, в которой ряды заданы уникальными значениями RCOL

\*и столбцы заданы уникальными значениями CCOL. Списки самих уникальных значений выдаются как вектор-столбец NAME2

\*(значения RCOL) и вектор-ряд NAME3 (значения CCOL).

\*Аргумент SORT (скаляр) - сортировать ли результаты по возрастанию значений: если положительный,

\*то ряды в NAME1 и NAME2 будут сортированы в возрастающем порядке уникальных значений RCOL и столбцы в

\*NAME1 и NAME3 будут сортированы в возрастающем порядке уникальных значений CCOL;

\*если же аргумент неположительный, то порядок рядов и столбцов в результатах определяется очередностью

\*встречи уникальных значений в RCOL и CCOL, соответственно (при просмотре этих столбцов сверху вниз).

\*Эта функция может требовать от компьютера много оперативной памяти при массивных данных.

ПРИМЕР.

matrix.

get var1 /variable= v1.

get var2 /variable= v2.

!KO\_crosstab(var1%var2%1%tab%vals1%vals2).

print tab.

print vals1.

print vals2.

end matrix.

### ФИКТИВНЫЕ ПЕРЕМЕННЫЕ ВЗАИМОДЕЙСТВИЯ И МНОГОМЕРНАЯ ЧАСТОТНАЯ АГРЕГАЦИЯ [!KO\_aggrtab]

\*/\*!KO\_aggrtab(cols%sort%empty%name1%name2%name3)\*/\*.

\*Version 1.

\*Делает частотную кросстабуляцию для любого числа переменных.

\*Выдает результаты в виде агрегированного массива (а не в виде кросс-таблиц).

\*(Подобным образом частоты выдает вне-матричная команда AGGREGATE.)

\*Функция также выдает двоичный массив фиктивных переменных, соответствующих взаимодействию

\*высшего порядка.

\*COLS - категориальные данные, где переменные - столбцы; столбцов должно быть минимум два и они

\*должны не быть константами.

\*Функция выдает частотный вектор-столбец NAME2, где каждый элемент соответствует одному сочетанию значений

\*всех переменных, т.е. это частота в ячейке многомерной таблицы, задаваемой переменными.

\*Сочетания самих уникальных значений переменных содержатся в NAME3, где ряды отвечают рядам NAME2,

\*а столбцы отвечают переменным, столбцам COLS.

\*Вектор NAME2 это суммы в столбцах выдаваемого массива NAME1, который есть фиктивные переменные: они

\*соответствуют вышеозначенным сочетаниям уникальных значений. Т.е. столбцы NAME1 соответственны рядам

\*NAME2 и NAME3.

\*Аргумент SORT (скаляр) - сортировать ли результаты по возрастанию значений: если положительный,

\*то столбцы в NAME1 и ряды в NAME2 и NAME3 будут сортированы в возрастающем порядке уникальных значений

\*переменных; если же аргумент неположительный, то упомянутый порядок в результатах определяется очередностью

\*встречи уникальных значений в данных (при просмотре столбцов COLS сверху вниз).

\*Аргумент EMPTY (скаляр) - если положительный, из результатов будут выпущены столбцы (в NAME1) и ряды

\*(в NAME2 и NAME3) с частотой 0 в NAME1. Если же аргумент неположительный, результаты будут

\*выданы полными.

\*Эта функция может требовать от компьютера много оперативной памяти при массивных данных.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Трехмерная частотная кросстабуляция (агрегация) в MATRIX и командой AGGREGATE.

set mxloops 10000.

matrix.

get data /variables= gender agegroup marriage.

!KO\_aggrtab(data%1%1%dummy%freq%codes).

print {codes,freq} /clabels= 'gender' 'agegr' 'marri' 'n'.

end matrix.

dataset declare aggr.

aggregate /outfile= 'aggr' /break= gender agegroup marriage /freq= n.

### ФИКТИВНЫЕ ПЕРЕМЕННЫЕ ВЗАИМОДЕЙСТВИЯ И МНОГОМЕРНАЯ ЧАСТОТНАЯ АГРЕГАЦИЯ (ДИХОТОМИЧЕСКИЕ ДАННЫЕ) [!KO\_baggrtab]

\*/\*!KO\_baggrtab(cols%empty%name1%name2%name3)\*/\*.

\*Делает частотную кросстабуляцию для любого числа дихотомических переменных.

\*Переменные должны быть кодированы двоично (1 vs 0); если у вас другие значения, сделайте их двоичными.

\*Данная функция эквивалентна /\*!aggrtab\*/, но рассчитана только на все переменные COLS

\*двоичные (0 vs 1). Она быстрее, чем /\*!aggrtab\*/, и бережет память.

\*Аргумент EMPTY и результаты NAME1, NAME2, NAME3 - см. в описании /\*!aggrtab\*/.

\*Порядок значений в результатах всегда сортирован так: 0 1, поэтому аргумента SORT нет.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ЭЛЕМЕНТАРНЫЕ ПЕРЕМЕННЫЕ И ЧАСТОТЫ ЗНАЧЕНИЙ В КАТЕГОРИАЛЬНОМ НАБОРЕ МНОЖЕСТВЕННОГО ОТВЕТА [!KO\_mrfreq]

\*/\*!KO\_mrfreq(mrc%dupl%sort%name1%name2%name3)\*/\*.

\*Version 1.

\*Эта функция подобна /\*!KO\_freq\*/ - выдает двоичные переменные и частоты значений - но в качестве

\*входящих принимает не одну категориальную переменную, а набор таковых; у этих переменных общий

\*пул значений. Это "категориальный набор множественного ответа". Если каждое наблюдение это респондент,

\*он мог, каким образом, выбрать несколько, а не один ответ.

\*MRC - входящие данные, столбцы которых категориальные переменные, составляющие упомянутый набор.

\*Нуль в MRC считается "наполнителем", т.е. пустой ячейкой. Всякое же иное значение есть валидное.

\*Допускается, чтобы некоторые ряды (наблюдения) или некоторые столбцы (переменные) были пусты.

\*Аргумент DUPL (скаляр) - если положительный, то каждое уникальное значение засчитывается в ряду MRC

\*не более раза, т.е. повторение одного и того же ответа одним респондентом, если есть, игнорируется;

\*а если аргумент неположительный, то все повторения ответов респондентом засчитываются как валидные.

\*Функция выдает:

\*NAME1 - матрица, по одному столбцу на каждое уникальное валидное значение MRC. Значение

\*(i,j) в NAME1 это сколько раз респондент i дал ответ j (если DUPL неположителен), либо 0 или 1

\*(если DUPL положителен); таким образом, при положительном DUPL NAME1 есть "двоичный набор

\*множественного ответа".

\*NAME2 - вектор-столбец частот уникальных значений MRC (это суммы в столбцах NAME1);

\*NAME3 - вектор-столбец, список уникальных значений MRC.

\*Аргумент SORT (скаляр) - сортировать ли результаты по возрастанию значений: если положительный,

\*то столбцы в NAME1 и ряды в NAME2 и NAME3 будут сортированы в возрастающем порядке уникальных значений MRC;

\*если же аргумент неположительный, то порядок рядов и столбцов в результатах определяется очередностью

\*встречи уникальных значений MRC (при просмотре MRC сверху вниз, сначала 1-й столбец, потом 2-й...).

\*Эта функция может требовать от компьютера много оперативной памяти при массивных данных.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

get mrc /variables= v1 to v6 /miss= 0. /\*Categorical vars with common pool of values

!KO\_mrfreq(mrc%1%1%bin%freq%codes).

print {codes,freq}. /\*Values and their frequencies

save bin /out= \*. /\*Binary variables (each corresponding to a unique value)

end matrix.

### ПРОЦЕНТ ПРАВИЛЬНОЙ КЛАССИФИКАЦИИ [!KO\_classres]

\*/\*!KO\_classres(tab%name1%name2%name3).\*/\*.

\*Version 1.

\*Берет квадратную частотную k x k кросс-таблицу TAB, являющую результаты классификации

\*наблюдений по k классам. Ряды таблицы - наблюдаемые классы, столбцы таблицы - эти же

\*классы предсказанные, идущие в том же порядке. Таким образом, частоты правильного

\*предсказания образуют диагональ таблицы. (Вы можете получить таблицу TAB из переменных

\*"наблюдаемый класс" и "предсказанный класс" функцией /\*!KO\_crosstab\*/.)

\*Выдает ряд NAME1 ("Общий Процент"), столбец NAME2 ("Процент Правильных"), скаляр NAME3

\*(угловое значение, "Общий Процент Правильных").

ПРИМЕР.

matrix.

get obs /variable= obs. /\*Grouping variable with (say) 2 classes

get pred /variable= pred. /\*Corresponding variable with these 2 classes predicted

/\*(both variables must have the same class codes

!KO\_crosstab(obs%pred%1%tab%name2%name3). /\*Produce square classification table:

/\*frequency crosstabulation with observed classes as rows and predicted –

/\*as columns;

/\*the counts of correctly classified cases is the diagonal of the table

/\*Note that argument SORT is set to a positive value (1)

print tab.

!KO\_classres(tab%overall%correct%ocorrect). /\*Assess the classification results

print {tab,correct;overall,ocorrect}

/rlabels= 'Class1' 'Class2' 'Overall%'

/clabels= 'Class1' 'Class2' '%Correct'

/title 'Classification table: Observed x Predicted classes'.

end matrix.

### ЧАСТОТЫ КОНКОРДАНТНЫХ, ДИСКОРДАНТНЫХ И СВЯЗАННЫХ ПАР [!KO\_concdisc]

\*/\*!KO\_concdisc(col1%col2%name1%name2%name3%name4%name5)\*/\*.

\*Version 1.

\*Берет векторы-столбцы COL1 и COL2, которые должны не быть константами и должны быть одинаковой длины

\*(это две мерные или порядковые переменные) и выдает 5 скалярных величин, необходимых для вычисления

\*таких коэффициентов связи между порядковыми переменными, как Гамма Гудмана-Краскела, коэффициент Сомера,

\*корреляция Кендалла, и т.д.

\*Эти величины:

\*NAME1 - число конкордантных пар (P);

\*NAME2 - число дискордантных пар (Q);

\*NAME3 - число пар со связью (ties) по первой переменной, COL1 (Tx);

\*NAME4 - число пар со связью (ties) по второй переменной, COL2 (Ty);

\*NAME5 - число пар со связью (ties) по обеим переменным сразу (Txy).

\*Если COL1 и COL2 категориальные (не континуальные) данные, другая ф-ция, /\*!KO\_concdisct\*/, вычисляющая

\*те же величины, может оказаться более скоростной.

ПРИМЕР.

matrix.

get var1 /variable= v1.

get var2 /variable= v2.

!KO\_concdisc(var1%var2%p%q%t\_v1%t\_v2%t\_v12).

print {p,q,t\_v1,t\_v2,t\_v12}.

end matrix.

### ЧАСТОТЫ КОНКОРДАНТНЫХ, ДИСКОРДАНТНЫХ И СВЯЗАННЫХ ПАР (ИЗ ТАБЛИЦЫ) [!KO\_concdisct]

\*/\*!KO\_concdisct(tab%name1%name2%name3%name4%name5)\*/\*.

\*Version 2.

\*Берет частотную кросс-таблицу TAB, полученную для двух мерных или порядковых переменных

\*(т.е. ряды и столбцы в таблице упорядочены по возрастанию уникальных значений переменных)

\*и выдает 5 скалярных величин, необходимых для вычисления таких коэффициентов связи между

\*порядковыми переменными, как Гамма Гудмана-Краскела, коэффициент Сомера, корреляция Кендалла, и т.д.

\*Эти величины:

\*NAME1 - число конкордантных пар (P);

\*NAME2 - число дискордантных пар (Q);

\*NAME3 - число пар со связью (ties) по первой переменной, образующей ряды таблицы (Tx);

\*NAME4 - число пар со связью (ties) по второй переменной, образующей столбцы таблицы (Ty);

\*NAME5 - число пар со связью (ties) по обеим переменным сразу (Txy).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Вычисление некоторых коэффициентов связи между двумя порядковыми переменными.

matrix.

get var1 /variable= v1.

get var2 /variable= v2.

!KO\_crosstab(var1%var2%1%tab%vals1%vals2).

print tab.

!KO\_concdisct(tab%p%q%t\_v1%t\_v2%t\_v12).

compute gamma= (p-q)/(p+q).

compute somer= (p-q)/((p+q+t\_v1+p+q+t\_v2)/2).

compute taub= (p-q)/sqrt((p+q+t\_v1)\*(p+q+t\_v2)).

print gamma /title 'Goodman-Kruskal Gamma'.

print somer /title 'Somer D (symmetric)'.

print taub /title 'Kendall tau-b correlation'.

end matrix.

### МАТРИЦА ПУТАНОСТИ СОЧЛЕНСТВА ДЛЯ ДВУХ РАЗБИЕНИЙ [!KO\_ccm]

\*/\*!KO\_ccm(part1%part2%name)\*/\*.

\*Version 1.

\*Матрица (или таблица) путаности сочленства (comembership confusion matrix/table) это такая 2x2 матрица

\*путаности (confusion matrix), где элементом счета выступает пара объектов (а не объект). Эта матрица

\*является основой для оценки сходства двух классифицаций (разбиений) одних и тех же объектов.

\*Пусть есть n объектов, которые были классифицированы двумя способами, I и II. Тогда в матрице путаницы

\*сочленства ячейка (1,1) содержит число пар, в которых оба объекта совстречаются в одном классе

\*как в классификации I, так и классийикации II; ячейка (1,2) содержит число пар, в которых оба объекта

\*совстречаются в классификации I, но не в классификации II; ячейка (2,1) содержит число пар, в которых

\*оба объекта совстречаются в классификации II, но не в классификации I; ячейка (2,2) содержит число пар,

\*в которых оба объекта не совстречаются в одном классе ни в той, ни в другой классификации.

\*Сумма в такой таблице равна числу пар объектов, n(n-1)/2.

\*Входящие PART1 и PART2 - два массива с одинаковым числом рядов (ряды - это объекты), являющие собой

\*разбиения/классификации I и II соответственно. Массив может быть либо столбцом с любыми значениями, либо

\*несколькими столбцами с двоичными (1 и 0) значениями. Когда классификация задана одним столбцом,

\*то функция принимает его за категориальную переменную: каждое уникальное значение обозначает класс.

\*Когда классификация задана более чем одним столбцом, функция принимает их за набор двоичных переменных,

\*каждая обозначает класс, а значения обозначают принадлежность ему (1=принадлежит, 0=не принадлежит).

\*Двоичный набор может быть любой, например содержать ряды или столбцы, состоящие из одних нулей или из

\*одних единиц; т.е. двоичным набором можно передать не только классификацию, где классы непересекаемы

\*составом объектов, но любую, в том числе с пустыми классами или объектами, не принадлежащими классам.

\*В том случае, если ваша классификация представляет только один класс, то задайте ее как два столбца:

\*один двоичный (принадлежит - не принадлежит), а другой пустой, из нулей.

\*Результат NAME - матрица (или таблица) путаности сочленства.

ПРИМЕР.

matrix.

get part1 /variables= groups1. /\*Categorical grouping: partition I

get part2 /variables= bin1 to bin4. /\*Set of binary variables: partition II

!KO\_ccm(part1%part2%ct).

print ct /title 'Confusion table for pairs of cases, I x II'

/rlabels 'InSameGr' 'NotSame' /clabels 'InSameGr' 'NotSame'.

end matrix.

### СЧЕТ ЗНАЧЕНИЙ В РЯДАХ (ТОЧНОЕ СОВПАДЕНИЕ) [!KO\_count1]

\*/\*!KO\_count1(data%vals%name)\*/\*.

\*Version 2.

\*Подсчитывает в рядах DATA число элементов, совпадающих с любым из значений списка VALS.

\*Выдает столбец NAME длиной как число рядов DATA.

\*Эта функция подобна внематричной команде COUNT, в которой значения заданы списком.

\*VALS - вектор (столбец или ряд) любой длины. Значения в списке могут повторяться: это не влияет на

\*результат, но замедляет работу.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### СЧЕТ ЗНАЧЕНИЙ В РЯДАХ (ПОПАДАНИЕ В ДИАПАЗОН) [!KO\_count2]

\*/\*!KO\_count2(data%low%high%name)\*/\*.

\*Version 2.

\*Подсчитывает в рядах DATA число элементов, попадающих в любой из диапазонов от LOW до HIGH (включительно).

\*Выдает столбец NAME длиной как число рядов DATA.

\*Эта функция подобна внематричной команде COUNT, в которой значения заданы диапазоном(ами) от - до.

\*LOW и HIGH - два скаляра или два вектора одинаковой длины. Каждая пара их соответствующих элементов

\*LOW(i) и HIGH(i) есть диапазон; HIGH(i) в норме должно быть >= LOW(i). Диапазоны могут пересекаться.

\*Чтобы подсчитать все значения <=HIGH, установите LOW на заведомо слишком малое число.

\*Чтобы подсчитать все значения >=LOW, установите HIGH на заведомо слишком большое число.

\*Если аргументы LOW и HIGH тождественны, функция даст тот же результат, что /\*!KO\_count1\*/.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Счет значений внутри наблюдений и эквивалентная внематричная команда COUNT.

matrix.

get data /variables= v1 to v10.

!KO\_count2(data%{-1,6,7,12}%{3,8,9,12}%name).

print name.

end matrix.

count x= v1 to v10 (-1 thru 3, 6 thru 8, 7 thru 9, 12).

list x.

### ХИ-КВАДРАТ (ПИРСОНА И ОТНОШЕНИЯ ПРАВДОПОДОБИЯ) ДВУМЕРНОЙ ТАБЛИЦЫ СОПРЯЖЕННОСТИ [!KO\_chitab]

\*/\*!KO\_chitab(tab%type%name)\*/\*.

\*Version 1.

\*Вычисляет статистику хи-квадрат, говорящую о связи рядов и столбцов в двумерной

\*таблице сопряженности. Выдает скаляр NAME.

\*Аргумент TYPE (скаляр) - какую версию статистики выдать: хи-квадрат Пирсона (TYPE неположителен)

\*или хи-квадрат отношения правдоподобия (TYPE положителен).

\*Таблица TAB должна содержать неотрицательные числа, и суммы в ее рядах и ее столбцах

\*должны быть все ненулевые.

### ДОЛИ И ОСТАТКИ В ДВУМЕРНОЙ ТАБЛИЦЕ СОПРЯЖЕННОСТИ [!KO\_cells]

\*/\*!KO\_cells(tab%stat%name)\*/\*.

\*Version 1.

\*Берет двумерную таблицу сопряженности (например частотную перекрестную таблицу) TAB с

\*(в норме) неотрицательными элементами и (обязательно) ненулевыми суммами в рядах и столбцах.

\*Выдает таблицу этого же размера NAME, содержащую в ячейках требуемую статистику.

\*Требуемая статистика - аргумент STAT, ключевое заглавными буквами:

\*"TABPR" - доли от суммы таблицы

\*"ROWPR" - доли от суммы в рядах

\*"COLPR" - доли от суммы в столбцах

\*"EXPECT" - ожидаемые (при независимости рядов и столбцов) значения

\*"RESID" - остатки

\*"SRESID" - стандартизованные остатки

\*"ASRESID" - приведенные стандартизованные остатки.

\*Кавычки или апострофы вокруг ключевого слова необязательны.

### МАТРИЦА КВАДРАТНЫХ ХИ-КВАДРАТ-РАССТОЯНИЙ В ТАБЛИЦЕ СОПРЯЖЕННОСТИ [!KO\_schitab]

\*/\*!KO\_schitab(tab%name)\*/\*.

\*Version 1.

\*Вычисляет квадратные хи-квадрат-расстояния между рядами таблицы сопряженности TAB.

\*Выдает квадратную симметрическую матрицу NAME.

\*TAB это матрица с неотрицательными элементами, и суммы в ее рядах и ее столбцах должны

\*быть все ненулевые.

\*Хи-квадрат расстояние между рядами это взвешенное евклидово расстояние между рядными долями,

\*где весами служат обращенные суммы в столбцах. Расстояние (квадратное) между рядами i и i' есть:

\*N \* Sum\_c[1/Nj \* (Nij/Ni-Ni'j/Ni')^2], где Nij значение в ячейке, Ni сумма в ряду i, Nj сумма в

\*столбце j, N общая сумма, Sum\_c сумма сквозь все столбцы.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### МАТРИЦА ХИ-КВАДРАТОВ МЕЖДУ РЯДЯМИ ЧАСТОТНЫХ ДАННЫХ ПОПАРНО [!KO\_schi2c]

\*/\*!KO\_schi2c(data%type%name)\*/\*.

\*Version 1.

\*Берет данные DATA размером не менее 2x2 с неотрицательными значениями, которые рассматривает как частоты.

\*Суммы в рядах и столбцах должны быть больше нуля.

\*Для каждых двух рядов DATA попарно вычленяет табличку 2xNC (NC - число столбцов DATA) и вычисляет

\*статистику хи-квадрат (или фи-квадрат) Пирсона этой таблички. Эта статистика может считаться мерой различия

\*между двумя данными рядями: хи-квадрат расстояние (квадратное).

\*Выдается квадратная симметричная матрица NAME таких расстояний между всеми рядями DATA.

\*Это те же расстояния, только в квадрате, что выдает команда PROXIMITIES /MEASURE=CHISQ (или /MEASURE=PH2).

\*Не следует путать эти "расстояния" с хи-квадрат расстояниями, выдаваемыми ф-цией /\*!KO\_schitab\*/.

\*TYPE - скаляр: если неположительный, вычисляется хи-квадрат; если положительный, вычисляется фи-квадрат

\*(хи-квадрат делится на общую частоту 2xNC таблички), фи-квадрат нечувствителен к неодинаковости сумм

\*в рядах DATA.

\*Если в какой-либо 2xNC табличке встретится пустой столбец, функция игнорирует его и пытается вычислить

\*на основании оставшихся столбцов.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

get data /vari= v1 to v4.

!KO\_schi2c(data%0%chi).

print sqrt(chi).

end matrix.

PROXIMITIES v1 to v4

/VIEW= CASE

/MEASURE= CHISQ

/STANDARDIZE= NONE.

### РАССТОЯНИЯ ДО ЦЕНТРОИДОВ ГРУПП [!KO\_dtoc]

\*/\*!KO\_dtoc(data%bin%name)\*/\*.

\*Version 1.

\*Берет данные DATA с любым числом рядов (точек) и столбцов (переменных).

\*И берет одну или более двоичных (1 vs 0) переменных, образующих матрицу BIN с числом рядов как в DATA.

\*Каждый столбец BIN означает группу, и значение 1 означает, что точка принадлежит ей.

\*В целом, группы могут пересекаться или непересекаться составом точек (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей.

\*Результат NAME - матрица точки x группы, содержащая квадратные евклидовы расстояния между точками

\*и центроидами групп.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

ПРИМЕР.

matrix.

get data /vari= v1 to v4.

get gr /var= gr. /\*Grouping variable

!KO\_freq(gr%1%bin%freq%codes). /\*Convert it to dummies

!KO\_dtoc(data%bin%sqdev).

print sqdev. /\*Sq. distances from all cases to all group centroids

!KO\_aggr(data%bin%MEAN%centr).

print centr. /\*Display group centroids (means): groups X variables

end matrix.

### РАССТОЯНИЯ МЕЖДУ ЦЕНТРОИДАМИ ГРУПП

Используйте функцию /\*!KO\_aggr\*/, чтобы вычислить средние (центроиды) групп, затем вычислите евклидовы расстояния между ними функцией /\*!KO\_seuclid\*/.

### РАССТОЯНИЯ ДО ЦЕНТРОИДОВ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАССТОЯНИЙ) [!KO\_dtocfrd]

\*/\*!KO\_dtocfrd(dis%bin%self%name)\*/\*.

\*Version 2.

\*Берет квадратную симметричную матрицу расстояний между точками DIS. Расстояния должны быть

\*квадратные евклидовы; во всяком случае, ф-ция воспримет их как квадратные евклидовы.

\*И берет одну или более двоичных (1 vs 0) переменных, образующих матрицу BIN с числом рядов как в DIS.

\*Каждый столбец BIN означает группу, и значение 1 означает, что точка принадлежит ей.

\*В целом, группы могут пересекаться или непересекаться составом точек (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей.

\*Результат NAME - матрица точки x группы, содержащая квадратные евклидовы расстояния между точками

\*и центроидами групп.

\*Аргумент SELF - цифра 0 или 1 (не имя и не выражение; цифру можно взять в кавычки/апострофы).

\*Если 1, то при вычислении расстояния между точкой i и центроидом группы, которой она принадлежит

\*(и которая кроме нее содержит еще точки), точка i участвует в определении центроида. Если 0, тогда

\*точка i не участвует в определении центроида: она рассматривается в этот момент как внешняя точка.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

### РАССТОЯНИЯ МЕЖДУ ЦЕНТРОИДАМИ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАССТОЯНИЙ) [!KO\_dbwcfrd]

\*/\*!KO\_dbwcfrd (dis%bin%name)\*/\*.

\*Version 1.

\*Берет квадратную симметричную матрицу расстояний между точками DIS. Расстояния должны быть

\*квадратные евклидовы; во всяком случае, ф-ция воспримет их как квадратные евклидовы.

\*И берет не менее двух двоичных (1 vs 0) переменных, образующих матрицу BIN с числом рядов как в DIS.

\*Каждый столбец BIN означает группу, и значение 1 означает, что точка принадлежит ей.

\*В целом, группы могут пересекаться или непересекаться составом точек (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей.

\*Результат NAME - квадратная симметричная матрица квадратных евклидовых расстояний между центроидами

\*групп точек.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

ПРИМЕР.

matrix.

get data /vari= v1 to v10. /\*Variables

get clu /var= clu. /\*Categorical grouping variable

!KO\_freq(clu%1%bin%n%code). /\*Convert it to the set of dummy variables

!KO\_seuclid(data%d). /\*Squared euclidean distances between cases

!KO\_dbwcfrd (d%bin%d\_c).

print d\_c. /\*Squared euclidean distances between group centroids

end matrix.

### УСРЕДНЕННЫЕ РАССТОЯНИЯ ДО ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_datofrd]

\*/\*!KO\_datofrd(dis%bin%self%name)\*/\*.

\*Version 1.

\*Среднее расстояние от каждой точки до точек каждой из предложенных групп.

\*Берет квадратную симметричную матрицу расстояний (различий) между точками DIS.

\*Это должна быть неотрицательная матрица с нулевой диагональю.

\*И берет одну или более двоичных (1 vs 0) переменных, образующих матрицу BIN с числом рядов как в DIS.

\*Каждый столбец BIN означает группу, и значение 1 означает, что точка принадлежит ей.

\*В целом, группы могут пересекаться или непересекаться составом точек (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей.

\*Результат NAME - матрица точки x группы: элемент ij в ней это усредненное расстояние между

\*точкой i и точками группы j.

\*Аргумент SELF - цифра 0 или 1 (не имя и не выражение; цифру можно взять в кавычки/апострофы).

\*Если 1, то расстояние точки до самой себя (нуль), когда она принадлежит группе, считается, т.е. входит

\*в усреднение. Если 0, то расстояние точки до самой себя не считается, т.е. не входит в усреднение,

\*кроме случая, когда точка - единственный член группы.

\*Если ваша матрица DIS это сходства, переведите их сначала в различия любым способом, каким хотите.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

### УСРЕДНЕННЫЕ РАССТОЯНИЯ МЕЖДУ ГРУППАМИ (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dabwfrd]

\*/\*!KO\_dabwfrd(dis%bin%self%name)\*/\*.

\*Version 1.

\*Расстояния межгрупповой средней связи, т.е. средние расстояния между точками от каждых двух групп.

\*Берет квадратную симметричную матрицу расстояний (различий) между точками DIS.

\*Это должна быть неотрицательная матрица с нулевой диагональю.

\*И берет не менее двух двоичных (1 vs 0) переменных, образующих матрицу BIN с числом рядов как в DIS.

\*Каждый столбец BIN означает группу, и значение 1 означает, что точка принадлежит ей.

\*В целом, группы могут пересекаться или непересекаться составом точек (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей.

\*Результат NAME - квадратная симметричная матрица группы x группы, содержащая усредненные

\*расстояния между точками разных групп. На диагонали стоят усредненные расстояния внутри групп.

\*Аргумент SELF - цифра 0 или 1 (не имя и не выражение; цифру можно взять в кавычки/апострофы).

\*Если 1, то расстояние точки до самой себя (нуль), когда она принадлежит группе (или обеим группам),

\*считается, т.е. входит в усреднение. Если 0, то расстояние точки до самой себя не считается, т.е.

\*не входит в усреднение, кроме случая, когда точка - единственный член группы (или единственный член

\*в обеих группах, когде речь идет о разных группах).

\*Если ваша матрица DIS это сходства, переведите их сначала в различия любым способом, каким хотите.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

### РАССТОЯНИЯ ДО ДАЛЬНИХ СОСЕДЕЙ ИЗ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dtoffrd]

\*/\*!KO\_dtoffrd(dis%bin%alsoind%name)\*/\*.

\*Version 1.

\*Расстояние от каждой точки до ее дальнего соседа каждой из предложенных групп.

\*Берет квадратную симметричную матрицу расстояний (различий) между точками DIS.

\*Это должна быть неотрицательная матрица с нулевой диагональю.

\*И берет одну или более двоичных (1 vs 0) переменных, образующих матрицу BIN с числом рядов как в DIS.

\*Каждый столбец BIN означает группу, и значение 1 означает, что точка принадлежит ей.

\*В целом, группы могут пересекаться или непересекаться составом точек (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей.

\*Аргумент ALSOIND - цифра 0 или 1 (не имя и не выражение; цифру можно взять в кавычки/апострофы).

\*Если вам нужны только расстояния до дальних соседей, укажите 0. Результат NAME будет матрица

\*точки x группы: элемент ij в ней это расстояние (значение из DIS) между точкой i и ее дальним соседом

\*из группы j. Если вы также хотите знать номера этих дальних соседей (их индексы рядов в DIS и BIN),

\*укажите ALSOIND как 1. Тогда в матрице NAME будет вдвое больше столбцов: к матрице с расстояниями до

\*дальних соседей будет справа пришита матрица, содержащая номера этих точек, являющихся дальними соседями.

\*Если у точки i существует более одного дальнего соседа в группе j, показывается номер последнего из них

\*(т.е. у кого номер больше).

\*Только если группа состоит из одной точки, ее сосед в этой группе - она сама.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если ваша матрица DIS это сходства, переведите их сначала в различия любым способом, каким хотите.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

ПРИМЕР. Расстояние от каждой точки до своего дальнего соседа (групп нет).

matrix.

get mx /vari= var1 to var100. /\*Dissimilarity matrix 100 cases

compute bin= make(nrow(mx),1,1). /\*There is no groups in this example,

/\*so simply create column of ones

!KO\_dtoffrd(mx%bin%1%name).

print name(:,1:ncol(bin))

/title 'Distance from every case to its farthest neighbour'.

print name(:,(ncol(bin)+1):ncol(name))

/title 'The case number of the farthest neighbour'.

end matrix.

### РАССТОЯНИЯ МЕЖДУ ДАЛЬНИМИ СОСЕДЯМИ ИЗ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dbwffrd]

\*/\*!KO\_dbwffrd(dis%bin%alsoind%name)\*/\*.

\*Version 1.

\*Расстояния полной связи между группами, т.е. наибольшие расстояния между точками каждых двух групп.

\*Берет квадратную симметричную матрицу расстояний (различий) между точками DIS.

\*Это должна быть неотрицательная матрица с нулевой диагональю.

\*И берет не менее двух двоичных (1 vs 0) переменных, образующих матрицу BIN с числом рядов как в DIS.

\*Каждый столбец BIN означает группу, и значение 1 означает, что точка принадлежит ей.

\*В целом, группы могут пересекаться или непересекаться составом точек (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей.

\*Аргумент ALSOIND - цифра 0 или 1 (не имя и не выражение; цифру можно взять в кавычки/апострофы).

\*Если вам нужны только расстояния между дальними соседями, укажите 0. Результат NAME будет квадратная

\*симметричная матрица группы x группы: элемент ij в ней это расстояние (значение из DIS) между самыми

\*дальними друг от друга точками, одна из группы i, другая из группы j. На диагонали - расстояния

\*между самыми разудаленными точками внутри каждой группы.

\*Если вы также хотите знать номера точек, являющихся теми дальними соседями (их индексы рядов в DIS и BIN),

\*укажите ALSOIND как 1. Тогда в матрице NAME столбцов будет вдвое больше плюс еще один: к матрице с

\*расстояниями между дальними соседями будет справа пришита матрица, содержащая номера этих точек, являющихся

\*дальними соседями. В этой матрице (к-рая будет пришита), и которая асимметрична, элемент ij это номер точки,

\*являющейся дальним соседом со стороны группы i (т.е. группы-ряда). Например, элементы (2,4)=10 и (4,2)=6

\*означают, что максимальное расстояние между группами 2 и 4 это расстояние между точкой 10 (она из группы 2)

\*и точкой 6 (она из группы 4). Что касается номеров точек-дальних соседей внутри каждой группы, то номер

\*одного из двух стоит на диагонали данной пришитой матрицы с номерами, а номер второго вынесен в отдельный

\*столбец, пришитый справа всего. Таким образом, если k это число столбцов BIN (число групп), то при ALSOIND=1

\*NAME имеет 2\*k+1 столбцов.

\*Точка может выйти своим собственным соседом только в том случае, если в обоих сопоставляемых группах она

\*единственный член группы.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если ваша матрица DIS это сходства, переведите их сначала в различия любым способом, каким хотите.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

ПРИМЕР. Расстояния полной связи (между дальними соседями) между группами.

matrix.

get mx /vari= var1 to var100. /\*Dissimilarity matrix 100 cases

get clu /var= clu. /\*Categorical grouping variable (in this example, groups

/\*will be disjoint, defined by a categorical variable)

!KO\_freq(clu%1%dummy%n%code). /\*Convert it to the set of dummy variables

!KO\_dbwffrd(mx%dummy%1%name).

print name(:,1:nrow(name))

/title 'Distances between farthest neighbours between (and within) groups'.

print name(:,(nrow(name)+1):(2\*nrow(name)))

/title 'Case numbers of the farthest neighbours'.

print /title 'Case number of one farthest neighbour within groups'.

print name(:,ncol(name)) /space 0

/title '(the number of the other one is on the diagonal of the prev. table)'.

end matrix.

### РАССТОЯНИЯ ДО БЛИЖНИХ СОСЕДЕЙ ИЗ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dtonfrd]

\*/\*!KO\_dtonfrd(dis%bin%alsoind%name)\*/\*.

\*Version 1.

\*Расстояние от каждой точки до ее ближнего соседа каждой из предложенных групп.

\*Эта функция такая же, как функция /\*!KO\_dtoffrd\*/, но выдает расстояния до ближних, а не до дальних соседей.

### РАССТОЯНИЯ МЕЖДУ БЛИЖНИМИ СОСЕДЯМИ ИЗ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dbwnfrd]

\*/\*!KO\_dbwnfrd(dis%bin%alsoind%name)\*/\*.

\*Version 1.

\*Расстояния единичной связи между группами, т.е. наименьшие расстояния между точками каждых двух групп.

\*Эта функция такая же, как функция /\*!KO\_dbwffrd\*/, но выдает расстояния между ближними,

\*а не между дальними соседями.

### РАССТОЯНИЯ ДО МЕДОИДОВ ГРУПП (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАЗЛИЧИЙ) [!KO\_dtomfrd]

\*/\*!KO\_dtomfrd(dis%bin%name1%name2)\*/\*.

\*Version 1.

\*Расстояние от каждой точки до медоида каждой из предложенных групп.

\*(Медоид это та точка в группе точек, сумма расстояний от которой до точек группы минимальна.)

\*Берет квадратную симметричную матрицу расстояний (различий) между точками DIS.

\*Это должна быть неотрицательная матрица с нулевой диагональю.

\*И берет одну или более двоичных (1 vs 0) переменных, образующих матрицу BIN с числом рядов как в DIS.

\*Каждый столбец BIN означает группу, и значение 1 означает, что точка принадлежит ей.

\*В целом, группы могут пересекаться или непересекаться составом точек (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей.

\*Результат NAME1 - матрица точки x группы, содержащая расстояния между точками и медоидами групп.

\*NAME2 это ряд, показывающий номера точек, являющихся медоидами в группах.

\*Если в группе более одной точки являются медоидами (как, например, в группе из двух точек),

\*то функция назначает точку с наибольшим индексом быть медоидом группы.

\*Матрица расстояний между медоидами групп есть DIS(NAME2,NAME2).

\*Если ваша матрица DIS это сходства, переведите их сначала в различия любым способом, каким хотите.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

### ВНУТРИГРУППОВЫЕ СУММЫ КВАДРАТОВ ОТКЛОНЕНИЙ (ВЫЧИСЛЕНИЕ ИЗ МАТРИЦЫ РАССТОЯНИЙ) [!KO\_sswfrd]

\*/\*!KO\_sswfrd(dis%bin%name)\*/\*.

\*Version 1.

\*Берет квадратную симметричную матрицу расстояний между точками DIS. Расстояния должны быть

\*квадратные евклидовы; во всяком случае, ф-ция воспримет их как квадратные евклидовы.

\*И берет одну или более двоичных (1 vs 0) переменных, образующих матрицу BIN с числом рядов как в DIS.

\*Каждый столбец BIN означает группу, и значение 1 означает, что точка принадлежит ей.

\*В целом, группы могут пересекаться или непересекаться составом точек (в последнем случае BIN

\*является набором фиктивных, dummy, переменных). Всякий столбец BIN должен содержать по меньшей

\*мере одну единицу. Столбец может, как частный случай, не содержать нулей.

\*Результат NAME - столбец длиной как число столбцов BIN, показывающий сумму квадратов отклонений

\*от центроида (SSW) в каждой из этих групп.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

ПРИМЕР. Вычисление SSwithin из данных и из расстояний, во всей выборке.

matrix.

get vars /variables= v1 v2 v3.

compute bin= make(nrow(vars),1,1). /\*One group, the whole sample

!KO\_aggr(vars%bin%SSDEV%ssdev).

print rsum(ssdev). /\*Multivariate SSwithin

!KO\_seuclid(vars%d). /\*Matrix of squared euclidean distances

!KO\_sswfrd(d%bin%ssw).

print ssw. /\*Multivariate SSwithin (same value)

end matrix.

### K БЛИЖНИХ СОСЕДЕЙ (ВЫПИСКА), ВЕРСИЯ "RANDOM" [!KO\_knnr]

\*/\*!KO\_knnr(dis%k%name1%name2)\*/\*.

\*Version 1.

\*Берет матрицу расстояний (различий) DIS (R x C) между точками-рядами и точками-столбцами.

\*Для каждой из C точек-столбцов находит K ближних (ближайших) к ней точек-рядов и выписывает их

\*в NAME1 (номера, индексы их) и NAME2 (расстояния до них). Элементы в NAME1 и NAME2

\*взаимосоответственны, соседи идут в столбце сверху вниз в порядке отдаления от точки-столбца.

\*DIS - прямоугольная или квадратная матрица с неотрицательными элементами.

\*В том случае, если DIS представляет собой не два, а один набор точек с расстояниями между друг другом,

\*DIS будет квадратной симметричной, и вы можете захотеть игнорировать ее диагональные элементы

\*(расстояния точек до себя); тогда поставьте на диагональ какое-нибудь число, заведомо большее, чем

\*остальные элементы матрицы.

\*K - число ближних соседей, положительный скаляр <=R (обычно K<<R).

\*NAME1 и NAME2 имеют размер K x C.

\*Если DIS это дискретные значения, т.е. содержат ties, результат может частично зависеть от зерна

\*случайных чисел (см. ниже). Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Данная функция выгодна, когда K<<R. Если K велико и приближается к R, выгоднее (с т.з. скорости) может

\*быть использование сортировки /\*!KO\_sort\*/ каждого столбца DIS с последующим отбором K первых рядов

\*результата.

\*Различие между /\*!KO\_knnr\*/ и /\*!KO\_knnp\*/.

\*Эти функции эквивалентны, если DIS - континуальные значения (но /\*!KO\_knnp\*/ тогда несколько быстрее).

\*Если DIS - дискретные значения, т.е. с ties, эти функции проявляют алгоритмическое различие,

\*заключающееся в следующем. Если точек, претендующих на ближнее соседство, окажется

\*больше, чем K, /\*!KO\_knnr\*/ выберет из tied соседей случайно, чтобы осталось ровно K

\*ближних соседей; а /\*!KO\_knnp\*/ выдаст всех ближних соседей, числом больше K. Таким образом, в

\*ситуации дискретных DIS результат /\*!KO\_knnr\*/ может зависеть от зерна случайных чисел, а /\*!KO\_knnp\*/

\*может выдать для точки соседей больше, чем K. Пример. Пусть K=2, и ближние соседи точки имеют до нее

\*расстояния 2 3 3 3 4. /\*!KO\_knnr\*/ выдаст два ближних соседа - с расстояниями 2 и 3, выбрав из трех "3"

\*одно случайно. /\*!KO\_knnp\*/ выдаст четыре ближних соседа: 2 3 3 3.

ПРИМЕР. См. прямо ниже.

### K БЛИЖНИХ СОСЕДЕЙ (ВЫПИСКА), ВЕРСИЯ "PLUS" [!KO\_knnp]

\*/\*!KO\_knnp(dis%k%name1%name2)\*/\*.

\*Version 1.

\*Берет матрицу расстояний (различий) DIS (R x C) между точками-рядами и точками-столбцами.

\*Для каждой из C точек-столбцов находит K ближних (ближайших) к ней точек-рядов и выписывает их

\*в NAME1 (номера, индексы их) и NAME2 (расстояния до них). Элементы в NAME1 и NAME2

\*взаимосоответственны, соседи идут в столбце сверху вниз в порядке отдаления от точки-столбца.

\*DIS - прямоугольная или квадратная матрица с неотрицательными элементами.

\*В том случае, если DIS представляет собой не два, а один набор точек с расстояниями между друг другом,

\*DIS будет квадратной симметричной, и вы можете захотеть игнорировать ее диагональные элементы

\*(расстояния точек до себя); тогда поставьте на диагональ какое-нибудь число, заведомо большее, чем

\*остальные элементы матрицы.

\*K - число ближних соседей, положительный скаляр <=R (обычно K<<R).

\*Если DIS это континуальные значения, то NAME1 и NAME2 будут иметь размер K x C.

\*Если DIS это дискретные значения, число рядов в NAME1 и NAME2 может оказаться больше K (см. об этом

\*ниже). Значение 0 в NAME1 означает пустую ячейку и, соответственно, пустую ячейку в этой позиции в NAME2.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Данная функция выгодна, когда K<<R. Если K велико и приближается к R, выгоднее (с т.з. скорости) может

\*быть использование сортировки /\*!KO\_sort\*/ каждого столбца DIS с последующим отбором K первых рядов

\*результата.

\*Различие между /\*!KO\_knnr\*/ и /\*!KO\_knnp\*/.

\*Эти функции эквивалентны, если DIS - континуальные значения (но /\*!KO\_knnp\*/ тогда несколько быстрее).

\*Если DIS - дискретные значения, т.е. с ties, эти функции проявляют алгоритмическое различие,

\*заключающееся в следующем. Если точек, претендующих на ближнее соседство, окажется

\*больше, чем K, /\*!KO\_knnr\*/ выберет из tied соседей случайно, чтобы осталось ровно K

\*ближних соседей; а /\*!KO\_knnp\*/ выдаст всех ближних соседей, числом больше K. Таким образом, в

\*ситуации дискретных DIS результат /\*!KO\_knnr\*/ может зависеть от зерна случайных чисел, а /\*!KO\_knnp\*/

\*может выдать для точки соседей больше, чем K. Пример. Пусть K=2, и ближние соседи точки имеют до нее

\*расстояния 2 3 3 3 4. /\*!KO\_knnr\*/ выдаст два ближних соседа - с расстояниями 2 и 3, выбрав из трех "3"

\*одно случайно. /\*!KO\_knnp\*/ выдаст четыре ближних соседа: 2 3 3 3.

ПРИМЕР. Континуальные входящие расстояния.

set mxloops 1E6.

set seed 56986875.

matrix.

compute d= uniform(8,6).

!KO\_knnr(d%4%ind%dist).

print ind.

print dist.

!KO\_knnp(d%4%ind%dist).

print ind.

print dist.

end matrix.

ПРИМЕР. Дискретные входящие расстояния.

set mxloops 1E6.

set seed 56986875.

matrix.

compute d= rnd(uniform(8,6)\*5).

!KO\_knnr(d%4%ind%dist).

print ind.

print dist.

!KO\_knnp(d%4%ind%dist).

print ind.

print dist.

end matrix.

### ВНУТРИКЛАССОВЫЙ КОЭФФИЦИЕНТ КОРРЕЛЯЦИИ (ОДНОФАКТОРНЫЙ) [!KO\_iccow]

\*/\*!KO\_iccow(col%dummy%type%name)\*/\*.

\*Version1.

\*Вычисляет внутриклассовую корреляцию (ICC) в ситуации однофакторного случайного дизайна:

\*скаляр NAME. Классы могут быть неравночисленными (unbalanced).

\*COL - столбец количественных данных.

\*DUMMY - двоичные фиктивные переменные (минимум две), помечающие классы (группы): каждая переменная

\*соответствует группе, значение 1 = наблюдение принадлежит ей, 0 = не принадлежит ей.

\*В каждом ряду DUMMY должна быть ровно одна единица.

\*В каждом столбце DUMMY должно быть минимум две единицы (при TYPE=PAIR2 или PAIR3), либо минимум одна

\*единица, и хотя бы в одном столбце больше одной (при остальных TYPE).

\*TYPE - версия оценивателя (estimator) ICC, ключевое слово заглавными буквами (опционально в кавычках

\*или апострофах) [см. Donner, 1986: A review of inference procedures...; Shieh, 2012: A comparison

\*of two indices...]:

\*"ANOVA" - ANOVA-оцениватель.

\*"CETASQ" - оцениватель Corrected Eta-square.

\*"PAIR1" - оцениватель Combinatorial pairing r Pearson без взвешивания классов, классы большего

\*размера будут информационно слишком превалировать.

\*"PAIR2" - то же, но с взвешиванием классов: классам придаются равные веса важности, несмотря на

\*разночисленность.

\*"PAIR3" - то же с промежуточным, компромиссным вариантом взвешивания: большие классы будут умеренно

\*превалировать.

\*Если все классы (группы) одного размера (т.е. balanced), то PAIR1=PAIR2=PAIR3=CETASQ.

\*NAME в данных иногда бывает отрицательным, хотя ICC в популяции неотрицателен; вы можете

\*воспользоваться советом занулять отрицательные значения.

\*Фиктивные переменные DUMMY вы можете создать из категориальной группирующей переменной

\*с помощью функции /\*!KO\_freq\*/.

### ВНУТРИКЛАССОВЫЙ КОЭФФИЦИЕНТ КОРРЕЛЯЦИИ (ОДНОФАКТОРНЫЙ, СБАЛАНСИРОВАННЫЕ КЛАССЫ) [!KO\_iccowb]

\*/\*!KO\_iccowb(data%type%name)\*/\*.

\*Version1.

\*Эта функция вычисляет тот же ICC, что ф-ция /\*!KO\_iccow\*/, но рассчитана на частный случай

\*равночисленных (balanced) классов. Входящие тут должны быть оформлены в широком (wide) формате:

\*DATA это матрица данных (не меньше 2x2) классы x замеры, т.е. каждый класс это ряд, все ряды

\*содержат одно и то же число значений. Не следует придавать столбцам DATA смысл регулярных

\*оценщиков, поскольку в данном случае принимается однофакторный случайный дизайн.

\*TYPE - версия оценивателя (estimator) ICC, ключевое слово заглавными буквами (опционально в кавычках

\*или апострофах) [см. Donner, 1986: A review of inference procedures...]

\*"ANOVA" - ANOVA-оцениватель.

\*"PAIR" - оцениватель Combinatorial pairing r Pearson.

\*Т.к. группы одного размера, pairing оцениватель тождествен оценивателю Corrected Eta-square и

\*оценивателю максимального правдоподобия, и он несмещенный, в отличие от ANOVA-оценивателя.

\*NAME в данных иногда бывает отрицательным, хотя ICC в популяции неотрицателен; вы можете

\*воспользоваться советом занулять отрицательные значения.

### КОЭФФИЦИЕНТ ДИАГОНАЛЬНОСТИ МАТРИЦЫ [!KO\_diagns]

\*/\*!KO\_diagns(mat%name)\*/\*.

\*Version 1.

\*Один из возможных коэффициентов, измеряющий степень концентрации больших (по абсолютной величине)

\*значений матрицы вблизи ее главной диагонали. Колеблется в пределах от 0 до 1.

\*Пусть MAT произвольная матрица, тогда коэффициент есть 1-D/Dmax, где D это взвешенное манхэттенское

\*физическое расстояние элементов от главной диагонали, равное Sum[abs(MAT)&\*OFF], где OFF это

\*матрица размером с MAT, главная диагональ в которой состоит из 0, пара прилегающих к ней диагоналей

\*(т.е. отстоящих от нее на порядок 1) состоят из 1, пара диагоналей, отстоящих от нее на порядок 2

\*состоят из 2, и так далее. (&\* это перемножение соответствующих элементов, а Sum это сумма элементов.)

\*Dmax это максимально возможное для данной MAT расстояние, т.е. наблюдаемое, когда все большие элементы

\*MAT концентрируются у лево-нижнего и право-верхнего углов матрицы; Dmax вычисляется так же, как D,

\*но после сортировки всех значений MAT по абс. величине и сортировке всех значений OFF по их величине.

\*Если MAT не квадратна, то не одна, а несколько диагоналей в ней претендуют называться "главной диагональю".

\*Это все диагонали (идущие слева-сверху вправо-вниз) одинаковой максимальной длины. Например,

\*в матрице размером 4x6 это три диагонали, начинающиеся соответственно с позиций (1,1), (1,2) и (1,3).

\*Функция выдаст, соответственно, три коэффициента, по одному в отношении каждой диагонали: результат NAME

\*это вектор-столбец, содержащий столько коэффициентов, сколько "главных диагоналей" помещается в матрицу,

\*и 1-й коэффициент относится к диагонали, начинающейся с позиции (1,1).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### РОБАСТНЫЕ К ВЫБРОСАМ СРЕДНЯЯ И ДИСПЕРСИЯ (ADP) [!KO\_robustadp]

\*/\*!KO\_robustadp(col%threshold%name1%name2)\*/\*.

\*Version 1.

\*Для мерной переменной COL (столбец) вычисляет робастные по отношению к возможным выбросам в ней

\*значения средней (NAME1) и дисперсии (NAME2).

\*Функция использует равноинтервальный алгоритм SPSS-команды ADP (см. SPSS Statistics Algorithms -

\*Automated Data Preparation Algorithms - Outlier Identification and Handling).

\*THRESHOLD - число между 0 и 1. Это максимальная доля крайних наблюдений, которые алгоритм

\*удержит от влияния на вычисляемые робастные статистики, посчитав первые потенциальными выбросами.

\*Например, THRESHOLD=0.05 исключит из влияния не более 5% крайних наблюдений.

\*Суть алгоритма проста: данные разбиваются на интервалы шириной в 1 sd; отсекаются крайние интервалы,

\*включающие в себя долю наблюдений, приближенную снизу к THRESHOLD; робастные статистики вычисляются

\*как обычные средняя и дисперсия на оставшихся наблюдениях.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

get x /variable= x.

!KO\_robustadp(x%.05%robmean%robvrnc).

compute robsd= sqrt(robvrnc).

print robmean /format= f8.6.

print robsd /format= f8.6.

compute flag= x<robmean-robsd\*3 or x>robmean+robsd\*3.

print {x,flag}. /\*Flag as outliers values that are farther

/\*than the 3 robust sd from the robust mean

end matrix.

### РОБАСТНЫЕ К ВЫБРОСАМ СРЕДНЯЯ И ДИСПЕРСИЯ (LTS) [!KO\_robustlts]

\*/\*!KO\_robustlts(col%threshold%name1%name2)\*/\*.

\*Version 1.

\*Для мерной переменной COL (столбец) вычисляет робастные по отношению к возможным выбросам в ней

\*значения средней (NAME1) и дисперсии (NAME2).

\*Функция использует алгоритм LTS (Least Trimmed Squares, см. Rousseeuw, Leroy (1987),

\*"Robust regression and outlier detection", p. 171-172).

\*THRESHOLD - число между 0 и 1 (невключительно). Это доля крайних наблюдений, которые алгоритм

\*удержит от влияния на вычисляемые робастные статистики, посчитав первые потенциальными выбросами.

\*Например, THRESHOLD=0.05 исключит из влияния 5% крайних наблюдений.

\*Суть алгоритма: N наблюдений сортируются по возрастанию значений; N-trunc(THRESHOLD\*N) это ширина

\*полосы наблюдений, на которых вычисляются средняя и дисперсия, эта полоса начинается слева и

\*сдвигается каждый раз на одно наблюдение вправо. Функция выдает в качестве результата наименьшую

\*вычисленную так дисперсию и соответствующую ей среднюю.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

get x /variable= x.

!KO\_robustlts(x%.05%robmean%robvrnc).

compute robsd= sqrt(robvrnc).

print robmean /format= f8.6.

print robsd /format= f8.6.

compute flag= x<robmean-robsd\*3 or x>robmean+robsd\*3.

print {x,flag}. /\*Flag as outliers values that are farther

/\*than the 3 robust sd from the robust mean

end matrix.

### УСЕЧЕННАЯ СРЕДНЯЯ [!KO\_trimmean]

\*/\*!KO\_trimmean(col%left%right%%name%)\*/\*.

\*Version 1.

\*Берет столбец данных COL и вычисляет усеченную среднюю NAME.

\*LEFT - доля наблюдений, которые отсечь слева (со стороны низких значений).

\*RIGHT - доля наблюдений, которые отсечь справа (со стороны высоких значений).

\*LEFT+RIGHT должно быть меньше 1.

\*Если число отсекаемых наблюдений (N\*LEFT или N\*RIGHT) есть число дробное,

\*пограничное наблюдение отсекается частично - на столько, какова дробная часть.

### M-ОЦЕНИВАТЕЛИ ПОЛОЖЕНИЯ [!KO\_mestim]

\*/\*!KO\_mestim(col%method%const%conv%name)\*/\*.

\*Version 1.

\*Для мерной переменной (столбец данных COL) вычисляет робастный M-оцениватель положения.

\*METHOD - ключевое слово заглавными буквами (опционально можно в кавычках или апострофах):

\*"HUBER", "ANDREW" или "TUKEY".

\*CONST - положительный скаляр (весовая константа). Обычно употребляемые значения следующие:

\*1.339 для HUBER, 1.34 для ANDREW, 4.685 для TUKEY.

\*CONV - коэффициент, регулирующий схождение на итерациях вычисления, малый положительный скаляр

\*(к примеру, 0.005 или 0.001).

\*Алгоритм - см. SPSS Statistics Algorithms - EXAMINE - M-Estimation (Robust Location Estimation).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

get x /variable= x.

!KO\_mestim(x%HUBER%1.339%.005%m).

print m /title "Huber's M-estimator of location".

end matrix.

### АЛЬФА КРОНБАХА [!KO\_cronalpha]

\*/\*!KO\_cronalpha(cov%name)\*/\*.

\*Version 1.

\*Вычисляет меру внутренней согласованности (один из показателей надежности) - альфу Кронбаха.

\*COV - p x p (p>2) матрица ковариаций (или корреляций) между переменными (пунктами теста).

\*В норме всем ковариациям следует быть положительными. Корреляции вместо ковариаций отвечают

\*переменным стандартизованным.

\*Альфа NAME - положительный (в норме) коэффициент между 0 и 1.

\*Отрицательное значение коэффициента возможно при отрицательных некоторых ковариациях.

### ОМЕГА МАКДОНАЛЬДА [!KO\_mcdomega]

\*/\*!KO\_mcdomega(cov%name)\*/\*.

\*Version 1.

\*Вычисляет меру внутренней согласованности (один из показателей надежности) - омегу Макдональда.

\*Используется метод Hancock and Ann, 2020, - см. SPSS Statistics Algorithms - Reliability -

\*Estimation of McDonald's Omega.

\*COV - p x p (p>2) матрица ковариаций (или корреляций) между переменными (пунктами теста).

\*В норме всем ковариациям следует быть положительными. Корреляции вместо ковариаций отвечают

\*переменным стандартизованным.

\*Омега NAME - положительный коэффициент между 0 и 1.

\*NAME = -1 означает, что омега не смогла быть вычислена из-за неположительных ковариаций.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

# СТАТИСТИЧЕСКИЕ ПРЕОБРАЗОВАТЕЛЬНЫЕ ФУНКЦИИ

# 

### ЦЕНТРАЦИЯ СТОЛБЦОВ ДАННЫХ [!KO\_center]

\*/\*!KO\_center(data%name)\*/\*.

\*Version 1.

\*Центрует столбцы DATA (приводит среднее в них к 0).

\*Выдает данные NAME.

ПРИМЕР.

matrix.

get vars /variables= v1 v2 v3.

!KO\_center(vars%vars).

save vars /outfile= \* /variables= v1 v2 v3.

end matrix.

### Z-СТАНДАРТИЗАЦИЯ СТОЛБЦОВ ДАННЫХ [!KO\_zscore]

\*/\*!KO\_zscore(data%name)\*/\*.

\*Version 1.

\*Стандартизует столбцы DATA (приводит среднее в них к 0, а дисперсию к 1).

\*Выдает данные NAME.

### Z-СТАНДАРТИЗАЦИЯ СТОЛБЦОВ ДАННЫХ (DF=N) [!KO\_zscore2]

\*/\*!KO\_zscore2(data%name)\*/\*.

\*Version 1.

\*Стандартизует столбцы DATA (приводит среднее в них к 0, а дисперсию к 1).

\*Стандартное отклонение, стандартизующее данные, посчитано на "df=n" (а не "df=n-1").

\*Выдает данные NAME.

### МАСШТАБИРОВАНИЕ СТОЛБЦОВ ДАННЫХ К SS=1 [!KO\_scale]

\*/\*!KO\_scale(data%name)\*/\*.

\*Version 1.

\*Масштабирует столбцы DATA (приводит сумму квадратов значений, SS, к 1).

\*Выдает данные NAME.

\*Если надо сделать SS=число\_рядов DATA, умножьте результат на sqrt(число\_рядов).

### ЦЕНТРАЦИЯ СТОЛБЦОВ ДАННЫХ, ПО ГРУППАМ [!KO\_gcenter]

\*/\*!KO\_gcenter(data%dummy%name)\*/\*.

\*Version 1.

\*Центрует столбцы DATA (приводит среднее в них к 0), по группам наблюдений.

\*Выдает данные NAME.

\*DUMMY - двоичные фиктивные переменные, помечающие группы (каждая переменная соответствует группе,

\*значение 1 = наблюдение принадлежит ей, 0 = не принадлежит ей).

\*Всякий столбец DUMMY должен содержать по меньшей мере одну единицу.

\*Группы должны быть непересекающиеся составом: сумма в рядах DUMMY не может превосходить единицу.

\*Если в DUMMY есть ряды, полные нулей, т.е. наблюдения вне групп, то данные для них останутся не тронуты.

\*Фиктивные переменные DUMMY вы можете создать из категориальной группирующей переменной

\*с помощью функции /\*!KO\_freq\*/.

### Z-СТАНДАРТИЗАЦИЯ СТОЛБЦОВ ДАННЫХ, ПО ГРУППАМ [!KO\_gzscore]

\*/\*!KO\_gzscore(data%dummy%df%name)\*/\*.

\*Version 2.

\*Стандартизует столбцы DATA (приводит среднее в них к 0, а дисперсию к 1), по группам наблюдений.

\*Выдает данные NAME.

\*DUMMY - двоичные фиктивные переменные, помечающие группы (каждая переменная соответствует группе,

\*значение 1 = наблюдение принадлежит ей, 0 = не принадлежит ей).

\*Группы должны быть непересекающиеся составом, и каждое наблюдение должно принадлежать группе,

\*т.е. сумма в каждом ряду DUMMY должна равняться единице.

\*Всякий столбец DUMMY должен содержать по меньшей мере две единицы.

\*DF - скаляр. Если положительный, стандартизующее стандартное отклонение будет посчитано на "df=n-1"

\*(тогда в каждом столбце DUMMY должно быть не менее 2 единиц); если неположительный, стандартное

\*отклонение будет посчитано на "df=n" (в каждом столбце DUMMY должно быть не менее 1 единицы).

\*Фиктивные переменные DUMMY вы можете создать из категориальной группирующей переменной

\*с помощью функции /\*!KO\_freq\*/.

### МАСШТАБИРОВАНИЕ СТОЛБЦОВ ДАННЫХ, ПО ГРУППАМ [!KO\_gscale]

\*/\*!KO\_gscale(data%dummy%scale%name)\*/\*.

\*Version 1.

\*Масштабирует столбцы DATA, по группам наблюдений.

\*Выдает данные NAME.

\*SCALE - скаляр. Если неположительный, то сумма квадратов значений переменной в каждой группе

\*будет приведена к 1; а если положительный, то - к численности в данной группе (т.е. средний квадрат будет 1).

\*DUMMY - двоичные фиктивные переменные, помечающие группы (каждая переменная соответствует группе,

\*значение 1 = наблюдение принадлежит ей, 0 = не принадлежит ей).

\*Всякий столбец DUMMY должен содержать по меньшей мере одну единицу.

\*Группы должны быть непересекающиеся составом, и каждое наблюдение должно принадлежать группе,

\*т.е. сумма в каждом ряду DUMMY должна равняться единице.

\*Фиктивные переменные DUMMY вы можете создать из категориальной группирующей переменной

\*с помощью функции /\*!KO\_freq\*/.

### ЛИНЕЙНОЕ ПЕРЕШКАЛИРОВАНИЕ [!KO\_rescale]

\*/\*!KO\_rescale(data%type%val1%val2%name)\*/\*.

\*Version 1.

\*Линейно перешкалирует переменные данных (столбцы DATA) к требуемым величинам статистик в этих данных,

\*таких как минимум, максимум, средняя, ст. отклонение.

\*Данные DATA должны состоять не менее чем из двух рядов и переменные (столбцы) должны не быть

\*константами.

\*Каждая переменная перешкалируется независимо; результат NAME в каждом столбце будет иметь

\*линейно преобразованные данные, точно удовлетворяющие двум заданным значениям VAL1 и VAL2

\*двух соответственных, согласно TYPE, статистик.

\*VAL1 и VAL2 - скаляры либо векторы-ряды длиной как число столбцов DATA; вектор(ы) означает, что

\*переменные надо привести каждую к своему заданному значению статистики.

\*TYPE (ключевое слово заглавными буквами, опционально в кавычках или апострофах) задает,

\*к каким двум статистикам привести данные в переменных:

\*"MIN\_MAX" - к заданным минимуму (VAL1) и максимуму (VAL2), (VAL1<VAL2)

\*"MIN\_MEAN" - к заданным минимуму (VAL1) и средней (VAL2), (VAL1<VAL2)

\*"MAX\_MEAN" - к заданным максимуму (VAL1) и средней (VAL2), (VAL1>VAL2)

\*"MIN\_SD" - к заданным минимуму (VAL1) и стандартному отклонению (VAL2), (VAL2 положительное)

\*"MAX\_SD" - к заданным максимуму (VAL1) и стандартному отклонению (VAL2), (VAL2 положительное)

\*"MEAN\_SD" - к заданным средней (VAL1) и стандартному отклонению (VAL2), (VAL2 положительное).

ПРИМЕР.

matrix.

get vars /variables= v1 v2 v3 /names= names.

compute varmin= cmin(vars).

print varmin. /\*Minimal values, row vector

!KO\_rescale(vars%MIN\_MEAN%varmin%10%newvars). /\*Rescale variables to have same

/\*mean 10 while keeping their original minimal values

/\*(10 must be greater than any of those)

print cmin(newvars).

print (csum(newvars)/nrow(newvars)).

save newvars /out= \* /names= names.

end matrix.

### ДВУВХОДОВАЯ ЦЕНТРАЦИЯ

Это есть центрация элементов матрицы MAT по рядам и столбцам вместе: MAT-R\*C/N, где R - столбец рядных сумм, C - ряд столбцовых сумм, N - общая сумма. Двувходовая центрация пропорционально нечувствительна к умножению исходной матрицы на константу, а не к прибавлению константы. Вы можете получить такую центрацию ф-цией /\*!KO\_cells\*/, затребовав в ней RESID – частотные остатки математически есть двувходовая центрация таблицы чисел.

### КОНВЕРТАЦИЯ КОВАРИАЦИОННОЙ МАТРИЦЫ В КОРРЕЛЯЦИОННУЮ [!KO\_covcorr]

\*/\*!KO\_covcorr(cov%name)\*/.

\*Version 1.

\*Конвертирует ковариационную матрицу в корреляционную (или sscp матрицу в косинусную).

\*Полученная матрица NAME имеет единицы на диагонали.

\*Входящая матрица COV должна иметь положительные диагональные элементы.

### КОНВЕРТАЦИЯ КОРРЕЛЯЦИОННОЙ МАТРИЦЫ В КОВАРИАЦИОННУЮ [!KO\_corrcov]

\*/\*!KO\_corrcov(corr%dg%name)\*/.

\*Version 1.

\*Конвертирует корреляционную матрицу в ковариационную (или косинусную матрицу в sscp матрицу).

\*Полученная матрица NAME имеет на диагонали вектор DG.

\*Входящая матрица CORR должна иметь единицы на диагонали.

\*DG - вектор-столбец положительных значений, длиной как число рядов/столбцов CORR.

### ДВОЙНАЯ ЦЕНТРАЦИЯ МАТРИЦЫ КВАДРАТНЫХ РАССТОЯНИЙ [!KO\_dcenter]

\*/\*!KO\_dcenter(dis%name)\*/\*.

\*Version 1.

\*Переводит матрицу квадратных различий DIS в матрицу сходств NAME, который суть скалярные произведения.

\*Если DIS это квадратные различия между точками облака, то на диагонали NAME - квадратные евклидовы

\*расстояния от точек до центроида облака, а внедиагональные элементы - скалярные произведения

\*между векторами, исходящими из центроида и кончающимися на точках.

\*Если различия это квадратные евклидовы расстояния, то данная операция есть "геометрически корректный"

\*перевод их в сходства. Именно, тогда NAME=X\*t(X), где матрица X - любые данные (точки на измерения,

\*причем измерения, столбцы, центрованы), имеющие между точками кв. евклидовы расстояния, равные DIS.

\*В матрице-двойном центрате NAME рядные и столбцовые суммы равны 0.

\*DIS должна быть квадратной симметрической с неотрицательными значениями, с 0 на диагонали.

\*Для обратной конвертации используйте противоположную функцию /\*!KO\_sdcosth\*/.

### КОНВЕРТАЦИЯ МАТРИЦЫ УГЛОВЫХ СХОДСТВ В КВАДРАТНЫЕ РАССТОЯНИЯ ПО ТЕОРЕМЕ КОСИНУСОВ [!KO\_sdcosth]

\*/\*!KO\_sdcosth(sim%name)\*/\*.

\*Version 1.

\*Переводит матрицу сходств SIM, которые принимает за скалярные произведения, в матрицу квадратных

\*различий NAME. Диагональные значения SIM используются как квадраты расстояний от точек до точки начала

\*пространства. Если SIM это скалярные произвеления и SIM положительно полуопределена, то различия NAME это

\*квадратные евклидовы расстояния. Если SIM не положительно полуопределенная, NAME иногда могут оказаться

\*квадратными евклидовыми (сходиться в евклидовом пространстве).

\*Данная операция - "геометрически корректная" конвертация ковариаций, корреляций или косинусов в

\*соответствующие им квадратные евклидовы расстояния.

\*SIM должна быть квадратной симметричной с положительными значениями на диагонали.

\*Эта функция противоположна функциям /\*!KO\_dcenter\*/ и /\*!KO\_dscosth\*/.

### ПЕРЕКЛЮЧЕНИЕ МАТРИЦЫ СКАЛЯРНЫХ ПРОИЗВЕДЕНИЙ НА НОВУЮ ДИАГОНАЛЬ (СОХРАНЯЯ УГОЛ) [!KO\_swdiag1]

\*/\*!KO\_swdiag1(sim%diag%name)\*/.

\*Version 1.

\*Пересчитывает элементы матрицы SSCP-типа (= матрицы скалярных произведений; к ней относятся,

\*в частности, ковариационная, корреляционная, косинусная матрицы) на заданные диагональные значения.

\*При этом неизменными удерживаются углы (косинусы) между векторами.

\*Этой функцией можно, например, превратить ковариационную матрицу в корреляционную или наоборот.

\*SIM должна квадратной симметричной, с положительными элементами на диагонали. Матрица трактуется

\*как матрица близостей SSCP-типа (матрица скалярных произведений).

\*DIAG - вектор (ряд или столбец) длиной как число рядов/столбцов SIM, содержащий положительные элементы;

\*это новая диагональ для SIM.

\*Результат NAME это SIM с диагональю DIAG и пересчитанными внедиагональными элементами.

\*Если входящая SIM грамова (все собственные числа неотрицательны), NAME тоже грамова.

ПРИМЕР.

matrix.

compute vars= uniform(20,8). /\*Some columns, they are the vectors

compute sp= sscp(vars). /\*Their sscp matrix, the scalar products

print sp.

compute newdiag= uniform(1,8)\*2+1. /\*Let this be the new diagonal for it

/\*All values must be positive

print newdiag.

!KO\_swdiag1(sp%newdiag%newsp). /\*Switch sp onto the new diagonal

print newsp.

print eval(newsp). /\*The new matrix will always be gramian (provided the initial

/\*one was)

\*Now check the cosines (angles) between the vectors.

!KO\_cosine(vars%csn). /\*The cosines in the initial matrix

/\*(can get them right from the data)

print csn.

!KO\_covcorr(newsp%newcsn). /\*What the cosines became after we replaced the diagonal

/\*(we can get it from newsp exactly like we convert a

/\*covariance matrix into correlation)

print newcsn.

print abs(csn-newcsn). /\*The cosines appear to be preserved

end matrix.

### ПЕРЕКЛЮЧЕНИЕ МАТРИЦЫ СКАЛЯРНЫХ ПРОИЗВЕДЕНИЙ НА НОВУЮ ДИАГОНАЛЬ (СОХРАНЯЯ РАССТОЯНИЕ) [!KO\_swdiag2]

\*/\*!KO\_swdiag2(sim%diag%name)\*/.

\*Version 1.

\*Пересчитывает элементы матрицы SSCP-типа (= матрицы скалярных произведений; к ней относятся,

\*в частности, ковариационная, корреляционная, косинусная матрицы) на заданные диагональные значения.

\*При этом неизменными удерживаются евклидовы расстояния между концами векторов.

\*SIM должна квадратной симметричной, с положительными элементами на диагонали. Матрица трактуется

\*как матрица близостей SSCP-типа (матрица скалярных произведений).

\*DIAG - вектор (ряд или столбец) длиной как число рядов/столбцов SIM, содержащий в норме положительные

\*элементы; это новая диагональ для SIM.

\*Результат NAME это SIM с диагональю DIAG и пересчитанными внедиагональными элементами.

\*В зависимости от DIAG (и, конечно, от входящей SIM), NAME может оказаться грамовой

\*(все собственные числа неотрицательны) или неграмовой.

ПРИМЕР.

matrix.

compute vars= uniform(20,8). /\*Some columns, they are the vectors

compute sp= sscp(vars). /\*Their sscp matrix, the scalar products

print sp.

compute newdiag= uniform(1,8)\*2+1. /\*Let this be the new diagonal for it

/\*Normally the values should be positive (though mathematically there is

/\*no such requirement)

print newdiag.

!KO\_swdiag2(sp%newdiag%newsp). /\*Switch sp onto the new diagonal

print newsp.

print eval(newsp). /\*The new matrix may be gramian or nongramian depending on newdiag

/\*(and whether the initial matrix was)

\*Now check the euclidean distances between the vectors’ endpoints.

!KO\_seuclid(t(vars)%d). /\*The (squared) distances in the initial matrix

/\*(can get them right from the data)

print d.

!KO\_sdcosth(newsp%newd). /\*What the distances became after we replaced the diagonal

/\*(we can get it from newsp by convertion through

/\*the cosine theorem

print newd.

print abs(d-newd). /\*The distances appear to be preserved

end matrix.

### ПЕРЕКЛЮЧЕНИЕ MSCP-МАТРИЦЫ НА НОВЫЙ ЦЕНТРОИД ДАННЫХ [!KO\_swcentr]

\*/\*!KO\_swcentr(sim%oldmean%newmean%name)\*/.

\*Version 1.

\*Пересчитывает элементы MSCP матрицы, заданной столбцами неких данных, если тем данным сдвигом поменять

\*центроид (многомерную среднюю). Функция не нуждается в самих данных. Допустим, что между некоторыми

\*переменными была вычислена MSCP-матрица (это SSCP-матрица - т.е. матрица t(данные)\*данные, - деленная на

\*число наблюдений в данных); а также вычислен вектор средних у переменных. Затем данные утеряны. Какова

\*была бы новая MSCP-матрица после того, как переменным поменять произвольно средние путем прибавления к

\*каждой переменной некоторой константы? Функция отвечает на этот вопрос в отсутствие самих данных.

\*Например, с помощью этой Функции можно восстановить MSCP-матрицу исходных данных из ковариационной

\*матрицы (которая, как известно, тоже является MSCP-матрицей), если изначальные средние у переменных

\*известны.

\*SIM - квадратная симметрическая MSCP матрица между столбцами некоторых данных; средние в тех столбцах

\*записаны как вектор (ряд или столбец) OLDMEAN. После сдвига этого центроида на новые координаты NEWMEAN

\*новая MSCP-матрица, отвечающая сдвинутым данным, и есть матрица NAME, которая выдается.

\*Учтите, что результатная матрица не обязана быть грамовой (положительно [полу]-определенной) -

\*грамовость будет зависеть от входящей матрицы SIM и реалистичности OLDMEAN и NEWMEAN. Функция

\*не следит за реалистичностью входящих. Когда входящие статистики происходят от реальных переменных,

\*проблем с грамовостью должно не быть.

ПРИМЕР. Конвертация MSCP-матрицы в ковариационную и обратно: демонстрация.

matrix.

get vars /variables= v1 v2 v3 v4.

compute n= nrow(vars).

compute mscp= sscp(vars)/n.

!KO\_mean(vars%mean).

print mscp /title 'MSCP matrix of some variables'.

print mean /title 'Means (centroid) of those variables'.

!KO\_swcentr(mscp%mean%{0,0,0,0}%mscp0). /\*Translate the given MSCP onto the new

/\*centroid, in this example, zero

print mscp0 /title 'MSCP matrix shifted to the zero centroid...'.

!KO\_cov2(vars%cov2).

print cov2 /title '... is (by definition) the cov matrix (computed on "df=n")'.

!KO\_swcentr(mscp0%{0,0,0,0}%mean%mscp\_). /\*Now translate it back

print mscp\_ /title 'Shifting back the cov matrix into the original MSCP'.

end matrix.

ПРИМЕР. Вопрос о положительной определенности.

matrix.

compute mx= {1.8, 0.5, -0.2, 0.2;

0.5, 3.1, -0.1, 0.4;

-0.2, -0.1, 2, 0.3;

0.2, 0.4, 0.3, 1.6}. /\*A MSCP matrix

print mx.

print all(eval(mx)>0). /\*It is gramian (positive definite)

!KO\_swcentr(mx%{1.2,0,-1.1,0}%{0,0,0,0}%mx\_). /\*Assuming the variables had means

/\*{1.2,0,-1.1,0}, what would be their covariance matrix

/\*(i.e. MSCP with centroid 0)?

print mx\_. /\*It would be this

print all(eval(mx\_)>0). /\*Unfortunately, it is nongramian, that is, impossible

/\*in real variables; so, either MX or the means were unrealistic; Sorry

end matrix.

matrix.

compute mx= {3.1, 0.8, -0.2, 0.4;

0.8, 2.1, -0.1, 0.2;

-0.2, -0.1, 2.3, 0.3;

0.4, 0.2, 0.3, 1.9}.

print mx.

print all(eval(mx)>0).

!KO\_swcentr(mx%{1.2,0,-1.1,0}%{0,0,0,0}%mx\_).

print mx\_.

print all(eval(mx\_)>0). /\*Under these MX and the means, there is no problem,

/\*the cov matrix MX\_ could really exist

end matrix.

### ФИКТИВНЫЕ (DUMMY) ПЕРЕМЕННЫЕ

Перекодировать категориальную переменную в набор двоичных индикаторных (фиктивных) - см. функцию /\*!KO\_freq\*/. Если сортировать порядок фиктивных переменных по возрастанию кодов категорий не требуется, вы можете также воспользоваться встроенной функцией design(). Для обратной операции – превратить набор двоичных фиктивных переменных в одну категориальную переменную, используйте операцию dummy\*t(1:ncol(dummy)), где dummy – матрица фиктивных переменных.

### НАБОР ПЕРЕМЕННЫХ МНОЖЕСТВЕННОГО ОТВЕТА

Перекодировать категориальный набор множественного ответа (несколько категориальных переменных с общим пулом кодов) в двоичный набор множественного ответа (набор двоичных переменных) - см. функцию /\*!KO\_mrfreq\*/. Для обратной операции – превратить набор двоичных переменных в несколько категориальных переменных, составляющих категориальный набор множественного ответа, используйте функцию /\*!KO\_ram\*/ (см. там пример).

### РАНЖИРОВАНИЕ ЗНАЧЕНИЙ

Используйте встроенные функции grade() и rnkorder().

### РИДИТЫ ПОРЯДКОВЫХ ГРАДАЦИЙ [!KO\_ridit]

\*/\*!KO\_ridit(vec%name)\*/\*.

\*Version 1.

\*Вычисляет ридит-баллы порядковых градаций.

\*VEC - вектор (ряд или столбец), содержащий частоты (или доли) градаций в порядковой переменной.

\*1-й элемент - частота в 1-й градации, 2-й элемент - частота во 2-й градации, и так далее.

\*NAME - ридиты градаций.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ЭКСПОНЕНЦИАЛЬНОЕ НОРМИРОВАНИЕ (SOFTMAX-ФУНКЦИЯ) [!KO\_nef]

\*/\*!KO\_nef(data%name)\*/\*.

\*Version 1.

\*Экспоненциально нормирует элементы в столбцах DATA.

\*Результат NAME - в каждом столбце значения преобразованы экспоненциально так, что

\*они лежат в диапазоне [0,1] и их сумма равна 1.

### ЛОГ-ОТНОШЕНЧЕСКАЯ ЦЕНТРАЦИЯ (CENTERED LOGRATIO TRANSFORM) [!KO\_clr]

\*/\*!KO\_clr(data%name)\*/\*.

\*Version 1.

\*Делает centered logratio преобразование элементов в столбцах DATA. Значения в DATA

\*должны быть положительными.

\*Преобразование заключается в приведении сначала суммы в столбце к 1, затем делении

\*на геометрическую среднюю в столбце, и взятии логарифма.

\*Результат NAME - в каждом столбце значения имеют нулевую среднюю.

### ПРЕОБРАЗОВАНИЕ БОКСА-КОКСА [!KO\_boxcox]

\*/\*!KO\_boxcox(col%grid%name1%name2)\*/\*.

\*Version 1.

\*Преобразует количественную переменную COL монотонно степенно так, чтобы распределение в ней стало

\*ближе к нормальному. Поиск оптимальной величины степенного параметра лямбда делается поиском по

\*решетке (перебором) так же, как это делает SPSS-команда ADP, осуществляющая такое преобразование.

\*COL - входящая переменная, столбец положительных значений. Если имеются неположительрные значения,

\*прибавьте предварительно константу, чтобы сделать все значения положительными (как вариант, можно

\*всегда, независимо от наличия или отсутствия неположительных значений, сделать

\*COMPUTE COL= COL-cmin(COL)+1, так поступает, например, команда ADP).

\*GRID - вектор (ряд или столбец) из трех значений - это решетка для перебора значений лямбды:

\*{min,max,step}. Max должно быть > Min, Step положительно, и (Max-Min)/Step должно быть целым.

\*Например, часто используют решетку {-3,3,0.5}.

\*Результат:

\*NAME2 - выбранная степень лямбда преобразования Бокса-Кокса.

\*NAME1 - преобразованные ею данные COL. (Команда ADP еще дополнительно z-стандартизует ее,

\*но данная ф-ция этого не делает).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Преобразование Бокса-Кокса, как делает команда ADP.

matrix.

get y /variable= y.

compute y= y-cmin(y)+1.

!KO\_boxcox(y%{-3,3,0.5}%y\_%lambda).

!KO\_zscore(y\_%y\_).

print lambda.

save y\_ /out= \*.

end matrix.

### МАТРИЦА ОТБЕЛИВАНИЯ (СФЕРИЗАЦИИ) [!KO\_white]

\*/\*!KO\_white(cov%method%name)\*/\*.

\*Version 1.

\*Отбеливанием или сферизацией называется преобразование многомерных данных в "сферическое" облако.

\*Функция берет ковариационную или корреляционную матрицу COV и вычисляет матрицу отбеливания

\*(сферизации) NAME такую, что если COV соответствуют некоторые переменные X, то переменные Z=X\*NAME

\*будут иметь единичную ковариационную матрицу. Таким образом, отбеленные данные это декореллированные

\*и стандартизованные переменные, а функция выдает матрицу преобразования для этого.

\*COV - квадратная p x p симметричная матрица ковариаций или корреляций.

\*METHOD - метод отбеливания, ключевое заглавное слово (можно взять в кавычки или апострофы):

\*"ZCA" - нулевофазный компонентный метод; выдает NAME (симметричную при этом методе), минимизирующую

\*квадратные различия между значениями Z и X. Употребляйте этот метод, если нужно, чтобы переменные Z

\*минимально отличались от соответствующих им переменных X.

\*"PCA" - главнокомпонентный метод; выдает NAME, максимизирующую учет дисперсии X в первых столбцах Z;

\*Z есть ни что иное, как главные компоненты X, и значения Z это стандартизованные главнокомпонентные

\*баллы. Употребляйте этот метод, если пожелаете снизить размерность (число столбцов) Z по сравнению

\*с X, с наименьшими информационными потерями.

\*"CHOL" - метод Холецкого; выдает NAME (верхнетреугольную при этом методе), обеспечивающую полную

\*корреляцию Z1 с X1 и ортогональность Zp со всеми X, кроме Xp. Употребляйте, если нужно, чтобы Z1

\*была связана со всеми X, особенно с X1; Z2 не была связана с X1; Z3 не была связана с X1 и X2, и т.д.

\*При методах "ZCA" и "PCA" отбеленные переменные Z и их отношения с исходными переменными X будут

\*зависеть от того, ковариационной или корреляционной является COV (поэтому существенно, ковариационную

\*или корреляционную COV вы введете в функцию). Пусть матрица R есть ковариации (если COV ковариационная)

\*или корреляции (если COV корреляционная) между X-переменными (ряды R) и Z-переменными (столбцы R).

\*Тогда:

\*-метод "ZCA" максимизирует диагональные элементы R (они положительны).

\*-при методе "PCA" R есть матрица главнокомпонентных нагрузок, так что данный метод максимизирует

\*в первую очередь сумму квадратов элементов 1-го столбца R (т.е. ковариации/корреляции Z1 со всеми X),

\*во вторую очередь сумму квадратов элементов ее 2-го столбца (т.е. Z2 со всеми X), и т.д.

\*При методе "CHOL" отбеленные переменные Z будут одинаковы что при ковариационной, что при

\*корреляционной COV (поэтому не важно, корреляционную или ковариационную матрицу вы введете в функцию).

\*Этот метод ведет к корреляционной матрице R такой, что она есть нижняя треугольная с положительной

\*диагональю и элементом (1,1)=1, в последнем же ее столбце элементы кроме (p,p) - нули.

\*Важно отметить, что для всех методов должно соблюдаться правило: если COV - корреляционная, то

\*при отбеливании Z=X\*NAME столбцы X должны быть стандартизованными переменными (а не сырыми или

\*центрованными).

\*Методы "ZCA" и "PCA" связаны между собой через поворот данных так, что ZCA-отбеливание эквивалентно

\*PCA-отбеливанию с последующим прокрустовым поворотом Z обратно к X, и матрица такого поворота есть

\*собственные векторы COV.

ПРИМЕР. ZCA-отбеливание, максимизирующее корреляции.

matrix.

get x /variables= x1 x2 x3 x4.

!KO\_corr(x%corr). /\*Correlations

!KO\_white(corr%ZCA%w).

print w. /\*Whitening matrix

!KO\_zscore(x%x). /\*With correlation matrix as basis of whitening, X must be standard

compute z= x\*w. /\*Z is whitened X

!KO\_cov(z%covz).

print covz. /\*Covariances in Z are identity matrix

!KO\_corr({x,z}%corrxz).

print corrxz(1:4,5:8) /rlab= 'x1' 'x2' 'x3' 'x4' /clab= 'z1' 'z2' 'z3' 'z4'.

/\*Correlations between original and whitened variables:

/\*diagonal is maximized

save z /outfile= \* /variables= z1 z2 z3 z4.

end matrix.

### ПРЕОБРАЗОВАТЬ ПЕРЕМЕННЫЕ ТОЧНО К ЗАДАННЫМ КОВАРИАЦИЯМ [!KO\_tocov]

\*/\*!KO\_tocov(ingot%type%cov%method%name)\*/\*.

\*Version 2.

\*Преобразование многомерных количественных данных INGOT такое, чтобы наблюдаемые ковариации (или

\*корреляции или другие sscp-меры связи) между этими переменными стали точно равны заданным пользователем.

\*Аргумент INGOT - n наблюдений x p переменных (n>p) матрица данных со значеними, служащими "сырьем" для

\*получения вариат, связи между которыми станут удовлетворять матрице COV. INGOT могут быть случайные

\*порожденные данные, которые вы ходите привести к нужной коррелированности между ними, или INGOT могут

\*быть существующие анализируемые данные, которые вы хотите преобразовать минимально, заставив их

\*воспроизвести заданную коррелированность (см. METHOD ниже). Столбцы INGOT должны быть неколлинеарны.

\*Аргумент TYPE - ключевое слово заглавными буквами (опционально можно в кавычках или апострофах).

\*Обозначьте, какого типа целевую матрицу вы задаете как аргумент COV:

\*"COV" - ковариационная матрица (на "df=n-1");

\*"COV2" - ковариационная матрица (на "df=n") или mscp матрица;

\*"CORR" - корреляционная матрица или матрица косинус-близости.

\*(Все перечисленные матрицы родственны: они являются матрицами вида sscp(переменные)/"df".)

\*Аргумент COV - p x p квадратная симметричная положительно определенная матрица ассоциаций (ее тип

\*вы обозначили в TYPE). Это есть та целевая матрица, для которой надо получить из INGOT переменные,

\*удовлетворяющие этой матрице.

\*Результат NAME - данные n x p, столбцы которых это вариаты, точно воспроизводящие заданные

\*коэффициенты COV.

\*Пользуйтесь следующими правилами задания входящих:

\*-Если целевая матрица COV ковариационная (на "df=n-1"), то type="COV", столбцы INGOT должны быть

\* центрованы (центрировать переменные можно ф-цией /\*!KO\_center\*/).

\*-Если целевая матрица COV ковариационная (на "df=n"), то type="COV2", столбцы INGOT должны быть центрованы.

\*-Если целевая матрица COV это средние квадраты и кросспроизведения (mscp), то type="COV2", INGOT это

\* сырые данные как есть.

\*-Если целевая матрица COV корреляционная (единица должна быть на диагонали), то type="CORR", столбцы INGOT

\* должны быть центрованы. На выходе переменные сохранят свой начальный масштаб (дисперсию).

\*-Если целевая матрица COV это косинус-близости (единица должна быть на диагонали), то type="CORR", столбцы

\* INGOT это сырые данные как есть. На выходе переменные сохранят свой начальный масштаб (средний квадрат).

\*Аргумент METHOD - скаляр, определяющий алгоритм подгонки. Если он положительный, то при преобразовании

\*данных будет применены компонентные нагрузки целевой матрицы, прокрустово повернутые к нагрузкам

\*данных INGOT. Используйте этот вариант, если хотите, чтобы значения NAME минимально отличались, в смысле

\*суммарного квадрата разниц от значений INGOT: mssq(INGOT-NAME)=min. Другими словами, если стоит задача -

\*"подправить" INGOT, доведя наличные данные до нужного коррелирования.

\*Если аргумент METHOD неположительный, то при преобразовании данных будет применен корень Холецкого

\*из целевой матрицы. Используйте этот более быстрый вариант, если хотите просто создать данные, т.е. если

\*для вас INGOT - лишь "болванка" случайных значений, которые заранее порождены. Этот вариант не преследует

\*задачу минимального изменения INGOT.

\*Замечание: если INGOT это значения из нормального распределения, то NAME это данные тоже с

\*многомерно-нормальным распределением; в ином же случае распределение в NAME не обязано соотноситься

\*с распределением в INGOT.

ПРИМЕР. Случайные данные из нормального распределения с нужными наблюдаемыми корреляциями.

matrix.

compute mx= {1, 0.5, -0.2, 0.2;

0.5, 1, -0.1, 0.4;

-0.2, -0.1, 1, 0.3;

0.2, 0.4, 0.3, 1}. /\*Matrix of required correlations (4 variables)

print mx.

!KO\_normal(100%4%data). /\*Generate 100 cases x 4 variables of normal random data

/\*(it is the raw “ingot” variables)

!KO\_center(data%data). /\*Center exactly to zero means (because mx is correlation

/\*or covariance matrix)

!KO\_tocov(data%CORR%mx%0%vars). /\*Obtaining the random variables with correlations=mx;

/\*(they are centered), ready; but let us set them the following

compute mean= {2.0,3.6,2.1,-1.8}. /\*means

compute sd= {0.7,0.9,1.5,1.9}. /\*and st. deviations

!KO\_rescale(vars%"MEAN\_SD"%mean%sd%vars).

!KO\_corr(vars%r).

print r.

save vars /outfile= \*.

end matrix.

ПРИМЕР. Подправить данные, чтобы они имели нужные ковариации.

matrix.

compute mx= {1.8, 0.5, -0.2, 0.2;

0.5, 2.1, -0.1, 0.4;

-0.2, -0.1, 1, 0.3;

0.2, 0.4, 0.3, 0.6}. /\*Matrix of required covariances

print mx.

get data /variables= v1 v2 v3 v4. /\*Variables we want to “correct”

!KO\_cov(data%cov).

print cov. /\*Their observed covariances

!KO\_mean(data%mean). /\*Their means

!KO\_center(data%data). /\*Center exactly to zero means (because mx is correlation

/\*or covariance matrix)

!KO\_tocov(data%COV%mx%1%vars). /\*Transforming the variables towards covariances=mx;

/\*they are centered; and let us restore them

compute vars= vars+mean(make(1,nrow(vars),1),:). /\*their original means

!KO\_cov(vars%newcov).

print newcov.

save vars /outfile= \*.

end matrix.

### ПРЕОБРАЗОВАТЬ ПЕРЕМЕННУЮ ТОЧНО К ЗАДАННЫМ КОВАРИАЦИЯМ С ДРУГИМИ ПЕРЕМЕННЫМИ [!KO\_ytocov]

\*/\*!KO\_ytocov(yingot%xvars%type%cov%name)\*/\*.

\*Version 1.

\*Преобразует переменную YINGOT массива так, чтобы ее наблюдаемые ковариации (или корреляции или другие

\*sscp-меры связи) с переменными XVARS данных стали точно равны заданным пользователем. Результат NAME -

\*преобразованная YINGOT, измененная минимально в смысле наименьших квадратов: mssq(YINGOT-NAME)=min.

\*XVARS не изменятся.

\*Аргумент INGOT - столбец n наблюдений, это количественная переменная-"сырье" для получения переменной

\*NAME. Вы можете породить YINGOT как случайные значения или это могут быть сушествующие анализируемые

\*данные.

\*Аргумент XVARS - одна или более количественных переменных массива, n x p; эти p столбцов должны быть

\*неколлинеарны.

\*Аргумент TYPE - ключевое слово заглавными буквами (опционально можно в кавычках или апострофах).

\*Обозначьте, какого типа целевой вектор вы задаете как аргумент COV:

\*"COV" - ковариации (на "df=n-1");

\*"COV2" - ковариации (на "df=n") или средние кросспроизведения;

\*"CORR" - корреляции или косинус-близости.

\*(Все перечисленные виды родственны: они являются коэффициентами типа sscp(переменные)/"df".)

\*Аргумент COV - вектор-ряд собственно целевых коэффициентов (их тип вы обозначили в TYPE). Это

\*те коэффициенты связи с XVARS, ради которых будет преобразована YINGOT в NAME так, чтобы точно

\*удовлетворять им. Длина ряда COV равна p+1. Первые p значений порядково отвечают p столбцам

\*(переменным) XVARS, а последнее, добавочное значение это масштаб для переменной NAME (см. ниже).

\*Пользуйтесь следующими правилами задания входящих:

\*-Если ваши целевые коэффициенты COV это ковариации (на "df=n-1"), то type="COV", последним

\* элементом вектора COV должна быть желаемая дисперсия NAME (положительное значение), YINGOT и столбцы

\* XVARS должны быть центрованы (центрировать переменные можно ф-цией /\*!KO\_center\*/).

\*-Если ваши целевые коэффициенты COV это ковариации (смещенные, на "df=n"), то type="COV2", последним

\* элементом вектора COV должна быть желаемая смещенная дисперсия NAME (положительное значение), YINGOT и

\* столбцы XVARS должны быть центрованы.

\*-Если ваши целевые коэффициенты COV это средние кросспроизведения, то type="COV2", последним

\* элементом вектора COV должен быть желаемый средний квадрат NAME (положительное значение), YINGOT и

\* XVARS это сырые данные как есть.

\*-Если ваши целевые коэффициенты COV это корреляции, то type="CORR", последним элементом вектора COV должна

\* быть единица, YINGOT и столбцы XVARS должны быть центрованы. NAME сохранит начальный масштаб

\* (дисперсию) YINGOT.

\*-Если ваши целевые коэффициенты COV это косинус-близости, то type="CORR", последним элементом вектора COV

\* должна быть единица, YINGOT и XVARS это сырые данные как есть. NAME сохранит начальный масштаб

\* (средний квадрат) YINGOT.

\*Замечание1: Если заданные коэффициенты COV своей величиной не допускают создание корректной

\*переменной NAME из переменной YINGOT (по причине возникшей неграмовой, не положительно-определенной

\*конфигурации), то функция выдаст NAME= скаляр 0.

\*Замечание2: если YINGOT это значения из нормального распределения, то NAME это данные тоже с

\*нормальным распределением; в ином же случае распределение в NAME не обязано соотноситься

\*с распределением в YINGOT.

ПРИМЕР. Привести Y к нужному коррелированию с одним или несколькими X.

matrix.

get y /variable= y. /\*Variable to modify

get x /variables= x1 x2 x3.

!KO\_center(y%y). /\*Centering required

!KO\_center(x%x). /\*because coefficients are correlations (or covariances)

!KO\_ytocov(y%x%CORR%{.32,-.22,0,1}%y#). /\*Run the function

/\*three target correlations, plus 1 as the last element (corr. of Y

/\*with itself)

do if nrow(y#)=1. /\*If the result is 0 scalar

-print /title 'Sorry, The input coefficients are unrealistic.'.

else.

-!KO\_corr({y#,x}%coef).

-print coef(1,2:ncol(coef)).

-!KO\_variance(y#%vrnc).

-print vrnc.

-save {y#,x} /outfile= \* /variables= y# x1 x2 x3.

end if.

end matrix.

### НОРМАЛЬНОЕ ОБЛАКО В РАВНОМЕРНЫЙ ШАР [!KO\_unifball]

\*/\*!KO\_unifball(data%alpha%name1%name2)\*/\*.

\*Version 1.

\*Берет одно- или многомерные данные DATA, которые функция ожидает быть из

\*стандартного нормального распределения. Функция уменьшает эксцесс данных, поджимая

\*периферию (хвосты) сильнее, чем околоцентральные области. При параметре ALPHA=1

\*нормальное облако превращается в равномерное, в (гипер)шар.

\*DATA - данные (наблюдения x переменные) из нормального распределения со средней 0 и

\*дисперсией 1; если переменных более одной, функция ожидает, что они независимы, некоррелируют.

\*(Вы можете породить случайные стандартные ортогональные нормальные данные функцией /\*!KO\_normal\*/,

\*а стандартизовать переменные, если необходимо - функцией /\*!KO\_zscore\*/.)

\*ALPHA - скаляр, параметр степени поджатия периферии (хвостов) данных. Чем он выше, тем сильнее

\*функция уменьшит эксцесс в данных. При параметре 1 нормальное облако превратится в равномерное,

\*причем если данные многомерны, это будет равномерный (гипер)шар. При параметре 0 данные останутся

\*как были на входе. При параметре между 0 и 1 вы получите данные с эксцессом промежуточным между

\*нормальным и равномерным. Параметр выше 1 уже делает данные бимодальными (вогнутыми в центре).

\*Формула преобразования данных следующая:

\*Y\_i = X\_i \* coef,

\*coef = 1 + alpha \* (sqrt(p+2)\*cdf.chisq(d^2,p)^(1/p)/d - 1),

\*где Y\_i и X\_i - значения точки i данных на выходе (NAME1) и на входе (DATA);

\*коэффициент поджатия coef выдается как NAME2;

\*p - размерность, число столбцов DATA;

\*d - евклидово расстояние от точки i в DATA до начала координат, т.е. до 0 (согласно сказанному

\*выше, центроид DATA должен лежать в 0 или стремиться к 0);

\*cdf.chisq(val,df) - ф-ция кумулятивного распределения хи-квадрат;

\*фактор sqrt(p+2) сохраняет исходную ожидаемую изменчивость (т.е. 1 для каждого измерения);

\*а если ALPHA=1, то гипершар NAME1 ожидает радиус sqrt(p+2).

\*(См.: https://stats.stackexchange.com/questions/79919).

\*Коррелирующие переменные. Хотя функция уменьшит эксцесс в таком облаке и сделает эллипсоид

\*более шарообразным, она не добъется равномерного распределения под ALPHA=1. Функция выполняет

\*свою роль, только когда входящие переменные некоррелируют. Если вы хотите коррелирующие переменные

\*из нормального распределения превратить к эллипс с равномерным распределением внутри него,

\*сделайте сначала отбеливание данных (сферизацию плюс стандартизацию), например, посредством

\*анализа главных компонент (ф-ция /\*!KO\_pcomp\*/). Затем примените данную функцию, и в конце сделайте

\*преобразование, обратное тому анализу главных компонент - см. ПРИМЕР.

ПРИМЕР.

matrix.

!KO\_normal(3000%2%data). /\*Some data from normal distribution

/\*3000 points, 2 uncorrelated variables

!KO\_zscore(data%data). /\*It is expected that the data are from standard

/\*normal variates; if not, standardize;

/\*You may prefer to standardize it anyway

!KO\_unifball(data%1%ball%coef). /\*Turn the spherical normal cloud into the uniform

/\*ball (alpha=1) cloud

\*print (ball/(coef\*make(1,ncol(ball),1)). /\*(This is the back-transform)

save {data,ball} /outfile= \* /variables= norm1 norm2 unif1 unif2.

end matrix.

GRAPH /SCATTERPLOT(BIVAR)= norm1 WITH norm2 /title 'Initial normal cloud'.

GRAPH /SCATTERPLOT(BIVAR)= unif1 WITH unif2

/title 'Resultant uniform spherical cloud'.

DESCRIPTIVES VARIABLES= norm1 norm2 unif1 unif2.

/\*Note that mean and st dev almost preserved

ПРИМЕР.

matrix.

!KO\_normal(3000%1%data). /\*Some data from normal distribution

/\*3000 points, 1 variable

/\*!KO\_zscore(data%data).

!KO\_unifball(data%.5%trans%coef). /\*Turn the normal data into the one with kurtosis

/\*lesser than normal ("halfway" towards uniformity, alpha=0.5)

save {data,trans} /outfile= \* /variables= norm trans.

end matrix.

GRAPH /HISTOGRAM(NORMAL)= norm.

GRAPH /HISTOGRAM(NORMAL)= trans.

DESCRIPTIVES VARIABLES= norm trans.

ПРИМЕР. Коррелированные переменные.

matrix.

!KO\_mvnorm(3000%chol({1,.7,.5;.7,1.2,.2;.5,.2,.8})%0%data). /\*3 normal variates with

/\*the specified population covariances and mean 0, 3000 cases

!KO\_cov(data%cov). /\*Let's look at the

print cov. /\*observed covariances in the data

!KO\_pcomp(data%1%0%1%name1%load%name3%name4%pcscores). /\*Perform PCA in order to

/\*whiten the data (i.e., make the dimensions uncorrelated and with

/\*variances 1); Because parameter OUT=1, pcscores (pr. component scores) are

/\*already that standard

!KO\_unifball(pcscores%1%pcscores%coef). /\*So we can apply KO\_unifball to them;

/\*aplha=1, thus we've turned the normal cloud into the uniform ball

/\*(with name pcscores)

compute trans= pcscores\*t(load). /\*With the help of PC loadings, rotate that result

/\*back into the correlated state of original data

!KO\_cov(trans%cov\_trans). /\*Let's look at the covariances in

print cov\_trans. /\*the transformed data: reasonably close to what there was at input

save {data,trans} /outfile= \* /variables= norm1 norm2 norm3 unif1 unif2 unif3.

end matrix.

GRAPH /SCATTERPLOT(XYZ)= norm1 WITH norm2 WITH norm3 /title 'Initial normal cloud'.

GRAPH /SCATTERPLOT(XYZ)= unif1 WITH unif2 WITH unif3

/title 'Resultant uniform cloud'.

DESCRIPTIVES VARIABLES= norm1 norm2 norm3 unif1 unif2 unif3.

# СТАТИСТИЧЕСКИЕ АНАЛИТИЧЕСКИЕ ФУНКЦИИ

### ЛИНЕЙНАЯ РЕГРЕССИЯ [!KO\_regress]

\*/\*!KO\_regress(dv%ivs%const%name1%name2%name3)\*/\*.

\*Version 1.

\*Линейная регрессия наименьших квадратов (без тестов значимости).

\*Берет зависимую переменную DV и одну или более независимых переменных IVS (последние должны быть неколлинеарны).

\*Аргумент CONST (скаляр) - нужна модель с константой (CONST положителен) или без константы (CONST неположителен).

\*Результаты:

\*NAME1 - SSregression, SSresiduals, наблюдаемый коэффициент детерминации R-square.

\*NAME2 - двухстолбцовая матрица регрессионных коэффициентов: в 1-м столбце сырые коэффициенты; во 2-м столбце

\*стандартизованные коэффициенты (beta). Если модель была с константой, то число рядов в NAME2 на 1 больше

\*числа столбцов IVS, и константа находится в последнем ряду.

\*NAME3 - предсказанные значения (сырые) зависимой переменной.

ПРИМЕР.

matrix.

get y /variables= y.

get x /variables= x1 x2 x3 x4 /names= names.

!KO\_regress(y%x%1%summary%coef%pred).

print summary /clabels= 'SSregr' 'SSresid' 'Rsq'.

print coef /rnames= names /clabels= 'b' 'beta'.

print pred.

end matrix.

### ГЛАВНЫЕ КОМПОНЕНТЫ [!KO\_pcomp]

\*/\*!KO\_pcomp(data%df%m%out%name1%name2%name3%name4%name5)\*/\*.

\*Version 4.

\*Анализ главных компонент.

\*Берет данные DATA (где ряды - точки данных, столбцы - переменные, оси пространства) и выдает главные

\*компоненты этих данных.

\*Внимание: данные анализируются "как есть", поэтому если нужна предварительная центрация или

\*стандартизация, используйте прежде соответствующие функции. (Центрация соответствует анализу

\*ковариаций, а стандартизация - анализу корреляций.)

\*Аргумент DF (скаляр) - если положительный, разлагаться будет матрица sscp(DATA)/(n-1), а если

\*неположительный, то матрица sscp(DATA)/n, где n это число рядов в DATA.

\*Аргумент M (скаляр) - сколько первых главных компонент выделять: укажите целое положительное число

\*от 1 до число\_столбцов DATA. Или укажите любое неположительное число, тогда выделены будут все

\*ненулевые компоненты.

\*Аргумент OUT (скаляр) управляет формой выдачи результатов.

\*Результаты:

\*NAME1 содержит в 1-м столбце собственные числа разложенной матрицы,

\*и во 2-м столбце - доли объясненной дисперсии (или масштаба), т.е. доли от следа

\*разложенной матрицы.

\*NAME2 - собственные векторы (если OUT неположителен) либо нагрузки (если OUT положителен);

\*сумма квадратов в столбцах нагрузок = собственные числа. Нагрузки это коэффициенты предсказания

\*переменных масштабированными компонентами. Собственные векторы это коэффициенты предсказания

\*переменных сырыми компонентами, или косинусы углов поворота.

\*NAME3 - перешкалированные нагрузки. Перешкалированная или стандартизованная нагрузка это нагрузка,

\*деленная на масштаб переменной - на корень из соответствующего диагонального элемента разложенной

\*матрицы. Этим делением снят эффект неодинаковости масштабов (дисперсий) переменных на нагрузки.

\*Перешкалированная нагрузка это косинус-сходство или корреляция между компонентой и переменной.

\*Квадраты элементов собственных векторов можно понимать как величину вкладов переменных в компоненты.

\*Квадраты элементов перешкалированных нагрузок можно понимать как величину вкладов компонент

\*в переменные.

\*NAME4 - регрессионные коэффициенты вычисления главнокомпонентных баллов по переменным.

\*Если OUT неположителен, эта матрица тождественна собственным векторам NAME2. Если OUT положителен,

\*эта матрица вычисляется из нагрузок NAME2. Матрицу NAME4 вы можете использовать для подсчета

\*главнокомпонентных баллов данных, не участвовавших в анализе.

\*NAME5 - главнокомпонентные баллы. Они есть DATA\*NAME4. Если OUT неположителен, это сырые баллы:

\*сумма квадратов в столбцах NAME5 = собственные числа матрицы sscp(DATA)). Если OUT положителен, это

\*масштабированные баллы: сумма квадратов в столбцах NAME5 = n-1 либо n, в зависимости от аргумента DF.

ПРИМЕР. Анализ главных компонент на основе корреляций.

matrix.

get vars /variables= v1 to v9.

!KO\_zscore(vars%vars). /\*Standardize the variables, so that their analysis will

/\*be the analysis of correlations

!KO\_pcomp(vars%1%3%1%eival%loads%rloads%b%scores). /\*Run the analysis

/\*with extraction of 3 first components;

/\*The requested mode of output:

print eival. /\*Eigenvalues (component variances) and % of variance explained

print loads. /\*Loadings

print rloads. /\*Rescaled loadings (they are = loadings if you analyze correlations)

print scores. /\*Scaled (standardized) component scores

print b. /\*Matrix of coefficients to compute the scores

end matrix.

### ГЛАВНЫЕ КОМПОНЕНТЫ (ДЛЯ N<P) [!KO\_pcomp2]

\*/\*!KO\_pcomp2(data%df%m%out%name1%name2%name3%name4%name5)\*/\*.

\*Version 2.

\*Анализ главных компонент.

\*Эта функция тождественна ф-ции /\*!KO\_pcomp\*/, но быстрее ее, когда число столбцов в данных больше

\*чем число рядов. И медленнее ее, когда число столбцов в данных меньше числа рядов.

### ГЛАВНЫЕ КОМПОНЕНТЫ (ВХОДЯЩИЕ - МАТРИЦА) [!KO\_pca]

\*/\*!KO\_pca(cov%w%m%out%name1%name2%name3%name4)\*/\*.

\*Version 2.

\*Анализ главных компонент (с опцией взвешивания переменных).

\*Эта функция, в отличие от /\*!KO\_pcomp\*/, берет в качестве входящих не данные, а уже готовую матрицу

\*(ковариаций и т.п.), и она сама не вычисляет главнокомпонентных баллов.

\*Есть опция придать неодинаковые веса (важности) переменным, т.е. исполнить взвешенный PCA.

\*Входящие:

\*COV - квадратная симметричная матрица коэффициентов sscp-типа - т.е. корреляционная, ковариационная,

\*косинусная или сырая sscp-матрица коэффициентов связи между переменными.

\*W - вектор длиной как число рядов/столбцов COV, содержащий веса переменным; либо неположительный

\*скаляр (что значит - не давать веса, результаты будут тождественны с /\*!KO\_pcomp\*/). Веса должны

\*быть положительными числами, находящимися в нужном вам соотношении (только соотношение величин весов

\*играет роль). Вес ровно 0 не разрешен, но вы можете придать очень близкий к нулю положительный вес,

\*если хотите сделать некую переменную "пассивной", supplementary.

\*Аргументы M и OUT - те же, что в функции /\*!KO\_pcomp\*/.

\*Результаты:

\*NAME1, NAME2, NAME3, NAME4 - аналогичны тем, что в функции /\*!KO\_pcomp\*/.

\*Матрицу коэффициентов компонентных баллов NAME4 вы можете использовать для вычисления

\*главнокомпонентных баллов, каковые будут равны DATA\*NAME4, где DATA - данные наблюдения x переменные

\*(если COV это ковариационная матрица, переменные должны быть центрованы, если COV это корреляционная

\*матрица, переменные должны быть z-стандартизованы, и т.п., т.е. должно быть соответствие с матрицей).

\*Ввод неграмовой COV:

\*Если среди M выделяемых компонент случится таковая с отрицательным собственным числом в NAME1,

\*т.е. мнимая, то соответствующие ей результаты в NAME2, NAME3, NAME4 будут посчитаны, словно эта

\*компонента реальная (с положительным собственным числом), но эти ее результаты следует

\*считать мнимыми.

ПРИМЕР. Взвешенный анализ главных компонент.

matrix.

get vars /variables= v1 to v9.

!KO\_cov(vars%cov).

compute w= {1,1.2,1,1.8,2,1,2.3,1,1}. /\*Weights (importances) of the variables

!KO\_pca(cov%w%3%1%eival%loads%rloads%b). /\*Run the analysis with

/\*extraction of 3 first components;

/\*The requested mode of output:

print eival. /\*Eigenvalues (component variances) and % of variance explained

print loads. /\*Loadings

print rloads. /\*Rescaled loadings (they are = loadings if you analyze correlations)

print b. /\*Matrix of coefficients to compute the scores

!KO\_center(vars%vars).

print (vars\*b). /\*Component scores

end matrix.

### ОРТОГОНАЛЬНОЕ ПРОКРУСТОВО ВРАЩЕНИЕ [!KO\_procr]

\*/\*!KO\_procr(fit%targ%refl%iso%name1%name2)\*/\*.

\*Version 1.

\*Осуществляет ортогональный прокрустов поворот подгоночной конфигурации FIT в целевую

\*конфигурацию TARG. Подогнанная (повернутая) подгоночная конфигурация выдается как NAME2,

\*а матрица поворота как NAME1.

\*FIT и TARG - матрицы одинакового размера. Ряды в них - это точки данных, а столбцы - измерения (оси)

\*пространства. Один и тот же ряд в двух матрицах есть пара взаимосоответственных точек.

\*Измерения пространства у FIT и TARG у каждой свои. Прокрустово вращение это такой поворот осей FIT

\*в оси TARG, который обеспечивает максимальную совмещенность взаимосоответственных точек.

\*NAME2 это координаты точек FIT в пространстве осей TARG после поворота.

\*Если число измерений в FIT и TARG неодинаково, добавьте к одной из этих матриц заполненные нулем

\*столбцы, т.к. функция требует, чтобы число столбцов было там и там одинаково.

\*В матрице поворота NAME1 ряды отвечают осям FIT, а столбцы отвечают осям TARG.

\*REFL - скаляр; если положительный, повороту разрешено быть с отражением (если это улучшит подгонку);

\*если неположительный, отражение запрещено (подгонка может оказаться хуже).

\*ISO - скаляр; если положительный, после поворота осуществляется дополнительная операция

\*изомасштабирования для лучшей подгонки; если неположительный, изомасштабирования не делается. Этот

\*аргумент не влияет на матрицу поворота NAME1.

\*Функция берет данные FIT и TARGET "как есть". Если требуется совмещение центроидов и/или уравнивание

\*масштабов двух этих конфигураций до операции прокрустова поворота, сделайте эти преобразования заранее.

### ЛИНЕЙНЫЕ ДИСКРИМИНАНТЫ [!KO\_discrim]

\*/\*!KO\_discrim(data%dummy%m%out%name1%name2%name3%name4%name5)\*/\*.

\*Version 2.

\*Осуществляет линейный дискриминантный анализ (без классификации и тестов значимости):

\*выдает дискриминанты.

\*Берет данные DATA (ряды - точки, столбцы - оси пространства) и группирование DUMMY.

\*Столбцы DATA (переменные) должны быть уже центрованы (или стандартизованы). Столбцы DATA должны быть

\*не коллинеарны.

\*DUMMY - двоичные фиктивные переменные, помечающие группы (каждая переменная соответствует группе,

\*значение 1 = наблюдение принадлежит ей, 0 = не принадлежит ей).

\*Всякий столбец DUMMY должен содержать по меньшей мере одну единицу.

\*Группы должны быть непересекающиеся составом, и каждое наблюдение должно принадлежать группе,

\*т.е. сумма в каждом ряду DUMMY должна равняться единице.

\*Аргумент M (скаляр) - сколько дискриминант выделять. Укажите целое положительное число от 1 до

\*min(число\_групп-1,число\_переменных). Либо укажите любое неположительное число: тогда выделятся все

\*ненулевые дискриминанты.

\*Аргумент OUT (скаляр) управляет формой выдачи результатов.

\*Результаты:

\*NAME1 - двухстолбцовая матрица: собственные числа в 1-м столбце, канонические корреляции во 2-м.

\*NAME2 - собственные векторы с нормированными столбцами (если OUT неположителен) либо сырые

\*(нестандартизованные) канонические дискриминантные коэффициенты (если OUT положителен).

\*NAME3 - стандартизованные канонические дискриминантные коэффициенты (подходят для интерпретации

\*дискриминант).

\*NAME4 - объединенные внутригрупповые (pooled within-group) корреляции между переменными и

\*дискриминантами (матрица структуры, подходит для интерпретации дискриминант).

\*NAME5 - дискриминантные баллы, они есть DATA\*NAME2, т.е посчитаны с помощью нормированных собственных

\*векторов (если OUT неположителен) либо с помощью сырых дискриминантных коэффициентов

\*(если OUT положителен); в последнем случае баллы имеют то свойство, что их объединенная

\*внутригрупповая ковариационная матрица (pooled within-group covariance matrix) есть единичная.

\*Если к матрице нестандартизованных коэфициентов необходим вектор констант (для получения нецентрованных

\*дискриминант), то он равен -csum(mdiag(MEAN)\*NAME2), где MEAN это вектор средних исходных переменных

\*и NAME2 это нестандартизованные канонические дискриминантные коэффициенты.

\*Фиктивные переменные DUMMY вы можете создать из категориальной группирующей переменной

\*с помощью функции /\*!KO\_freq\*/.

\*Для классификации используйте например ф-цию /\*!KO\_gaclass\*/.

ПРИМЕР.

matrix.

get vars /variables= v1 to v9.

get group /variable= group.

!KO\_center(vars%vars). /\*You normally SHOULD center variables before

!KO\_freq(group%1%dummy%freq%codes). /\*Grouping variable into dummies

!KO\_discrim(vars%dummy%0%1%cancorr%coef%scoef%corr%scores).

print cancorr. /\*Eigenvalues and canonical correlations

print scoef. /\*Standardized coefficients

print corr. /\*Correlations (structure matrix)

save scores /outfile= \*.

end matrix.

ПРИМЕР. Дискриминантный анализ с последующим варимакс-вращением дискриминант (то же самое вы можете сделать SPSS-командой DISCRIMINANT через ее синтаксис).

set mxloops 1E6.

matrix.

get vars /variables= v1 v2 v3 v4 v5.

get group /variable= cluster\_.

\*Extract discriminants.

!KO\_center(vars%vars).

!KO\_freq(group%1%dummy%freq%codes).

!KO\_discrim(vars%dummy%0%1%cancorr%coef%scoef%corr%scores).

print scoef /title 'Standardized coefficients' /format= f8.5.

print corr /title 'Correlations (structure matrix)' /format= f8.5.

\*----.

\*Perform varimax rotation of the structure matrix.

print /title '--- Varimax-rotated will be Structure matrix ---'.

!KO\_ortrot(corr%1%VARIMAX%{.00001,50}%0%rotcorr%rotmat1%name3).

print rotmat1 /title 'Varimax rotation (transformation) matrix' /format= f8.5.

print rotcorr /title 'Rotated structure matrix' /format= f8.5.

print (scoef\*rotmat1) /title 'Corresponding st. coefficients matrix' /format= f8.5.

print (scores\*rotmat1)(1:10,:) /title 'Corresponding discr. scores (first 10 shown)'

/format= f8.5.

\*----.

\*Another approach is to perform varimax rotation of the coefficients matrix.

print /title '--- Varimax-rotated will be St. coefficients matrix ---'.

!KO\_gcov(vars%dummy%CORR%POOL%pooledr). /\*See Note below; Get the pooled

/\*within-group corr matrix of variables

!KO\_image(pooledr%1%image). /\*Diagonal of its image matrix contains squared

/\*multiple correlation coefficients (to use in the custom

/\*Kaiser normalization)

!KO\_ortrot(scoef%diag(image)%VARIMAX%{.00001,50}%0%rotwei%rotmat2%name3).

print rotmat2 /title 'Varimax rotation (transformation) matrix' /format= f8.5.

print rotwei /title 'Rotated st. coefficients matrix' /format= f8.5.

print (corr\*rotmat2) /title 'Corresponding structure matrix' /format= f8.5.

print (scores\*rotmat2)(1:10,:) /title 'Corresponding discr. scores (first 10 shown)'

/format= f8.5.

end matrix.

Note. We follow SPSS Algorithms (DISCRIMINANT) expressing that, when varimax rotating the st. discriminant coefficients, Kaiser normalization is done by multiple correlation coefficients; therefore we first have to get to know these.

### ГАУССОВ КЛАССИФИКАТОР [!KO\_gaclass]

\*/\*!KO\_gaclass(data%grmean%grcov%prior%name1%name2%name3)\*/\*.

\*Version 2.

\*Берет данные DATA (ряды - точки, столбцы - оси пространства) и характеристики k (k>=2)

\*классов (групп) в виде координат их центроидов GRMEAN и ковариационных матриц GRCOV.

\*На основе квадратных махаланобисовых расстояний (z^2) точек данных до центроидов групп

\*вычисляет вероятности Prob принадлеждости точек к группам и приписывает каждую точку к группе.

\*Гауссов классификатор - один из примеров байесовского классификатора мерными переменными,

\*и основан на гауссовой функции Prob ~ exp(-z^2/2), которой описывается, упомянем это, и

\*колоколообразная кривая нормального распределения, - поэтому гауссов классификатор оптимален

\*для зачисления наблюдений в нормально распределенные или по кр. мере колоколообразные популяции.

\*В частности, данный классификатор часто применяется в дискриминантном анализе.

\*(В дискриминантном анализе классификация наблюдений делается по данным, которые есть

\*дискриминантные баллы.)

\*Входящие:

\*DATA - n точек (наблюдений) x p переменных (измерений). Это точки, которые надо классифицировать.

\*GRMEAN - k классов x p переменных. Это средние (центроиды) k классов (групп).

\*GRCOV - p x p ковариационная матрица или матрицы классов. Ковариационная матрица/ы должна быть

\*положительно определенной. Есть три варианта задать GRCOV:

\*(i) k подшитых одна под другой ковариационных матриц, - т.е. задать свою ков. матрицу для каждого

\*класса; GRCOV будет иметь размер k\*p x p.

\*(ii) одна p x p ковариационная матрица на все классы, обычно это усредненная ков. матрица групп

\*(pooled within group covariance matrix).

\*(iii) указать скаляр 0 вместо матрицы, - тогда p x p единичная матрица будет

\*принята для каждой группы (в этой ситуации махаланобисовы расстояния превратятся в евклидовы).

\*GRMEAN и GRCOV могут как проистекать из выборки DATA, так и не проистекать. Если GRMEAN и GRCOV

\*вычислены на массиве наблюдений, составляющих ряды DATA, то значит задача: классифицировать

\*анализируемые наблюдения к наблюдаемым классам. (Ковариационные матрицы групп и их pooled

\*матрицы можно получить из данных функцией /\*!KO\_gcov\*/.) Если GRMEAN и GRCOV не выведены из

\*массива n наблюдений DATA, то значит задача: классифицировать "новые" (или "удержанные") наблюдения

\*к "предложенным" классам.

\*PRIOR - вектор-столбец длиной k; это априорные вероятности приписания к классам (они служат

\*как веса при вычислении апостериорных вероятностей Prob). Задайте неотрицательные значения, их

\*сумма должна быть положительной. Величина самих чисел не играет роли, т.к. функция на входе

\*нормирует вектор к сумме его элементов =1.

\*Результаты:

\*NAME1 - квадратные махаланобисовы расстояния между точками DATA и центроидами классов, n x k.

\*NAME2 - апостериорные вероятности Prob приписания точек к классам, n x k (сумма в каждом ряду =1).

\*NAME3 - результат приписания наблюдений, n x 2. 1-й столбец показывает номер класса, к которому

\*классификатор приписал наблюдение - это класс, для которого Prob максимально в данном ряду NAME2.

\*2-й столбец показывает номер класса, для которого Prob в данном ряду NAME2 - 2-е по

\*величине значение.

\*Ошибки ковариационной матрицы. Если GRCOV содержит ковариационную матрицу не положительно

\*определенную, функция не сработает правильно. Либо SPSS сообщит об ошибке со стороны "INV"

\*или со стороны "SQRT", либо некоторые кв. махаланобисовы расстояния окажутся на выходе отрицательны.

\*Мы рекомендуем перед пуском функции проверять ков. матрицы: eval(m), где m - ков. матрица.

\*Все собственные числа должны быть положительны.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

get data /variables= v1 v2 v3. /\*Scale variables which cases to classify

\*Two classes are suggested.

compute grmean= {-5.2, .6, 3.1;

2.4, -1.2, 3.3}. /\*The means of the classes

compute grcov= {2.1, 1.3, .4;

1.3, 1.8, -1.1;

.4, -1.1, 2.0;

2.6, 1.8, .1;

1.8, 2.3, -1.3;

.1, -1.3, 1.9}. /\*The covariances in the classes (2 stacked matrices)

compute prior= {.45;.55}. /\*Apriori probabilities (saliences) of the classes

print grmean /rlabels= 'class1' 'class2' /clabels= 'v1' 'v2' 'v3'.

print grcov /rlabels= 'v1' 'v2' 'v3' 'v1' 'v2' 'v3' /clabels= 'v1' 'v2' 'v3'.

print prior /rlabels= 'class1' 'class2'.

\*Let us make sure the cov. matrices are positive definite.

print eval(grcov(1:3,:)).

print eval(grcov(4:6,:)).

\*Classify data cases to the classes.

!KO\_gaclass(data%grmean%grcov%prior%smah%post%class).

print smah /title 'Sq. Mahalanobis distances' /clabels= 'class1' 'class2'.

print post /title 'Posterior probabilities' /clabels= 'class1' 'class2'.

print class /title 'Class of prediction' /clabels= 'TheClass' '2ndBest'.

end matrix.

ПРИМЕР. Дискриминантный анализ с классификацией.

matrix.

get vars /variables= v1 v2 v3 v4. /\*Analysis variables

get group /variable= group. /\*Grouping variable (say, k=3 classes)

\*-----.

\*Let us hold out two first cases from the analysis.

compute hold= vars(1:2,:). /\*These two cases will be treated as "new cases"

/\*to classify

compute vars= vars(3:nrow(vars),:). /\*So cut them out from the data

compute group= group(3:nrow(group)). /\*and from here

\*-----.

\*Do with the analysis data.

!KO\_freq(group%1%dummy%freq%codes). /\*Grouping variable into dummies

compute k= ncol(dummy).

print {codes,freq} /title 'Groups in the analysis' /clabels 'Code' 'Count'.

!KO\_mean(vars%mean). /\*Means of the variables

compute vars= vars-make(nrow(vars),1,1)\*mean. /\*Normally SHOULD center variables

/\*before applying /\*!KO\_discrim\*/

!KO\_discrim(vars%dummy%0%1%cancorr%coef%scoef%corr%dis). /\*Discriminant analysis

/\*extracting min(3-1, 4)=2 discriminant functions; we will need

/\*their scores "dis"; because in discriminant analysis, the discr

/\*functions are used as classifiers

\*Obtain the discriminant scores also for the hold-out cases.

compute hold= hold-make(nrow(hold),1,1)\*mean. /\*Center first by

/\*the mean of the analysis variables

compute holddis= hold\*coef. /\*And compute them discriminant scores

\*Compute group means of discriminant scores for the analysis cases.

!KO\_aggr(dis%dummy%MEAN%grmean).

print grmean /title 'Means of the 2 discriminants in the 3 groups.'.

\*We need a covariance matrix or matrices of discriminant scores. This:.

!KO\_gcov(dis%dummy%COV%BOTH%grcov). /\*outputs, stacked, the pooled

/\*within-group covariance matrix and the group covariance matrices

print grcov /title 'Pooled and separate covariance matrices of discriminants'.

/\*Note that the pooled within-group covariance matrix of discriminant

/\*scores is the identity matrix

\*-----.

\*Perform classification of both the analysis and the hold-out cases.

!KO\_gaclass({holddis;dis}%grmean%0%make(k,1,1)%smah%post%class).

/\*In this run, we use the pooled within-group cov matrix

/\*and equal prior probabilities [this setting is default in DISCRIMINANT]

/\*Since the pooled matrix for discriminants is always the identity one,

/\*we could just set GRCOV argument to scalar 0

print /title 'POOLED MATRIX AND EQUAL PRIORS USED'.

print /title 'Cases 1 and 2 were hold-out: no influence, just classified.'.

print smah /title 'Sq. Mahalanobis (= sq. Euclidean) distances'

/clabels= 'class1' 'class2' 'class3' /format= f8.3.

print post /title 'Posterior probabilities'

/clabels= 'class1' 'class2' 'class3' /format= f8.5.

print class /title 'Group of assignment' /clabels= 'TheClass' '2ndBest'.

\*-----.

!KO\_gaclass({holddis;dis}%grmean%grcov(3:nrow(grcov),:)%freq%smah%post%class).

/\*In this run, we use individual within-group cov matrices,

/\*and prior probabilities proportional to the group sizes

print /title 'SEPARATE MATRIX AND UNEQUAL PRIORS USED'.

print /title 'Cases 1 and 2 were hold-out: no influence, just classified.'.

print smah /title 'Sq. Mahalanobis distances'

/clabels= 'class1' 'class2' 'class3' /format= f8.3.

print post /title 'Posterior probabilities'

/clabels= 'class1' 'class2' 'class3' /format= f8.5.

print class /title 'Group of assignment' /clabels= 'TheClass' '2ndBest'.

end matrix.

The 1st run of /\*!KO\_gaclass\*/ classification was equivalent to this run of DISCRIMINANT:

DISCRIMINANT

/GROUPS= group (1 3)

/VARIABLES= v1 v2 v3 v4

/SELECT= selvar (1) /\*Selection variable with all cases=1 but for cases 1 & 2

/ANALYSIS ALL

/SAVE= CLASS SCORES PROBS

/PRIORS EQUAL /\*Equal priors

/PLOT= CASES

/CLASSIFY= NONMISSING POOL /\*Pooled cov. matrix to use (identity one).

The 2nd run of /\*!KO\_gaclass\*/ classification was equivalent to this run of DISCRIMINANT:

DISCRIMINANT

/GROUPS= group (1 3)

/VARIABLES= v1 v2 v3 v4

/SELECT= selvar (1) /\*Selection variable with all cases=1 but for cases 1 & 2

/ANALYSIS ALL

/SAVE= CLASS SCORES PROBS

/PRIORS SIZE /\*Priors proportional to group sizes

/PLOT= CASES

/CLASSIFY= NONMISSING SEPAR /\*Separate group matrices to use.

### ГАУССОВ КЛАССИФИКАТОР (С ОПЦИЕЙ "СКОЛЬЗЯЩИЙ КОНТРОЛЬ-1") [!KO\_gaclass2]

\*/\*!KO\_gaclass2(data%dummy%method%prior%name1%name2%name3)\*/\*.

\*Version 1.

\*Берет данные DATA (ряды - точки, столбцы - оси пространства) и групповую принадлежность

\*этих наблюдений, DUMMY. На основе квадратных махаланобисовых расстояний точек данных до

\*центроидов групп вычисляет вероятности Prob принадлеждости точек к группам и приписывает каждую

\*точку к группе.

\*Данная функция отличается от /\*!KO\_gaclass\*/ - тоже гауссов классификатор - следующим:

\*- средние групп (координаты центроидов) и ковариации между переменными не вводятся, функция

\* берет их из данных;

\*- нет опции использовать для каждой группы отдельную ковариационную матрицу, используется лишь

\* объединенная (pooled) внутригрупповая ковариационная матрица;

\*- все наблюдения должны иметь групповую принадлежать;

\*- есть опция leave-one-out классификации (кросс-валидации).

\*Ф-ция /\*!KO\_gaclass2\*/ является слегка переписанным старым макросом "disclass", некогда

\*прилагавшимся к SPSS, и дает те же результаты.

\*Входящие:.

\*DATA - n точек (наблюдений) x p переменных (измерений). Это точки, которые надо классифицировать.

\*DUMMY - n x k: их наблюдаемая принадлежность к k классам (группам), двоичные фиктивные переменные

\*(каждая переменная соответствует группе, значение 1 = наблюдение принадлежит ей,

\*0 = не принадлежит ей). Всякий столбец DUMMY должен содержать по меньшей мере одну единицу.

\*Группы должны быть непересекающиеся составом: сумма в рядах DUMMY не может превосходить единицу.

\*В DUMMY не может быть рядов, полных нулей - т.е. наблюдений вне групп. (Используйте /\*!KO\_gaclass\*/

\*для классификации наблюдений "новых", групповая принадлежность которых неизвестна.)

\*Аргумент METHOD - скаляр. Если неположительный, то функция исполнит обычную классификацию -

\*классификацию без кросс-валидации. Если же положительный, то функция исполнит кросс-валидацию

\*"скользящий контроль-1" (синонимы: leave-one-out classification, U-method cross-validation,

\*jackknifing cross-validation). При этом методе каждое наблюдение, перед тем как оно

\*классифицируется, временно исключается из выборки и рассматривается как проверочный массив

\*размером 1, а остальные n-1 наблюдений образуют обучающий массив, из которого выводится

\*правило классификации для того наблюдения. Каждое наблюдение, таким образом, не влияет на

\*собственную классификацию.

\*PRIOR - вектор-столбец длиной k; это априорные вероятности приписания к классам (они служат

\*как веса при вычислении апостериорных вероятностей Prob). Задайте неотрицательные значения, их

\*сумма должна быть положительной. Величина самих чисел не играет роли, т.к. функция на входе

\*нормирует вектор к сумме его элементов =1.

\*Результаты:

\*NAME1 - квадратные махаланобисовы расстояния между точками DATA и центроидами классов, n x k.

\*NAME2 - апостериорные вероятности Prob приписания точек к классам, n x k (сумма в каждом ряду =1).

\*NAME3 - результат приписания наблюдений, n x 2. 1-й столбец показывает номер класса, к которому

\*классификатор приписал наблюдение - это класс, для которого Prob максимально в данном ряду NAME2.

\*2-й столбец показывает номер класса, для которого Prob в данном ряду NAME2 - 2-е по

\*величине значение.

\*Ошибка ковариационной матрицы. Если объединенная внутригрупповая ковариационная (или рассеяния)

\*матрица (pooled within-group covariance/scatter matrix) окажется вырождена, SPSS сообщит об ошибке

\*со стороны "INV". Попробуйте в этой ситуации слегка изменить входящие значения VARS, например,

\*добавить крохотный случайный шум. (Получить для проверки упомянутую объединенную матрицу

\*можно функцией /\*!KO\_gcov\*/ или /\*!KO\_bwscat\*/.)

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Фиктивные переменные DUMMY вы можете создать из категориальной группирующей переменной

\*с помощью функции /\*!KO\_freq\*/.

ПРИМЕР. Обычная классификация.

set mxloops 1E6.

matrix.

get vars /variables= v1 v2 v3 v4.

get gr /variable= class.

!KO\_freq(gr%1%dummy%freq%codes). /\*Grouping variable into dummies

print {t(1:ncol(dummy)),codes,freq} /title 'Classes (groups) observed'

/clabels 'GrNumber' 'Code' 'Count'.

compute prior= make(ncol(dummy),1,1). /\*Let priors for the classes be equal

\*Perform usual classification.

!KO\_gaclass2(vars%dummy%0%prior%smah%post%class).

print smah /title 'Sq. Mahalanobis distances, cases x classes' /format= f8.5.

print post /title 'Posterior probabilities, cases x classes' /format= f8.5.

print class /title 'Group of assignment' /clabels= 'TheClass' '2ndBest'.

\*Compare the input classification with the predicted.

compute grnum= dummy\*t(1:ncol(dummy)). /\*Dummies to categorical variable

!KO\_crosstab(grnum%class(:,1)%0%count%nums1%nums2).

print count /title 'Observed (rows) vs Predicted (cols) classification'.

print /title '(Classes go by their number "GrNumber" order)' /space= 0.

\*-----.

\*Get the same results with /\*!KO\_gaclass\*/.

!KO\_aggr(vars%dummy%MEAN%grmean). /\*Observed group means

!KO\_gcov(vars%dummy%COV%POOL%poolcov). /\*Pooled within-group cov matrix

!KO\_gaclass(vars%grmean%poolcov%prior%smah%post%class).

print smah /title 'Sq. Mahalanobis distances, cases x classes' /format= f8.5.

print post /title 'Posterior probabilities, cases x classes' /format= f8.5.

print class /title 'Group of assignment' /clabels= 'TheClass' '2ndBest'.

end matrix.

ПРИМЕР. Классификация скользящий контроль-1.

set mxloops 1E6.

matrix.

get vars /variables= v1 v2 v3 v4.

get gr /variable= class.

!KO\_freq(gr%1%dummy%freq%codes). /\*Grouping variable into dummies

print {t(1:ncol(dummy)),codes,freq} /title 'Classes (groups) observed'

/clabels 'GrNumber' 'Code' 'Count'.

compute prior= make(ncol(dummy),1,1). /\*Let priors for the classes be equal

!KO\_gaclass2(vars%dummy%1%prior%smah%post%class).

print smah /title 'Sq. Mahalanobis distances, cases x classes' /format= f8.5.

print post /title 'Posterior probabilities, cases x classes' /format= f8.5.

print class /title 'Group of assignment' /clabels= 'TheClass' '2ndBest'.

\*Compare the input classification with the predicted.

compute grnum= dummy\*t(1:ncol(dummy)). /\*Dummies to categorical variable

!KO\_crosstab(grnum%class(:,1)%0%count%nums1%nums2).

print count /title 'Observed (rows) vs Predicted (cols) classification'.

print /title '(Classes go by their number "GrNumber" order)' /space= 0.

print /title 'The Predicted classification was Leave-one-out' /space= 0.

end matrix.

### КАНОНИЧЕСКИЕ КОРРЕЛЯЦИИ [!KO\_cancorr]

\*/\*!KO\_cancorr(set1%set2%m%out%name1%name2%name3%name4%name5%name6)\*/\*.

\*Version 2.

\*Осуществляет канонический корреляционный анализ (без тестов значимости): выдает канонические

\*вариаты.

\*Берет два неодинаковых набора переменных - SET1 (n x p1) и SET2 (n x p2), с одними и теми же

\*наблюдениями (рядами). Переменные (столбцы) должны быть внутри наборов неколлинеарны.

\*Переменные в норме должны быть уже центрованы (или стандартизованы), чтобы получился классический

\*канонический корреляционный анализ.

\*Аргумент M (скаляр) - сколько канонических корреляций (т.е. пар канонических вариат) выделять.

\*Укажите целое положительное число от 1 до min(p1,p2). Либо укажите любое неположительное число: тогда

\*выделятся все ненулевые канонические корреляции: m будет равно их числу.

\*Аргумент OUT (скаляр) управляет формой выдачи результатов.

\*Результаты:

\*NAME1 - двустолбцовая матрица: собственные числа в 1-м столбце, канонические корреляции во 2-м.

\*NAME2 - сырые (нестандартизованные) канонические коэффициенты (если OUT положителен) либо они же с

\*нормированными столбцами (если OUT неположителен).

\*NAME3 - стандартизованные канонические коэффициенты (подходят для интерпретации вариат).

\*NAME4 - нагрузки, или матрица структуры - это корреляции между переменными и вариатами внутри

\*наборов (подходят для интерпретации вариат).

\*NAME5 - кросс-нагрузки - это корреляции между переменными и вариатами между наборами.

\*Размеры NAME2, NAME3, NAME4, NAME5 одинаковы, p1+p2 x m, т.е. переменные двух наборов x канонические

\*пары, т.е. канонические корреляции.

\*NAME6 - баллы канонических вариат (n x 2\*m), они есть SET1\*NAME2(первые p1 рядов) - первые их m

\*столбцов, и SET2\*NAME2(последние p2 рядов) - последние их m столбцов. Если OUT положителен, получатся

\*стандартизованные баллы (ст. откл. в каждой вариате =1); если OUT неположителен, получатся

\*сырые баллы (вообще, существуют разные версии вычисленных сырых канонических баллов). В любом случае

\*внутри каждого набора вариаты всегда некоррелируют своими баллами, а между наборами корреляции

\*ненулевые есть только в парах соответствующих вариат - и они равны каноническим корреляциям NAME1.

ПРИМЕР.

matrix.

get set1 /variables= u1 to u5 /names= names1.

get set2 /variables= w1 w2 w3 /names= names2.

compute names= {names1,names2}.

!KO\_center(set1%set1). /\*Normally should center

!KO\_center(set2%set2). /\*all the variables

!KO\_cancorr(set1%set2%0%1%cancorr%coef%scoef%load%crload%scores).

print cancorr /title 'Eigenvalues and canonical correlations'

/clabels 'Eigen' 'CanCorr' /rlabels 'I' 'II' 'III' /format= f8.4.

print coef /title 'Canonical coefficients' /clabels 'I' 'II' 'III' /rnames= names.

print scoef /title 'Standardized canonical coefficients'

/clabels 'I' 'II' 'III' /rnames= names.

print load /title 'Canonical loadings' /clabels 'I' 'II' 'III' /rnames= names.

print crload /title 'Canonical cross-loadings'

/clabels 'I' 'II' 'III' /rnames= names.

print scores(1:10,:) /title 'Canonical scores (first 10 shown)'

/clabels 'I,set1' 'II,set1' 'III,set1' 'I,set2' 'II,set2' 'III,set2'.

print /title 'Redundancy analysis:'.

compute p1= ncol(set1).

compute p2= ncol(set2).

compute p= p1+p2.

print (cssq(load(1:p1,:))/p1)

/title "Prop. of variance in set1 explained by set1's variates".

print (cssq(load((p1+1):p,:))/p2)

/title "Prop. of variance in set2 explained by set2's variates".

print (cssq(crload(1:p1,:))/p1)

/title "Prop. of variance in set1 explained by set2's variates".

print (cssq(crload((p1+1):p,:))/p2)

/title "Prop. of variance in set2 explained by set1's variates".

end matrix.

### ГЛАВНЫЕ КООРДИНАТЫ [!KO\_pcoord]

\*/\*!KO\_pcoord(dis%m%method%name1%name2)\*/\*.

\*Version 1.

\*Анализ главных координат (= метрическое многомерное шкалирование Торгерсона).

\*Берет матрицу различий DIS, которые принимает за квадратные (sic!) расстояния, и укладывает эти

\*объекты в евклидово пространство, выдавая координаты объектов как NAME2.

\*Измерения (столбцы NAME2) являются некоррелированными переменными.

\*DIS долна быть квадратной симметричной, число\_объектов x число\_объектов.

\*M (целый скаляр) - нужная размерность евклидового пространста: целое число от 1 до число\_объектов,

\*либо любое неположительное число (тогда M будет = все реально существующие

\*нелишние измерения).

\*Аргумент METHOD (скаляр) это метод разложения матрицы-двойного центрата, промежуточного продукта DIS:

\*eigen-разложение (при METHOD положительно) или svd-разложение (если METHOD неположительно).

\*Все собственные или сингулярные числа от этих разложений выдаются как вектор NAME1.

\*Различие этих методов состоит в следующием: возможные отрицательные собственные числа двойного

\*центрата обнаруживаются eigen-разложением, и функция не использует их далее, но корректирует ими сумму

\*положительных собственных чисел. Отрицательные собственные числа не распознаются svd-разложением и выходят

\*как младшие положительные сингулярные числа. С точки зрения точности репродукции итоговыми евклидовыми

\*расстояниями (расстояниями между объектами в полученной конфигурации NAME2) входящих различий DIS вариант

\*с eigen-разложением как правило предпочтителен.

\*Если DIS это квадратные евклидовы расстояния, тогда оба метода одинаковы, причем координаты NAME2 тогда

\*тождественны сырым комнонентным баллам, которые были бы получены анализом главных компонент тех данных

\*(центрованных), из которых можно вычислить матрицу расстояний DIS. (Это свидетельствует от тесной

\*родственной связи между анализом главных координат и анализом главных компонент.).

### БИПЛОТ [!KO\_biplot]

\*/\*!KO\_biplot(data%rw%cw%norm%m%name1%name2%name3%name4)\*/\*.

\*Version 1.

\*Биплот с опцией взвешивания рядов и столбцов.

\*Выдает координаты точек-рядов и точек-столбцов, и другие статистики.

\*Эта функция также может считаться делающей взвешенный анализ главных компонент (см. ниже).

\*DATA - произвольная матрица данных (не менее двух рядов и столбцов). Анализ берет данные как есть,

\*поэтому при необходимости заранее сделайте нужную стандартизацию.

\*RW - вес для каждого ряда DATA (вектор-столбец) либо неположительный скаляр (что значит нет

\*взвешивания, программа придаст равный вес всем рядам).

\*CW - вес для каждого столбца DATA (вектор-ряд) либо неположительный скаляр (что значит нет

\*взвешивания, программа придаст равный вес всем столбцам).

\*Задаваемые вами веса могут быть любыми неотрицательными числами, только соотношение величин весов

\*играет роль. Заданный вес 0 для ряда/столбца означает, что данный ряд/столбец должен быть пассивным

\*(добавочным, supplementary). Всякий положительный вес означает активный ряд/столбец.

\*NORM - вектор из двух чисел от 0 до 1. Это коэффиенты нормирования, или наделения инерцией, координат

\*точек-рядов (первое число) и точек-столбов (второе число). Коэффициент 0 означает

\*"стандартное нормирование", когда масштаб координат стандартизован к 1. Коэффициент 1 означает

\*"главное нормирование", когда весь масштаб собственных чисел измерений отдан масштабу координат.

\*Например, {0,1} есть стандартное нормирование для точек-рядов и главное нормирование для

\*точек-столбцов; {1,1} есть главное нормирование для тех и других; {.5,.5} есть так называемое

\*"симметричное нормирование".

\*M - сколько измерений (главных осей) выделять: целое число от 1 до

\*min(число\_активных\_рядов,число\_активных\_столбцов) DATA. Либо укажите любое неположительное

\*число - тогда выделены будут все ненулевые измерения.

\*Результаты:

\*NAME1 - двухстолбцовая матрица: собственные числа (масштаб, инерция) измерений (1-й столбец) и доля

\*объясненной суммарной инерции в DATA (2-й столбец).

\*В NAME2, NAME3, NAME4 ряды это точки биплота: сначала идут точки-ряды DATA, потом, под ними,

\*точки-столбцы DATA.

\*NAME2 - двухстолбцовая матрица: масса точки (1-й столбец) и инерция точки (2-й столбец).

\*Масса активного ряда или столбца пропорциональна его весу. Масса пассивного ряда или столбца

\*пропорциональна усредненному весу активных рядов или столбцов, соответственно.

\*Инерция точки это ее масштаб в нормированной (взвешенной) матрице DATA. Если взвешивания не было

\*или веса всех активных рядов/столбцов были равными, тогда инерция ряда/столбца есть его сумма

\*квадратов в DATA, деленная на r\*c (r число активных рядов, c число активных столбцов).

\*NAME3 - координаты точек рядов и столбцов в пространстве M измерений. Характер нормирования инерцией

\*(аргумент NORM) сказывается на координатах.

\*NAME4 - вклады точек в инерцию измерений (первые M столбцов) и вклады измерений в инерцию точек

\*(последние M столбцов). Вклады точек в измерения это квадраты эл-тов собственных векторов. Точки с

\*высоким вкладом в инерцию измерения это точки, кому преимущественно обязано измерение своим появлением.

\*У пассивных точек вклад 0, т.к. они не влияют на формирование измерений. Вклад измерения в инерцию

\*точки - это показатель полноты объяснения данной точки данным измерением; высокий вклад означает,

\*что точка хорошо репрезентируется данным измерением и его достаточно для ее "объяснения".

\*ОТНОШЕНИЕ к анализу главных компонент (PCA). Биплот и PCA почти тождественны: единственное различие

\*состоит в том, что биплот приводит сумму весов столбцов входящих данных к 1, а PCA - к числу

\*(активных) столбцов. Это делает результаты двух методов пропорционально тождественными, и данную

\*функцию можно использовать для исполнения анализа главных компонент, в том числе PCA с взвешиванием

\*важности столбцов и/или рядов данных.

\*Конкретно, пусть C это число активных столбцов, тогда верны следующие отношения для пересчета:

\*1) собс. числа PCA = собс. числа биплота \* C;

\*2) нагрузки = координаты столбцов под "главным нормированием" столбцов;

\*3) стандартизованные компонентные баллы = координаты рядов под "стандартным нормированием" рядов;

\*4) собственные векторы PCA = координаты столбцов под "стандартным нормированием" столбцов / sqrt(C);

\*5) сырые компонентные баллы = координаты рядов под "главным нормированием" рядов \* sqrt(C).

\*О родственности biplot, PCA и correspondence analysis - https://stats.stackexchange.com/q/141754/3277.

ПРИМЕР. Карта восприятия 13 стран (с добавочными категориями).

data list list /country (a8) LIVING CLIMATE FOOD SECUR HOSPIT INFRA (6f2).

begin data

Italy 7 8 9 5 3 7

Spain 7 9 9 5 2 8

Croatia 5 6 6 6 5 6

Brazil 5 8 7 3 2 3

Russia 6 2 2 3 7 6

Germany 8 3 2 8 7 9

Turkey 5 8 9 3 1 3

Morocco 4 7 8 2 1 2

Peru 5 6 6 3 4 4

Nigeria 2 4 4 2 3 2

France 8 4 7 7 9 8

Mexico 2 5 5 2 3 3

SouthAfr 4 4 5 3 3 3

end data.

variable level all (sca) country (nom).

dataset name data13countries.

matrix.

get data /variables= living climate food secur hospit infra

/names= varname.

get country /variable= country.

compute point= {country;t(varname)}.

!KO\_center(data%data). /\*Perhaps center the features

compute rw= {1;1;0;1;1;1;1;1;1;1;1;1;0}. /\*Weights for rows: Croatia and

/\*South Africa will be declared supplementary rows (no influence

/\*on the definition of dimensions), the rest will have equal influence

!KO\_biplot(data%rw%0%{.5,.5}%2%eival%masine%coord%contr).

/\*Apply biplot to extract coordinates in 2 dimensions;

/\*Let us spread inertia (variability) half between rows and columns

print eival /title 'Inertia of dimensions' /clab 'Eigenval' 'Prop'.

print masine /title 'Points mass and inertia' /clab 'Mass' 'Inertia' /rname= point.

print coord /title 'Points coordinates in dimensions I and II'

/clab 'I' 'II' /rname= point.

print contr /title 'Contributions' /clab '->I' '->II' '<-I' '<-II' /rname= point.

save {point,coord} /outfile= \* /variables= point dim1 dim2 /string= point.

end matrix.

GRAPH /SCATTERPLOT(BIVAR)= dim1 WITH dim2 BY point (IDENTIFY).

### ПРОСТОЙ АНАЛИЗ СООТВЕТСТВИЙ [!KO\_corresp]

\*/\*!KO\_corresp(tab%sta%rsup%csup%norm%m%name1%name2%name3%name4)\*/\*.

\*Version 1.

\*Простой (двувходовый) анализ соответствий. Является приложением идеи биплота к анализу таблицы

\*сопряженности.

\*Выдает координаты точек-рядов и точек-столбцов, и другие статистики.

\*TAB - таблица сопряженности (не менее двух рядов и столбцов). Это любая таблица с неотрицательными

\*элементами, в которой значение в ячейке имеет смысл сопряженности, близости между данным рядом и

\*данным столбцом. В TAB суммы в рядах от активных столбцов должны быть ненулевыми и суммы с столбцах от

\*активных рядов должны быть ненулевыми (активные ряды/столбцы это не пассивные).

\*STA - нужная стандартизация для таблицы, целое число от 1 до 6:

\*1 - двухвходовая центрация и взвешенное нормирование

\*2 - двухвходовая центрация и простое нормирование

\*3 - центрация рядов и простое нормирование

\*4 - центрация столбцов и простое нормирование

\*5 - уравнивание рядных сумм, центрация рядов и простое нормирование

\*6 - уравнивание столбцовых сумм, центрация столбцов и простое нормирование

\*Опция (1) есть хи-квадратная модель анализа, она оптимальна, если TAB это частотная таблица. Остальные

\*пять опций на выбор это евклидова модель анализа, которая подходит, если TAB это не частоты

\*(а, например, суммы или средние). Хи-квадратная модель подразумевает биплот со взвешиванием.

\*RSUP - вектор номеров рядов, которые надо считать пассивными (добавочными, supplementary), либо любой

\*неположительный скаляр, если пассивные не нужны.

\*CSUP - аналогично, вектор номеров столбцов, которые надо считать пассивными, либо любой

\*неположительный скаляр. Ряды и столбцы TAB, не являющиеся пассивными, считаются активными.

\*NORM - вектор из двух чисел от 0 до 1. Это коэффиенты нормирования, или наделения инерцией, координат

\*точек-рядов (первое число) и точек-столбов (второе число). Коэффициент 0 означает

\*"стандартное нормирование", когда масштаб координат стандартизован к 1. Коэффициент 1 означает

\*"главное нормирование", когда весь масштаб собственных чисел измерений отдан масштабу координат.

\*Например, {0,1} есть стандартное нормирование для точек-рядов и главное нормирование для

\*точек-столбцов; {1,1} есть главное нормирование для тех и других; {.5,.5} есть так называемое

\*"симметричное нормирование".

\*M - сколько измерений выделять: целое число от 1 до

\*min(число\_активных\_рядов-1,число\_активных\_столбцов-1) TAB. Либо укажите любое неположительное

\*число - тогда выделены будут все ненулевые измерения.

\*Результаты (они того же вида, что в биплоте /\*!KO\_biplot\*/):

\*NAME1 - двухстолбцовая матрица: собственные числа (масштаб, инерция) измерений (1-й столбец) и доля

\*объясненной суммарной инерции в преобразованной стандартизацией TAB (2-й столбец).

\*В NAME2, NAME3, NAME4 ряды это точки биплота: сначала идут точки-ряды TAB, потом, под ними,

\*точки-столбцы TAB.

\*NAME2 - двухстолбцовая матрица: масса точки (1-й столбец) и инерция точки (2-й столбец).

\*Масса активного ряда или столбца пропорциональна его весу. Масса пассивного ряда или столбца

\*пропорциональна усредненному весу активных рядов или столбцов, соответственно.

\*Инерция точки это ее масштаб (сумма квадратов) в стандартизованной TAB.

\*NAME3 - координаты точек рядов и столбцов в пространстве M измерений. Характер нормирования инерцией

\*(аргумент NORM) сказывается на координатах.

\*NAME4 - вклады точек в инерцию измерений (первые M столбцов) и вклады измерений в инерцию точек

\*(последние M столбцов). Вклады точек в измерения это квадраты эл-тов собственных векторов.

\*Точки с высоким вкладом в инерцию измерения это точки, кому преимущественно обязано измерение своим

\*появлением. У пассивных точек вклад 0, т.к. они не влияют на формирование измерений. Вклад измерения

\*в инерцию точки - это показатель полноты объяснения данной точки данным измерением; высокий вклад

\*означает, что точка хорошо репрезентируется данным измерением и его достаточно для ее "объяснения".

ПРИМЕР. Анализ соответствий частотной таблицы.

matrix.

get vars /variables= v1 v2. /\*take two numeric categorical variables

!KO\_crosstab(vars(:,1)%vars(:,2)%1%table%rowcat%colcat). /\*obtain the crosstabulation

print rowcat /title 'Row categories'.

print colcat /title 'Column categories'.

print table /title 'Frequency crosstabulation'.

!KO\_corresp(table%1%0%0%{.5,.5}%2%eival%masine%coord%contr).

/\*run correspondence analysis with: chi-square model, no passive rows

/\*or columns, symmetric normalization, extraction of 2 dimensions

print eival /title 'Inertia of dimensions' /clab 'Eigenval' 'Prop'.

print masine /title 'Points mass and inertia' /clab 'Mass' 'Inertia'.

print {{rowcat;t(colcat)},coord} /title 'Points coordinates in dimensions I and II'

/clab 'Point' 'I' 'II'.

print contr /title 'Contributions' /clab '->I' '->II' '<-I' '<-II'.

end matrix.

### ФАКТОРЫ МЕТОДОМ ГЛАВНЫХ ОСЕЙ [!KO\_paf]

\*/\*!KO\_paf(cov%comm%m%conv%name1%name2%name3%name4%name5)\*/\*.

\*Version 1.

\*Линейный факторный анализ (только извлечение факторных нагрузок из матрицы; функция не делает

\*вращений факторов и вычисления факторных баллов). Метод главных осей применяется для экстракции.

\*Входящие:

\*COV - квадратная p x p симметричная матрица коэффициентов sscp-типа - т.е. корреляционная, ковариационная,

\*косинусная или сырая sscp-матрица коэффициентов связи между переменными.

\*COMM - вектор-столбец начальных оценок общностей, либо неположительный скаляр. Если вектор, то он должен

\*быть длиной как диагональ COV (т.е. равен числу переменных p) и содержать положительные числа, не

\*превышающие соответствующие значения этой диагонали (т е масштабы, дисперсии переменных).

\*Если COMM неположительный скаляр, функция будет использовать в качестве начальных общностей образы

\*(images) матрицы COV - они как правило являются лучшими кандидатами на начальные общности. Эта опция

\*требует положительно определенную COV.

\*M - сколько факторов выделять, укажите целое положительное число, меньшее чем число переменных

\*(вы должны заранее решить, используя известные методы оценки, сколько выделять факторов).

\*CONV - параметры итеративного схождения: вектор из двух чисел, например {.001,25}. Первое это порог

\*изменения общностей - неотрицательное число; если изменение будет меньше, общности считаются

\*стабилизировавшимися и итерации прекращаются. Второе это максимально позволенное число итераций -

\*задайте целое положительное число.

\*Результаты:

\*NAME1 - M x 4 матрица масштабов (дисперсий) выделенных факторов и долей объясненного ими полного масштаба.

\*1-й столбец это масштабы (дисперсии) факторов как есть, равные M первым собственным числам

\*редуцированной COV на последней итерации. 2-й столбец это доля этих масштабов от полного масштаба,

\*равного следу входящей COV. 3-й столбец это перешкалированные масштабы (дисперсии), т.е. такие, из

\*которых удален эффект неодинаковости масштабов (дисперсий) переменных. 4-й столбец относится к 3-му как

\*2-й относится к 1-му.

\*Если в NAME1 окажется фактор с отрицательной дисперсией, то он мнимый; однако увеличение числа итераций

\*часто переводит его в реальный; либо уменьшите M.

\*NAME2 - p x M матрица факторных нагрузок, как они есть. Их столбцовые суммы квадратов равны масштабам

\*факторов, а рядные суммы квадратов - итоговым общностям.

\*NAME3 - перешкалированные нагрузки. Это нагрузки, из которых удален эффект неодинаковости масштабов

\*(дисперсий) переменных; они суть корреляции (или косинусы) между факторами и переменными и удобны для

\*интерпретации факторов.

\*NAME4 - начальные и итоговые оценки общностей, p x 4. Столбцы слева направо: начальные как есть,

\*итоговые как есть, начальные перешкалированные, итоговые перешкалированные. Перешкалированная общность

\*это общность, деленная на масштаб (дисперсию) переменной, и является долей масштаба переменной,

\*объясняемой общими факторами.

\*Если COV это корреляционная или косинусная матрица, то "сырые" и перешкалированные результаты

\*в NAME1, в NAME2/NAME3, в NAME4 будут тождественны.

\*NAME5 - отчет о схождении, вектор из двух чисел, который можно сравнить с аргументом CONV:

\*первое число это максимальное изменение общностей, случившееся на последней проделанной итерации и

\*которое, ожидается, должно быть меньше порога в CONV. Второе число - число проделанных итераций; если

\*это число больше на 1 чем заданное в CONV, значит, было проделано максимальное число итераций, но

\*заказанная точность схождения так и не была достигнута.

\*Случай Хейвуда. Если на какой-либо итерации общность переменной превысит масштаб переменной, итерации

\*обрываются, результаты выдаются существующие на момент обрыва. О возникшем случае Хейвуда видно по

\*значению, превышающему единицу, в последнем столбце NAME4; кроме того, функция ради внимания

\*даст второму числу в NAME5 отрицательный знак. Результаты в случае Хейвуда следует признать

\*несостоявшимися.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. См. пример ниже.

### ОРТОГОНАЛЬНЫЕ АНАЛИТИЧЕСКИЕ ВРАЩЕНИЯ [!KO\_ortrot]

\*/\*!KO\_ortrot(load%kaiser%method%conv%reord%name1%name2%name3)\*/\*.

\*Version 2.

\*Делает разные виды ортогональных аналитических вращений, обобщенно известных как "ортомакс",

\*которые обычно употребляются для вращения экстрагированных факторов в факторном анализе.

\*LOAD - входящая матрица нагрузок ортогональных факторов (p переменных x m факторов,

\*p и m >1), которую надо вращать (результат вращения - матрица NAME1). LOAD это не обязательно

\*факторные нагрузки, это могут быть просто координаты каких-то p точек в пространстве m

\*перпендикулярных осей.

\*KAISER - скаляр или вектор-столбец длиной p.

\*Если неположительный скаляр, то нормирования Кайзера не будет. Иначе - будет. Нормирование

\*Кайзера заключается в изменениии сумм квадратов в рядах LOAD перед вращением (а после вращения

\*результат подвергнется обратному разнормированию).

\*Если KAISER - положительный скаляр, будет обычное нормирование Кайзера, уравнивающее суммы

\*квадратов в рядах LOAD путем деления на sqrt(общности).

\*Если KAISER - вектор, то он должен состоять из положительных чисел; будет произвольное

\*нормирование Кайзера, делящее LOAD на sqrt(эти числа).

\*METHOD - метод вращения, кл. слово заглавными буквами (можно взять в кавычки или апострофы).

\*Это критерий, который максимизируется при данном методе:

\*"QUARTIMAX" - квартимакс; он равен сумме итоговых нагрузок (итоговых координат), возведенных

\*в 4-ю степень: msum(NAME1^4).

\*Остальные методы имеют формулу критерия p\*Q-c\*V, где Q это критерий квартимакс, а V это сумма

\*квадратов дисперсий итоговых факторов, столбцов NAME1: rsum(cssq(NAME1)^2).

\*Коэффициент c определяет метод:

\*"VARIMAX" - варимакс: c=1

\*"EQUAMAX" - эквамакс: c=m/2

\*"PARSIMAX" - парзимакс: c=p\*(m-1)/(p+m-2)

\*"FACPARS" - факпарз (factor parsimony): c=p

\*"CUSTOM c" - произвольное значение c: задайте ему значение (число или имя скаляра). Значение может

\*быть любым. Чем выше c, тем дальше конфигурация будет от "генерального фактора".

\*С приближением c к +бесконечности дисперсии факторов, cssq(NAME1), будут равными.

\*С приближением c к -бесконечности дисперсии факторов будут равны таковым, как если подвергнуть LOAD

\*вращению в свои главные компоненты (без предварительной центрации).

\*CONV - параметры итеративного схождения: вектор из двух чисел, например {.0001,50}. Первое это

\*порог изменения критерия - неотрицательное число; если изменение будет меньше порога, критерий

\*считается стабилизировавшимся и итерации прекращаются (см. NAME3). Второе - это максимально

\*позволенное число итераций - задайте целое положительное число.

\*REORD - скаляр. Если положительный, то факторы после вращения, столбцы NAME1, будут упорядочены

\*по убыванию их дисперсий (т.е. сумм квадратов в столбцах), а также в каждом столбце знак нагрузкам

\*будет назначен так, чтобы сумма нагрузок была в нем положительна. Если скаляр неположительный,

\*эти две послевращательные операции делаться не будут.

\*Результаты:

\*NAME1 - итоговая матрица нагрузок (координат).

\*NAME2 - матрица ортогонального поворота, так что NAME1=LOAD\*NAME2.

\*NAME3 - история итераций, это вектор значений критерия на выходе из итераций. 1-е значение в

\*векторе это величина критерия для входящей LOAD; 2-е значение это величина критерия после 1-й

\*итерации вращения, и т.д. Последнее значение в векторе это величина критерия на выходе из последней

\*потребовавшейся итерации. Всего итераций, таким образом, nrow(NAME3)-1.

\*Значения критерия растут от итерации к итерации вплоть до стабилизации или исчерпания предела

\*итераций. У квартимакса и варимакса значения всегда положительные, но у остальных методов они могут

\*быть и отрицательными, это зависит от входящих данных.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Замечание: если факторный/главнокомпонентный анализ делался на ковариациях (а не корреляциях), то

\*нагрузки бывают двух видов – сырые и перешкалированные (последние получаются делением первых на

\*ст. отклонения переменных). Больше смысла вращать к "простой структуре" – именно перешкалированные

\*нагрузки, т.к. интерпретируют обычно их. Впрочем, если вы делаете вращение с нормированием Кайзера, то

\*результаты вращения перешкалированных и сырых нагрузок будут эквивалентны, т.к. NAME2 тогда одна и та же.

ПРИМЕР. Факторный анализ с ортогональным вращением факторов (варимакс).

set mxloops 10000.

matrix.

get vars /variables= v1 to v30.

!KO\_corr(vars%r). /\*Correlations

\*Factor extraction.

!KO\_paf(r%0%3%{.001,25}%fvrnc%load%rload%comm%conv). /\*Extract 3 factors

print conv /title 'Extraction Convergence report'.

print fvrnc /title 'Extraction SS loadings (factor variances)'

/clab= 'Eival' 'Prop' 'RescEiv' 'Prop'.

print load /title 'Extraction Loadings'.

\*Factor rotation.

!KO\_ortrot(load%1%VARIMAX%{.0001,50}%1%rotload%rotm%history).

/\*Varimax with Kaiser normalization

print rotload /title 'Rotation Loadings (varimax with kaiser)'.

print cssq(rotload) /title 'Rotation SS loadings (factor variances)'.

print rotm /title 'Rotation (transformation) matrix'.

print history /title 'Iteration history of the rotation criterion'.

end matrix.

### КОСОУГОЛЬНОЕ ВРАЩЕНИЕ ПРОМАКС И ПРОМАЙ [!KO\_promax]

\*/\*!KO\_promax(ort%kaiser%k%name1%name2%name3)\*/\*.

\*Version 1.

\*Делает вращение промакс или промай ортогональной конфигурации ORT, которую функция принимает за

\*"уже близкую к простой структуре". ORT это матрица факторных нагрузок, уже повернутая

\*к "простой структуре" ортогонально; в классическом варианте - методом варимакс (используйте для

\*этого ф-цию \*/\*!KO\_ortrot\*/). Промакс рассматривает ORT^K как целевую конфигурацию - более идеальную

\*"простую структуру", чем достигнуто в ORT (возведение элементов ORT в степень K>1 с сохранением их

\*знака сближает малые нагрузки с нулем скорее, чем большие нагрузки, "заостряя" паттерн) и пытается

\*приблизить ORT к ORT^K, ослабив ортогональность осей (факторов) неортогональным прокрустовым

\*вращением, причем ORT^K принимается за целевое отображение (нагрузки, а не за корреляции).

\*Итог: косоугольное отображение NAME1, полученное из ортогонального ORT.

\*Промай похож на промакс, но создает целевую для вращения конфигурацию не возведением в степень K,

\*а другим способом (см. Mulaik, Foundations of factor analysis, 2010).

\*Входящие:

\*ORT - p x m (p>=m) матрица нагрузок или каких-то координат в ортогональной (декартовой) системе.

\*ORT должна быть полноранговой, т.е. t(ORT)\*ORT обратимой. ORT рассматривается как близкая к

\*"простой структуре", но еще ортогональная.

\*KAISER - скаляр. Если положительный, то целевая конфигурация будет основана на ORT, подвергнутой

\*нормированию Кайзера. Это рекомендованный вариант (нормирование Кайзера это приведение сумм

\*квадратов в рядах к единице). Если скаляр неположительный, то целевая конфигурация будет основана

\*на ORT как она есть. В последнем случае предпочтительно, чтобы значения в ORT не превышали 1 по

\*абс. величине.

\*K - скаляр, промаксный параметр, степень для элементов ORT. В промаксе смысл имеют только

\*степени >1. Чаще используют K от 2 до 4.

\*Если вы укажете неположительное значение для K, то ф-ция осуществит метод промай вместо промакса.

\*Конкретное значение K тогда не играет роли, т.к. промай не возводит элементы в степень K.

\*Результаты:

\*NAME3 - матрица промаксного (или промайного) преобразования (де-ортогонализации осей).

\*NAME1 - результатная матрица факторного отображения, = ORT\*NAME3.

\*NAME2 - матрица итоговых корреляций между факторами (косинусы углов между неортогнальными осями

\*в конфигурации NAME1); она равна inv(sscp(NAME3)).

\*Матрица факторной структуры есть NAME1\*NAME2.

ПРИМЕР. Промакс после анализа главных компонент.

set mxloops 10000.

matrix.

get data /variables= v1 to v10. /\*Some variables

!KO\_center(data%data). /\*We will do PCA on covariances (not correlations),

/\*therefore center (not standardize) the data

!KO\_pcomp(data%1%3%1%eival%load%rload%ssccoef%scores). /\*PCA extracting 3 principal

/\*components

print load /title 'Extracted loadings' /format f8.3. /\*The raw

print rload /title 'Extracted loadings (rescaled)' /format f8.3. /\*and the rescaled

/\*loadings differ when covariances were analyzed

!KO\_ortrot(rload%1%VARIMAX%{.0001,50}%0%rotload%q%history). /\*Perform varimax

/\*(promax normally is based on it); it is better to rotate

/\*rescaled loadings because we base interpretations on rescaled ones

print rotload /title 'Varimax-rotated (rescaled) loadings - input to promax'

/format f8.3.

!KO\_promax(rotload%1%3%pattern%corr%q\_). /\*Perform promax (with power 3) based on

/\*varimax-rotated loadings

print pattern /title 'Pattern matrix (promax)' /format f8.3.

print (pattern\*corr) /title 'Structure matrix (promax)' /format f8.3.

print corr /title 'Factor (component) correlations' /format f8.3.

print (rotload\*q\_) /format f8.3. /\*(This is what the pattern matrix is)

print (rload\*q\*q\_) /format f8.3. /\*(and this) Tip: keep the REORD argument

/\*of KO\_ortrot on 0, for this expression to work properly

end matrix.

\*Equivalent job by FACTOR command:.

\*Note: sign of loadings and order of factors may differ from the results above,

\*what is normal.

FACTOR

/VARIABLES v1 to v10 /ANALYSIS v1 to v10

/PRINT INITIAL EXTRACTION ROTATION

/CRITERIA FACTORS(3) ITERATE(25) /EXTRACTION PC

/CRITERIA ITERATE(50) /ROTATION PROMAX(3)

/METHOD= COVARIANCE.

\*Note: When METHOD=COVARIANCE, SPSS rotates the rescaled loadings (and obtains

\*the raw ones by back-transforming).

### КОЭФФИЦИЕНТЫ ФАКТОРНЫХ БАЛЛОВ [!KO\_fsc]

\*/\*!KO\_fsc(cov%load%fcorr%method%name1%name2)\*/\*.

\*Version 1.

\*Вычисляет матрицу коэффициентов факторных баллов (factor scores), с помощью которой можно получить

\*оценки значений факторов, выделенных в факторном анализе, как переменных. Имеется несколько методов

\*вычисления ее.

\*Входящие, необходимые для всех методов METHOD:

\*COV - анализировавшаяся в факторном анализе корреляционная или ковариационная симметричная матрица

\*размером p x p.

\*LOAD - p x m (1<=m<p) матрица факторных нагрузок. Если факторы подверглись косоугольному вращению,

\*то это должна быть матрица факторного отображения. Если COV это ковариации (не корреляции), LOAD это

\*"сырые" (не "перешкалированные") нагрузки.

\*Входящее, необходимое не для всех методов METHOD:

\*FCORR - m x m матрица корреляций между факторами. Если факторы ортогональны или если m=1, вы можете

\*указать вместо матрицы произвольный скаляр.

\*Метод вычисления факторных баллов METHOD (ключевое заглавное слово, опционально можно в кавычках

\*или апострофах):

\*"REG" - регрессионный метод Терстоуна (Томпсона); максимизирует валидность баллов.

\*"REG2" - тот же метод, но в качестве внедиагональных элементов COV будет использовать воссозданные

\*факторами значения.

\*"REG3" - метод Хорста, или "идеализированных переменных"; факторные баллы вычисляются точно так, как

\*компонентные баллы в главнокомпонентном анализе; эквивалентен методу "REG" с использованием вместо COV

\*воссозданной редуцированной матрицы. Не нуждается в FCORR (укажите любой скаляр).

\*"BART" - метод Бартлета (Bartlett); минимизирует смещенность баллов. Не нуждается в FCORR

\*(укажите любой скаляр).

\*Следующие три метода называются корреляционно-сохраняющими, потому что добиваются корреляций между

\*факторными баллами таких же, какие между факторами:

\*"AR" - метод Андерсона-Рубина (Anderson-Rubin). Не нуждается в FCORR (укажите любой скаляр).

\*Этот метод всегда дает некоррелированные баллы, даже для косоугольных факторов.

\*"MAR" - метод Макдональда-Андерсона-Рубина (McDonald-Anderson-Rubin) является обобщением предыдущего,

\*годящимся и для косоугольных факторов.

\*"GREEN" - метод Грина (Green). Его можно рассматривать как приближение к "MAR", которому он становится

\*тождествен, если все общности у переменных одинаковы.

\*Для методов REG, MAR, GREEN необходима невырожденная COV.

\*Результаты:

\*NAME1 - коэффициенты для вычисления факторных баллов; последние есть DATA\*NAME1, где DATA -

\*анализировавшиеся факторным анализом переменные (центрованные, если факторный анализ базировался на

\*ковариациях, или стандартизованные, если он базировался на корреляциях).

\*NAME2 - перешкалированные (стандартизованные) коэффициенты, т.е. такие, из которых устранен эффект

\*неодинаковости дисперсий переменных. Если COV это корреляции, то NAME2=NAME1.

\*Если вместо собственно факторного анализа применялся анализ главных компонент, то компонентные баллы,

\*одни и те же, выдаются методами REG, REG3, GREEN.

ПРИМЕР. Факторный анализ (без вращения) и факторные баллы.

set mxloops 10000.

matrix.

get vars /variables= v1 to v30.

!KO\_corr(vars%r). /\*Correlations

!KO\_paf(r%0%3%{.001,25}%fvrnc%load%rload%comm%conv). /\*Extract 3 factors

print conv /title 'Convergence report'.

print fvrnc /title 'Extraction SS loadings (factor variances)'

/clab= 'Eival' 'Prop' 'RescEiv' 'Prop'.

print load /title 'Loadings'.

!KO\_fsc(r%load%0%REG%b%stb). /\*Regression method to compute factor scores

print b /title 'Factor score coefficients'.

!KO\_zscore(vars%vars). /\*(Correlation-based analysis implies standardized data)

save (vars\*b) /out= \*. /\*Factor scores

end matrix.

### КЛАСТЕРИЗАЦИЯ МЕТОДОМ K-СРЕДНИХ [!KO\_kmeans]

\*/\*!KO\_kmeans(data%ini%iter%empty%name1%name2%name3%name4)\*/\*.

\*Version 1.

\*Кластерный анализ методом k-средних.

\*Берет мерные данные DATA (n наблюдений x p переменных) и кластеризует наблюдения в заданное число

\*кластеров k (k<n). Инициальные центры пользователь должен предложить.

\*INI - матрица инициальных центров (средних) размером k кластеров x p переменных (столбцов DATA).

\*ITER - проделать столько итераций (неотрицательный целый скаляр, например 10). ITER=0 соответствует

\*классификации без итераций, т.е. приписыванию наблюдений к инициальным центрам без поправления последних.

\*EMPTY - цифра (не имя и не выражение, цифру можно опционально взять в кавычки или апострофы) 0 или 1.

\*Это задание способа решения проблемы пустых кластеров, если таковые возникнут:

\*"1" - избавиться от пустого кластера, уменьшив k; k уже не сможет вернуться к исходному значению.

\*"0" - оставить пустому кластеру его предшествующий центр, в надежде что в дальнейшем кластер наполнится

\*(чего может и не произойти).

\*Пустые кластеры бывают при плохих инициальных центрах, а также сильно дискретных данных.

\*Результаты:

\*NAME1 - получившееся кластерное членство, в виде n x k двоичной матрицы, т.е. k фиктивных переменных,

\*соответствующих кластерам. Каждое наблюдение приписано одному кластеру: в этом столбце стоит 1.

\*Вы можете создать единую категориальную переменную кластерного членства так: NAME1\*t(1:ncol(NAME1)).

\*NAME2 - k x p матрица окончательных, перевычисленных средних: это центроиды кластеров NAME1.

\*NAME3 - k x p матрица предокончательных средних: это последние центры, к которым были классифицированы

\*наблюдения.

\*NAME4 - n x 1 столбец квадратных евклидовых расстояний от наблюдений до центров NAME3, к которым

\*наблюдения были в последний раз приписаны. Например, если наблюдение согласно NAME1 оказалось

\*зачислено в кластер 3, то его расстояние NAME4 до центра этого кластера есть расстояние до центра,

\*к-рый есть 3-й ряд в NAME3, именно на основании этого расстояния наблюдение было к нему приписано.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1000.

matrix.

get data /variables= v1 to v4.

compute ini= {1,-2,3,3;

-2,2,8,-3;

4,2,0,-5}.

!KO\_kmeans(data%ini%10%1%dummy%fin%pre%dpre).

print pre /title 'Classification centres'.

print fin /title 'Recalculated centroids'.

save {dummy\*t(1:ncol(dummy)),sqrt(dpre)} /out= \* /variables= clu dpre.

end matrix.

### БЛОК-ДИАГОНАЛИЗАЦИЯ МЕТОДОМ VAT/IVAT [!KO\_vat]

\*/\*!KO\_vat(mat%method%priority%name1%name2)\*/\*.

\*Version 1.

\*Переупорядочивает ряды/столбцы квадратной матрицы расстояний MAT методом VAT (и IVAT).

\*Это один из способов (Bezdek, Hathaway, "VAT: a tool for...", 2002; Havens, Bezdek, "An efficient

\*formulation of...", 2012) блок-диагонализации матрицы, целью которой является сосредоточить малые

\*значения расстояний вблизи диагонали матрицы. Если блок-диагонализованную матрицу отобразить на

\*графике "тепловая карта" (heatmap), то можно получить представление о присутствии кластерной

\*структуры в данных и примерно оценить число кластеров.

\*Матрица после блок-диагонализации (переупорядочивания рядов/столбцов) - результат NAME1.

\*Входящие:

\*MAT - квадратная симметричная неотрицательная матрица расстояний (или стоимостей) с нулевой

\*диагональю.

\*METHOD - кл. слово заглавными буквами (опционально в кавычках или апострофах):

\*"VAT" - сделать переупорядочивание VAT-алгоритмом. Он тесно связан с алгоритмом построения

\*минимального остовного дерева Прима. Порядок будет записан в векторе NAME2.

\*"IVAT" - сделав переупорядочивание VAT, затем сделать iVAT-операцию. Она определенным образом

\*заменяет некоторые расстояния в матрице другими ее расстояниями, уменьшая разнообразие расстояний

\*в матрице. Эффект iVAT-замены тот, что на "тепловой карте" (1) контраст между междукластерными и

\*внутрикластерными расстояниями усилится, помогая визуально обнаружить кластеры; (2) повысится

\*обнаруживаемость кластеров цепочечной структуры (включая сильно вытянутые, древовидные, кольцевидные).

\*iVAT-операция по своему эффекту тождественна обработке матрицы, полученной в VAT, алгоритмом

\*Флойда-Уоршалла в варианте "определение легчайших проходов".

\*Таким образом, method=IVAT возвращает матрицу с переупорядоченными+измененными (более дискретными)

\*значениями, нежели были в MAT.

\*PRIORITY - цифра (не имя и не выражение) 0 или 1 (опционально в кавычках или апострофах).

\*Это технический параметр, потенциально сказывающийся только если в MAT есть одинаковые значения

\*(ties). Он определяет порядок поиска/выбора элемента. 0 это поиск "от меньшего к большему": элемент

\*с меньшим индексом будет предпочтен. 1 это поиск "от большего к меньшему": элемент с большим индексом

\*будет предпочтен (этот вариант чуть быстрее). PRIORITY не меняет валидности результата, но

\*результаты в условиях ties могут различаться, вследствие того, что порядок (NAME2) может стать другим.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

compute d=

{0, 0.73, 0.19, 0.71, 0.16;

0.73, 0, 0.59, 0.12, 0.78;

0.19, 0.59, 0, 0.55, 0.19;

0.71, 0.12, 0.55, 0, 0.74;

0.16, 0.78, 0.19, 0.74, 0}. /\*Distance matrix

print d /format= f5.2.

\*Perform VAT.

!KO\_vat(d%VAT%1%vat%ord). /\*Apply VAT algoritm

print vat /format= f5.2. /\*VAT-reordred rows/columns

print ord. /\*The reordering followed this sequence

compute ord2= ord(ord). /\*(So you can restore

print vat(ord2,ord2) /format= f5.2. /\*the original order, if needed)

\*Perform iVAT.

!KO\_vat(d%IVAT%1%ivat%ord). /\*It is VAT reordering, then iVAT operation

print ivat /format= f5.2.

save vat /outfile= \*. /\*Save VAT- or

\*save ivat /outfile= \*. /\*iVAT-processed matrix to submit to a plotting

/\*command which can draw heatmap of a matrix

end matrix.

### K-БЛИЖНИХ СОСЕДЕЙ (С ЗАВИСИМОЙ ПЕРЕМЕННОЙ) [!KO\_knnpred]

\*/\*!KO\_knnpred(dv%dis%k%stat%name1%name2%name3%name4)\*/\*.

\*Version 1.

\*Анализ K-ближних соседей в его базовой форме (без подбора K и без отбора предикторов).

\*Функция берет в качестве входящих не переменные-предикторы (features), а уже вычисленную матрицу

\*расстояний между точками (наблюдениями); таким образом функция расстояния может быть любой,

\*какую выберет пользователь.

\*Идентификация ближних (ближайших) соседей делается так, как делает функция /\*!KO\_knnr\*/ (см.) -

\*/\*!KO\_knnpred\*/ базируется на ней.

\*Входящие:

\*DIS - прямоугольная или квадратная матрица R x C с неотрицательными элементами. Это расстояния

\*между R точками-рядами (training set, массив обучения) и C точками-столбцами (удержанные наблюдения

\*aka массив проверки, holdout cases aka test set). Если у вас есть только обучающий массив

\*(т.е. он же и проверочный), вы введете квадратную симметричную матрицу, диагональ которой должна

\*быть "закрыта" каким-нибудь заведомо бОльшим числом, чем остальные элементы матрицы

\*(чтобы расстояния точек до себя самих "выбыли из игры").

\*DV - зависимая (target) переменная, столбец длиной R. Она может быть категориальной или

\*количественной. Конечная цель анализа K-ближних соседей - предсказать значения этой переменной

\*у точек-столбцов (удержанных наблюдений, точек массива проверки) исходя того, какие значения

\*этой переменной имеют ближние к ним точки-соседи из массива обучения (см. аргумент STAT).

\*K - число ближних соседей, положительный целый скаляр (обычно K<<R).

\*STAT - статистика, являющаяся функцией предсказания значений DV у точек-рядов: заглавное кл. слово

\*(его можно взять в кавычкми или апострофы):

\*"MEAN" - средняя DV у ближних соседей (DV это количественный признак);

\*"MEDIAN" - медиана DV у ближних соседей (DV это количественный признак),

\*медиана вычисляется методом Averaging Empirical;

\*"MODE" - мода DV у ближних соседей (DV это категориальный или дискретный признак).

\*Есть также взвешенные версии этих трех статистик: "WMEAN", "WMEDIAN", "WMODE". Тогда статистика

\*вычисляется взвешенная, где весами выступают расстояния до ближайших соседей, обращенные

\*в сходства по формуле 1/(d+1). И таким образом, больший вес в предсказании придается тем из K

\*ближних соседей, которые ближе к точке предсказания.

\*Внимание: в случае MODE и WMODE переменная DV должна быть только из положительных значений.

\*STAT = MODE или WMODE это "классификация KNN-анализом".

\*Если мод (предсказанных категорий) несколько одинаковых, функция выбирает одну с наибольшей

\*частотой в переменной DV.

\*Результаты:

\*NAME1 - матрица K x C, содержащая номера K ближних соседей у каждой точки-столбца.

\*NAME2 - матрица K x C, содержащая сами расстояния до ближних соседей. В том случае,

\*если STAT= "WMEAN", "WMEDIAN" или "WMODE", то содержатся не расстояния, а полученные из них

\*сходства - веса, причем они нормированы к сумме 1 в каждом столбце.

\*NAME3 - матрица K x C, содержащая значения DV у K ближних соседей.

\*NAME4 - вектор-ряд длиной C, содержащий предсказание DV для каждой точки-столбца, т.е. элемент

\*i есть f(NAME3(:,i)), где f это функция STAT.

\*Примечание. Формула получения сходств из расстояний, 1/(d+1), даст более крутое падение веса

\*от ближайшего из K соседей к более отдаленному из них, если разница между соответствующими

\*двумя d велика, нежели когда она мала. Поэтому (при взвешенном анализе) результат может зависеть

\*от общей величинности (от диапазона) расстояний в данных. Вы можете захотеть умножить DIS

\*на константу, вздув или сжав так расстояния, чтобы повлиять на результат.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Вся выборка - обучающий массив.

set mxloops 1E6.

matrix.

get y /variable= y. /\*Scale dependent variable (target)

get x /variables= x1 x2 x3. /\*Independent scale variables (features)

!KO\_seuclid(x%d). /\*Compute matrix of (squared) euclidean distances

/\*between all data points

release x. /\*(we don't need X anymore)

call setdiag(d,1E300). /\*Because a point cannot be a neighbour to itself,

/\*cover the diagonal with some supervalue

compute k= 3. /\*We're interested in K=3 nearest neighbours as predictor points

!KO\_knnpred(y%d%k%MEAN%nei%dnei%ynei%ypred). /\*Run the KNN prediction;

/\*let the prediction function be the mean

\*Prediction result:.

print {y,t(ypred)} /title 'Actual and predicted (MEAN) Y for each case'

/clabels 'Obs\_Y' 'Pred\_Y' /format= f8.4.

\*Concomitant output:.

print t(ynei) /title 'K=3 Nearest neigbours (their Y values) to each case'

/format= f8.4.

print t(nei) /title 'K=3 Nearest neigbours (their indices) to each case'.

print t(dnei) /title 'K=3 Nearest neigbours (their distances) to each case'.

end matrix.

\*Equivalent run of SPSS prodedure KNN. All the cases are the training set.

KNN y (MLEVEL=S) WITH x1 x2 x3

/RESCALE COVARIATE=NONE

/MODEL NEIGHBORS=FIXED(K=3) METRIC=EUCLID FEATURES=ALL

/CRITERIA PREDICTED=MEAN WEIGHTFEATURES=NO

/PARTITION TRAINING=100 HOLDOUT=0

/SAVE PREDVAL.

ПРИМЕР. Есть обучающие наблюдения (TRAIN=1) и удержанные (holdout, TRAIN=0) наблюдения.

set mxloops 1E6.

matrix.

get y /variable= y. /\*Scale dependent variable (target)

get x /variables= x1 x2 x3. /\*Independent scale variables (features)

get set /variable= train. /\*Set identifier;

/\*Holdout cases will be predicted by training cases

!KO\_indices2(set%set1%set0). /\*Get indices of train cases and of holdout cases

!KO\_pwminkr(x(set1,:)%x(set0,:)%1%{1,1,1}%d). /\*Compute rectangular matrix

/\*of distances

/\*between training cases (row points) and holdout cases (column points);

/\*Let the distances be Manhattan ones

compute k= 3. /\*We're interested in K=3 nearest neighbours as predictor points

!KO\_knnpred(y%d%k%"MEDIAN"%nei%dnei%ynei%ypred). /\*Run the KNN prediction;

/\*let the prediction function be the median

\*Prediction result for holdout cases:.

print {y(set0),t(ypred)}

/title 'Actual and predicted (MEDIAN) Y for each holdout case'

/clabels 'Obs\_Y' 'Pred\_Y' /format= f8.4.

print t(ynei)

/title 'K=3 Nearest training neigbours (their Y values) to each holdout case'

/format= f8.4.

print t(nei)

/title 'K=3 Nearest training neigbours (their indices) to each holdout case'.

print t(dnei)

/title 'K=3 Nearest training neigbours (their distances) to each holdout case'.

\*If needed, do analysis within training set cases as well.

!KO\_pwmink(x(set1,:)%1%{1,1,1}%d). /\*Compute square matrix of Manhattan distances

/\*between training cases only

call setdiag(d,1E300). /\*Cover diagonal entries by some supervalue

!KO\_knnpred(y%d%k%"MEDIAN"%nei%dnei%ynei%ypred). /\*Run the KNN prediction as before

\*Prediction result for training cases:.

print /title '----------------'.

print {y(set1),t(ypred)}

/title 'Actual and predicted (MEDIAN) Y for each training case'

/clabels 'Obs\_Y' 'Pred\_Y' /format= f8.4.

print t(ynei)

/title 'K=3 Nearest training neigbours (their Y values) to each training case'

/format= f8.4.

print t(nei)

/title 'K= 3 Nearest training neigbours (their indices) to each training case'.

print t(dnei) /title 'K=3 Nearest training neigbours (their distances) to each case'.

end matrix.

\*Equivalent run of SPSS prodedure KNN. Variable TRAIN splits cases in training and holdout.

KNN y (MLEVEL=S) WITH x1 x2 x3

/RESCALE COVARIATE=NONE

/MODEL NEIGHBORS=FIXED(K=3) METRIC=CITYBLOCK FEATURES=ALL

/CRITERIA PREDICTED=MEDIAN WEIGHTFEATURES=NO

/PARTITION VARIABLE=train

/SAVE PREDVAL.

ПРИМЕР. Как Пример первый, но зависимая переменная категориальная (будет классификация).

set mxloops 1E6.

matrix.

get y /variable= ypos. /\*Categorical dependent variable (target) w/ positive values

get x /variables= x1 x2 x3. /\*Independent scale variables (features)

!KO\_seuclid(x%d). /\*Compute matrix of (squared) euclidean distances

/\*between all data points

release x. /\*(we don't need X anymore)

call setdiag(d,1E300). /\*Because a point cannot be a neighbour to itself,

/\*cover the diagonal with some supervalue

compute k= 3. /\*We're interested in K=3 nearest neighbours as predictor points

!KO\_knnpred(y%d%k%MODE%nei%dnei%ynei%ypred). /\*Run the KNN prediction;

/\*the prediction function be the MODE (knn analysis performs classification)

\*Prediction result:.

print {y,t(ypred)} /title 'Actual and predicted (MEAN) Y for each case'

/clabels 'Obs\_Y' 'Pred\_Y' /format= f8.4.

\*Concomitant output:.

print t(ynei)

/title 'K=3 Nearest neigbours (their Y values) to each case' /format= f8.4.

print t(nei) /title 'K=3 Nearest neigbours (their indices) to each case'.

print t(dnei) /title 'K=3 Nearest neigbours (their distances) to each case'.

end matrix.

\*Equivalent run of SPSS prodedure KNN. All the cases are the training set.

KNN ypos (MLEVEL=N) WITH x1 x2 x3

/RESCALE COVARIATE=NONE

/MODEL NEIGHBORS=FIXED(K=3) METRIC=EUCLID FEATURES=ALL

/CRITERIA WEIGHTFEATURES=NO

/PARTITION TRAINING=100 HOLDOUT=0

/SAVE PREDVAL.

### НАИВНЫЙ БАЙЕСОВ КЛАССИФИКАТОР [!KO\_nbclass]

\*/\*!KO\_nbclass(dv%ivs%name1%name3%name3)\*/\*.

\*Version 1.

\*Наивный байесов классификатор для категориальных данных в его базовой форме - без

\*отбора предикторов.

\*DV - зависимая категориальная (или дискретная) переменная.

\*IVS - независимые, предикторные категориальные (или дискретные) переменные.

\*DV имеет n1 наблюдений (рядов), IVS имеет n1+n2 (n2>=0) наблюдений (рядов). Первые n1 наблюдений -

\*это обучающая выборка (training sample). Идущие ниже n2 наблюдений в IVS, если есть - это

\*удержанная или тестовая выборка (holdout или test sample), т.е. наблюдения, DV-значения которых

\*мы или не знаем, или не подаем их в построение модели.

\*Результаты:

\*NAME1 - вектор-ряд k значений зависимой переменной, т.е. коды классов.

\*NAME2 - n1+n2 x k матрица, предсказанных (апостериорных) вероятностей принадлежности наблюдений к

\*классам; сумма вероятностей =1 в каждом ряду матрицы.

\*NAME3 - n1+n2 столбец предсказанных классов (значений DV); предсказанный класс это класс с

\*максимальной вероятностью в ряду NAME2. Если этот максимум одинаков для нескольких элементов в

\*ряду (ties), то выбирается класс с бОльшим значением NAME1.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. В массиве данных 100 наблюдений. Последние 10 наблюдений - пропуски в переменной Y.

set mxloops 1E6.

matrix.

get y /variable= y /missing= omit. /\*Dependent (target) categorical variable

/\*with, say, 3 categories (classes)

/\*It has 10 missing values in the end - those cases are holdout ones,

/\*so we discard them

get x /variables= x1 x2 x3 x4. /\*Categorical predictors, all 100 cases

!KO\_nbclass(y%x%yvals%predp%predc). /\*Run the analysis

print yvals /title "Classes' codes".

print {predc,predp} /title 'Predicted Class and classes Predicted Probabilities'

/clabels= 'PredCl' 'Prob1' 'Prob2' 'Prob3'. /\*Predictions for all cases,

/\*the 10 holdout cases are the last ones

\*Let's observe classification results for the training sample.

!KO\_crosstab(y%predc(1:90)%1%tab%name2%name3).

!KO\_classres(tab%overall%correct%ocorrect).

print {tab,correct;overall,ocorrect} /format= f8.1

/rlabels= 'Class1' 'Class2' 'Class3' 'Overall%'

/clabels= 'Class1' 'Class2' 'Class3' '%Correct'

/title 'Classification table: Observed x Predicted classes'.

end matrix.

\*Equivalent run of NAIVEBAYES command.

NAIVEBAYES y BY x1 x2 x3 x4 /SUBSET NOSELECTION /SAVE PREDVAL PREDPROB.

# ФУНКЦИИ СЛУЧАЙНЫХ ДАННЫХ

### СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ СТАНДАРТНОГО РАВНОМЕРНОРГО РАСПРЕДЕЛЕНИЯ

Используйте встроенную функцию uniform().

### СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ СТАНДАРТНОГО НОРМАЛЬНОГО РАСПРЕДЕЛЕНИЯ [!KO\_normal]

\*/\*!KO\_normal(nr%nc%name)\*/\*.

\*Version 1.

\*Порождает матрицу NAME размером NR x NC, элементы которой - случайные

\*значения из стандартного нормального распределения.

\*NR, NC - целые положительные скаляры.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

\*Для генерации данных из популяции с коррелирующими нормальными переменными используйте

\*более общую функцию /\*!KO\_mvnorm\*/.

### СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ НОРМАЛЬНОГО РАСПРЕДЕЛЕНИЯ С ЗАДАННЫМИ ПАРАМЕТРАМИ [!KO\_rvnorm]

\*/\*!KO\_rvnorm(mean%sd%name)\*/\*.

\*Version 1.

\*Порождает случайное число из нормального распределения со средней MEAN и стандартным отклонением SD.

\*MEAN и SD - матрицы любого, одинакового размера. Результат NAME - того же размера.

\*Элемент ij NAME содержит случайное значение из нормального распределения с параметрами ij MEAN и ij SD.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

### СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ РАВНОМЕРНОГО РАСПРЕДЕЛЕНИЯ С ЗАДАННЫМИ ПАРАМЕТРАМИ [!KO\_rvunif]

\*/\*!KO\_rvunif(min%max%name)\*/\*.

\*Version 1.

\*Порождает случайное число из равномерного распределения с минимумом MIN и максимумом MAX.

\*MIN и MAX - матрицы любого, одинакового размера. Результат NAME - того же размера.

\*Элемент ij NAME содержит случайное значение из равномерного распределения с параметрами ij MIN и ij MAX

\*(ij MIN должен быть не больше ij MAX).

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

### СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ ХИ-КВАДРАТ РАСПРЕДЕЛЕНИЯ [!KO\_chisq]

\*/\*!KO\_chisq(nr%nc%df%name)\*/\*.

\*Version 1.

\*Порождает матрицу NAME размером NR x NC, элементы которой - случайные

\*значения из хи-квадрат распределения с числом степеней свободы DF.

\*NR, NC, DF - целые положительные скаляры.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

### СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ БИНОМИАЛЬНОГО РАСПРЕДЕЛЕНИЯ [!KO\_binom]

\*/\*!KO\_binom(nr%ntrials%prob%name)\*/\*.

\*Version 1.

\*Порождает вектор-столбец NAME длиной NR, элементы которого - случайные значения из

\*биномиального распределения с параметрами число испытаний NTRIALS и вероятностью успеха PROB.

\*NTRIALS (целое положительное) и PROB (между 0 и 1) - либо два скаляра, либо два вектора длиной NR.

\*Если это векторы, значит, разные значения NAME можно породить под разными значениями параметров.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### СЛУЧАЙНЫЕ ЗНАЧЕНИЯ ИЗ КАТЕГОРИАЛЬНОГО РАСПРЕДЕЛЕНИЯ [!KO\_categ]

\*/\*!KO\_categ(nr%nc%kprob%name)\*/\*.

\*Version 2.

\*Порождает матрицу NAME размером NR x NC, элементы которой - случайные

\*значения (коды от 1 до k) из k-категориального распределения. (Категориальное распределение

\*это мультиномиальное распределение с числом испытаний 1. Распределение Бернулли это 2-категориальное

\*распределение.)

\*NR, NC - целые положительные скаляры.

\*KPROB - либо вектор длиной k, содержащий вероятности для k категорий, либо целый скаляр, равный k.

\*Если KPROB - вектор вероятностей, это должны быть неотрицательные числа, хотя бы одно положительное.

\*Числа не обязаны давать в сумме 1: функция сама приведет их сумму к 1; достаточно, чтобы числа

\*находились в соотношениях тех же, как вероятности.

\*Если KPROB - скаляр, он должен быть целым положительным (дробное значение будет обрезано до целого).

\*Он есть k, и k вероятностей полагаются равными. Когда цель - равновероятное распределение, выгоднее

\*такое задание KPROB, нежели заданее вектором вероятностей, т.к. будет быстрее.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

!KO\_categ(5%3%{.6,.3,.1}%name).

print name.

end matrix.

ПРИМЕР. Случайный выбор наблюдений с возвращением (sampling with replacement).

set mxloops 100000.

matrix.

get data /variables= id v1. /\*Case id and some data

loop i= 1 to 5. /\*5 random samples to draw

-!KO\_categ(10%1%nrow(data)%selcases).

/\*Genetrate, say, 10 indices (case numbers) to select

-print data(selcases,:). /\*The random sample of size 10

end loop.

end matrix.

Для выборки без возвращения используйте !KO\_catwor или !KO\_sample.

### ВЫБОРКА ИЗ КАТЕГОРИАЛЬНОГО РАСПРЕДЕЛЕНИЯ БЕЗ ВОЗВРАЩЕНИЯ [!KO\_catwor]

\*/\*!KO\_catwor(n%kprob%stock%method%name1%name2)\*/\*.

\*Version 1.

\*Набирает случайную выборку N объектов из ограниченной совокупности объектов, принадлежащих

\*разным категориям. Эта функция выдает случайные значения из k-категориального распределения

\*(k - число категорий объектов), как и функция /\*!KO\_categ\*/, но она делает выборку без возвращения

\*из ограниченной популяции, тогда как /\*!KO\_categ\*/ делает выборку с возвращением.

\*Т.к. объекты извлекаются один за другим и не возвращаются "на склад" (в ограниченную популяцию),

\*вероятности выбора ("реализации") для разных категорий меняются по ходу процесса. Эти вероятности

\*определяются, с одной стороны, заданными вероятностями "спроса", а с другой стороны, текущими

\*вероятностями "наличия на складе", так сказать.

\*Входящие:

\*N - целый положительный скаляр: требуемый размер выборки.

\*KPROB - вектор длиной k, содержащий вероятности для k категорий объектов. Это должны быть

\*неотрицательные числа, хотя бы одно положительное. Числа не обязаны давать в сумме 1: функция

\*сама приведет их сумму к 1; достаточно, чтобы числа находились в соотношениях тех же,

\*как вероятности. KPROB надо понимать как исходные вероятности "спроса"; т.е. это вероятности

\*выбора, если бы популяция была бесконечной.

\*STOCK - вектор длиной k и той же ориентации что KPROB, содержащий частоты для k категорий

\*объектов - это наблюдаемое распределение нашей ограниченной совокупности ("склада"), откуда будут

\*изыматься объекты в выборку. Частоты должны быть неотрицательными целыми числами, хотя бы одно

\*положительное. Если вы введете дробные значения, функция сперва обрежет их до целых.

\*Если веса KPROB задают категориям вероятности "спроса", то частоты STOCK задает им начальные

\*вероятности "наличия".

\*METHOD - цифра (не переменная и не выражение; цифру опционально можно закавычить или заапострофить).

\*Это задание метода определения текущих вероятностей выбора (реализации):

\*"1" - метод "Исходный спрос, если осталось". Вероятность PROB(i) выбрать сейчас объект категории i

\*пропорциональна KPROB(i)\*LEFT(i), где LEFT(i)=1, если объекты категории i еще остались в популяции,

\*иначе LEFT(i)=0. Если частоты STOCK все достаточно велики, чтобы не исчерпаться до окончания

\*процедуры выборки, то этот метод эквивалентен применению ф-ции /\*!KO\_categ\*/.

\*"2" - метод "Исходный спрос \* сколько осталось". Вероятность PROB(i) выбрать сейчас объект категории i

\*пропорциональна KPROB(i)\*FLEFT(i), где FLEFT(i) это доля категории i среди оставшихся пока в

\*популяции объектов. Это классический случайный выбор без возвращения.

\*"3" - метод "Плавающий спрос \* сколько осталось". Вероятность PROB(i) выбрать сейчас объект

\*категории i пропорциональна PROB\_(i)\*FLEFT(i), где PROB\_(i) это PROB(i), вычисленное перед выбором

\*предыдущего объекта в выборку (перед выбором 1-го объекта PROB\_(i)=KPROB(i)). Это "заостряющий" метод,

\*ведущий к быстрому по ходу выбора смещению предпочтения в сторону одной категории, как правило с

\*большей представленностью "на складе" ограниченной популяции.

\*"4" - метод "Плавающий не-спрос \* сколько осталось". Вероятность PROB(i) выбрать сейчас объект

\*категории i пропорциональна (1-PROB\_(i))\*FLEFT(i) (перед выбором 1-го объекта 1-PROB\_(i)=KPROB(i)).

\*Этот метод может быть интересен как "сглаживающая" альтернатива методу "2": выбор из малопредставленных

\*категорий будет немного повышен, а выбор из многопредставленных категорий будет несколько понижен, по

\*сравнению с тем методом.

\*Результаты:

\*NAME1 - вектор-столбец длиной N, содержащий коды категорий (номера из последовательности 1,2,...,k)

\*созданной случайной выборки (и последовательность тоже случайна). В том случае, если объектов в

\*популяции не хватило для выбора без возвращения N объектов, длина NAME1 будет меньше N. Если не

\*удалось взять ни одного объекта, NAME1=0.

\*NAME2 - вектор-столбец длиной k, содержащий оставшиеся в STOCK частоты.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если KPROB это равные вероятности, а STOCK это вектор из одних единиц или двоичный (1 и 0), то эквивалентно

\*будет воспользоваться функцией /\*!KO\_sample\*/, которая быстрее.

ПРИМЕР. Выбрать случайно 50 наблюдений из массива, взвешенного частотно. Кроме того, затребовано, чтобы чем значение данных больше по величине, тем вероятнее было его попадение в выборку.

set mxloops 10000000.

matrix.

get data /variables= id v1. /\*Case id and data V1

get freq /variables= freq. /\*also, frequency variable with positive integers

/\*defines how many duplicates of each case exist:

/\*this is the “stock” vector

compute kprob= abs(data(:,2)). /\*While the “demand” vector will reflect the

/\*magninute of V1 data values

!KO\_catwor(50%kprob%freq%"2"%sample%freq2).

save data(sample,:) /outfile= \* /variables= id v1.

print {data(:,1),freq2}. /\*How many case duplicates did not enter the sample.

end matrix.

\*In the saved sample if a case was selected more than once (due to duplicates

\*freq) it occupies own row. If you need to aggregate the duplicates, run:.

AGGREGATE /OUTFILE= \* /BREAK= id /v1= FIRST(v1) /freq= N.

### СЛУЧАЙНО ВЫБРАТЬ N ЭЛЕМЕНТОВ ВЕКТОРА [!KO\_sample]

\*/\*!KO\_sample(n%cases%name1%name2)\*/\*.

\*Version 1.

\*Случайно выбирает N разных элементов из последовательности номеров 1,2,...,k.

\*Эта функция удобна для формирования случайной подвыборки ("без возвращения") из массива,

\*а также для случайной сортировки наблюдений массива.

\*Входящие:

\*N - целый положительный скаляр: сколько разных элементов выбрать.

\*CASES - "популяция" элементов: либо целый положительный скаляр, либо двоичный (0 или 1) вектор.

\*Если скаляр, то он есть k, и выбор будет из чисел 1,2,...,k. Если двоичный вектор, то его

\*длина есть k и номера элементов соответствуют числам 1,2,...,k; значение 1 у элемента в векторе

\*разрешает выбрать номер этого элемента, а значение 0 - запрещает. Таким образом, вектор

\*CASES является фильтром. Вектор, состоящий из одних единиц, эквивалентен заданию CASES скаляром.

\*Вектор должен содержать минимум одну единицу.

\*Результаты:

\*NAME1 - вектор-столбец длиной N, содержащий случайно выбранные номера, и в случайном порядке.

\*NAME2 - двоичный вектор-столбец длиной k, где 1 помечает не выбранный (оставшийся) элемент CASES,

\*а 0 помечает выбранный (изъятый) элемент.

\*Если N>=k, то длина NAME1 будет k, а NAME2 будет состоять из одних нулей.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если ваша популяция CASES взвешенная, т.е. состоит из частот (а не 0 или 1), или если ее элементы

\*неравновероятны к выбору, используйте функцию /\*!KO\_catwor\*/.

ПРИМЕР. Выбрать случайно 30 наблюдений из массива. Затем выбрать еще 10 (из еще не выбранных).

set mxloops 10000000.

matrix.

get data /variables= id v1. /\*Case id and data

!KO\_sample(30%nrow(data)%selected%left).

print data(selected,:).

!KO\_sample(10%left%selected%left).

print data(selected,:).

end matrix.

ПРИМЕР. Эквивалентно предыдущему, с удалением выбранных наблюдений из данных.

set mxloops 10000000.

matrix.

get data /variables= id v1. /\*Case id and data

!KO\_sample(30%nrow(data)%selected%left).

!KO\_split(data%selected%sample1%remnant).

print sample1.

!KO\_sample(10%nrow(remnant)%selected%left).

!KO\_split(remnant%selected%sample2%remnant).

print sample2.

print remnant.

end matrix.

ПРИМЕР. Сортировать наблюдения в случайном порядке.

set mxloops 10000000.

matrix.

get data /variables= id v1. /\*Case id and data

!KO\_sample(nrow(data)%nrow(data)%order%left).

print data(order,:).

end matrix.

### СЛУЧАЙНАЯ МАТРИЦА ИЗ РАСПРЕДЕЛЕНИЯ УИШАРТА [!KO\_wishart]

\*/\*!KO\_wishart(chl%n%name)\*/\*.

\*Version 1.

\*Выдает случайную матрицу NAME из распределения Уишарта с параметром масштаба C и степенями свободы N.

\*C это популяционная ковариационная матрица (в популяции предполагается многомерное нормальное

\*распределение). Аргументом CHL функции выступает не C, а ее корень Холецкого: верхняя треугольная

\*матрица, выдаваемая встроенной матричной функцией chol(C).

\*N (целый положительный скаляр) это размер выборки для порождения случайной матрицы NAME.

\*Последняя является матрицей рассеяния (scatter); чтобы перевести ее в ковариационную матрицу,

\*поделите ее на N.

\*Если в качестве CHL указать скаляр (положительный), то он считается популяционным стандартным

\*отклонением, и NAME/N будет случайным значением дисперсии.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

compute cov=

{ 2.8, 1.9, 0.7,-0.4;

1.9, 4.1,-1.0, 0.5;

0.7,-1.0, 2.4, 1.3;

-0.4, 0.5, 1.3, 2.6 }.

print cov.

compute ch= chol(cov).

!KO\_wishart(ch%100%scat).

print (scat/100).

end matrix.

### СЛУЧАЙНЫЕ ДАННЫЕ ИЗ НОРМАЛЬНОЙ ПОПУЛЯЦИИ С ЗАДАННЫМИ КОВАРИАЦИЯМИ [!KO\_mvnorm]

\*/\*!KO\_mvnorm(n%chl%mean%name)\*/\*.

\*Version 1.

\*Порождает данные NAME - случайную выборку размером N из нормально распределенной популяции, имеющей

\*ковариации C и центроид MEAN.

\*Аргумент CHL функции это не сама ковариационная матрица C (размером p x p) между переменными,

\*а ее корень Холецкого: верхняя треугольная матрица (того же размера), выдаваемая встроенной матричной

\*функцией chol(C).

\*N - число наблюдений для порождения, целый положительный скаляр.

\*MEAN - скаляр либо вектор-ряд длиной p: это средние переменных в популяции.

\*Результат NAME матрица случайных данных n наблюдений x p переменных из нормальной популяции с указанными

\*ковариациями между переменными и средними переменных.

\*Как частный случай, C может быть размером 1 x 1; тогда C это дисперсия (а CHL это ст. отклонение).

\*Если матрица C это корреляции, а не ковариации, аргумент MEAN держите как 0, полученные переменные NAME

\*вы можете затем умножить на популяционные стандартные отклонения и потом прибавить популяционные средние;

\*но более удобный путь - конвертируйте корреляционную матрицу в ковариационную функцией /\*!KO\_corrcov\*/,

\*после чего примените /\*!KO\_mvnorm\*/ с заданием популяционных средних как MEAN в ней.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

ПРИМЕР. Породить 500-наблюденные случайные данные из нормальной 4-переменной коррелированной популяции.

matrix.

compute mx= {12.2, -2.3, 5.6, -3.2;

-2.3, 17.7, 1.1, -3.7;

5.6, 1.1, 23.1, 5.9;

-3.2, -3.7, 5.9, 5.8}. /\*Population covariances, positive

/\*definite matrix

print mx /title 'Population covariance matrix'.

!KO\_mvnorm(500%chol(mx)%0%vars).

!KO\_cov(vars%cov).

print cov /title 'Sample covarinces in the generated data'.

!KO\_mean(vars%mean).

print mean /title 'Means in the generated data'.

save vars /out= \*.

end matrix.

ПРИМЕР. Входяшая популяционная матрица это MSCP. Создать 10000-наблюденные случайные нормальные данные для нее.

matrix.

compute mx= {21.8, 9.5, 2.2, 4.2;

9.5, 32.1, -3.1, 5.4;

2.2, -3.1, 24.3, 3.3;

4.2, 5.4, 3.3, 11.6}. /\*Population MSCP matrix, 4 variables

compute mean= {3.1,3.8,-1.1,2.4}. /\*Population means

print mx.

print mean.

!KO\_swcentr(mx%mean%{0,0,0,0}%mx). /\*Since MVNORM function needs covariance matrix,

/\*convert the MSCP matrix into the covariance

/\*(hoping it will be positive definite)

compute n= 10000.

!KO\_mvnorm(n%chol(mx)%mean%vars). /\*Generate the sample, and take to the needed

/\*population mean

print (sscp(vars)/n). /\*So that the input MSCP matrix is satisfied

save vars /out= \*.

end matrix.

### СОЗДАТЬ СЛУЧАЙНЫЕ ДАННЫЕ ТОЧНО С ЗАДАННЫМИ КОВАРИАЦИЯМИ

Для получения массива случайных данных с точно как надо ковариирующими или коррелирующими переменными примените функцию /\*!KO\_tocov\*/ к случайно порожденным данным. Если вы хотите, чтобы данные были многомерно-нормальными, породите значения из нормального распределения (ф-ция /\*!KO\_normal\*/). Чтобы придать к конце желаемые средние и дисперсии (или максимум, минимум) переменным, используйте функцию /\*!KO\_rescale\*/. См. пример при функции /\*!KO\_tocov\*/.

# ФУНКЦИИ ПЕРЕКОДИРОВКИ/ЗАМЕНЫ

### ПЕРЕКОДИРОВКА ЗНАЧЕНИЙ (ТОЧНОЕ СОВПАДЕНИЕ) [!KO\_recode1]

\*/\*!KO\_recode1(data%filter%old%new%name)\*/\*.

\*Version 1.

\*В матрице данных DATA любого размера заменяет значения списка OLD на соответствующие значения списка NEW.

\*Выдает матрицу NAME, являющуюся матрицей DATA с перекодированными значениями.

\*Эта функция подобна внематричной команде RECODE, в которой значения для перекодировки заданы списком.

\*OLD и NEW - векторы-ряды. i-е значение OLD, если найдено в DATA, перекодируется в i-е значение NEW.

\*Оба эти вектора - одинаковой длины, либо NEW может быть на один элемент длиннее; в последнем случае все прочие

\*кроме OLD значения данных будут перекодированы в это последнее значение NEW.

\*Если некое значение встречается в OLD больше раза, оно перекодируется в NEW-значение, соответствующее первой

\*(самой левой) его встрече в векторе OLD.

\*Аргумент FILTER позволяет запретить перекодировать значения некоторых рядов DATA. FILTER либо скаляр,

\*либо столбец длиной как число рядов DATA. Если ненулевой скаляр - никакого запрета нет; если нулевой

\*скаляр - то полный запрет (функция не сработает, NAME будет равна DATA). Если столбец, то 0 в нем запрещает

\*перекодировку в данном ряду DATA, а иное значение - разрешает.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Перекодировка в MATRIX и эквивалентно вне MATRIX.

matrix.

get vars /variables= v1 to v5.

!KO\_recode1(vars%1%{2,4,5}%{-2,-4,-5,999}%vars).

print vars.

end matrix.

recode v1 to v5 (2=-2) (4=-4) (5=-5) (else=999).

list v1 to v5.

ПРИМЕР. Перекодировка в MATRIX и эквивалентно вне MATRIX.

matrix.

get vars /variables= v1 to v5.

get filt /variable= filter.

!KO\_recode1(vars%filt%{2,4,5}%{-2,-4,-5}%vars).

print vars.

end matrix.

do if filter<>0.

-recode v1 to v5 (2=-2) (4=-4) (5=-5).

end if.

list v1 to v5.

### ПЕРЕКОДИРОВКА ЗНАЧЕНИЙ (ПОПАДАНИЕ В ДИАПАЗОН) [!KO\_recode2]

\*/\*!KO\_recode2(data%filter%low%high%new%name)\*/\*.

\*Version 1.

\*В матрице данных DATA любого размера заменяет значения, попадающие в диапазон от LOW до HIGH (включительно),

\*на соответствующие значения списка NEW.

\*Выдает матрицу NAME, являющуюся матрицей DATA с перекодированными значениями.

\*Эта функция подобна внематричной команде RECODE, в которой значения для перекодировки заданы

\*диапазоном(ами) от... до....

\*LOW, HIGH и NEW - векторы-ряды. LOW(i) и HIGH(i) есть диапазон; HIGH(i) в норме должно быть >= LOW(i).

\*Всякое значение DATA, лежащее в диапазоне от LOW(i) до HIGH(i), будет

\*перекодировано в значение NEW(i). LOW и HIGH должны быть одинаковой длины. NEW может быть на один элемент

\*длиннее; в последнем случае все прочие, кроме перекодированных, значения данных будут перекодированы в

\*это последнее значение NEW.

\*Если некоторые диапазоны, указанные векторами LOW и HIGH, взаимонакладываются, их общие значения

\*перекодируется в NEW-значение, соответствующее самому раннему (левому в векторах) из этих диапазонов.

\*Например, если LOW есть {6,5} и HIGH есть {11,10}, то есть заданы пересекающиеся диапазоны 6-11 или 5-10,

\*NEW же есть {23,37}, то "спорные" значения данных - от 6 до 10 - будут перекодированы в 23, а не в 37,

\*т.к. диапазон 6-11 задан "раньше" (левее в векторах), чем 5-10. (Так же ведет себя вне-матричная команда

\*RECODE.)

\*Аргумент FILTER позволяет запретить перекодировать значения некоторых рядов DATA. FILTER либо скаляр,

\*либо столбец длиной как число рядов DATA. Если ненулевой скаляр - никакого запрета нет; если нулевой

\*скаляр - то полный запрет (функция не сработает, NAME будет равна DATA). Если столбец, то 0 в нем запрещает

\*перекодировку в данном ряду DATA, а иное значение - разрешает.

\*Если векторы LOW и HIGH идентичны, эта функция даст тот же результат, что /\*!KO\_recode1\*/.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Перекодировка в MATRIX и эквивалентно вне MATRIX.

matrix.

get vars /variables= v1 to v5.

get filt /variable= filter.

!KO\_recode2(vars%filt%{2,4,5}%{3,4.999,5}%{2.5,4.5,5}%newvars).

print newvars.

end matrix.

do if filter<>0.

-recode v1 to v5 (2 thru 3 =2.5) (4 thru 4.999 =4.5) (5=5).

end if.

list v1 to v5.

### РАЗНЫЕ ОДНОПЕРЕМЕННЫЕ СТАТИСТИКИ, ПО ГРУППАМ: ЗАМЕНА ИМИ ИСХОДНЫХ ЗНАЧЕНИЙ [!KO\_aggrv]

\*/\*!KO\_aggrv(data%bin%stat%check%value1%value2%name)\*/\*.

\*Version 1.

\*Эта функция вычисляет требуемую описательную статистику по группам - точно так, как функция /\*!KO\_aggr\*/,

\*но в качестве результата NAME выдает не таблицу группы X переменные, а массив размером с DATA, в котором

\*исходные значения DATA заменены на соответствующие сводные статистики: если ряд (наблюдение)

\*принадлежит группе i и только ей, то его значения в NAME это есть ряд i в таблице, выдаваемой

\*функцией /\*!KO\_aggr\*/. Таким образом, /\*!KO\_aggrv()\*/ делает то же, что SPSS команда AGGREGATE под опцией

\*MODE=ADDVARIABLES.

\*Аргументы DATA, BIN, STAT - точно те же, что в функции /\*!KO\_aggr\*/ - см.

\*Аргумент CHECK (цифра 0 или 1, не имя и не выражение):

\*Если вы знаете, что суммы во всех рядах BIN равны 1, т.е. что каждое наблюдение принадлежит ровно одной

\*группе, так что BIN это фиктивные (dummy) переменные, установите CHECK на 0. Если же вы допускаете, что

\*суммы в некоторых рядах BIN могут быть не равны 1, установите CHECK на 1.

\*Аргументы VALUE1 и VALUE2 игнорируются, если СHECK есть 0.

\*VALUE1 (число, не имя и не выражение): задайте значение, котороое вставить в NAME тем наблюдениям, которые

\*не принадлежат ни одной группе (т.е. у которых сумма в этом ряду BIN =0).

\*VALUE2: задайте значение (число, не имя и не выражение), котороое вставить в NAME тем наблюдениям, которые

\*принадлежат более чем одной группе (т.е. у которых сумма в этом ряду BIN >1); либо укажите заглавными

\*буквами ключевое слово "AVER" - что значит требование у наблюдения, принадлежащего более чем одной

\*группе и значит располагающего более чем одним посчитанным значением статистики, усреднить эти его

\*значения статистики.

\*Значения аргументов CHECK, VALUE1, VALUE2 можно опционально взять в кавычки или апострофы.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если ваше группирование задается категориальной переменной, создайте из нее фиктивные двоичные

\*переменные BIN с помощью функции /\*!KO\_freq\*/.

ПРИМЕР. Заменить индивидуальные данные групповыми средними.

matrix.

get vars /vari= v1 v2 /names= names.

print data /cnames= names /format= f8.4.

get grvar /variable= group. /\*Categorical grouping variable

!KO\_freq(grvar%1%dummy%freq%codes). /\*Create dummy variables

print codes.

print dummy /title 'Group membership'.

!KO\_aggr(vars%dummy%MEAN%table).

print table /title 'Group means' /cnames= names /format= f8.4.

!KO\_aggrv(vars%dummy%MEAN%0%-999%9999%addvars).

print addvars /title 'Individual values replaced with corresponding group means'

/cnames= names /format= f8.4.

end matrix.

ПРИМЕР. То же, но группы могут пересекаться составом.

matrix.

get vars /vari= v1 v2 /names= names.

print data /cnames= names /format= f8.4.

compute bin= rnd(uniform(nrow(vars),3)). /\*Binary grouping variables

print bin /title 'Group memberships'.

!KO\_aggr(vars%bin%MEAN%table).

print table /title 'Group means' /cnames= names /format= f8.4.

!KO\_aggrv(vars%bin%MEAN%1%-999%AVER%addvars).

print addvars /title 'Individual values replaced with corresponding group means'

/cnames= names /format= f8.4.

print title 'If a case belongs to >1 groups the group means were averaged' /space= 0.

print title 'If a case belongs to no groups it got value -999' /space= 0.

end matrix.

# ФУНКЦИИ ПОИСКА/ПОМЕТКИ

### ПОЗИЦИИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ В ВЕКТОРЕ [!KO\_indices]

\*/\*!KO\_indices(vec%name)\*/\*.

\*Version 2.

\*Берет вектор (ряд или столбец) VEC и выдает вектор-ряд NAME, содержащий список индексов

\*(порядковых номеров, позиций) ненулевых элементов в VEC. Индексы идут по возрастающей.

\*Если ненулевых элементов в VEC нет, ф-ция выдает скаляр 0.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Какой-то анализ раздельно по группам индивидов (без сортировки).

set mxloops 10000. /\*Some functions might need many cycles.

matrix.

get vars /variables= v1 v2 /names= names. /\*Get some variables, memorize names

get group /variables= gr. /\*Get the grouping variable (also numeric)

!KO\_freq(group%1%dummies%freq%codes). /\*Get to know group codes and group sizes

release dummies. /\*(This we won't need in this example)

print {codes,freq} /clab 'Codes' 'Freq'.

loop i= 1 to nrow(codes). /\*For each group

-do if freq(i)>=3. /\*with at least 3 individuals

- !KO\_indices(group=codes(i)%casenums). /\*get the indices (case numbers list),

- compute data= vars(casenums,:). /\*extract the group and

- !KO\_kurtosis(data%kurt). /\*perform any analysis for it

/\*(in this example, compute kurtosis for our variables)

- print codes(i) /title 'Group code'.

- print kurt /cnames= names /title 'Kurtosis in the group' /space= 0.

-end if.

end loop.

end matrix.

### ПОЗИЦИИ НЕНУЛЕВЫХ И НУЛЕВЫХ ЭЛЕМЕНТОВ В ВЕКТОРЕ [!KO\_indices2]

\*/\*!KO\_indices2(vec%name1%name2)\*/\*.

\*Version 1.

\*Берет вектор (ряд или столбец) VEC и выдает вектор-ряд NAME1, содержащий список индексов

\*(порядковых номеров, позиций) ненулевых элементов в VEC, и вектор-ряд NAME2, содержащий список

\*индексов нулевых элементов в VEC. Индексы идут по возрастающей.

\*Если ненулевых элементов в VEC нет, NAME1 будет скаляр 0. Если нулевых элементов в VEC нет,

\*NAME2 будет скаляр 0.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ПОЗИЦИИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ В РЯДАХ ИЛИ В СТОЛБЦАХ МАТРИЦЫ

Если вы хотите выписать индексы ненулевых элементов в множестве векторов, т.е. в матрице, вы можете использовать /\*!KO\_indices\*/ для каждого ее ряда или столбца. Но выгоднее с точки зрения скорости – использвать для этой цели /\*!KO\_ram\*/ (выписать индексы ненулевых элементов из рядов матрицы) или /\*!KO\_vram\*/ (выписать индексы ненулевых элементов из столбцов матрицы). Или используйте функцию /\*!KO\_indicesm\*/.

### ПОЗИЦИИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ В МАТРИЦЕ (И САМИ ЭЛЕМЕНТЫ) [!KO\_indicesm]

\*/\*!KO\_indicesm(mat%ANDVAL%name)\*/\*.

\*Version 1.

\*Берет матрицу любого размера MAT и выдает двухстолбцовую матрицу NAME, содержащую список индексов

\*(порядковых номеров, позиций) ненулевых элементов в MAT. Индексы идут по возрастающей.

\*Номер ряда записан в 1-м столбце NAME, а номер столбца - во 2-м столбце NAME.

\*Аргумент ANDVAL позволяет выписать и сами ненулевые значения, в качестве 3-го столбца NAME.

\*ANDVAL - цифра (не имя и не выражение) 0 или 1 (цифру можно взять в кавычки или апострофы).

\*Если 1, то будет выписка значений.

\*Если ненулевых элементов в MAT нет, ф-ция выдает скаляр 0.

\*Эта функция работает немного быстрее на широких, чем на высоких матрицах.

\*Эта функция близка к /\*!KO\_nzlist\*/, но выдает результат немного в другом виде.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 10000.

matrix.

compute mat= uniform(15,8).

compute mat= mat&\*(mat>.7).

print mat.

!KO\_indicesm(mat%1%name).

print name /clabels= 'row' 'col' 'val'.

end matrix.

### ПОЗИЦИЯ ПЕРВОЙ ВСТРЕЧИ ЗНАЧЕНИЯ В РЯДУ/РЯДАХ [!KO\_indx]

\*/\*!KO\_indx(mat%val%name)\*/\*.

\*Version 1.

\*Берет MAT - матрицу либо вектор-ряд. Выдает индекс (позицию) первой (самой левой)

\*встречи значения VAL в каждом ряду MAT.

\*VAL может быть скаляром или вектором-столбцом длиной как число рядов MAT; в последнем случае

\*каждому ряду MAT отвечает свое значение VAL.

\*Результат NAME - вектор-столбец с числом рядов как в MAT.

\*Позиция 0 означает, что VAL в ряду не найдено.

### ПОЗИЦИЯ ПОСЛЕДНЕЙ ВСТРЕЧИ ЗНАЧЕНИЯ В РЯДУ/РЯДАХ [!KO\_rindx]

\*/\*!KO\_rindx(mat%val%name)\*/\*.

\*Version 1.

\*Берет MAT - матрицу либо вектор-ряд. Выдает индекс (позицию) последней (самой правой)

\*встречи значения VAL в каждом ряду MAT.

\*VAL может быть скаляром или вектором-столбцом длиной как число рядов MAT; в последнем случае

\*каждому ряду MAT отвечает свое значение VAL.

\*Результат NAME - вектор-столбец с числом рядов как в MAT.

\*Позиция 0 означает, что VAL в ряду не найдено.

### ПОЗИЦИЯ СЛУЧАЙНОЙ ВСТРЕЧИ ЗНАЧЕНИЯ В РЯДУ/РЯДАХ [!KO\_randindx]

\*/\*!KO\_randindx(mat%val%name)\*/\*.

\*Version 1.

\*Берет MAT - матрицу либо вектор-ряд. Выдает индекс (позицию) случайной встречи значения VAL

\*в каждом ряду MAT. Т.е. если VAL встречается в ряду k раз, одна из этих позиций будет выбрана

\*случайно.

\*VAL может быть скаляром или вектором-столбцом длиной как число рядов MAT; в последнем случае

\*каждому ряду MAT отвечает свое значение VAL.

\*Результат NAME - вектор-столбец с числом рядов как в MAT.

\*Позиция 0 означает, что VAL в ряду не найдено.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

### ПОЗИЦИЯ (ПАРА ИНДЕКСОВ) ОДНОГО ЗНАЧЕНИЯ В МАТРИЦЕ [!KO\_ij]

\*/\*!KO\_ij(mat%val%priority%name)\*/\*.

\*Version 2.

\*Находит в матрице MAT один элемент, равный значению VAL (скаляр).

\*Выдает в качестве NAME позицию (индексы i,j) такого элемента в MAT.

\*Если элементов, равных VAL, найдено более одного, то выдается позиция одного из них согласно

\*аргументу PRIORITY. Если не найдено ни одного, выдается {0,0}.

\*PRIORITY - кл. слово заглавными буквами (можно взять в кавычки или апострофы):

\*"TBLR" - поиск сверху вниз, слева направо: выдается позиция в наиболее верхнем ряду

\*наиболее левого элемента.

\*"LRTB" - поиск слева направо, сверху вниз: выдается позиция в наиболее левом столбце

\*наиболее верхнего элемента.

\*"BTRL" - поиск снизу вверх, справа налево: выдается позиция в наиболее нижнем ряду

\*наиболее правого элемента.

\*"RLBT" - поиск справа налево, снизу вверх: выдается позиция в наиболее правом столбце

\*наиболее нижнего элемента.

\*"RANDOM" - из всех позиций VAL выдать одну по случайному выбору.

\*Есть крохотная разница в скорости исполнения; если позиционная приоритетность сама не

\*важна вам, но вы жадны на предмет скорости, выберите RLBT как самое быстрое.

ПРИМЕР.

matrix.

compute x= {0,0,1,0,0;1,0,0,0,0;0,0,0,0,0;0,0,0,1,0;0,0,0,0,0;0,1,0,0,0}.

print x.

!KO\_ij(x%1%TBLR%ij).

print ij.

!KO\_ij(x%1%LRTB%ij).

print ij.

!KO\_ij(x%1%BTRL%ij).

print ij.

!KO\_ij(x%1%RLBT%ij).

print ij.

!KO\_ij(x%1%RANDOM%ij).

print ij.

end matrix.

### ПОМЕТКА В РЯДУ И СТОЛБЦЕ НЕ БОЛЕЕ ОДНОГО ЭЛЕМЕНТА, РАВНОГО ЗНАЧЕНИЮ [!KO\_prime]

\*/\*!KO\_prime(mat%val%name)\*/\*.

\*Version 1.

\*Берет матрицу или вектор MAT и значение VAL.

\*Выдает двоичную матрицу NAME размером с MAT, в которой единицей помечены элементы, равные VAL, но функция

\*оставляет в каждом ряду и столбце не более одной такой пометки. Другими словами, единица в NAME

\*эксклюзивно спаривает ряды и столбцы: "один ряд - один столбец".

\*Это не алгоритм оптимального спаривания: результат может зависеть от того, в каком направлении вести поиск

\*элементов, равных VAL. Данная функция просматривает ряды сверху вниз, а элементы в ряду справа налево.

\*Если вы хотите другой порядок, просто измените соответственно этому порядок рядов и/или столбцов в MAT.

\*Аргумент VAL может быть не только скаляром, но и вектором-столбцом длиной как число рядов MAT.

\*В последнем случае в каждом ряду MAT ищется и помечается свое значение, а дальше

\*(стирание "лишних" пометок) функция действует так же, как при скаляре.

\*Функция работает несколько быстрее с широкой матрицей, чем с высокой матрицей.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 10000.

matrix.

compute m= rnd(uniform(9,7)\*10).

print m.

!KO\_prime(m%0%flag).

print flag /title 'Elements equal 0 are highlighted, at most one per row and col'.

end matrix.

### ИНДИКАЦИЯ НАБЛЮДЕНИЙ С ЗАДАННЫМ ПРОФИЛЕМ ЗНАЧЕНИЙ [!KO\_indic]

\*/\*!KO\_indic(mat%valvecs%name)\*/\*.

\*Version 1.

\*Данные MAT - n наблюдений x p переменных (n>=1, p>=1).

\*Векторы значений VALVECS - m векторов x p значений (m>=1), 1-е значение в векторе относится к 1-й

\*переменной данных, 2-е значение - ко 2-й переменной, и так далее.

\*Выдает двоичную индикаторную матрицу NAME, n x m, в которой единица на позиции (i,j) означает, что

\*i-е наблюдение (ряд) в MAT совпадает с j-м вектором (рядом) в VALVECS.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

matrix.

compute data=

{3, 2, 1, 0;

3, 1, 0, 3;

2, 2, 2, 4;

0, 1, 3, 0;

1, 1, 1, 1;

1, 0, 3, 2;

2, 3, 2, 2;

3, 2, 2, 2;

1, 1, 2, 1;

2, 4, 2, 4;

3, 3, 2, 2;

0, 2, 4, 4;

4, 2, 3, 2;

3, 0, 0, 1;

4, 2, 3, 1;

1, 1, 1, 1;

1, 3, 2, 3;

3, 3, 1, 2;

0, 4, 4, 1;

1, 3, 3, 2}.

print data.

compute valvecs=

{1,1,1,1;

1,2,3,4;

0,2,4,4;

1,0,3,2;

2,2,2,4}.

print valvecs.

!KO\_indic(data%valvecs%name).

print name.

end matrix.

### ПОМЕТКА ПРЯМЫХ ЦЕПОЧЕК [!KO\_runs]

\*/\*!KO\_runs(mat%maxw%dir%hl%name)\*/\*.

\*Version 2.

\*В двоичной матрице MAT помечает прямые цепочки подряд идущих единиц.

\*Цепочки какой длины каким значением будут помечены, зависит от аргумента

\*"максимальный вес" MAXW (неотрицательный целый скаляр).

\*Если MAXW>0, то цепочка длиной >=MAXW будет помечена числом MAXW,

\*цепочка длиной MAXW-1 будет помечена числом MAXW-1,

\*цепочка длиной MAXW-2 будет помечена числом MAXW-2, и так далее.

\*Если MAXW задано неположительное (любое такое число), то цепочка будет помечена числом,

\*равным ее длине, какой бы длинной она ни оказалась. Иными словами, неположительное

\*MAXW есть отказ от предела "максимальный вес".

\*Аргумент DIR (скаляр) - цепочки какого направления интересуют: вертикальные, в столбцах

\*(положительный аргумент) или горизонтальные, в рядах (неположительный аргумент).

\*Аргумент HL - цифра 1 или 0 (не имя и не выражение, цифру можно взять в кавычки или апострофы).

\*Этот аргумент о том, как помечать. Если HL=1, то все члены цепочки будут помечены одним числом;

\*если HL=0, то они будут помечены порядковыми номерами.

\*Например, если MAXW=0, то цепочка 1 1 1 1 1 будет помечена как 5 5 5 5 5 при HL=1 и

\*как 1 2 3 4 5 при HL=0. А если MAXW=3, то она будет помечена как 3 3 3 3 3 при HL=1 и

\*как 1 2 3 3 3 при HL=0.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ПОМЕТКА ПРЯМЫХ ЦЕПОЧЕК (ДРУГОЙ АЛГОРИТМ) [!KO\_runs2]

\*/\*!KO\_runs2(mat%maxw%dir%name)\*/\*.

\*Version 1.

\*В двоичной матрице MAT помечает прямые цепочки подряд идущих единиц.

\*Цепочки какой длины каким значением будут помечены, зависит от аргумента

\*"максимальный вес" MAXW (положительный скаляр от 2 до

\*число\_рядов или число\_столбцов матрицы - в зависимости от DIR):

\*единицы цепочек длиной >=MAXW обратятся в значение MAXW; единицы цепочек длиной MAXW-1

\*обратятся в значение MAXW-1; единицы цепочек

\*длиной MAXW-2 обратятся в значение MAXW-2, и т.д.

\*Аргумент DIR (скаляр) - цепочки какого направления интересуют: вертикальные, в столбцах

\*(положительный аргумент) или горизонтальные, в рядах (неположительный аргумент).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Ф-ция /\*!KO\_runs\*/ исполняет ту же работу, что и данная ф-ция, но реализована по-другому. /\*!KO\_runs2\*/

\*отличается от /\*!KO\_runs\*/ следующим:

\*- аргумент MAXW должен быть положительным;

\*- аргумент HL отсутствует;

\*- скорость у /\*!KO\_runs2\*/ зависит от MAXW, а у /\*!KO\_runs\*/ - нет.

\*- скорость у /\*!KO\_runs2\*/ немного выше, чем у /\*!KO\_runs\*/, если MAXW не выше 3 и MAT

\*мала; во всех прочих случаях /\*!KO\_runs\*/ быстрей и поэтому она обычно предпочтительней.

### ПОМЕТКА КОСЫХ ЦЕПОЧЕК [!KO\_slant]

\*/\*!KO\_slant(mat%maxw%dir%hl%name)\*/\*.

\*Version 2.

\*В двоичной матрице MAT помечает косые цепочки подряд идущих единиц.

\*Цепочки какой длины каким значением будут помечены, зависит от аргумента

\*"максимальный вес" MAXW (неотрицательный целый скаляр).

\*Если MAXW>0, то цепочка длиной >=MAXW будет помечена числом MAXW,

\*цепочка длиной MAXW-1 будет помечена числом MAXW-1,

\*цепочка длиной MAXW-2 будет помечена числом MAXW-2, и так далее.

\*Если MAXW задано неположительное (любое такое число), то цепочка будет помечена числом,

\*равным ее длине, какой бы длинной она ни оказалась. Иными словами, неположительное

\*MAXW есть отказ от предела "максимальный вес".

\*Аргумент DIR (скаляр) - косые цепочки какого направления интересуют: верх-лево/низ-право

\*(положительный аргумент) или верх-право/низ-лево (неположительный аргумент).

\*Аргумент HL - цифра 1 или 0 (не имя и не выражение, цифру можно взять в кавычки или апострофы).

\*Этот аргумент о том, как помечать. Если HL=1, то все члены цепочки будут помечены одним числом;

\*если HL=0, то они будут помечены порядковыми номерами.

\*Например, если MAXW=0, то цепочка 1 1 1 1 1 будет помечена как 5 5 5 5 5 при HL=1 и

\*как 1 2 3 4 5 при HL=0. А если MAXW=3, то она будет помечена как 3 3 3 3 3 при HL=1 и

\*как 1 2 3 3 3 при HL=0.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Ф-ция /\*!KO\_slant2\*/ исполняет ту же работу, что и данная ф-ция, но реализована по-другому.

ПРИМЕР.

matrix.

compute mat= rnd(uniform(9,11)). /\*random binary matrix

print mat.

!KO\_slant(mat%2%1%1%chains).

print chains /\*All slant chains highlighted as 2.

!KO\_slant(mat%4%1%1%chains).

print chains. /\*Slant chains of length 2 highlighted as 2,

/\*of length 3 highlighted as 3,

/\*of length 4+ highlighted as 4

!KO\_slant(mat%0%1%0%chains).

print chains. /\*Slant chains were highlighted as

/\*1 2 3 … chain\_length

end matrix.

### ПОМЕТКА КОСЫХ ЦЕПОЧЕК (ДРУГОЙ АЛГОРИТМ) [!KO\_slant2]

\*/\*!KO\_slant2(mat%maxw%dir%name)\*/\*.

\*Version 1.

\*В двоичной матрице MAT помечает косые цепочки подряд идущих единиц.

\*Цепочки какой длины каким значением будут помечены, зависит от аргумента

\*"максимальный вес" MAXW (положительный скаляр от 2 до

\*min(число\_рядов,число\_столбцов) матрицы): единицы цепочек длиной >=MAXW обратятся в

\*значение MAXW; единицы цепочек длиной MAXW-1 обратятся в значение MAXW-1; единицы цепочек

\*длиной MAXW-2 обратятся в значение MAXW-2, и т.д.

\*Аргумент DIR (скаляр) - косые цепочки какого направления интересуют: верх-лево/низ-право

\*(положительный аргумент) или верх-право/низ-лево (неположительный аргумент).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Ф-ция /\*!KO\_slant\*/ исполняет ту же работу, что и данная ф-ция, но реализована по-другому. /\*!KO\_slant2\*/

\*отличается от /\*!KO\_slant\*/ следующим:

\*- аргумент MAXW должен быть положительным;

\*- аргумент HL отсутствует;

\*- скорость у /\*!KO\_slant2\*/ зависит от MAXW, а у /\*!KO\_slant\*/ - нет.

\*- скорость у /\*!KO\_slant2\*/ немного выше, чем у /\*!KO\_slant\*/, если MAXW=2, либо если MAXW= 3-4 и MAT

\*мала; во всех же прочих случаях /\*!KO\_slant\*/ быстрей и поэтому она обычно предпочтительней.

### ПОИСК В ГЛУБИНУ В ОДНОДОЛЬНОМ ГРАФЕ [!KO\_dfs]

\*/\*!KO\_dfs(adj%start%stop%forest%out%name)\*/\*.

\*Version 1.

\*Осуществляет обход/поиск связанных вершин в глубину (Depth-first search) в однодольном невзвешенном

\*графе ADJ, стартуя с вершины START. Опциональная вершина STOP - пункт остановки обхода: дойдя до нее,

\*посещение вершин прекращается. START и STOP - скаляры, номера вершин, они должны различаться. Если STOP

\*задавать не хотите, т.е. обход графа должен идти максимально далеко, установите этот аргумент на

\*значение 0.

\*Аргумент OUT - цифра 1 или 0 (не имя и не выражение; цифру можно взять в кавычки или апострофы).

\*Если 0, результат NAME будет содержать только один столбец, показывающий, сверху вниз, порядок

\*посещения вершин. Если 1, будут дополнительные вычисления и NAME будет содержать еще столбцы:

\*2-й столбец показывает родителя данной посещенной (указанной в 1-м столбце) вершины. Под родителем

\*разумеется тут самая ранопосещенная вершина, указывающая на (т.е. имеющая смежником) нашу вершину.

\*3-й столбец показывает расстояние посещенной вершины от вершины START (корневой вершины); это

\*расстояние есть длина кратчайшего пути от START до нашей вершины через уже посещенные вершины

\*(в DFS это расстояние от START до нашей вершины не обязательно глобально минимально).

\*4-й столбец помечает единицей начало новой цепи в посещениях. Цепь это непрерывная последовательность

\*вглубь графа, где каждая следующая посещенная вершина является смежной предыдущей вершине и это ребро

\*алгоритм прошел (учел). Новая цепь начинается, когда дальше двигаться нельзя (т.к. еще непосещенных

\*смежных вершин больше нет); тогда новая цепь начинается с еще не посещенной вершины, лежащей "под"

\*некоторой "развилкой" путей; алгоритм возвращается к развилке, чтобы с этой вершины начать новый,

\*другой путь вглубь. DFS-алгоритм имеет целью максимизировать длину цепи.

\*5-й столбец добавляется, если FOREST=1 (см.). Здесь помечается единицей начало нового дерева.

\*Аргумент FOREST - цифра 1 или 0 (не имя и не выражение; цифру можно взять в кавычки или апострофы).

\*Если 0, алгоритм ограничится обходом вглубь, стартуя только из одной вершины - заданной как START. NAME

\*будет представлять единственное DF-дерево (Depth-first tree), идущее из корня START. Если аргумент 1,

\*то - в случае если дерево из корня START не смогло посетить все вершины графа - берется дополнительная

\*корневая вершина (из еще непосещенных вершин), и дерево еще непосещенных вершин строится из нее.

\*Так повторяется, пока наконец все вершины графа не окажутся посещены. NAME в этом случае будет

\*представлять не одно дерево, а коллекцию деревьев, называемую DF-лесом (Depth-first forest).

\*Начало (вершина-корень) нового дерева в лесу помечается единицей в 5-м столбце NAME (вы должны задать

\*OUT=1, чтобы в NAME было несколько столбцов). Отличие нового дерева от новой цепи то, что дерево

\*начинается из нового начала, и расстояния в новом дереве отмеряются уже от нового начала, тогда как

\*расстояния в новой цепи продолжают отмеряться от старого начала - от корня того дерева, которому

\*принадлежит данная цепь. Т.о., цепи это пути, вложенные в деревья.

\*Аргумент ADJ - входящий граф. Граф может быть любым - ненаправленным или направленным, с циклами или

\*без, с петлями или без, быть деревом или сетью. Он задается списком смежности (adjacency list) ADJ.

\*Каждый ряд ADJ это вершина графа, и ряд содержит номера вершин, которые с ней смежны (при направленном

\*графе это те, на которые она указывает); номера должны быть записаны слева направо утрамбованно.

\*Например, если из 2-й вершины идут ребра к вершинам 1,4 и 5, то 2-й ряд должен выглядеть как

\*{1,4,5,0,...,0}, где пустоты добиты нулями. Порядок номеров может быть любой: {4,5,1,0,...,0},

\*например, тоже корректно. Число столбцов ADJ должно быть достаточно для того, чтобы вместить информацию

\*о всех ребрах графа. Список смежности соответствует квадратной двоичной матрице, в которой единица в

\*ячейке (i,j) обозначают наличие ребра (дуги) из вершины i в вершину j. Список смежности является другим

\*способом представить такую матрицу. Если ваши входящие - двоичная матрица, вы можете получить из нее

\*список смежности функцией /\*!KO\_ram\*/.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. DFS и BFS.

set mxloops 1E6.

matrix.

compute mat=

{0, 0, 0, 1, 0, 1, 0, 1;

1, 1, 0, 0, 0, 0, 0, 1;

0, 0, 1, 0, 0, 0, 0, 1;

1, 0, 0, 0, 0, 0, 1, 0;

1, 0, 0, 0, 0, 1, 0, 0;

0, 1, 0, 0, 1, 0, 1, 0;

0, 0, 0, 0, 0, 0, 0, 0;

0, 0, 0, 1, 0, 0, 1, 0}. /\*A directed graph given by a asymmetric matrix

/\*(1 means the edge is directed from the vertex-row to the vertex-column)

print mat.

!KO\_ram(mat%IND%0%0%Adj%name). /\*Produce corresponding adjacency list

print Adj.

!KO\_dfs(Adj%1%0%'1'%'1'%dfs\_order). /\*Run DFS starting from vertex 1;

/\*forest=1, which means all vertices sooner or later will sure be visited

print dfs\_order. /\*The results: Tree one started with vertex 1 and comprised

/\*seven vertices; Tree two is the remaining vertex 3;

/\*The 1st chain was the 1->4->7 pass into depth, then broke;

/\*The 2nd chain was the 6->2->8 pass into depth, then broke

/\*(and note that the "parent" displayed for vertex 8 is vertex 1 dispite

/\*that the actual, chain order was 2->8, not 1->8);

/\*The 3rd chain was vertex 5, one more child of 6

!KO\_bfs(Adj%1%0%'1'%'1'%bfs\_order). /\*Run BFS starting from vertex 1;

/\*forest=1, which means all vertices sooner or later will sure be visited

print bfs\_order. /\*The results: Tree one started with vertex 1 and comprised

/\*seven vertices; Tree two is the remaining vertex 3;

/\*The 1st fan of siblings was the 4,6,8 pass into breadth, then broke;

/\*The 2nd fan was 7, the only yet unvisited child of vertex 4, then broke;

/\*The 3rd fan was siblings 2,5, children of vertex 6

end matrix.

ПРИМЕР.

set mxloops 1E6.

matrix.

compute Adj=

{4, 0, 0, 0;

1, 3, 7, 8;

1, 2, 3, 4;

6, 0, 0, 0;

7, 0, 0, 0;

2, 3, 0, 0;

7, 0, 0, 0;

4, 7, 0, 0}. /\*Adjacency list

print Adj.

!KO\_dfs(Adj%1%0%0%1%order). /\*Run DFS starting from vertex 1;

/\*forest=0, which means we are interested in specifically one DF tree

print order. /\*The results:

/\*The 1st chain was the 1->4->6->2->3 pass into depth, then broke

/\*(and note that the "parent" displayed for vertex 3 is vertex 6 dispite

/\*that the actual, chain order was 2->3, not 6->3, and the shortest

/\*distance captured from 1 to 3 is thus length 3, not 4);

/\*The 2nd chain was just 7 (child of 2, and the dead end);

/\*The 3rd chain was just 8 (child of 2, and the dead end since vertices

/\*4 and 7 were already visited);

/\*All vertices except 5 (which is nobody’s child) got visited

end matrix.

### ПОИСК В ШИРИНУ В ОДНОДОЛЬНОМ ГРАФЕ [!KO\_bfs]

\*/\*!KO\_bfs(adj%start%stop%forest%out%name)\*/\*.

\*Version 1.

\*Осуществляет обход/поиск связанных вершин в ширину (Breadth-first search) в однодольном невзвешенном

\*графе ADJ, стартуя с вершины START. Опциональная вершина STOP - пункт остановки обхода: дойдя до нее,

\*посещение вершин прекращается. START и STOP - скаляры, номера вершин, они должны различаться. Если STOP

\*задавать не хотите, т.е. обход графа должен идти максимально далеко, установите этот аргумент на

\*значение 0.

\*Аргумент OUT - цифра 1 или 0 (не имя и не выражение; цифру можно взять в кавычки или апострофы).

\*Если 0, результат NAME будет содержать только один столбец, показывающий, сверху вниз, порядок

\*посещения вершин. Если 1, будут дополнительные вычисления и NAME будет содержать еще столбцы:

\*2-й столбец показывает родителя данной посещенной (указанной в 1-м столбце) вершины. Под родителем

\*разумеется тут самая ранопосещенная вершина, указывающая на (т.е. имеющая смежником) нашу вершину.

\*3-й столбец показывает расстояние посещенной вершины от вершины START (корневой вершины); это

\*расстояние есть длина кратчайшего пути от START до нашей вершины через посещенные вершины

\*(в BFS это расстояние от START до нашей вершины непременно глобально минимально).

\*4-й столбец помечает единицей начало нового веера в посещениях. Веер это непрерывнная

\*последовательность вширь графа, т.е. перечисление сибсов (детей одного ранее посещенного родителя).

\*Новый веер начинается, когда сибсов больше нет. BFS-алгоритм имеет целью максимизировать размер веера.

\*5-й столбец добавляется, если FOREST=1 (см.). Здесь помечается единицей начало нового дерева.

\*Аргумент FOREST - цифра 1 или 0 (не имя и не выражение; цифру можно взять в кавычки или апострофы).

\*Если 0, алгоритм ограничится обходом вширь, стартуя только из одной вершины - заданной как START. NAME

\*будет представлять единственное BF-дерево (Breadth-first tree), идущее из корня START. Если аргумент 1,

\*то - в случае если дерево из корня START не смогло посетить все вершины графа - берется дополнительная

\*корневая вершина (из еще непосещенных вершин), и дерево еще непосещенных вершин строится из нее.

\*Так повторяется, пока наконец все вершины графа не окажутся посещены. NAME в этом случае будет

\*представлять не одно дерево, а коллекцию деревьев, называемую BF-лесом (Breadth-first forest).

\*Начало (вершина-корень) нового дерева в лесу помечается единицей в 5-м столбце NAME (вы должны задать

\*OUT=1, чтобы в NAME было несколько столбцов). Отличие нового дерева от нового веера то, что дерево

\*начинается из нового начала, и расстояния в новом дереве отмеряются уже от нового начала, тогда как

\*расстояния в новом веере продолжают отмеряться от старого начала - от корня того дерева, которому

\*принадлежит данный веер. Т.о., веера это пути, вложенные в деревья.

\*Аргумент ADJ - входящий граф. Граф может быть любым - ненаправленным или направленным, с циклами или

\*без, с петлями или без, быть деревом или сетью. Он задается списком смежности (adjacency list) ADJ.

\*Каждый ряд ADJ это вершина графа, и ряд содержит номера вершин, которые с ней смежны (при направленном

\*графе это те, на которые она указывает); номера должны быть записаны слева направо утрамбованно.

\*Например, если из 2-й вершины идут ребра к вершинам 1,4 и 5, то 2-й ряд должен выглядеть как

\*{1,4,5,0,...,0}, где пустоты добиты нулями. Порядок номеров может быть любой: {4,5,1,0,0,0,0},

\*например, тоже корректно. Число столбцов ADJ должно быть достаточно для того, чтобы вместить информацию

\*о всех ребрах графа. Список смежности соответствует квадратной двоичной матрице, в которой единица в

\*ячейке (i,j) обозначают наличие ребра (дуги) из вершины i в вершину j. Список смежности является другим

\*способом представить такую матрицу. Если ваши входящие - двоичная матрица, вы можете получить из нее

\*список смежности функцией /\*!KO\_ram\*/.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. См. прямо выше.

### АЛГОРИТМ МАКСИМАЛЬНЫХ КЛИК БРОНА-КЕРБОША (PIVOT-ВЕРСИЯ) [!KO\_bronkerb]

\*/\*!KO\_bronkerb(adj%return%name)\*/\*.

\*Version 1.

\*Эта функция исполняет алгоритм Брона-Кербоша для перечисления максимальных клик в ее pivot-версии.

\*(Эквивалентную, но более медленную базовую версию алгоритма - см. функцию /\*!KO\_bronkerb2\*/.)

\*Входящая матрица ADJ - это ненаправленный невзвешенный граф в виде "списка смежности"

\*(adjacency list). Каждый ряд ADJ это вершина графа, и ряд содержит номера вершин, с которыми эта

\*вершина смежна (связана). Номера идут слева направо утрамбованно. Например, пусть вершина под

\*номером 2 связана с вершинами 1, 4, 5. Тогда 2-й ряд ADJ должен выглядеть так: {1,4,5[,0...]},

\*где [,0...] - это дозаполнение ряда нулями, если необходимо. Число столбцов в ADJ может быть любым

\*достаточным для того, чтобы разместить список всех смежников в графе. Порядок номеров вершин в ряду

\*произволен: например, можно написать тот же ряд как {5,1,4[,0...]}. Вершина в графе не может

\*быть смежна себе (т.е., напр. во 2-м ряду не может присутствовать вершина 2). Изолированные вершины

\*в графе не допускаются (т.е. ряд не может состоять из нулей - но см. ПРИМЕР). Список смежности ADJ

\*соответствует квадратной симметричной двоичной матрице смежности с нулевой диагональю, в которой

\*единицы обозначают наличие связи, ADJ является другим способом ее представить. Если ваши входящие -

\*такая двоичная матрица, то вы можете получить из нее список смежности функцией /\*!KO\_ram\*/.

\*Результат NAME - двухстолбцовая матрица. Ее первый столбец это номера вершин, входящих в максимальные

\*клики. Ее 2-й столбец это идентификатор клик - их номера в порядке их обнаружения алгоритмом.

\*Аргумент RETURN - кл. слово заглавными буквами (можно также в кавычках или апострофах):

\*"ALL" - позволить алгоритму выполниться до конца; NAME будет перечислять все максимальные клики,

\*существующие в графе;

\*"COUNT n" - выдать n найденных первыми максимальных клик и остановиться; n это целое >0, может быть

\*цифрой, именем или выражением (примеры заданий: COUNT 10, COUNT x, COUNT abs(z)\*2);

\*"SIZEEQ n" - выдать одну максимальную клику, состоящую из n вершини, и остановиться; n это целое >1,

\*может быть цифрой, именем или выражением;

\*"SIZEGE n" - выдать одну максимальную клику, состоящую из >=n вершин, и остановиться; n это целое >1,

\*может быть цифрой, именем или выражением.

\*В двух последних случаях - если клик затребованного размера не нашлось, результат будет {0,0}.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1E9.

matrix.

compute Adj=

{2,9,0,0,0;

1,3,0,0,0;

2,9,4,8,0;

3,5,6,7,8;

4,6,0,0,0;

4,5,7,8,0;

4,6,8,0,0;

3,4,6,7,0;

1,2,3,0,0}. /\*Adjacency list

print Adj.

!KO\_bronkerb(adj%ALL%name).

print name /title 'All maximal cliques'.

!KO\_bronkerb(adj%SIZEGE 4%name).

print name /title 'One maximal clique of size>=4'.

end matrix.

ПРИМЕР.

set mxloops 1E9.

matrix.

compute mat=

{0, 1, 1, 1, 1, 0, 0, 0;

1, 0, 0, 0, 0, 0, 0, 1;

1, 0, 0, 0, 1, 0, 1, 1;

1, 0, 0, 0, 0, 1, 0, 0;

1, 0, 1, 0, 0, 1, 1, 0;

0, 0, 0, 1, 1, 0, 1, 0;

0, 0, 1, 0, 1, 1, 0, 0;

0, 1, 1, 0, 0, 0, 0, 0}. /\*A symmetric adjacency matrix w/ zero diagonal

/\*and no empty rows/columns

print mat.

!KO\_ram(mat%IND%0%0%Adj%name). /\*Produce corresponding adjacency list

print Adj.

!KO\_bronkerb(adj%ALL%name).

print name /title 'All maximal cliques'.

end matrix.

ПРИМЕР. Когда в графе есть изолированные вершины, их можно удалить, но удобнее приписать им в соседи несуществующую вершину (некий номер выше числа вершин в графе).

set mxloops 1E9.

matrix.

compute Adj=

{2,0,0,0,0;

1,3,0,0,0;

2,4,8,0,0;

3,6,7,8,0;

0,0,0,0,0;

4,7,8,0,0;

4,6,8,0,0;

3,4,6,7,0;

0,0,0,0,0}. /\*Graph with 2 isolated vertices, 5th and 9th

print Adj.

compute Adj(5,1)= 99. /\*Assign them a nonexistent vertex

compute Adj(9,1)= 99. /\*as the only neighbour

print Adj.

!KO\_bronkerb(adj%ALL%name).

print name. /\*Verices 5 and 9 came out as "cliques" of their own

end matrix.

### АЛГОРИТМ МАКСИМАЛЬНЫХ КЛИК БРОНА-КЕРБОША (БАЗОВАЯ ВЕРСИЯ) [!KO\_bronkerb2]

\*/\*!KO\_bronkerb2(adj%return%name)\*/\*.

\*Version 1.

\*Эта функция исполняет алгоритм Брона-Кербоша для перечисления максимальных клик в ее базовой версии

\*и она эквивалентна функции /\*!KO\_bronkerb\*/, но медленнее ее, поэтому не предпочтительна - кроме

\*случаев очень разреженного графа, когда она бывает немного быстрее. Аргументы - те же,

\*что /\*!KO\_bronkerb\*/. Порядок нахождения клик и порядок вершин внутри клик, т.е. порядок рядов

\*в результате NAME - обычно будет другой, чем при использовании /\*!KO\_bronkerb\*/, но список и состав

\*клик при RETURN=ALL всегда такой же. Порядок вершин в кликах тут всегда - по возрастанию их номеров.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ВЫПУКЛАЯ ОБОЛОЧКА: 2D АЛГОРИТМ ДЖАРВИСА [!KO\_chjarv]

\*/\*!KO\_chjarv(data%name)\*/\*.

\*Version 1.

\*Очерчивает выпуклую оболочку (выпуклый многоугольник из всех периферических точек) у облака точек DATA.

\*Базовый алгоритм "заворачивания подарка", называемый также алгоритмом Джарвиса, рассчитан на

\*2-мерный случай. DATA - n x 2 матрица данных, координат n точек (n>=3) на плоскости.

\*Результат NAME - список номеров точек, составляющих выпуклую оболочку. Точки в NAME

\*идут против часовой стрелки в данных.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Для очерчивания выпуклой оболочки вы можете использовать также функцию /\*!KO\_chgrah\*/.

ПРИМЕР. См. прямо ниже.

### ВЫПУКЛАЯ ОБОЛОЧКА: 2D АЛГОРИТМ ГРЭМА [!KO\_chgrah]

\*/\*!KO\_chgrah(data%name)\*/\*.

\*Version 1.

\*Очерчивает выпуклую оболочку (выпуклый многоугольник из всех периферических точек) у облака точек DATA.

\*Алгоритм Грэма (Грэхема) рассчитан на 2-мерный случай.

\*DATA - n x 2 матрица данных, координат n точек (n>=3) на плоскости. В данных должно быть не менее трех

\*несовпадающих точек.

\*1-й столбец NAME - список номеров точек, составляющих выпуклую оболочку. Если в оболочке есть

\*совпадающие точки, то только один экземпляр из них показывается в NAME; однако 2-й столбец NAME

\*осведомляет о числе экземпляров, в каком существует каждая из точек. Точки в NAME идут по часовой

\*стрелке в данных.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Для очерчивания выпуклой оболочки вы можете использовать также функцию /\*!KO\_chjarv\*/. Эти две

\*функции эквивалентны по результату, хотя алгоритмы разные. /\*!KO\_chjarv\*/ показывает все точки оболочки,

\*включая совпадающие, в одном столбце результата.

ПРИМЕР.

set mxloops 1E9.

matrix.

!KO\_normal(35%2%data). /\*Some points 2D coordinates

\*compute data= rnd(data\*3). /\*[If you want data to be discrete, with some

/\*points potentially coinciding (tied) and/or some points collinear\*]

!KO\_chjarv(data%jarv). /\*Run Jarvis algorithm

print jarv. /\*Result: points of the convex hull

!KO\_chgrah(data%grah). /\*Run Graham algorithm

print grah. /\*Result: points of the convex hull

save data /out= \*.

end matrix.

GRAPH /SCATTERPLOT(BIVAR)= COL1 WITH COL2. /\*You may check yourself that the results

/\*are indeed the polygon outlining the data cloud (activate data point labels)

\*Коллинеарные (collinear) точки – точки, лежащие ровно на прямой линии.

### ВЕЛИЧИНЫ ОТСТОЯНИЙ ЭЛЕМЕНТОВ ОТ ДИАГОНАЛИ МАТРИЦЫ [!KO\_diagoff]

\*/\*!KO\_diagoff(nr%nc%diag%name)\*/\*.

\*Version 1.

\*Данная функция помечает нулем диагональ матрицы размером NR x NC, а остальные

\*элементы помечает величиной отстояния от диагонали. Пометки под диагональю имеют (для удобства)

\*отрицательный знак.

\*Когда матрица неквадратная, можно говорить, что у нее более одной диагонали.

\*Аргумент DIAG (слово заглавными буквами, опционально в кавычках/апострофах) позволяет

\*выбрать, какую диагональ помечать нулем и отсчитывать от нее отстояния:

\*"FIRST" - первая, левая или главная диагональ (начинается в верхнелевом углу матрицы)

\*"LAST" - последняя или правая диагональ (кончается в нижнеправом углу матрицы)

\*"MIDDLE" - срединная диагональ, она находится посредине между первой и последней;

\*в том случае если abs(NR-NC) есть число нечетное, то срединных диагоналей - две.

\*Если матрица квадратна (NR=NC), то диагональ единственная, тогда не важно, как задать DIAG.

### ВЕЛИЧИНЫ ОТСТОЯНИЙ ЭЛЕМЕНТОВ ОТ ПОЛОСЫ ДИАГОНАЛЕЙ МАТРИЦЫ [!KO\_diagboff]

\*/\*!KO\_diagboff(nr%nc%name)\*/\*.

\*Version 1.

\*Когда матрица неквадратная, можно говорить, что у нее более одной диагонали.

\*Первая диагональ это главная или левая диагональ: начинается их верхнелевого угла.

\*Последняя диагональ оканчивается в нижнеправом углу. Всего диагоналей abs(NR-NC)+1 штук,

\*они образуют полосу диагоналей.

\*Данная функция помечает нулем всю полосу диагоналей матрицы размером NR x NC, а остальные

\*элементы помечает величиной отстояния от этой полосы. Пометки под полосой имеют (для удобства)

\*отрицательный знак.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

# ФУНКЦИИ ПЕРЕСТРУКТУРИРОВАНИЯ

### СЧИТКА ВСТРАИВАЕМЫХ ДАННЫХ [!KO\_read]

\*/\*!KO\_read(nr nc name) data\*/\*.

\*Version 2.

\*Эта функция имеет особую форму задания. Она читает предложенные вами в синтаксисе значения

\*и конструирует из них матрицу NAME. Используйте ее для создания небольших матриц.

\*Функция эквивалентна явному созданию матрицы из предложенных элементов при помощи

\*операторов {,;}. Ее удобство как раз в том, что она не требует написания вами этих

\*операторов (на что уходит человеческое время). Таким образом, вы можете

\*просто подать массив значений, разделенных лишь пробелами, и функция создаст из них матрицу.

\*Аргументы (заметьте, аргументы в скобках идут через пробел, не через %):

\*NR - число рядов в создаваемой матрице (положительное целое: число или имя, не выражение)

\*NC - число столбцов в создаваемой матрице (положительное целое: число или имя, не выражение).

\*Вы вправе также задать один из этих двух аргументов символом ?. Тогда функция сама

\*вычислит его, разделив число вводимых значений на величину другого, заданного аргумента

\*(деление должно быть без остатка).

\*Сами встраиваемые (inline) данные DATA должны следовать после скобок. Не ставьте точку между

\*скобкой и DATA. Точку надо поставить после DATA.

\*Данные это массив значений через пробел(ы) или табуляцию. Не важно, на сколько рядов и столбцов

\*вы разбили массив значений: порядок создаваемой матрицы определяется аргументами NR, NC. Функция

\*читает значения как свободный (неструктурированный) текст, слева направо и строки сверху вниз.

\*Число значений должно равняться произведению NRxNC.

\*Чем может являться считываемое значение?:

\*-положительным, нулевым или отрицательным числом (примеры: 2.5 -3 +4 -.001 0.15);

\*-дробью или делением (примеры: 1/3 -3/16 3/-16);

\*-именем переменной (например: x);

\*-выражением в кавычках или апострофах (примеры: 'x\*2' "2/3" '-ln(4)').

\* Считывая значения, функция снимает с них кавычки/апострофы (используйте функцию /\*!KO\_read2()\*/,

\* если желаете сохранить кавычки/апострофы, т.е. считывать значения как текстовые).

\*Не применяйте данную функцию для считки большого массива (тысячи элементов), т.к.

\*она медленная (быстрее будет взять данные из синтаксиса командой DATA LIST и затем открыть

\*их в MATRIX).

\*Данная функция удобна для рутинной работы, но не для программирования алгоритмов.

ПРИМЕР.

matrix.

!KO\_read(6 4 mx)

.63042603368 -.3811789 -.37968575 -.3978886

.31876472299 .2149328 -.4382288945 .237998

.267827629 .51122279 .1124206621 -.27705817

-.41313506 .042735859 .699697012798 -.031

.7608927337 .1262283 -.18491719 .622383

.72269674851 -.2762052 .10438647 -.219443.

print mx.

end matrix.

ПРИМЕР.

matrix.

compute nr= 6.

compute x= 10.

!KO\_read(nr ? mx)

.63042603368 x -.37968575 -1/3

.31876472299 .2149328 -.4382288945 .237998

.267827629 .51122279 .1124206621 -.27705817

-.41313506 .042735859 x -.031

.7608927337 .1262283 -.18491719 -x

.72269674851 -.2762052 .10438647 "-.219443\*x".

print mx.

end matrix.

### СЧИТКА ВСТРАИВАЕМЫХ ДАННЫХ (БЕЗ РАСКАВЫЧКИ) [!KO\_read2]

\*/\*!KO\_read2(nr nc name) data\*/\*.

\*Version 2.

\*Эта функция тождественна /\*!KO\_read()\*/ с единственным отличием, что она не снимает

\*кавычки/апострофы со считываемого значения. Поэтому если значение взято в кавычки или апострофы,

\*то это значение берется как текстовой элемент в матрицу NAME.

### РАЗРЕЖЕННУЮ МАТРИЦУ В СПИСОК НЕНУЛЕВЫХ ЗНАЧЕНИЙ [!KO\_nzlist]

\*/\*!KO\_nzlist(mat%name1%name2)\*/\*.

\*Version 1.

\*Берет матрицу (или вектор) MAT любого размера. Выписывает из MAT все ненулевые элементы в один список.

\*Этот список есть 2-й столбец NAME2, а в 1-м столбце NAME2 записаны номера столбцов MAT, в которых

\*находились выписанные значения. NAME2 имеет столько рядов, сколько ненулевых элементов в MAT.

\*Вектор-столбец NAME1 имеет длину как число рядов MAT, в нем записано, сколько ненулевых элементов

\*в каждом ряду MAT.

\*Если в MAT нет ненулевых элементов, функция выдает NAME1 и NAME2 как нулевые скаляры.

\*Если входящую MAT транспонировать, результаты будут другие, но информационно эквивалентные.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Функция /\*!KO\_nzunlist\*/ противоположна данной и может восстановить MAT из NAME1 и NAME2.

ПРИМЕР. См. прямо ниже.

### СПИСОК В РАЗРЕЖЕННУЮ МАТРИЦУ [!KO\_nzunlist]

\*/\*!KO\_nzunlist(nnz%list%name)\*/\*.

\*Version 1.

\*Создает (разреженную) матрицу, вписывая в пустую матрицу значения из списка.

\*Берет ненулевой вектор-столбец NNZ, значения которого должны быть целыми неотрицательными.

\*Эти значения - количества вписываемых элементов в каждый ряд создаваемой матрицы NAME (NAME будет иметь столько

\*рядов, сколько рядов в NNZ).

\*Значения, вписываемые в матрицу NAME, функция берет из 2-го столбца двухстолбцовой матрицы LIST,

\*а информацию, в какой столбец NAME следует поместить данное вписываемое значение - берет из соответственнго

\*элемента 1-го столбца LIST. (Значения в 1-м столбце LIST должны быть все целые положительные. Сами вписываемые

\*значения - 2-й столбец LIST - могут быть любые.)

\*В 1-й ряд NAME будет вписано первые NNZ(1) значений из LIST, во 2-й ряд NAME будет вписано NNZ(2) следующих

\*значений из LIST, и так далее.

\*Создаваемая матрица NAME имеет столько столбцов, каково максимальное значение в 1-м столбце LIST.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Функция /\*!KO\_nzlist\*/ противоположна данной и может выписать значения из NAME, создавая NNZ и LIST, если

\*выписываемые значения - это все ненулевые значения, существующие в NAME.

ПРИМЕР.

matrix.

!KO\_normal(6%8%mat). /\*Generate some random matrix of values

compute mat= mat&\*(abs(mat)>1.5). /\*Small values turn to 0 (make sparsity)

print mat /title 'MAT'.

!KO\_nzlist(mat%nnz%list).

print nnz /title 'NNZ: Number of nonzero values in each row'.

print list /title 'LIST: All the nonzero values (with corresponding column number)'.

!KO\_nzunlist(nnz%list%newmat).

print newmat /title 'NEWMAT: Back compilation of the matrix'.

end matrix.

### УТРАМБОВКА ПО ГОРИЗОНТАЛИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ ИЛИ ИХ ИНДЕКСОВ [!KO\_ram]

\*/\*!KO\_ram(mat%indval%lr%rand%name1%name2)\*/\*.

\*Version 2.

\*Берет матрицу или вектор MAT и сдвигает в каждом ряду MAT все ненулевые элементы к левому или

\*правому краю, уплотняя их (между ними не будет нулей).

\*Аргумент INDVAL - ключевое заглавное слово (можно взять в кавычки или апострофы):

\*"VAL" - сдвигать сами ненулевые значения

\*"IND" - сдвигать индексы ненулевых значений (т.е. номера их столбцов)

\*"BOTH" - делать оба задания

\*Аргумент LR - скаляр. Если неположительный, сдвиг будет влево; если положительный, то вправо.

\*Результат NAME1 это матрица со сделанной утрамбовкой (сдвижкой) значений или их индексов,

\*размером как MAT. При INDVAL=BOTH выдается оба результата в стопку: матрица с индексами подшита

\*под матрицу со значениями.

\*Результат NAME2 это столбец длиной как число рядов MAT, показывающий число ненулевых элементов в

\*каждом ряду MAT/NAME1.

\*Аргумент RAND - цифра (не имя и не выражение) 0 или 1 (можно в кавычких или апострофах).

\*Если 1, то к сдвижке добавляется действие рандомизации: утрамбованные значения (и их индексы)

\*будут идти в случайном порядке, своем в каждом ряду NAME1.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1000000.

matrix.

compute m= rnd(uniform(12,10)\*10-4).

compute m= m&\*(abs(m)>3).

print m /title 'Matrix with zero and nonzero elements'.

!KO\_ram(m%VAL%0%0%name1%name2).

print name1 /title 'Rammed to the left (nonzero values)'.

print name2.

!KO\_ram(m%IND%0%0%name1%name2).

print name1 /title 'Rammed to the left (indices of nonzero values))'.

end matrix.

ПРИМЕР. Получить категориальный набор множественного ответа (с левовыровненным строением) из дихотомического набора множественного ответа (т.е. из набора двоичных переменных).

set mxloops 1000000.

matrix.

compute bin= rnd(uniform(20,8)).

print bin. /\*Binary data (dichotomous MR set)

!KO\_ram(bin%IND%0%0%name1%name2).

print name1. /\*Categorical MR set

print name1(:,1:cmax(name2)). /\*with empty columns dropped

end matrix.

### УТРАМБОВКА ПО ВЕРТИКАЛИ НЕНУЛЕВЫХ ЭЛЕМЕНТОВ ИЛИ ИХ ИНДЕКСОВ [!KO\_vram]

\*/\*!KO\_vram(mat%indval%tb%rand%name1%name2)\*/\*.

\*Version 1.

\*Эта функция как /\*!KO\_ram\*/, только сдвигает элементы по вертикали, а не горизонтали.

\*Берет матрицу или вектор MAT и сдвигает в каждом столбце MAT все ненулевые элементы к верхнему или

\*нижнему краю, уплотняя их (между ними не будет нулей).

\*Аргумент INDVAL - ключевое заглавное слово (можно взять в кавычки или апострофы):

\*"VAL" - сдвигать сами ненулевые значения

\*"IND" - сдвигать индексы ненулевых значений (т.е. номера их рядов)

\*"BOTH" - делать оба задания

\*Аргумент TB - скаляр. Если неположительный, сдвиг будет вверх; если положительный, то вниз.

\*Результат NAME1 это матрица со сделанной утрамбовкой (сдвижкой) значений или их индексов,

\*размером как MAT. При INDVAL=BOTH выдается оба результата в гармошку: матрица с индексами пришита

\*справа к матрице со значениями.

\*Результат NAME2 это ряд длиной как число столбцов MAT, показывающий число ненулевых элементов в

\*каждом столбце MAT/NAME1.

\*Аргумент RAND - цифра (не имя и не выражение) 0 или 1 (можно в кавычких или апострофах).

\*Если 1, то к сдвижке добавляется действие рандомизации: утрамбованные значения (и их индексы)

\*будут идти в случайном порядке, своем в каждом столбце NAME1.

\*Для регулирования зерна случайных чисел используйте внематричный генератор

\*случайных чисел (соответствует командам SET MTINDEX или SET SEED).

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### РАЗМНОЖЕНИЕ РЯДОВ [!KO\_propag]

\*/\*!KO\_propag(mat%times%name)\*/\*.

\*Version 1.

\*Берет матрицу или вектор любого размера MAT. Размножает ее ряды в количество копий,

\*указанное в TIMES. В выходящей матрице NAME копии каждого ряда идут смежно.

\*TIMES - либо положительный целый скаляр, либо вектор длиной как число рядов MAT, в котором указано

\*количество копий каждого ряда отдельно. Значения в векторе - неотрицательные целые, хотя бы одно

\*значение больше нуля. Нулевое значение означает не брать данный ряд MAT в NAME вообще.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1000000.

matrix.

compute m= rnd(uniform(6,4)\*10).

print m.

!KO\_propag(m%2%mmm).

print mmm.

!KO\_propag(m%{1,2,3,0,1,4}%mmm).

print mmm.

end matrix.

### ВЫБОР РЯДОВ (ФИЛЬТРАЦИЯ НАБЛЮДЕНИЙ)

Если у вас есть матрица данных DATA и есть двоичный вектор-столбец FILT длиной как число рядов там, служащий фильтром рядов, «наблюдений» (1= взять ряд, 0= не брать ряд), и вы хотите проделать что-либо (анализ, трансформации) лишь с «подвыборкой» взятых наблюдений или же хотите оставить от массива только эти взятые наблюдения, – то можете пойти двумя путями: (A) выписать индексы единиц, их позиции в FILT, функцией /\*!KO\_indices\*/ (это будет вектор IND) и получить нужную подвыборку как DATA(IND,:); либо (B) воспользоваться функцией /\*!KO\_propag\*/ с вектором FILT в качестве аргумента TIMES. Вариант (A) обычно несколько быстрее; кроме того, он позволит вставить выбранную подвыборку рядов (наблюдений) в матрицу DATA обратно, поскольку вектор позиций IND остался после ф-ции /\*!KO\_indices\*/. См. также функцию /\*!KO\_split\*/.

### ПРИБАВЛЕНИЕ КОНСТАНТЫ К ЭЛЕМЕНТАМ ТРЕУГОЛЬНИКА МАТРИЦЫ [!KO\_tradd]

\*/\*!KO\_tradd(mat%tr%val%name)\*/\*.

\*Version 1.

\*Прибавляет число VAL (скаляр) к элементам верхнего или нижнего треугольника MAT.

\*Матрица MAT не обязана быть квадратной, но должна быть размером не меньше 2x2.

\*TR - скаляр; если положительный, VAL прибавляется к элементам над главной диагональю;

\*если неположительный, то прибавляется к элементам под главной диагональю.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### УМНОЖЕНИЕ НА КОНСТАНТУ ЭЛЕМЕНТОВ ТРЕУГОЛЬНИКА МАТРИЦЫ [!KO\_trmult]

\*/\*!KO\_trmult(mat%tr%val%name)\*/\*.

\*Version 1.

\*Умножает элементы верхнего или нижнего треугольника MAT на число VAL (скаляр).

\*Матрица MAT не обязана быть квадратной, но должна быть размером не меньше 2x2.

\*TR - скаляр; если положительный, на VAL умножаются элементы над главной диагональю;

\*если неположительный, то умножаются элементы под главной диагональю.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ЭКСПОНЕНЦИИРОВАНИЕ КОНСТАНТОЙ ЭЛЕМЕНТОВ ТРЕУГОЛЬНИКА МАТРИЦЫ [!KO\_trexp]

\*/\*!KO\_trexp(mat%tr%power%name)\*/\*.

\*Version 1.

\*Экспоненциирует (возводит в степень) элементы верхнего или нижнего треугольника MAT.

\*Степень - POWER (скаляр). Ограничения:

\*Если в треугольнике MAT есть отрицательные значения, POWER должно быть целым числом.

\*Если в треугольнике MAT есть нулевые значения, POWER должно быть положительным числом.

\*Матрица MAT не обязана быть квадратной, но должна быть размером не меньше 2x2.

\*TR - скаляр; если положительный, то экспоненциируются элементы над главной диагональю;

\*если неположительный, то экспоненциируются элементы под главной диагональю.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### РАЗВЕРТКА ТРЕУГОЛЬНИКОВ КВАДРАТНОЙ МАТРИЦЫ В ВЕКТОРЫ [!KO\_unftri]

\*/\*!KO\_unftri(mat%name1%name2%name3)\*/\*.

\*Version 2.

\*Берет матрицу MAT, которая должна быть квадратной, и выдает 2-столбцовую матрицу NAME1,

\*в которой 1-й столбец есть развертка под-диагонального (нижне-левого) треугольника MAT

\*и 2-й столбец есть развертка над-диагонального (верхне-правого) треугольника MAT.

\*Пара элементов в одном ряду NAME1 это симметричные по положению элементы в MAT.

\*NAME2 это соответствующие NAME1 индексы (позиции) внедиагональных элементов в векторе,

\*который есть развертка (векторизация) матрицы MAT. 1-й ряд NAME2 относится к элементам

\*под-диагонального треугольника, а 2-й ряд - к элементам над-диагонального треугольника.

\*С помощью NAME2 вы быстро можете развернуть треугольники любой матрицы размером с MAT,

\*не прибегая вторично к данной функции.

\*NAME3 содержит памятку - номера рядов и столбцов MAT, откуда взяты элементы. Так, 1-я пара

\*индексов в NAME3 есть 2 и 1, что напоминает, что 1-я пара значений в NAME1 это: 2-й эл-т

\*из 1-го столбца и 2-й элемент из 1-го ряда.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Функция /\*!KO\_foltri\*/ противоположна данной: сворачивает пару векторов NAME1 в треугольники

\*квадратной матрицы MAT.

ПРИМЕР.

set mxloops 10000.

matrix.

compute m= {11,12,13,14,15;21,22,23,24,25;31,32,33,34,35;41,42,43,44,45;51,52,53,54,55}.

print m.

!KO\_unftri(m%unfold%posinvec%memo).

print unfold /clabels= 'LowTr' 'UppTr'.

print t(memo) /title 'Where pairs of symmetric elements came from:'.

print t(posinvec) /title 'Positions of elements after turning M to vector'.

print /title '------------'.

\*You can use indices "posinvec" to unfold quickly another matrix of the

\*same size as M.

compute mat= uniform(5,5).

print mat /format= f8.3.

compute mat= reshape(mat,25,1). /\*Reshape it into vector first

print mat /format= f8.3.

print {mat(posinvec(1,:)),mat(posinvec(2,:))}

/format= f8.3 /clabels= 'LowTr' 'UppTr'. /\*The unfolded triangles

end matrix.

### СВЕРТКА ДВУХ ВЕКТОРОВ В ТРЕУГОЛЬНИКИ КВАДРАТНОЙ МАТРИЦЫ [!KO\_foltri]

\*/\*!KO\_foltri(vecs%name1%name2%name3)\*/\*.

\*Version 2.

\*Берет матрицу VECS, в которой должно быть 2 стобца (векторы), и выдает квадратную

\*матрицу NAME1, в которой под-диагональный (нижне-левый) треугольник есть свертка 1-го вектора

\*и над-диагональный (верхне-правый) треугольник есть свертка 2-го вектора.

\*Пара элементов в одном ряду VECS становится симметричными по положению элементами NAME1.

\*Диагональ NAME1 состоит из нулей.

\*NAME2 это соответствующие NAME1 индексы (позиции) внедиагональных элементов в векторе,

\*который есть развертка (векторизация) матрицы NAME1. 1-й ряд NAME2 относится к элементам

\*под-диагонального треугольника, а 2-й ряд - к элементам над-диагонального треугольника.

\*NAME3 это скаляр, размер матрицы NAME1.

\*Если число рядов VECS не позволяет создать квадратную матрицу NAME1, то NAME3 окажется

\*дробным числом, и NAME1 и NAME2 будут выданы как нулевые скаляры.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Функция /\*!KO\_unftri\*/ противоположна данной: разворачивает треугольники квадратной матрицы NAME1

\*в пару векторов VECS.

ПРИМЕР.

set mxloops 10000.

matrix.

compute vecs= uniform(15,2).

print vecs.

!KO\_foltri(vecs%mat%posit%size).

print mat.

print posit.

\*You can use indices "posit" to fold quickly some other unfolded

\*triangles of the same size as "vecs".

compute vecs2= rnd(uniform(15,2)\*10).

print vecs2.

compute mat2= make(size\*\*2,1,0). /\*Initialize mat2 as empty vector of length size^2

compute mat2(posit(1,:))= vecs2(:,1). /\*Populate it with values of lower triangle

compute mat2(posit(2,:))= vecs2(:,2). /\*and of upper triangle

compute mat2= reshape(mat2,size,size). /\*Reshape into square matrix

print mat2.

end matrix.

### СИММЕТРИЗАЦИЯ КВАДРАТНОЙ МАТРИЦЫ: ЗАМЕНА ОДНОГО ТРЕУГОЛЬНИКА ВТОРЫМ [!KO\_symtri1]

\*/\*!KO\_symtri1(mat%tr%name)\*/\*.

\*Version 1.

\*Берет матрицу MAT, которая должна быть квадратной, и заменяет в ней один треугольник вторым;

\*выдает квадратную симметрическую матрицу NAME.

\*Аргумент TR заказывает, какой треугольник заместить:

\*TR неположительный скаляр => под-диагональный треугольник заместить над-диагональным;

\*TR положительный скаляр => над-диагональный треугольник заместить под-диагональным.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### СИММЕТРИЗАЦИЯ КВАДРАТНОЙ МАТРИЦЫ: ЗАМЕНА БОЛЬШЕГО ЭЛЕМЕНТА МЕНЬШИМ ИЛИ МЕНЬШЕГО БОЛЬШИМ [!KO\_symtri2]

\*/\*!KO\_symtri2(mat%el%name)\*/\*.

\*Version 2.

\*Берет матрицу MAT, которая должна быть квадратной, и заменяет в ней каждый внедиагональный элемент

\*симметричным ему по положению, если последний больше (или меньше) него;

\*выдает квадратную симметрическую матрицу NAME.

\*Аргумент EL заказывает, когда делать замещение:

\*EL неположительный скаляр => заместить элемент симметричным, если последний меньше него;

\*EL положительный скаляр => заместить элемент симметричным, если последний больше него.

ПРИМЕР.

matrix.

compute mat= {1,2,3,4;5,6,7,8;9,10,11,12;13,14,15,16}.

print mat.

!KO\_symtri2(mat%0%mat).

print mat.

end matrix.

### СОРТИРОВКА ВЕКТОРА [!KO\_sort]

\*/\*!KO\_sort(vec%ord%name1%name2)\*/\*.

\*Version 2.

\*Сортирует элементы в векторе (ряд или столбец) VEC по величине их значений.

\*Выдает сортированный вектор NAME1.

\*А также вектор NAME2, который есть номера элементов NAME1, расположенные в последовательности,

\*в какой элементы были до сортировки (с помощью NAME2 можно восстановить исходный порядок элементов).

\*NAME2 есть целочисленные ранги значений VEC.

\*Аргумент ORD заказывает порядок сортировки:

\*ORD неположительный скаляр => по убыванию; ORD положительный скаляр => по возрастанию.

\*Если хотите затем переупорядочить ряды внешнего массива так же, как оказались сортированы элементы в VEC,

\*используйте NAME2 в качестве COL в /\*!KO\_rsortc\*/ и задайте ORD как 0.

ПРИМЕР.

matrix.

compute vec= uniform(10,1).

print vec /title 'Vector'.

!KO\_sort(vec%0%svec%rnk).

print svec /title 'Sorted vector'.

print rnk /title 'Ranks which defined the sort'.

print svec(rnk) /title 'Restored order'.

end matrix.

ПРИМЕР. Сортировка значений во 2-м столбце матрицы.

matrix.

compute mat= {4,3,2;3,1,1;6,3,4;2,1,5;1,2,3;3,4,5;6,2,7;7,4,6;3,4,1;5,1,2;1,2,7}.

print mat.

!KO\_sort(mat(:,2)%-1%col%name2).

compute mat(:,2)= col.

print mat.

end matrix.

### ПРОСТАЯ СОРТИРОВКА РЯДОВ В МАТРИЦЕ [!KO\_rsort]

\*/\*!KO\_rsort(mat%cn%ord%name1%name2)\*/\*.

\*Version 2.

\*Сортирует ряды в матрице MAT по величине значений в ее столбце CN.

\*Выдает матрицу NAME1.

\*А также столбец NAME2, который есть номера рядов NAME1, расположенные в последовательности,

\*в какой ряды были до сортировки (с помощью NAME2 можно восстановить исходный порядок рядов).

\*NAME2 есть целочисленные ранги значений столбца CN (до сортировки).

\*CN это номер столбца, положительный целый скаляр.

\*Аргумент ORD заказывает порядок сортировки:

\*ORD неположительный скаляр => по убыванию; ORD положительный скаляр => по возрастанию.

\*Если хотите затем переупорядочить ряды внешнего массива так же, как оказались сортированы ряды в MAT,

\*используйте NAME2 в качестве COL в /\*!KO\_rsortc\*/ и задайте ORD как 0.

ПРИМЕР. Сортировка рядов матрицы по ее 2-му столбцу.

matrix.

compute mat= {4,3,2;3,1,1;6,3,4;2,1,5;1,2,3;3,4,5;6,2,7;7,4,6;3,4,1;5,1,2;1,2,7}.

print mat.

!KO\_rsort(mat%2%1%mat%rnk).

print mat.

print rnk. /\*Ranks which defined the sort

print mat(rnk,:). /\*Restore order

end matrix.

### СОРТИРОВКА РЯДОВ В МАТРИЦЕ ПО ЗНАЧЕНИЯМ ВНЕШНЕГО СТОЛБЦА [!KO\_rsortc]

\*/\*!KO\_rsortc(mat%col%ord%name1%name2)\*/\*.

\*Version 2.

\*Сортирует ряды в матрице MAT по величине значений столбца COL.

\*COL это вектор-столбец длиной как число рядов MAT.

\*Выдает матрицу NAME1.

\*А также столбец NAME2, который есть номера рядов NAME1, расположенные в последовательности,

\*в какой ряды были до сортировки (с помощью NAME2 можно восстановить исходный порядок рядов).

\*NAME2 есть целочисленные ранги значений COL.

\*Аргумент ORD заказывает порядок сортировки:

\*ORD отрицательный скаляр => по убыванию; ORD положительный скаляр => по возрастанию.

\*Если ORD есть 0, то COL должен содержать целочисленные ранги и порядок сортировки определяется

\*уже ими. При этом функция не тратит время на ранжирование. Этот вариант задания выгоден в смысле скорости,

\*когда вы хотите отсортировать ряды в нескольких матрицах (с одним и тем же числом рядов) в идентичном

\*порядке, но не хотите ради этого временно объединять матрицы. Тогда вы можете использовать

\*NAME2, полученный в сортировке первой матрицы, в качестве COL в сортировках остальных матриц.

ПРИМЕР.

set seed 453473.

matrix.

compute mat= {4,3,2;3,1,1;6,3,4;2,1,5;1,2,3;3,4,5;6,2,7;7,4,6;3,4,1;5,1,2;1,2,7}.

print mat.

compute sortby= uniform(nrow(mat),1).

print sortby. /\*External column to sort by its values

!KO\_rsortc(mat%sortby%1%smat%rnk).

print smat. /\*Matrix with rows sorted by values of the SORTBY column

print rnk. /\*Ranks which defined the sort

print smat(rnk,:). /\*(Restored initial sort order)

compute mat2= {112,67;84,209;314,475;24,199;251,87;146,403;274,88;97,150;251,282;333,460;72,302}.

print mat2. /\*One more matrix with the same number of rows as MAT, to sort samely

!KO\_rsortc(mat2%rnk%0%smat2%rnk). /\*To sort its rows by SORTBY quicker, you may use

/\*ranks RNK (instead of SORTBY) and set argument ORD to 0

print smat2.

end matrix.

### ИЕРАРХИЧЕСКАЯ СОРТИРОВКА РЯДОВ В МАТРИЦЕ [!KO\_hiesort]

\*/\*!KO\_hiesort(mat%ord%name1%name2)\*/\*.

\*Version 2.

\*Сортирует ряды в матрице MAT по величине значений во всех ее столбцах.

\*Ряды сортируются иерархически: по значениям 1-го столбца; внутри одинаковых

\*его значений - по значениям 2-го столбца; внутри одинаковых его значений - по

\*значениям 3-го столбца; и т.д. Эта функция подобна внематричной команде SORT CASES,

\*когда в ней указываешь несколько переменных.

\*Выдает матрицу NAME1.

\*А также столбец NAME2, который есть номера рядов NAME1, расположенные в последовательности,

\*в какой ряды были до сортировки (с помощью NAME2 можно восстановить исходный порядок рядов).

\*Аргумент ORD это вектор длиной как число столбцов MAT либо скаляр; он заказывает порядок сортировки

\*в каждом отдельном столбце:

\*n-й элемент ORD неположительный => сортировка в n-м столбце по убыванию;

\*n-й элемент ORD положительный => сортировка в n-м столбце по возрастанию.

\*ORD скаляр значит один и тот же порядок во всех столбцах.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Функция относительно медленная.

\*Если хотите затем переупорядочить ряды внешнего массива так же, как оказались сортированы ряды в MAT,

\*используйте NAME2 в качестве COL в /\*!KO\_rsortc\*/ и задайте ORD как 0.

ПРИМЕР. Иерархическая сортировка в MATRIX и вне MATRIX.

matrix.

compute mat= {4,3,2;3,1,1;6,3,4;2,1,5;1,2,3;3,4,5;6,2,7;7,4,6;3,4,1;5,1,2;1,2,7}.

print mat.

save mat /outfile= \* /variables= v1 v2 v3.

!KO\_hiesort(mat%{-1,1,1}%mat%name2).

print mat.

end matrix.

sort cases v1(D) v2(A) v3(A).

list v1 v2 v3.

### СОРТИРОВАТЬ В СЛУЧАЙНОМ ПОРЯДКЕ

Используйте функцию /\*!KO\_sample\*/.

### РАСТАСКИВАНИЕ МАТРИЦЫ НА ДВЕ [!KO\_split]

\*/\*!KO\_split(whole%dim%ind%name1%name2)\*/\*.

\*Version 2.

\*Из матрицы WHOLE вырезаются ее ряды либо столбцы под номерами IND,

\*образуя матрицу-вырезку NAME1. Оставшиеся ряды (столбцы) матрицы WHOLE образуют

\*матрицу-остаток NAME2. Функция выдает обе матрицы NAME1 и NAME2.

\*Аргумент DIM - ключевое слово (опционально в кавычках или апострофах):

\*"ROWS" - вырезать ряды (слово "меры" ниже - это значит ряды);

\*"COLS" - вырезать столбцы (слово "меры" ниже - это значит столбцы).

\*WHOLE должна иметь число мер более одной.

\*IND это вектор (ряд или столбец, все равно) длиной меньше, чем есть мер в WHOLE,

\*и содержащий целые положительные числа: номера мер в WHOLE, которые надо вырезать.

\*Эти числа не обязаны идти по возрастающей; их последовательность задает последовательность

\*мер, какая она будет в матрице-вырезке NAME1.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно

\*большое число командой SET MXLOOPS.

\*Функция /\*!KO\_merge\*/ противоположна данной и может вставить врезку NAME1 в основу NAME2

\*с восстановлением WHOLE.

ПРИМЕР. Временное расщепление матрицы на две для раздельных операций над ними.

matrix.

compute mat= {4,3,2;3,1,1;6,3,4;2,1,5;1,2,3;3,4,5;6,2,7;7,4,6;3,4,1;5,1,2;1,2,7}.

print mat.

compute cutrows= {2,3,5,9}.

!KO\_split(mat%ROWS%cutrows%cutting%remnant).

print cutting.

print remnant.

compute remnant= remnant\*10.

!KO\_rsort(cutting%3%1%cutting%name2).

print remnant.

print cutting.

!KO\_merge(remnant%ROWS%cutting%cutrows%mat).

print mat.

end matrix.

### СМЕШЕНИЕ ДВУХ МАТРИЦ [!KO\_merge]

\*/\*!KO\_merge(base%dim%slice%ind%name)\*/\*.

\*Version 2.

\*В матрицу-основу BASE добавляются все ряды или столбцы матрицы-врезки SLICE, образуя

\*объединенную матрицу NAME, которая и выдается. Ряды (столбцы), составляющие SLICE, займут в NAME

\*позиции, заказанные в IND.

\*Аргумент DIM - ключевое слово (опционально в кавычках или апострофах):

\*"ROWS" - вставлять SLICE как ряды в BASE (слово "меры" ниже - это значит ряды), число столбцов

\*в BASE и SLICE должно быть одинаково;

\*"COLS" - вставлять SLICE как столбцы в BASE (слово "меры" ниже - это значит столбцы), число рядов

\*в BASE и SLICE должно быть одинаково.

\*IND это вектор (ряд или столбец, все равно) длиной как число мер в SLICE, и содержащий

\*целые положительные числа: номера мер в NAME, на которые надо поставить меры матрицы-врезки SLICE.

\*Эти числа не обязаны идти по возрастающей; их последовательность задает последовательность врезки

\*мер SLICE между мерами BASE. Максимальное значение в IND не может превышать сумму числа мер BASE и SLICE.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно

\*большое число командой SET MXLOOPS.

\*Функция /\*!KO\_split\*/ противоположна данной и может расщепить NAME обратно на

\*вырезку SLICE и остаток BASE.

### РАЗДЕЛЕНИЕ МАТРИЦЫ ГОРИЗОНТАЛЬНО НА НЕСКОЛЬКО, ПО ГРУППАМ [!KO\_msplit]

\*/\*!KO\_msplit(mat%group%'maxcode')\*/\*.

\*Version 1.

\*Берет матрицу MAT (некоторое имя, но не выражение) и группирующую переменную GROUP (сортированную).

\*Расщепляет MAT горизонтально на матрицы, соответствующие группам в GROUP. Имена результатных

\*матриц определяются автоматически.

\*Функция отчасти похожа на команду SPLIT FILE.

\*Подробно:

\*GROUP - группирующая наблюдения переменная, столбец с целочисленными кодами групп: 1, 2, 3,...

\*(Дробные коды или меньше единицы - недопустимы.) Наблюдения (ряды) должны быть уже сортированы по

\*возрастанию кодов.

\*MAT - матрица (данные) с числом рядов как в GROUP, уже сортированными в соответствие с GROUP, которую

\*требуется расщепить горизонтально: одна матрица-результат на каждый код из GROUP.

\*Имена будут образованы из имени MAT и кода группы: MAT1, MAT2,...

\*'MAXCODE' - код последней наличной или интересующей вас группы (максимум 999). Это должно быть число

\*(не переменная и не выражение), его можно взять в кавычки или апострофы.

\*Если некоторые коды между 1 и maxcode отсутствуют в GROUP, матрицы для этих отсутствующих групп не будут

\*созданы.

\*Данная функция удобна для рутинной работы, но не для программирования алгоритмов.

ПРИМЕР.

sort cases by group. /\*Group codes must be 1, 2,..., sorted

matrix.

get data /variables= v1 to v3.

get group /variable= group.

!KO\_msplit(data%gr%'4').

print data1.

print data2.

print data3.

print data4.

end matrix.

### ПЕРЕСТРУКТУРИРОВАНИЕ "ПЕРЕМЕННЫЕ В НАБЛЮДЕНИЯ" [!KO\_vartocas]

\*/\*!KO\_vartocas(data%k%ord%name1%name2)\*/\*.

\*Version 1.

\*Переструктурирует каждую группу из K переменных (столбцов) DATA в единый стобец.

\*Таким образом каждое наблюдение (ряд) разбивается на K рядов.

\*Выдает переструктурированные данные NAME1.

\*Функция подобна SPSS команде VARSTOCASES.

\*DATA - матрица с числом столбцов, кратным K.

\*K - целый положительный скаляр; каждые K столбцов в DATA (каждая "группа") дадут один столбец в NAME1.

\*Скаляр ORD указывает, в каком порядке расположены в DATA переменные, дающие единый столбец:

\*расположены смежно (ORD положителен) или расположены с шагом число\_столбцов\_DATA/K (ORD неположителен).

\*Например, если число\_столбцов\_DATA=6 и K=2, то смежное расположение значит, что столбцы

\*1 и 2 дадут столбец, 3 и 4 дадут столбец, 5 и 6 дадут столбец. А при расположении через шаг 3

\*столбцы 1 и 4 дадут столбец, 2 и 5 дадут столбец, 3 и 6 дадут столбец.

\*Двухстолбцовая матрица NAME2 отражает этот порядок: она будет содержать

\*номера наблюдений DATA в своем 1-м столбце и порядковый индекс

\*переменной в "группе" (1,2,...,K) переменных в своем 2-м столбце.

\*Функция \*/\*!KO\_castovar\*/\* противоположна данной и переструктурирует NAME1 обратно в DATA.

### ПЕРЕСТРУКТУРИРОВАНИЕ "НАБЛЮДЕНИЯ В ПЕРЕМЕННЫЕ" [!KO\_castovar]

\*/\*!KO\_castovar(data%k%ord%name1%name2)\*/\*.

\*Version 1.

\*Переструктурирует каждую группу из K наблюдений (рядов) DATA в единый ряд.

\*Таким образом каждая переменная (столбец) разбивается на K столбцов.

\*Выдает переструктурированные данные NAME.

\*Функция подобна SPSS команде CASESTOVARS.

\*DATA - матрица с числом рядов, кратным K.

\*K - целый положительный скаляр; каждые K рядов в DATA (каждая "группа") дадут один ряд в NAME1.

\*Ряды в DATA должны быть заранее сортированы так, чтобы K рядов каждой группы были смежны.

\*Скаляр ORD задает, в каком порядке надо расположить в NAME1 переменные, образованные из

\*одной переменной DATA: смежно (ORD положителен) или же с шагом число\_столбцов\_DATA

\*(ORD неположителен). Например, если число\_столбцов\_DATA=3 и K=2, то смежное расположение значит,

\*что столбец 1 даст столбцы 1 и 2, столбец 2 даст столбцы 3 и 4, столбец 3 даст столбцы 5 и 6.

\*А при расположении через шаг 3 столбец 1 даст столбцы 1 и 4, столбец 2 даст столбцы 2 и 5,

\*столбец 3 даст столбцы 3 и 6.

\*Двухрядная матрица NAME2 отражает этот порядок: она будет содержать номера переменных DATA

\*в своем 1-м ряду и порядковый индекс наблюдения в "группе" (1,2,...,K) наблюдений в своем 2-м ряду.

\*Функция \*/\*!KO\_vartocas\*/\* противоположна данной и переструктурирует NAME1 обратно в DATA.

ПРИМЕР.

set seed 34534753.

matrix.

compute data= uniform(10,15)\*10.

print data /format= f6.2 /title '10 cases X 15 variables Data'.

!KO\_vartocas(data%3%1%vartocas%ord).

print {ord,vartocas} /format= f6.2 /clab= 'CaseNo' 'Index'

/title 'Vars-to-Cases, K=3'.

!KO\_castovar(vartocas%3%1%castovar%ord).

print {ord;castovar} /format= f6.2 /rlab= 'VarNo' 'Index'

/title 'Cases-to-Vars, K=3, back into Data'.

end matrix.

### СМЕЩЕНИЕ РЯДОВ ПО ГОРИЗОНТАЛИ [!KO\_shift]

\*/\*!KO\_shift(mat%rel%displ%name)\*/\*.

\*Version 1.

\*Берет матрицу MAT с не менее чем двумя рядами. Выдает матрицу NAME, в которой ряды

\*смещены по горизонтали - вправо или влево на величины (число элементов), указанные в векторе-столбце

\*DISPL. DISPL должен быть длиной на 1 меньше, чем число рядов MAT. Его значения - целые числа.

\*Отрицательное значение это смещение на столько-то элементов влево, положительное значение это

\*смещение на столько-то элементов вправо. Первый элемент это смещение ряда 2, 2-й элемент это смещение

\*ряда 3, и т.д.

\*Аргумент REL (скаляр) поясняет, относительно чего заданы смещения в DISPL. Если REL неположителен,

\*смещение каждого ряда это смещение относительно 1-го ряда. Если REL положителен, смещение каждого ряда

\*это смещение относительно предыдущего, вышележащего ряда.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Последовательное смещение наблюдений на 1 вниз.

matrix.

get vars /variables= v1 v2 v3 v4.

print vars.

compute shifts= make(1,ncol(vars)-1,1).

print shifts.

!KO\_shift(t(vars)%1%t(shifts)%vars2).

compute vars2= t(vars2).

print vars2.

end matrix.

### БЛОК-ВРЕЗКА [!KO\_blockic]

\*/\*!KO\_blockic(mat1%mat2%pos%name)\*/\*.

\*Version 1.

\*Берет матрицы MAT1 и MAT2 любого размера и соединяет их, врезая MAT2 снизу-справа

\*в MAT1, пристыковывая ее к позиции POS в MAT1. Конкретно, сначала матрица NAME создается как

\*блок матрица block(MAT1(1:I,1:J),MAT2). Потом, если в NAME остались ячейки, еще не

\*получившие значений из соответствующих им ячеек MAT1, они их получают.

\*POS - вектор длиной 2, содержащий позиции I и J в MAT1. I может быть от 1 до число\_рядов MAT1.

\*J может быть от 1 до число\_столбцов MAT1. Когда I и J оба максимальны, функция превращается

\*в функцию block().

\*Если вы хотите "перевернуть" задачу и врезать MAT2 сверху-слева в MAT1, то сначала

\*поменяйте порядок рядов и столбцоа в обеих матрицах на обратный, а в конце поменяйте порядок

\*рядов и столбцов в NAME на обратный.

ПРИМЕР.

matrix.

compute m1=

{46,25,15;

3,77,75;

50,40,19;

25,28,67;

57,25,88}.

compute m2=

{-70,-90;

-69,-35;

-8,-21}.

print m1.

print m2.

print {3,2}.

\*Block-incut (from bottom-right).

!KO\_blockic(m1%m2%{3,2}%m).

print m /title 'Block-incut (to position 3,2)'.

\*Now block-incut from top-left.

compute m1= m1(nrow(m1):1,ncol(m1):1).

compute m2= m2(nrow(m2):1,ncol(m2):1).

!KO\_blockic(m1%m2%{3,2}%m).

compute m= m(nrow(m):1,ncol(m):1).

print m

/title 'Reversed Block-incut (to position 3,2 meas. off the bottom-right corner)'.

end matrix.

# КОМБИНАТОРНЫЕ ФУНКЦИИ И МНОЖЕСТВА

### ДВУМЕСТНЫЕ ОПЕРАЦИИ С МНОЖЕСТВАМИ [!KO\_setdo]

\*/\*!KO\_setdo(set1%set2%oper%name)\*/\*.

\*Version 1.

\*Берет два множества SET1 и SET2 (произвольные матрицы) и выдает их объединение, пересечение,

\*разность или симметричную разность. Значение 0 в SET1 и SET2 зарезервировано как пустота, т.е.

\*не считается элементом множества. Всякое прочее значение является элементом и учитывается

\*однократно в результатном векторе-ряде NAME (т.е., например, пересечение между {1,2,2,2,3}

\*и {1,2,2,3,4} есть {1,2,3}, а не {1,2,2,3} или {1,2,2,2,3}).

\*OPER - кл. слово заглавными буквами (можно в кавычках/апострофах):

\*"UNION" - объединение двух множеств (значения, присутствующие хотя бы в одном из множеств);

\*"INTER" - пересечение двух множеств (значения, присутствующие в обоих множествах);

\*"DIFF" - разность двух множеств (значения, присутствующие в SET1, но не в SET2);

\*"SYMDIFF" - симметричная разность двух множеств (значения, присутствующие в SET1, но не в SET2,

\*либо в SET2, но не в SET1).

\*NAME = скаляр 0 означает пустое множество на выходе.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1E6.

matrix.

compute x= {-1;0;1;2;3}.

compute y= {1;2;3;4;5}.

!KO\_setdo(x%y%SYMDIFF%result).

print result.

end matrix.

### ДЕКАРТОВО ПРОИЗВЕДЕНИЕ ДВУХ МНОЖЕСТВ [!KO\_cartp]

\*/\*!KO\_cartp(set1%set2%ii%name)\*/\*.

\*Version 1.

\*Комбинирует элементы между двумя множествами, создавая декартово произведение. Есть опция не спаривать

\*элементы с одинаковыми порядковыми номерами.

\*Множества SET1 и SET2 это матрицы одного или разного размера с числом рядов >1.

\*Их ряды это есть элементы множеств.

\*Результат NAME - матрица, каждый ряд это уникальная пара элементов, первый - из SET1,

\*второй - из SET2. Декартово произведение это все возможные парные сочетания.

\*Столбцы NAME это объединение столбцов входящих матриц.

\*Аргумент II - скаляр. Если он неположителен, выдается обычное, полное декартово произведение.

\*Если II положителен, то пары, представляющие элементы с одинаковыми порядковыми номерами, исключены из результата.

\*Например, если SET1 это A,B,C и SET2 это 1,2,3,4, то будут опущены пары A1, B2 и C3.

ПРИМЕР.

matrix.

compute x= {1;2;3}.

compute y= {1;2;3;4;5}.

print x.

print y.

!KO\_cartp(x%y%0%prod1).

print prod1.

!KO\_cartp(y%x%0%prod2).

print prod2.

!KO\_cartp(x%y%1%prod3).

print prod3.

end matrix.

### ДЕКАРТОВО ПРОИЗВЕДЕНИЕ N МНОЖЕСТВ [!KO\_ncartp]

\*/\*!KO\_ncartp(setlist%name)\*/\*.

\*Version 1.

\*Комбинирует элементы между двумя или более множествами, создавая декартово произведение.

\*Множества это матрицы любого размера, ряды матрицы это элементы множества.

\*SETLIST это список (через пробел) имен матриц. Список может опционально быть в кавычках/апострофах.

\*Выдает матрицу NAME, ряды которой это элементы декартова произведения этих матриц, а столбцы это

\*объединение столбцов этих матриц.

ПРИМЕР.

matrix.

compute x= {'a';'b';'c'}.

compute y= {'1';'2';'3';'4'}.

compute z= {'+';'-'}.

!KO\_ncartp(x y z%prod).

print prod /format= a4.

end matrix.

### СОЧЕТАНИЯ ПО K ЭЛЕМЕНТОВ [!KO\_combk]

\*/\*!KO\_combk(set%k%name)\*/\*.

\*Version 1.

\*Образует сочетания по k элементов из множества n элементов.

\*Множество SET это столбец или матрица с n рядами: элементом множества выступает ряд.

\*K (целый скаляр от 2 до n) - число элементов в сочетании.

\*Результат NAME - матрица, каждый ряд которой есть сочетание элементов.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ВСЕ СОЧЕТАНИЯ ПО K=2,3... ЭЛЕМЕНТОВ [!KO\_allcomb]

\*/\*!KO\_allcomb(set%maxk%prefix)\*/\*.

\*Version 1.

\*Образует, из множества n элементов, сочетания по k=2, k=3, ..., k=MAXK элементов.

\*Выдает все эти сочетания соответственно как матрицы PREFIX2, PREFIX3, и т.д.

\*Т.е. эта функция подобна \*/\*!KO\_combk\*/\*, но выдает сразу все сочетания с разными k.

\*Множество SET это столбец или матрица с n рядами: элементом множества выступает ряд.

\*MAXK - максимальное число элементов в сочетании. Это должно быть целое число (не имя и не выражение)

\*от 2 до n. Опционально его можно взять в кавычки или апострофы.

\*PREFIX - приставка (опционально в кавычках или апострофах) в имена выдаваемых матриц - наборов сочетаний.

\*В выдаваемых матрицах каждый ряд есть сочетание элементов.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Совет: если вы хотите установить MAXK=n, но число n вам заранее не известно, задайте MAXK числом,

\*заведомо превышающим возможное n. Матрицы сверх PREFIXn не смогут быть созданы и ошибки не будет.

ПРИМЕР.

set mxloops 100000.

matrix.

compute set= {'a';'b';'c';'d';'e'}.

print set /format a2.

!KO\_allcomb(set%4%'name').

print name2 /format= a2.

print name3 /format= a2.

print name4 /format= a2.

end matrix.

### СОЧЕТАНИЯ ПО K ЭЛЕМЕНТОВ ИЗ РАЗНЫХ МНОЖЕСТВ [!KO\_dscombk]

\*/\*!KO\_dscombk(sets%last%k%name)\*/\*.

\*Version 1.

\*Образует сочетания по k элементов, принадлежащих разным множествам.

\*Множества SETS это столбец или матрица: элементом множества выступает ряд.

\*Множества должны быть собраны в SETS как стопка: сначала идут все элементы (ряды) одного множества,

\*под ними все элементы второго множества, и так далее. Множеств может быть любое количество и число

\*элементов в них может быть разное.

\*LAST - вектор (ряд или столбец) длиной как число множеств в SETS и в соответствие им, содержащий

\*индексы последних рядов каждого множества. Например, {3,6,11} означает: в SETS есть три множества,

\*с элементами, занимающими ряды 1:3, 4:6, 7:11. 11 (последний элемент последнего множества) должно

\*равняться числу рядов в SETS.

\*K (целый скаляр от 2 до nc, где nc это число множеств, длина вектора LAST) - число элементов в сочетании.

\*Результат NAME - матрица, каждый ряд которой есть сочетание из k элементов от разных множеств, не более

\*одного элемента от множества. Если K = число множеств, то сочетание есть "по элементу от всех множеств".

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### ВСЕ СОЧЕТАНИЯ ПО K=2,3... ЭЛЕМЕНТОВ ИЗ РАЗНЫХ МНОЖЕСТВ [!KO\_dsallcomb]

\*/\*!KO\_dsallcomb(sets%last%maxk%prefix)\*/\*.

\*Version 1.

\*Образует, из нескольких множеств, сочетания по k=2, k=3, ..., k=MAXK элементов, принадлежащих

\*разным множествам.

\*Выдает все эти сочетания соответственно как матрицы PREFIX2, PREFIX3, и т.д.

\*Т.е. эта функция подобна \*/\*!KO\_dscombk\*/\*, но выдает сразу все сочетания с разными k.

\*Множества SETS это столбец или матрица: элементом множества выступает ряд.

\*Множества должны быть собраны в SETS как стопка: сначала идут все элементы (ряды) одного множества,

\*под ними все елементы второго множества, и так далее. Множеств может быть любое количество и число

\*элементов в них может быть разное.

\*LAST - вектор (ряд или столбец) длиной как число множеств в SETS и в соответствие им, содержащий

\*индексы последних рядов каждого множества. Например, {3,6,11} означает: в SETS есть три множества,

\*с элементами, занимающими ряды 1:3, 4:6, 7:11. 11 (последний элемент последнего множества) должно

\*равняться числу рядов в SETS.

\*MAXK - максимальное число элементов в сочетании. Это должно быть целое число (не имя и не выражение)

\*от 2 до ns, где ns это число множеств, длина вектора LAST. Опционально число можно взять в кавычки

\*или апострофы.

\*PREFIX - приставка (опционально в кавычках или апострофах) в имена выдаваемых матриц - наборов сочетаний.

\*В выдаваемых матрицах каждый ряд есть сочетание из k элементов от разных множеств, не более

\*одного элемента от множества.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Совет: если вы хотите установить MAXK=ns, но число ns вам заранее не известно, задайте MAXK числом,

\*заведомо превышающим возможное ns. Матрицы сверх PREFIXns не смогут быть созданы и ошибки не будет.

ПРИМЕР.

set mxloops 100000.

matrix.

compute sets= {'a1';'a2';'b1';'b2';'b3';'c1'}.

print sets /format= a2.

compute last= {2;5;6}.

!KO\_dsallcomb(sets%last%'3'%'name').

print name2 /format= a2.

print name3 /format= a2.

end matrix.

### ГОРИЗОНТАЛЬНЫЕ АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ В СОЧЕТАНИЯХ ПЕРЕМЕННЫХ [!KO\_comboper]

/\*!KO\_comboper(vars%last%maxk%return%oper%weights%alpha%prefix%prefix\_res)\*/\*.

\*Version 1.

\*Столбцы матрицы VARS это числовые переменные (минимум 2 переменные).

\*Функция рассматривает все сочетания переменных по K переменных в сочетании; K пробегает

\*значения от 2 до MAXK. В каждом сочетании переменных вычисляется величина OPER горизонтально,

\*т.е. внутри каждого наблюдения. Это и есть результат.

\*Аргумент OPER - арифметическое действие, которое надо совершить в каждом сочетании переменных;

\*заглавное кл. слово (опционально можно в кавычках/апострофах):

\*"SUM" - сумма, операция сложения переменных: V(1)+V(2)+...+V(K)

\*"DIF" - разница, операция вычитания переменных V(1)-V(2)-...-V(K)

\*"PROD" - произведение, операция перемножения переменных V(1)\*V(2)\*...\*V(K)

\*"QUOT" - частное, операция деления переменных V(1)/V(2)/.../V(K) (нуждается в ненулевых данных)

\*"WSUM" - взвешенная сумма: w(1)\*V(1)+w(2)\*V(2)+...+w(K)\*V(K) (веса w - аргумент WEIGHTS)

\*"WDIF" - взвешенная разница: w(1)\*V(1)-w(2)\*V(2)-...-w(K)\*V(K) (веса w - аргумент WEIGHTS)

\*"EWMA" - конечное (K-е) значение экспоненциально взвешенной скользящей средней: S(K);

\* S(K)=(1-a)\*S(K-1)+a\*V(K), где S(K-1)=(1-a)\*S(K-2)+a\*V(K-1), где S(K-2)=...; и S(1)=V(1)

\* (экспоненциальный вес a - аргумент ALPHA).

\*Аргумент WEIGHTS - веса для операций WSUM и WDIF. Либо вектор-ряд длиной с число столбцов VARS,

\*либо матрица размером как VARS; второй случай означает, что для разных наблюдений (рядов VARS)

\*веса переменным заданы разные. Другие действия - не WSUM и WDIF - не пользуются аргументом:

\*задайте любое значение, например 0.

\*Аргумент ALPHA - экспоненциальный вес для операции EWMA, значение между 0 и 1. Это либо скаляр,

\*либо вектор-столбец длиной с число рядов VARS; второй случай означает, что для разных наблюдений

\*альфа-вес задан разный. Другие действия - не EWMA - не пользуются аргументом: задайте любое,

\*значение, например 0.

\*Аргументы LAST, MAXK, а также PREFIX - те же, что в функции /\*!KO\_dsallcomb\*/, на которой построена

\*данная функция.

\*Аргумент LAST - параметр для разбивки переменных на множества. Только переменные, принадлежащие

\*разным множествам, будут комбинироваться между собой. Это вектор (ряд или столбец) длиной как

\*число множеств и в соответствие им, содержащий индексы (номера) замыкающих переменных каждого

\*множества. Например, {3,6,11} означает: в VARS есть три множества, группы переменных со столбцами

\*1:3, 4:6, 7:11. 11 (последнее значение в LAST) должно равняться числу столбцов в VARS.

\*Если у вас нет групп переменных, т.е. каждая переменная есть свое собственное множество, задайте LAST

\*как 1:ncol(VARS), т.е. вектор из чисел 1,2,...,число\_переменных.

\*Аргумент MAXK - максимальное нужное вам значение K, число переменных в сочетании. Это должно быть

\*целое число (не имя и не выражение) от 2 до ns, где ns это число множеств, длина вектора LAST.

\*Аргумент RETURN - для каких значений K вы хотите получить результаты. Если для всех K от 2 до

\*min(MAXK,ns), т.е. все результаты, задайте RETURN как кл. слово ALL (заглавными).

\*В противном случае закажите список из требуемых K, целые через пробел, например: 2 3 5.

\*Тогда функция вычислит результаты только для этих K.

\*Результаты: PREFIX, PREFIX\_RES. Укажите на месте этих аргументов 2 приставки

\*для имен выдаваемых матриц, которые будут состоять из этих приставок и окончания в виде числа K.

\*Для каждого K будет выдана пара массивов:

\*- Номера переменных, образующих сочетания - матрица с приставкой PREFIX.

\* Ряды в PREFIX это разные сочетания переменных VARS, элементы в ряду это номера переменных в сочетании.

\*- Результаты операции OPER - матрица с приставкой PREFIX\_RES. Ряды в ней отвечают наблюдениям данных

\* (рядам VARS), а столбцы отвечает сочетаниям переменных (рядам PREFIX).

\*Задания аргументов MAXK, RETURN, PREFIX, PREFIX\_RES вы вправе взять в кавычки/апострофы.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Совет: если вы хотите установить MAXK=ns, но число ns вам заранее не известно, задайте MAXK числом,

\*заведомо превышающим возможное ns. Матрицы сверх PREFIXns не смогут быть созданы и ошибки не будет.

ПРИМЕР. Перемножение количественных переменных: получение их «взаимодействий» вплоть до 3-сторонних.

set mxloops 1E9.

matrix.

get vars /variables= v1 to v10.

!KO\_comboper(vars%1:ncol(vars)%3%ALL%PROD%0%0%Comb%Inter).

print vars.

print Comb2.

print Inter2.

print Comb3.

print Inter3.

save {Comb2,Comb3} /out= \*.

end matrix.

ПРИМЕР. Взвешенное суммирование количественных переменных из разных их наборов.

set mxloops 1E9.

matrix.

get vars /variables= v1 to v10.

compute weights= {2,.5,1.3,1.8,3,.6,1,1.1,.5,0} /\*10 weights for the 10 variables

!KO\_comboper(vars%{3,7,8,10}%4%3 4%WSUM%weights%0%Comb%wSum). /\*The sets are:

/\*v1-v3, v4-v7, v8, v9-v10 (positition of the last variable in each set

/\*was specified as LAST argument); combining will be only between sets;

/\*Combinations up to K=4 variables are requested, however, results for K=2

/\*will not be produced, because RETURN is "3 4"

print vars.

print Comb3.

print wSum3.

print Comb4.

print wSum4.

save {wSum3,wSum4} /out= \*.

end matrix.

### КОЛИЧЕСТВА НА БАЗЕ ГОРИЗОНТАЛЬНЫХ СУММ ИЛИ СРЕДНИХ В СОЧЕТАНИЯХ ПЕРЕМЕННЫХ [!KO\_turflike]

\*/\*!KO\_turflike(vars%last%maxk%return%tosum%tocnt%cond%prefix%prefix\_sum%prefix\_cnt)\*/\*.

\*Version 1.

\*Столбцы матрицы VARS это числовые переменные (минимум 2 переменные).

\*Функция рассматривает все сочетания переменных по K переменных в сочетании; K пробегает

\*значения от 2 до MAXK. В каждом сочетании переменных вычисляются статистики hs (см. ниже)

\*горизонтально, т.е. внутри каждого наблюдения. Затем функция (1) суммирует hs сквозь

\*наблюдения, выдавая результат PREFIX\_SUM; (2) подсчитывает наблюдения, где hs удовлетворяет

\*условию COND, выдавая результат PREFIX\_CNT. Таким образом, для каждого сочетания переменных,

\*столбцов из VARS, вы получите значение "column\_sum(hs)" и "column\_count(hs COND = True)".

\*Аргументы TOSUM и TOCNT. Здесь надо указать статистику hs в виде кл. слова заглавными буквами.

\*Можно заказать одну или разные статистики в этих двух аргументах. Выбор из следующих

\*статистик, вычислить внутри каждого наблюдения (ряда VARS):

\*"SUM" - сумма значений данных

\*"MEAN" - средняя значений данных

\*"SUMVLD" - сумма валидных значений данных

\*"MEANVLD" - средняя валидных значений данных

\*"NVLD" - количество валидных значений данных

\*В трех последних статистиках за невалидное значение принято значение: -999.

\*Если валидных данных в наблюдении нет, MEANVLD наблюдения оценивается как 0.

\*Аргумент COND - условие для подсчета числа наблюдений, в которых TOCNT-статистика удовлетворит ему.

\*Состоит из одного оператора и одного операнда. Операторами могут быть: GT (или >), LT (или <),

\*EQ (или =), GE, LE, NE. (>=, <=, <> использовать нельзя в данной функции.) Операндом может быть

\*число или имя некоторой матрицы или выражение; перед операндом может стоять отрицательный знак.

\*Примеры задания COND: GT 0, EQ -4.12, LE v3, LT x(:,1).

\*Операнд с именем. Именем может быть скаляр (число), вектор-столбец длиной =nrow(VARS),

\*вектор-ряд длиной >=min(MAXK,ns) [см. ниже об ns] либо матрица размером [nrow(VARS),>=min(MAXK,ns)].

\*Вектор-столбец позволяет вам задать разные значения операнда для разных наблюдений данных (рядов VARS).

\*Вектор-ряд позволяет вам задать разные значения операнда для разных K (числа переменных в сочетании);

\*K-й элемент в векторе-ряде будет отвечать данному K. Операнд в виде матрицы указанного выше размера

\*позволяет вам задать разные значения операнда и для разных наблюдений, и для разных K.

\*Аргументы LAST, MAXK, а также PREFIX - те же, что в функции /\*!KO\_dsallcomb\*/, на которой построена

\*данная функция.

\*Аргумент LAST - параметр для разбивки переменных на множества. Только переменные, принадлежащие

\*разным множествам, будут комбинироваться между собой. Это вектор (ряд или столбец) длиной как

\*число множеств и в соответствие им, содержащий индексы (номера) замыкающих переменных каждого

\*множества. Например, {3,6,11} означает: в VARS есть три множества, группы переменных со столбцами

\*1:3, 4:6, 7:11. 11 (последнее значение в LAST) должно равняться числу столбцов в VARS.

\*Если у вас нет групп переменных, т.е. каждая переменная есть свое собственное множество, задайте LAST

\*как 1:ncol(VARS), т.е. вектор из чисел 1,2,...,число\_переменных.

\*Аргумент MAXK - максимальное нужное вам значение K, число переменных в сочетании. Это должно быть

\*целое число (не имя и не выражение) от 2 до ns, где ns это число множеств, длина вектора LAST.

\*Аргумент RETURN - для каких значений K вы хотите получить результаты. Если для всех K от 2 до

\*min(MAXK,ns), т.е. все результаты, задайте RETURN как кл. слово ALL (заглавными).

\*В противном случае закажите список из требуемых K, целые через пробел, например: 2 3 5.

\*Тогда функция вычислит результаты только для этих K.

\*Результаты: PREFIX, PREFIX\_SUM, PREFIX\_CNT. Укажите на месте этих аргументов 3 приставки

\*для имен выдаваемых матриц, которые будут состоять из этих приставок и окончания в виде числа K.

\*Для каждого K будет выдана серия из 3 массивов:

\*- Номера переменных, образующих сочетания - матрица с приставкой PREFIX.

\* Ряды в PREFIX это разные сочетания переменных VARS, элементы в ряду это номера переменных в сочетании.

\*- Результаты суммирования TOSUM - вектор-столбец с приставкой PREFIX\_SUM. Ряды в нем отвечают рядам

\* в PREFIX.

\*- Результаты подсчета TOCNT - вектор-столбец с приставкой PREFIX\_CNT. Ряды в нем отвечают рядам

\* в PREFIX.

\*Если вы не желаете получать результат PREFIX\_SUM или PREFIX\_CNT, поствьте цифру 0 на место этого

\*аргумента.

\*Задания аргументов TOSUM, TOCNT, COND, MAXK, RETURN, PREFIX, PREFIX\_SUM, PREFIX\_CNT вы вправе

\*взять в кавычки/апострофы.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Совет: если вы хотите установить MAXK=ns, но число ns вам заранее не известно, задайте MAXK числом,

\*заведомо превышающим возможное ns. Матрицы сверх PREFIXns не смогут быть созданы и ошибки не будет.

\*Частный случай: если VARS - двоичные переменные (0 vs 1), TOSUM=TOCNT="SUM" и условие COND="GT 0",

\*то функция фактически выдает величины, вычисляемые в так. наз. TURF-анализе. PREFIX\_SUM даст частоты

\*"Объема" ("Frequency"), а PREFIX\_CNT даст частоты "Охвата" ("Reach").

ПРИМЕР. TURF-анализ.

set mxloops 1E9.

matrix.

get vars /variables= v1 to v10. /\*Ten binary variables (products)

!KO\_turflike(vars%1:nrow(vars)%10%ALL%SUM%SUM%gt 0%Comb%Freq%Reach).

/\*All variables can be combined with each other;

/\*K runs from 2 to 10; and results for every K is requested;

/\*Statistic “within-cases SUM” specified for the TOSUM arg, that

/\*will yield TURF “Frequency” counts (output with prefix Freq);

/\*Statistic “within-cases SUM” specified for the TOCNT arg with

/\*the condition COND to count if the sum is above zero, and that

/\*will yield TURF “Reach” (output with prefix Reach)

\*The results are: variable combinations as matrices Comb2 to Comb10;

\*“Reach” values as vectors Reach2 to Reach10;

\*“Frequency” values as vectors Freq2 to Freq10.

\*Organize the results:

\*For each K, concatenate the results and sort by “Reach”,

\*and within it - by “Frequency”;

!KO\_hiesort({Reach2,Freq2,Comb2}%0%K2%ind).

!KO\_hiesort({Reach3,Freq3,Comb3}%0%K3%ind).

!KO\_hiesort({Reach4,Freq4,Comb4}%0%K4%ind).

!KO\_hiesort({Reach5,Freq5,Comb5}%0%K5%ind).

!KO\_hiesort({Reach6,Freq6,Comb6}%0%K6%ind).

!KO\_hiesort({Reach7,Freq7,Comb7}%0%K7%ind).

!KO\_hiesort({Reach8,Freq8,Comb8}%0%K8%ind).

!KO\_hiesort({Reach9,Freq9,Comb9}%0%K9%ind).

!KO\_hiesort({Reach10,Freq10,Comb10}%0%K10%ind).

\*Print.

print /title '------ As Counts ------'.

print K2 /format= f6 /title 'K=2' /clabels 'Reach' 'Frequ' 'VarComb' '...'.

print K3 /format= f6 /title 'K=3' /clabels 'Reach' 'Frequ' 'VarComb' '...'.

print K4 /format= f6 /title 'K=4' /clabels 'Reach' 'Frequ' 'VarComb' '...'.

print K5 /format= f6 /title 'K=5' /clabels 'Reach' 'Frequ' 'VarComb' '...'.

print K6 /format= f6 /title 'K=6' /clabels 'Reach' 'Frequ' 'VarComb' '...'.

print K7 /format= f6 /title 'K=7' /clabels 'Reach' 'Frequ' 'VarComb' '...'.

print K8 /format= f6 /title 'K=8' /clabels 'Reach' 'Frequ' 'VarComb' '...'.

print K9 /format= f6 /title 'K=9' /clabels 'Reach' 'Frequ' 'VarComb' '...'.

print K10 /format= f6 /title 'K=10' /clabels 'Reach' 'Frequ' 'VarComb' '...'.

\*Also, transform counts to percents (“Reach” as % of cases,

\*“Frequency” as % of total responses):.

compute K2(:,1)= K2(:,1)/nrow(vars)\*100.

compute K3(:,1)= K3(:,1)/nrow(vars)\*100.

compute K4(:,1)= K4(:,1)/nrow(vars)\*100.

compute K5(:,1)= K5(:,1)/nrow(vars)\*100.

compute K6(:,1)= K6(:,1)/nrow(vars)\*100.

compute K7(:,1)= K7(:,1)/nrow(vars)\*100.

compute K8(:,1)= K8(:,1)/nrow(vars)\*100.

compute K9(:,1)= K9(:,1)/nrow(vars)\*100.

compute K10(:,1)= K10(:,1)/nrow(vars)\*100.

compute totalcnt= msum(vars).

compute K2(:,2)= K2(:,2)/totalcnt\*100.

compute K3(:,2)= K3(:,2)/totalcnt\*100.

compute K4(:,2)= K4(:,2)/totalcnt\*100.

compute K5(:,2)= K5(:,2)/totalcnt\*100.

compute K6(:,2)= K6(:,2)/totalcnt\*100.

compute K7(:,2)= K7(:,2)/totalcnt\*100.

compute K8(:,2)= K8(:,2)/totalcnt\*100.

compute K9(:,2)= K9(:,2)/totalcnt\*100.

compute K10(:,2)= K10(:,2)/totalcnt\*100.

\*Print.

print /title '------ As Percents ------'.

print K2 /format= f6.1 /title 'K=2' /clabels 'Reach%' 'Frequ%' 'VarComb' '...'.

print K3 /format= f6.1 /title 'K=3' /clabels 'Reach%' 'Frequ%' 'VarComb' '...'.

print K4 /format= f6.1 /title 'K=4' /clabels 'Reach%' 'Frequ%' 'VarComb' '...'.

print K5 /format= f6.1 /title 'K=5' /clabels 'Reach%' 'Frequ%' 'VarComb' '...'.

print K6 /format= f6.1 /title 'K=6' /clabels 'Reach%' 'Frequ%' 'VarComb' '...'.

print K7 /format= f6.1 /title 'K=7' /clabels 'Reach%' 'Frequ%' 'VarComb' '...'.

print K8 /format= f6.1 /title 'K=8' /clabels 'Reach%' 'Frequ%' 'VarComb' '...'.

print K9 /format= f6.1 /title 'K=9' /clabels 'Reach%' 'Frequ%' 'VarComb' '...'.

print K10 /format= f6.1 /title 'K=10' /clabels 'Reach%' 'Frequ%' 'VarComb' '...'.

end matrix.

ПРИМЕР. Несколько изощренный пример.

set mxloops 1E9.

set seed 74194798.

matrix.

\*Simulate some data with value -999 designating missing data.

compute n= 15.

compute p= 8.

compute vars= rnd(uniform(n,p)\*8).

compute miss= rnd(uniform(n,p)\*.6).

compute vars= vars&\*(not miss)+miss&\*(-999).

print vars. /\*The data, 8 variables (columns), 15 cases (rows)

\*Let our variables be split in 5 sets; variables will combine only between sets.

compute last= {1,4,6,7,8}. /\*The sets are:

/\*var 1, vars 2-4, vars 5-6, var 7, var 8

\*Let us have some operand condition variable X.

compute x= rnd(uniform(n,1)\*10).

print x.

\*Run the function.

!KO\_turflike(vars%last%8%3 4 5%MEANVLD%NVLD%ge x%K%SM%CSC).

/\*Variables belonging only to different sets will be combined together;

/\*Combinations from K=2 to K=8 variables are requested, however the number of

/\*sets 5 is less, so min(8,5)=5 is the upper value of K really possible;

/\*Moreover, results for K=2 won't be produced because instead of "ALL", just

/\*K= 3, 4, and 5 were demanded as RETURN argument;

/\*(The TOSUM argument) For each variable combination, mean of valid values

/\*will be computed in each case, and then summed across cases;

/\*the results are stored as vectors with prefix SM ("sum of means")

/\*(The TOCNT argument) For each variable combination, number of valid values

/\*will be counted in each case, and then the number of cases where this

/\*count is not below the threshold contained in variable X (COND argument)

/\*is counted; the results are stored as the vectors with prefix CSC

/\*("count of satisfying counts");

/\*The combinations themselves (variables' indices) are stored

/\*in matrices prefixed K.

\*Print results.

print {SM3,CSC3,K3} /format f8.4 /title 'K=3' /clabels 'TOSUM' 'TOCNT' 'Comb' '...'.

print {SM4,CSC4,K4} /format f8.4 /title 'K=4' /clabels 'TOSUM' 'TOCNT' 'Comb' '...'.

print {SM5,CSC5,K5} /format f8.4 /title 'K=5' /clabels 'TOSUM' 'TOCNT' 'Comb' '...'.

end matrix.

### АЛГОРИТМ APRIORI [!KO\_apriori]

\*/\*!KO\_apriori(mrc%minsup%name1%name2)\*/\*.

\*Version 1.

\*В данных типа множественного ответа Apriori-алгоритм подсчитывает наборы предметов

\*(предметы и их сочетания), которые не менее часты, чем порог MINSUP ("minimal support",

\*положительное целое). Выдаваемый счет называется "support" ("поддержка"). Поддержка это

\*частота подмножественная, т.е. число наблюдений, где предмет или сочетание предметов найдено

\*как поднабор. Например, в следующих четырех наблюдениях: {1,3}, {1,2,5}, {1,3,5}, {1,2,3,4}

\*поддержка сочетания {1,3} есть 3, а поддержка предмета {5} есть 2. Поддержка {1,3,5} есть 1.

\*MRC - данные в виде категориального набора множественного ответа, минимум один столбец.

\*Каждое наблюдение (ряд) это респондент или корзина, и значения в нем это коды ответов или

\*предметы, список. Значение 0 зарезервировано обозначать пустую (не использованную) ячейку.

\*Полностью нулевые ряды или столбцы данных допускаются. Если код встречается более раза в

\*наблюдении, он учитывается только раз.

\*Результаты - это частые наборы предметов (NAME1) и их соответствующая поддержка (NAME2).

\*Данная функция - не быстрая реализация Apriori-алгоритма, так что вы не станете применять ее

\*к многим тысячам или миллионам наблюдений.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1E6.

matrix.

compute data=

{1,2,0,0;

2,4,0,0;

2,3,0,0;

1,2,4,0;

1,3,0,0;

2,3,0,0;

1,3,0,0;

1,2,3,5;

1,2,3,0}.

print data.

!KO\_apriori(data%3%name1%name2).

print {name1,name2} /title 'Results for minsupport=3'

/clabels 'ItemSet' ' ' 'Support'.

end matrix.

# КОМБИНАТОРНЫЕ ОПТИМИЗАЦИОННЫЕ ФУНКЦИИ

### СПАРИВАНИЕ ВЕНГЕРСКИМ АЛГОРИТМОМ [!KO\_hungar]

\*/\*!KO\_hungar(cost%name)\*/\*.

\*Version 1.

\*Венгерский или Kuhn-Munkres алгоритм спаривает взаимно-однозначно элементы между двумя множествами

\*оптимально так, чтобы суммарная стоимость пар (или суммарное различие в парах) оказалась минимальна.

\*В терминологии графов, он создает оптимальное паросочетание (optimal matching) из двудольного

\*взвешенного или невзвешенного графа. (Отметим, что в случае невзвешенного графа вы можете использовать

\*для решения этой задачи и алгоритм Хопкрофта-Карпа /\*!KO\_hopckarp\*/.)

\*Функция принимает в качестве входящей n x m матрицу стоимостей COST с неотрицательными элементами.

\*Число рядов должно быть не меньше числа столбцов. Ряды - это одно множество элементов, столбцы - другое

\*множество элементов. Значения матрицы понимаются как "стоимость" пары ряд-столбец или как "различие"

\*между рядом и столбцом. Алгоритм находит каждому столбцу в пару один ряд.

\*Результат NAME - трехстолбцовая матрица с числом рядов как число столбцов COST; ряды в ней это

\*созданные пары. 2-й столбец ее это номер столбца входящей матрицы COST, 1-й столбец это номера ряда

\*COST, а 3-й столбец это значение COST на их пересечении. Алгоритм минимизирует сумму в 3-м столбце.

\*Если ваша входящая матрица являет собой не стоимости (различия), а выигрыши (сходства), переведите

\*их в стоимости (различия), обратив знак и затем прибавив константу, чтобы сделать значения

\*неотрицательными.

\*Скорость алгоритма выше, когда число различных значений в COST невелико, поэтому имеет смысл подумать

\*о предварительной дискретизации значений "стоимости".

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1000000.

matrix.

compute cost= rnd(uniform(10,8)\*100).

print cost /title 'Cost matrix'.

!KO\_hungar(cost%pairs).

print pairs /title 'Matching'.

print csum(pairs(:,3)) /title 'Overall cost (minimized)'.

end matrix.

### СПАРИВАНИЕ АЛГОРИТМОМ ХОПКРОФТА-КАРПА [!KO\_hopckarp]

\*/\*!KO\_hopckarp(adj%name)\*/\*.

\*Version 1.

\*Алгоритм Хопкрофта-Карпа (Hopcroft-Karp) так спаривает взаимно-однозначно элементы между двумя

\*множествами, имеющими между собой связи, чтобы максимизировать число пар, т.е. число учтенных связей.

\*В терминологии графов, он создает максимальное паросочетание (maximal matching) из двудольного

\*невзвешенного графа. (Отметим, что вы можете использовать для решения этой задачи и венгерский

\*алгоритм /\*!KO\_hungar\*/.)

\*Входящая матрица ADJ - это "список смежности" (adjacency list) размером n x m, где n и m - количества

\*элементов в двух множествах. Каждый ряд ADJ это элемент из множества "n", и он содержит номера

\*элементов множества "m", идущие слева направо утрамбованно. Например, пусть есть множества "n"

\*(1,2,...,n) и "m" (1,2,...,m=7), где, в частности, 2-й элемент из "n" имеет связи с элементами 1,4 и 5

\*множества "m". Тогда 2-й ряд ADJ должен выглядеть так: {1,4,5,0,0,0,0}. Порядок номеров не играет роли:

\*{4,5,1,0,0,0,0}, например, тоже корректно. Число столбцов ADJ должно быть m, поэтому пустоты добиты

\*нулями до 7. Список смежности соответствует двоичной матрице размером n x m, в которой единицы

\*обозначают наличие связи, и является другим способом ее представить. Если ваши входящие - двоичная

\*матрица, вы можете получить из нее список смежности функцией /\*!KO\_ram\*/.

\*Пустые (полностью из нулей) ряды в ADJ допускаются - они соответствуют изолированным, не имеющим связей,

\*элементам из "n", - но вы можете захотеть сначала удалить их в интересах быстродействия.

\*В ADJ должен быть хотя бы один ненулевой элемент.

\*Если n и m сильно различаются, выгоднее с точки зрения быстродействия иметь ADJ широкую (n<m), нежели

\*высокую (n>m).

\*Результат NAME - двухстолбцовая матрица, являющая максимальное паросочетание; 1-й столбец это элементы

\*множества "n", 2-й столбец - элементы множества "m"; каждый ряд это осуществленное спаривание.

\*Число рядов в NAME есть, т.о., величина паросочетания.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР. Хопкрофт–Карп и Венгерский алгоритм на одних и тех же данных.

set mxloops 1E6.

matrix.

compute n= 10.

compute m= 8.

compute mat= rnd(uniform(n,m)\*0.8). /\*Let this random binary NxM matrix represent

/\*a bipartite graph

print mat.

!KO\_ram(mat%IND%0%0%adj%name). /\*Turn it into corresponding NxM adjacency list

/\*!KO\_ram(mat%IND%0%1%adj%name). /\*[or might additionally randomize the order]

print adj.

!KO\_hopckarp(adj%match). /\*Perform matching by Hopcroft-Karp

print match. /\*The optimal (maximal) matching

print nrow(match). /\*Its size

\*-----.

!KO\_hungar(not mat%match2). /\*Now use Hungarian for the same task;

/\*Hungarian uses MAT as the input, and note that we reversed

/\*it (1->0, 0->1), to make Hungarian to maximize gain rather

/\*than minimize cost; also mind that Hungarian needs n>=m to

/\*process a matrix correctly

print match2. /\*The optimal (maximal) matching

/\*If the maximal matching is not unique for the input data,

/\*Hungarian and Hopcroft-Karp may return different matchings;

print csum(match2(:,3)=0). /\*but the size of the maximal matching is always the same

/\*(cost "0" in Hungarian corresponded to link "1" in MAT)

end matrix.

О скорости. Если данные сильно разреженные (т.е. между двумя множествами элементов связей весьма мало), Хопкрофт–Карп быстрее Венгерского алгоритма. В остальных случаях он как правило медленнее. И его скорость падает, когда n>>m.

### ПРОСТОЕ ЖАДНОЕ СПАРИВАНИЕ [!KO\_greedm]

\*/\*!KO\_greedm(mat%cg%name)\*/\*.

\*Version 1.

\*Простое жадное спаривание это пошаговый алгоритм, спаривающий элементы между двумя множествами так,

\*чтобы минимизировать (или максимизировать) сумму значений в парах. Это та же задача, что выполняет

\*венгерское спаривание (ф-ция /\*!KO\_hungar\*/), но /\*!KO\_greedm\*/ быстрее и не гарантирует глобального

\*оптимума в смысле итоговой суммы. Алгоритм находит наименьший (или наибольший - как затребовано)

\*элемент в матрице и записывает его ряд и столбец как пару (match), затем вырезает их из матрицы и

\*в оставшейся матрице опять находит наименьший элемент, спаривает столбец и ряд, удаляет их,

\*и так далее.

\*Входящие:

\*MAT - матрица размера не меньше чем 2x2 с любыми числами. Если число столбцов намного превышает число

\*рядов, возможно имеет смысл транспонировать ее сначала, это может дать выигрыш в скорости.

\*CG - цифра (не имя и не выражение) 0, 1 или 2 (цифру можно опционально взять в кавычки или апострофы).

\*Если "0", то функция будет искать минимальные значения, т.е. относиться к MAT как к стоимости (cost),

\*которую надо минимизировать в спаривании. Если "1", то функция будет искать максимальные значения,

\*т.е. относиться к MAT как к выигрышу (gain), который надо максимизировать в спаривании.

\*Если "2", то MAT это разреженная (sparse) неотрицательная gain-матрица: с положительными значениями

\*в качестве выигрышей и нулевыми элементами, запрещающими спаривание, хотя бы одно значение должно быть

\*больше 0.

\*Результат NAME: трехстолбцовая матрица, где ряды - это пары ряд-столбец, выбранные алгоритмом:

\*номер ряда входящей MAT записан в 1-м столбце NAME, номер спаренного с ним столбца входящей MAT записан

\*во 2-м столбце NAME, а значение, лежащее на их пересечении в MAT - в 3-м столбце NAME. Сумму в этом 3-м

\*столбце ф-ция и пытается минимизировать (максимизировать).

\*В случае CG="2" некоторые последние ряды в NAME могут быть пусты, поскольку разреженная MAT может

\*не дать спарить все ряды и столбцы.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1000000.

matrix.

compute gain= rnd(uniform(10,8)\*30).

print gain /title 'Gain matrix'.

!KO\_greedm(gain%'1'%pairs).

print pairs /title 'Matching'.

print csum(pairs(:,3)) /title 'Overall gain (attempted to maximize)'.

end matrix.

### ПРОСТОЕ ЖАДНОЕ СПАРИВАНИЕ (ДИСКРЕТНЫЕ ДАННЫЕ) [!KO\_greedm2]

\*/\*!KO\_greedm2(mat%cg%name)\*/\*.

\*Version 1.

\*Эта функция дает тот же результат, что /\*!KO\_greedm\*/, но она быстрее ее в ситуации, когда число

\*различных значений в MAT невелико по сравнению с размером MAT. Например, когда MAT это целые

\*числа из существенно ограниченного диапазона. И напротив, если различных значений в MAT много, как,

\*например, при континуальных данных, то /\*!KO\_greedm2\*/ будет медленнее, чем /\*!KO\_greedm\*/.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### АЛГОРИТМ ФЛОЙДА-УОРШАЛЛА: КРАТЧАЙШИЕ ПУТИ / ЛЕГЧАЙШИЕ ПРОХОДЫ [!KO\_flowar]

\*/\*!KO\_flowar(cost%task%path%name1%name2)\*/.

\*Version 1.

\*Берет квадратную матрицу COST с неотрицательными элементами и нулевой диагональю;

\*матрица может быть симметричной или асимметричной. Значения COST можно понимать как веса в графе

\*или связи в сети, и ряды/столбцы матрицы - это вершины графа или узлы в сети.

\*TASK (ключевое слово заглавными буквами, можно взять в кавычки или апострофы) - цель алгоритма:

\*"SHWAY" - найти, из каждой точки в каждую точку, кратчайший путь (Shortest path aka Shortest

\*way-модификация алгоритма). Значения COST принимаются за расстояния между точками:

\*COST(i,j) это расстояние из i в j.

\*Из каждой точки i в каждую точку j будет найден самый короткий путь. Результат NAME1 будет

\*содержать суммарную длину этого кратчайшего пути из всякой точки-ряда i во всякую точку-столбец j.

\*"EPASS" - найти, из каждой точки в каждую точку, путь из легчайших перегонов (Optimal routing

\*aka Easiest passes-модификация алгоритма). Значения COST принимаются за трудность преодоления

\*перегона из точки в точку: COST(i,j) это трудность из i в j.

\*Из каждой точки i в каждую точку j будет найден самый легкий путь:

\*все перегоны, из которых он будет состоять, минимально трудны. Результат NAME1 будет

\*содержать трудность самого трудного перегона на этом легчайшем пути из всякой точки-ряда i

\*во всякую точку-столбец j.

\*Сами найденные (т.е. оптимальные) пути, траектории, будут записаны в "свернутом виде" в матрицу NAME2.

\*Элемент (i,j) в NAME2 это номер точки, следующей за точкой i на всяком найденном пути, ведущем

\*к точке j как к конечной точке пути. См. ПРИМЕР, показывающий, как извлечь всю траекторию пути

\*из некоторой начальной точки U в некоторую конечную точку V=j.

\*Записывать траекторию или нет - зависит от аргумента PATH. Аргумент PATH - это цифра 1 или 0

\*(не имя и не выражение, цифру можно взять в кавычки или апострофы). Если 0, то траектории не будут

\*записаны и матрица NAME2 выйдет как нулевой скаляр.

\*Отсутствующие значения в COST и невозможный путь. Если во входящей матрице должно не быть связи

\*между некоторыми точками - например, перегона из точки i в точку j не существует, то поставьте в

\*ячейку (i,j) число X, не меньшее чем сумма всех существующих в матрице значений, т.е. X>=msum(COST).

\*Тогда, если в некоторой ячейке (U,V) NAME1 вы увидите число X, то значит, путь из начальной точки U

\*в конечную точку V - невозможен.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1E6.

matrix.

compute dist= rnd(uniform(8,8)\*50).

call setdiag(dist,0).

print dist /title 'Asymmetric Distance matrix'.

!KO\_flowar(dist%SHWAY%1%length%next). /\*Shortest ways will be searched for

print length /title 'Shortest path length from point-row to point-column'.

print next /title 'Paths are written here'.

\*Show extraction of a path.

compute u= 2. /\*Select two

compute v= 5. /\*arbitrary points

print u /title 'Extract the path starting from point U='.

print v /title 'and ending at point V=' /space= 0.

print length(u,v) /title '(the path length is:)' /space= 0.

compute path= u.

loop if u<>v.

-compute u= next(u,v).

-compute path= {path;u}.

end loop.

print path /title 'The path (point numbers):'.

end matrix.

### АЛГОРИТМ ФЛОЙДА-УОРШАЛЛА (СИММЕТРИЧНАЯ МАТРИЦА) [!KO\_sflowar]

\*/\*!KO\_sflowar(cost%task%name)\*/.

\*Version 1.

\*Эта функция тождественна /\*!KO\_flowar\*/, но рассчитана только на симметричную входящую

\*матрицу COST. Эта функция несколько быстрее, чем /\*!KO\_flowar\*/.

\*Аргумент PATH и результат NAME2 - не предусмотрены.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

### АЛГОРИТМ ДЕЙКСТРЫ: КРАТЧАЙШИЙ ПУТЬ / ЛЕГЧАЙШИЙ ПРОХОД [!KO\_dijkstra]

\*/\*!KO\_dijkstra(cost%start%stop%task%path%name1%name2)\*/.

\*Version 1.

\*Берет квадратную матрицу COST с неотрицательными элементами; матрица может быть симметричной или

\*асимметричной. Значения COST можно понимать как веса в графе или связи в сети, и ряды/столбцы

\*матрицы - это вершины графа или узлы в сети. Диагональные элементы COST игнорируются функцией

\*(они принимаются за 0).

\*START - скаляр; номер точки (вершины), из которой находить пути в другие точки.

\*TASK (ключевое слово заглавными буквами, можно взять в кавычки или апострофы) - цель алгоритма:

\*"SHWAY" - найти из точки START в каждую точку кратчайший путь (Shortest path aka Shortest

\*way-модификация алгоритма). Значения COST принимаются за расстояния между точками:

\*COST(i,j) это расстояние из i в j.

\*Из точки START в каждую точку j будет найден самый короткий путь. Результат NAME1 будет

\*содержать суммарную длину этого кратчайшего пути из точки-ряда START во всякую точку-столбец j.

\*"EPASS" - найти из точки START в каждую точку путь из легчайших перегонов (Optimal routing

\*aka Easiest passes-модификация алгоритма). Значения COST принимаются за трудность преодоления

\*перегона из точки в точку: COST(i,j) это трудность из i в j.

\*Из точки START в каждую точку j будет найден самый легкий путь:

\*все перегоны, из которых он будет состоять, минимально трудны. Результат NAME1 будет

\*содержать трудность самого трудного перегона на этом легчайшем пути из точки-ряда START

\*во всякую точку-столбец j.

\*Сами найденные (т.е. оптимальные) пути, траектории, будут записаны в "свернутом виде" в вектор NAME2.

\*Значение j-го элемента в NAME2 это номер точки, предшествующей точке j на всяком найденном пути,

\*ведущем к точке j как к конечной точке пути. См. ПРИМЕР, показывающий, как извлечь всю траекторию пути

\*из точки START в некоторую конечную точку V=j.

\*Записывать траекторию или нет - зависит от аргумента PATH. Аргумент PATH - это цифра 1 или 0

\*(не имя и не выражение, цифру можно взять в кавычки или апострофы). Если 0, то траектории не будут

\*записаны и NAME2 выйдет как нулевой скаляр.

\*Отсутствующие значения в COST и невозможный путь. Если во входящей матрице должно не быть связи

\*между некоторыми точками - например, перегона из точки i в точку j не существует, то поставьте в

\*ячейку (i,j) значение: -1. Тогда, если некоторый V-й элемент NAME1 оказался -1, то значит,

\*путь из START в конечную точку V - невозможен.

\*Аргумент STOP. Если вам нужны пути из START ко всем точкам, установите STOP на значение 0.

\*Если же вас интересует путь из START только в одну определенную точку, то укажите ее

\*номер тут. Тогда алгоритм прекратит работать, дойдя до этой точки; так можно сэкономить время.

\*Результат NAME1 выйдет тогда не вектором, а скаляром. STOP должно отличаться от START.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если вам нужны пути из всякой точки во всякую точку, используйте функцию /\*!KO\_flowar\*/.

ПРИМЕР. Кратчайший путь из одной вершины в одну другую вершину.

set mxloops 1E6.

matrix.

compute dist= rnd(uniform(8,8)\*50).

call setdiag(dist,0).

print dist /title 'Asymmetric Distance matrix'.

!KO\_dijkstra(dist%4%2%SHWAY%1%length%prev). /\*Shortest way from pt 4 to pt 2 will

/\*be searched for

print t(length) /title 'Shortest path length from point 4 to point 2'.

print t(prev) /title 'The path is written here'.

\*Show extraction of the path.

compute v= 2. /\*Indicate the STOP point: it was our specific goal in the function

compute path= 0. /\*(A prime to concatenate)

do if prev(v).

-loop if v.

- compute path= {v;path}.

- compute v= prev(v).

-end loop.

end if.

compute path= path(1:(nrow(path)-1)). /\*(Drop the priming zero)

print path /title 'The path (point numbers):'.

end matrix.

ПРИМЕР. Эквивалентность с алгоритмом Флойда–Уоршалла.

set mxloops 1E6.

matrix.

compute dist= rnd(uniform(8,8)\*50).

call setdiag(dist,0).

print dist /title 'Asymmetric Distance matrix'.

compute start= 1. /\*Let this be the start point

!KO\_dijkstra(dist%start%0%SHWAY%1%length%prev). /\*Shortest way from pt 1 to every

/\*other point

print t(length)

/title 'Shortest path lengths from point 1 to to every other point'.

print t(prev) /title 'The paths are written here in the "previous point" mode'.

\*Show extraction of the path to point, say, 5.

compute v= 5. /\*The end point: 5

print v /title 'Extract the path from the start point (1) to point V='.

print length(5) /title '(the path length is:)' /space= 0.

compute path= 0. /\*(A prime to concatenate)

do if prev(v).

-loop if v.

- compute path= {v;path}.

- compute v= prev(v).

-end loop.

end if.

compute path= path(1:(nrow(path)-1)). /\*(Drop the priming zero)

print path /title 'The path (point numbers):'.

\*----.

\*Now perform the same search by means of Floyd-Warshall algorithm.

print /title '---- Now try Floyd-Warshall ----'.

!KO\_flowar(dist%SHWAY%1%lengths%next). /\*Search shortest way from each point

/\*to each point

print t(lengths(start,:))

/title 'Shortest path lengths from point 1 to to every other point'.

/\*The lengths of the optimal routes returned by Floyd-Warshall are

/\*always the same as by Dijkstra, so the algorithms are equivalent

/\*optimizers

print t(next(start,:)) /title 'The paths are written here in the "next point" mode'.

\*Show extraction of a path.

compute u= start. /\*From our start point

compute v= 5. /\*to, say, point 5

print u /title 'Extract the path starting from point U='.

print v /title 'and ending at point V=' /space= 0.

print lengths(u,v) /title '(the path length is:)' /space= 0.

compute path= u.

loop if u<>v.

-compute u= next(u,v).

-compute path= {path;u}.

end loop.

print path /title 'The path (point numbers):'.

/\*The paths themselves (the points of the optimal routes) returned by

/\*Floyd-Warshall and Dijkstra may occasionally differ, because, though

/\*equivalent, these are different algorithms

end matrix.

Если ваш граф невзвешенный, кратчайшие пути из одной вершины в другие можно также получить функцией /\*!KO\_bfs\*/.

### АЛГОРИТМ МИНИМАЛЬНОГО ОСТОВНОГО ДЕРЕВА ПРИМА [!KO\_prim]

\*/\*!KO\_prim(mat%cg%start%name)\*/\*.

\*Version 1.

\*В однодольном взвешенном ненаправленном графе, заданным матрицей смежности MAT, находит

\*минимальное (или максимальное) остовное дерево, начиная с вершины START (скаляр, номер вершины).

\*MAT - квадратная n x n симметричная матрица. Ее диагональные элементы функция игнорирует.

\*Аргумент CG: задает понимание MAT. CG - цифра (не имя и не выражение) 0, 1 или 2

\*(цифру можно опционально взять в кавычки или апострофы).

\*Если CG=0, то MAT понимается как матрица стоимостей (cost) или расстояний между вершинами

\*полного взвешенного графа. Результат NAME это минимальное остовное дерево.

\*Если CG=1, то MAT понимается как матрица выигрышей (gain) или сходств между вершинами

\*полного взвешенного графа. Результат NAME это максимальное остовное дерево. В обоих этих случаях

\*значения в MAT могут иметь любой знак.

\*Если CG=2, то MAT понимается как разреженная (sparse) неотрицательная матрица выигрышей:

\*с положительными значениями в качестве выигрышей и нулевыми элементами, соответствующими

\*отсутствующим ребрам (неполный граф). Результат NAME тогда это максимальное остовное дерево из

\*вершины start в условиях неполного графа. Если неполный граф не является связным графом

\*(т.е. таким графом, где есть путь из всякой вершины во всякую вершину), то остовное дерево

\*покроет только его часть - подграф, содержащий вершину start, - и в результате в NAME

\*будут заполнены не все n-1 рядов (какие-то последние останутся пусты). Так алгоритм Прима

\*можно использовать с CG=2 для диагностики связности неполного графа.

\*Когда граф полный или неполный, но связный, остовное дерево покроет весь граф и величина

\*(сумма весов) дерева не зависит от выбора вершины start.

\*Результат NAME: имеет три столбца и n-1 рядов, ряды соответствуют ребрам, записанным в остовное

\*дерево. Первые два элемента в ряду это вершины i и j, смежные ребру, а третье значение это

\*вес ребра, т.е. значение (i,j) в MAT. Алгоритм Прима минимизирует (если MAT это cost-матрица) или

\*максимизирует (если MAT это gain-матрица) сумму в 3-м столбце NAME.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

\*Если ваша MAT это cost-матрица и граф неполный (некоторые ребра отсутствуют), переведите ее в

\*положительную gain-матрицу, обратив знак и прибавив положительную константу, а отсутствующие ребра

\*пометьте нулями; далее используйте CG=2.

ПРИМЕР. Полный граф стоимостей (или расстояний).

set mxloops 1E6.

matrix.

compute dist= uniform(8,8)\*50.

compute dist= rnd((dist+t(dist))/2).

call setdiag(dist,0).

print dist /title 'Square symmetric distance matrix'.

!KO\_prim(dist%0%4%mst). /\*Build minimum spanning tree, starting from point 4

print mst /title 'Minimum spanning tree includes:' /clab= 'Point' 'Point' 'Dist'.

print msum(mst(:,3)) /title 'The sum (minimized)'.

end matrix.

ПРИМЕР. Этот граф связный?

matrix.

compute g=

{0,1,0,0,1,0,0,1;

1,0,0,0,0,0,1,1;

0,0,0,0,0,1,0,0;

0,0,0,0,0,1,0,0;

1,0,0,0,0,0,1,0;

0,0,1,1,0,0,0,0;

0,1,0,0,1,0,0,0;

1,1,0,0,0,0,0,0}. /\*An incomplete (sparse) graph

/\*(all existing edge weights are equal, 1, in this example)

print g /title 'Square symmetric sparse gain matrix'.

!KO\_prim(g%2%1%mst). /\*Build maximum spanning tree, starting from point 1

print mst /title 'Maximum spanning tree:' /clab= 'Vert' 'Vert' 'Gain'.

/\*The algo exited before filling in all 7 rows of mst: G is disconnected

end matrix.

# ПОСЛЕДОВАТЕЛЬНОСТИ

### РАССТОЯНИЕ ЛЕВЕНШТЕЙНА (АЛГОРИТМ ВАГНЕРА-ФИШЕРА) [!KO\_levenshtein]

\*/\*!KO\_levenshtein(s1%s2%editcosts%name)\*/\*.

\*Version 1.

\*Расстояние Левенштейна между двумя последовательностями.

\*Это количество редакторских операций вставки, удаления и замещения, минимально необходимое для того,

\*чтобы превратить одну последовательность в другую.

\*S1 и S2 - две последовательности, векторы-ряды длиной n и m (векторы могут быть числовые

\*и/или текстовые).

\*EDITCOSTS - цены трех редакторских операций: вставки, удаления, замещения; вектор

\*из трех неотрицательных значений. Расстояние Левенштейна минимизирует суммарную цену всех

\*сделанных редакторских операций. 1-е значение - цена вставки в S1 (= удаления из S2);

\*2-е значение - цена удаления из S1 (= вставки в S2); 3-е значение - цена замещения

\*(замещение по эффекту эквивалентно удалению и вставке на это место).

\*Результат NAME - матрица размером m+1 x n+1. Ее последний (нижнеправый) элемент и есть

\*расстояние Левенштейна между S1 и S2.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1E6.

matrix.

compute s1= {'s','i','t','t','i','n','g'}.

compute s2= {'k','i','t','t','e','n'}.

!KO\_levenshtein(s1%s2%{1,1,1}%d).

print d(nrow(d),ncol(d)) /title 'Levenshtein distance'.

end matrix.

### ДЛИННЕЙШАЯ ОБЩАЯ ПОДПОСЛЕДОВАТЕЛЬНОСТЬ [!KO\_lcs]

\*/\*!KO\_lcs(s1%s2%subseq%name1%name2)\*/\*.

\*Version 1.

\*S1 и S2 - две последовательности, векторы-ряды длиной n и m (векторы могут быть числовые

\*и/или текстовые).

\*Результат NAME1 - матрица размером m+1 x n+1. Ее последний (нижнеправый) элемент есть

\*длина максимальной общей подпоследовательности между S1 и S2.

\*Сама же эта подпоследовательность - вектор-ряд NAME2.

\*SUBSEQ - цифра (не имя и не выражение) 1 или 0 (цифру можно взять в кавычки/апострофы).

\*Если 0, то максимальная общая подпоследовательность NAME2 не выдается, NAME2 будет скаляр 0.

\*Если длина максимальной общей подпоследоваетльности =0, NAME2 будет скаляр 0.

\*Для этой функции может требоваться предварительная установка предела числу циклов на достаточно большое

\*число командой SET MXLOOPS.

ПРИМЕР.

set mxloops 1E8.

matrix.

compute s1= {'s','i','t','t','i','n','g'}.

compute s2= {'k','i','t','t','e','n'}.

!KO\_lcs(s1%s2%1%mat%sub).

print mat.

compute length= mat(nrow(mat),ncol(mat)).

print length /title 'Length of the longest common subsequence'.

do if length>0. /\*Print the subsequence if it exists

print sub /format= a2 /title 'The longest common subsequence'.

end if.

end matrix.