

DataMinerXL - Microsoft Excel Add-In for Building Predictive Models

Version 2.00

www.DataMinerXL.com

Feb 2, 2021

Contents

1	Overview	1
1.1	Introduction	1
1.2	Installation of Add-Ins	1
1.3	Some Excel Tips	3
1.4	Function List	5
1.4.1	Utility Functions	5
1.4.2	Data Manipulation Functions	5
1.4.3	Basic Statistical Functions	6
1.4.4	Modeling Functions for All Models	6
1.4.5	Weight of Evidence Transformation Functions	7
1.4.6	Principal Component Analysis and Factor Analysis Functions	7
1.4.7	Linear Regression Functions	7
1.4.8	Partial Least Square Regression Functions	8
1.4.9	Logistic Regression Functions	8
1.4.10	Time Series Analysis Functions	8
1.4.11	Linear and Quadratic Discriminant Analysis Functions	9
1.4.12	Survival Analysis Functions	9
1.4.13	Correspondence Analysis Functions	9
1.4.14	Naive Bayes Classifier Functions	9
1.4.15	Tree-Based Model Functions	10
1.4.16	Clustering and Segmentation Functions	10
1.4.17	Neural Network Functions	10
1.4.18	Support Vector Machine Functions	10
1.4.19	Optimization Functions	10
1.4.20	Portfolio Optimization Functions	11
1.4.21	Control Theory Functions	11
1.4.22	Matrix Operation Functions	11
1.4.23	Fast Fourier Transform Functions	12

1.4.24	Numerical Integration Functions	13
1.4.25	Probability Functions	13
1.4.26	Excel Built-in Statistical Distribution Functions	14
1.5	Sample Spreadsheets	14
2	Utility Functions	17
2.1	version	17
2.2	function_list	17
3	Data Manipulation Functions	19
3.1	variable_list	19
3.2	subset	20
3.3	data_info	20
3.4	data_lookup	21
3.5	data_save	21
3.6	data_save_tex	22
3.7	data_load	22
3.8	data_partition	23
3.9	data_sort	23
3.10	data_fill	24
3.11	sort_file	24
3.12	merge_tables	25
3.13	rank_items	25
3.14	split_str	26
4	Basic Statistical Functions	27
4.1	ranks	28
4.2	ranks_from_file	28
4.3	freq	29
4.4	freq_from_file	29
4.5	freq_2d	30
4.6	freq_2d_from_file	30
4.7	means	31
4.8	means_from_file	31
4.9	univariate	32
4.10	univariate_from_file	32
4.11	percentiles	32
4.12	summary	33

4.13	summary_from_file	34
4.14	binning	35
4.15	QQ_plot	36
4.16	variable_corr_select	36
4.17	poly_roots	37
4.18	poly_prod	37
4.19	Lagrange_interpolation	38
4.20	three_moment_match_to_SLN	39
4.21	set	39
4.22	set_union	40
4.23	set_intersection	40
4.24	set_difference	41
5	Modeling Functions for All Models	43
5.1	model_bin_eval	43
5.2	model_bin_eval_from_file	44
5.3	model_cont_eval	44
5.4	model_cont_eval_from_file	45
5.5	model_eval	46
5.6	model_eval_from_file	47
5.7	model_score	48
5.8	model_score_node	49
5.9	model_score_from_file	49
5.10	model_save_scoring_code	50
6	Weight of Evidence Transformation Functions	53
6.1	woe_xcont_ybin	53
6.2	woe_xcont_ybin_from_file	54
6.3	woe_xcont_ycont	55
6.4	woe_xcont_ycont_from_file	56
6.5	woe_xcat_ybin	57
6.6	woe_xcat_ybin_from_file	58
6.7	woe_xcat_ycont	58
6.8	woe_xcat_ycont_from_file	59
6.9	woe_transform	60
6.10	woe_transform_from_file	60
7	Principal Component Analysis and Factor Analysis Functions	63

7.1	PCA	63
7.2	factor_analysis	63
8	Linear Regression Functions	65
8.1	linear_reg	65
8.2	linear_reg_from_file	66
8.3	linear_reg_forward_select	67
8.4	linear_reg_forward_select_from_file	68
8.5	linear_reg_score_from_coefs	68
8.6	linear_reg_piecewise	69
8.7	linear_reg_piecewise_from_file	69
8.8	poly_reg	70
9	Partial Least Square Regression Functions	73
9.1	pls_reg	73
9.2	pls_reg_from_file	74
10	Logistic Regression Functions	75
10.1	logistic_reg	75
10.2	logistic_reg_from_file	76
10.3	logistic_reg_forward_select	77
10.4	logistic_reg_forward_select_from_file	77
10.5	logistic_reg_score_from_coefs	78
11	Time Series Analysis Functions	79
11.1	ts_acf	80
11.2	ts_pacf	80
11.3	ts_ccf	81
11.4	Box_white_noise_test	81
11.5	Mann_Kendall_trend_test	82
11.6	ADF_test	83
11.7	ts_diff	84
11.8	ts_sma	85
11.9	lowess	85
11.10	natural_cubic_spline	86
11.11	garch	86
11.12	stochastic_process	87
11.13	stochastic_process_simulate	88

11.14Holt_Winters	89
11.15Holt_Winters_forecast	90
11.16HP_filter	92
11.17arima	92
11.18sarima	94
11.19arima_forecast	95
11.20sarima_forecast	96
11.21arima_simulate	97
11.22sarima_simulate	98
11.23arma_to_ma	100
11.24arma_to_ar	100
11.25acf_of_arma	101
12 Linear and Quadratic Discriminant Analysis Functions	103
12.1 LDA	103
12.2 QDA	104
13 Survival Analysis Functions	107
13.1 Kaplan_Meier	107
14 Correspondence Analysis Functions	109
14.1 corresp_analysis	109
15 Naive Bayes Classifier Functions	111
15.1 naive_bayes_classifier	111
15.2 naive_bayes_classifier_from_file	111
16 Tree-Based Model Functions	113
16.1 tree	113
16.2 tree_from_file	114
16.3 tree_boosting_logistic_reg	115
16.4 tree_boosting_logistic_reg_from_file	116
16.5 tree_boosting_ls_reg	117
16.6 tree_boosting_ls_reg_from_file	118
17 Clustering and Segmentation Functions	119
17.1 k_means	119
17.2 k_means_from_file	120
17.3 cmds	120

17.4 mds	121
18 Neural Network Functions	123
18.1 neural_net	123
18.2 neural_net_from_file	124
19 Support Vector Machine Functions	127
19.1 svm	127
19.2 svm_from_file	128
20 Optimization Functions	131
20.1 linear_prog	131
20.2 quadratic_prog	132
20.3 lcp	133
20.4 nls_solver	133
20.5 diff_evol_solver	134
20.6 diff_evol_min_solver	135
20.7 diff_evol_nls_solver	135
20.8 transportation_solver	136
20.9 assignment_solver	137
20.10netflow_solver	138
20.11maxflow_solver	138
20.12shortest_path_solver	139
21 Portfolio Optimization Functions	141
21.1 efficient_frontier	141
21.2 Black_Litterman	142
22 Control Theory Functions	145
22.1 pole_placement	145
23 Matrix Operation Functions	147
23.1 matrix_random	148
23.2 matrix_cov	149
23.3 matrix_cov_from_file	149
23.4 matrix_corr	150
23.5 matrix_corr_from_file	150
23.6 matrix_corr_from_cov	151
23.7 matrix_cov_from_corr	151

23.8 matrix_stdev_from_cov	151
23.9 matrix_prod	152
23.10matrix_directprod	152
23.11matrix_elementprod	153
23.12matrix_plus	153
23.13matrix_minus	154
23.14matrix_I	154
23.15matrix_t	154
23.16matrix_diag	155
23.17matrix_tr	155
23.18matrix_inv	156
23.19matrix_pinv	156
23.20matrix_complex_pinv	157
23.21matrix_solver	157
23.22matrix_tridiagonal_solver	158
23.23matrix_pentadiagonal_solver	158
23.24matrix_Sylvester_solver	159
23.25matrix_chol	160
23.26matrix_sym_eigen	160
23.27matrix_eigen	161
23.28matrix_complex_eigen	161
23.29matrix_svd	162
23.30matrix_LU	162
23.31matrix_QR	163
23.32matrix_complex_QR	163
23.33matrix_Schur	164
23.34matrix_complex_Schur	164
23.35matrix_sweep	165
23.36matrix_det	166
23.37matrix_exp	166
23.38matrix_complex_exp	167
23.39matrix_distance	167
23.40matrix_freq	168
23.41matrix_from_vector	168
23.42matrix_to_vector	169
23.43matrix_decimal_to_fraction	169

24 Fast Fourier Transform Functions	171
24.1 FFT	171
24.2 IFFT	172
25 Numerical Integration Functions	175
25.1 gauss_legendre	175
25.2 gauss_laguerre	176
25.3 gauss_hermite	176
25.4 integral	177
25.5 function_eval	178
25.6 prime_numbers	178
25.7 Halton_numbers	178
25.8 Sobol_numbers	179
25.9 Latin_hypcube	179
26 Probability Functions	181
26.1 prob_normal	181
26.2 prob_normal_inv	182
26.3 prob_normal_table	182
26.4 prob_t	183
26.5 prob_t_inv	183
26.6 prob_t_table	183
26.7 prob_chi	184
26.8 prob_chi_inv	184
26.9 prob_chi_table	185
26.10 prob_f	185
26.11 prob_f_inv	186
26.12 prob_f_table	186
26.13 Cornish_Fisher_expansion	186
27 Excel Built-in Statistical Distribution Functions	189
References	191
Index	192

Chapter 1

Overview

1.1 Introduction

This document describes DataMinerXL software, a Microsoft Excel add-in for building predictive models.

Add-in XLL is a DLL (Dynamic-Link Library) designed for Microsoft Excel. The algorithms in DataMinerXL library are implemented in C++. It serves as a core engine while Excel is focused on its role in creating a neat presentation or layout for input/output as a familiar user interface. By combining the strengths of both C++ and Excel, the calculation-intensive routines implemented in C++ are integrated into the convenient Excel environment. After the add-in is installed and loaded into Excel the functions in the add-in can be used exactly the same way as the built-in functions in Excel.

In the following, we first explain how to install add-ins and then introduce some tips of using Excel. The remaining of this document describes the details of each function in the DataMinerXL software. The theories and algorithms behind this software can be found in the book "Foundations of Predictive Analysis" in the [References](#).

1.2 Installation of Add-Ins

Q: How to install add-ins?

A: There are two add-ins in DataMinerXL software, DataMinerXL.xll and DataMinerXL_Utility.xla. The following steps will add add-ins in Excel.

For Excel 2010:

1. Open Excel, click the "File" menu and then click the "Options" button
2. Click the "Add-Ins" tab in the left pane and then click "Go..." button at the bottom of the window
3. The "Add-Ins" dialog box appears. Select/Check the add-in file you want to add from the "Add-Ins Available:" drop-down list or click "Browse..." to the folder you place the add-in files
4. Click the OK button(s)

For Excel 2007:

1. Open Excel, click the "Office Button" and then click the "Excel Options" button

2. Click the "Add-Ins" tab in the left pane and then click "Go..." button at the bottom of the window
3. The "Add-Ins" dialog box appears. Select/Check the add-in file you want to add from the "Add-Ins Available:" drop-down list or click "Browse..." to the folder you place the add-in files
4. Click the OK button(s)

For Excel 2003 or earlier versions:

1. Open Excel, under "Tools" menu, select "Add-Ins"
2. The "Add-Ins" dialog box appears. Select/Check the add-in file you want to add from the "Add-Ins Available:" drop-down list or click "Browse..." to the folder you place the add-in files
3. Click the OK button(s)

Q: How to remove or delete an add-in?

A: The following steps will remove an add-in in Excel.

For Excel 2010:

1. Find the add-in file you want to remove, rename the file or delete the file if you do not want it permanently
2. Open Excel, click the "File" menu and then click the "Options" button
3. Click the "Add-Ins" tab in the left pane and select the add-in you want to remove. Click "Go..." button at the bottom of the window
4. The "Add-Ins" dialog box appears. Uncheck the add-in file you want to remove from the "Add-Ins Available:" drop-down list
5. An alert dialog box appears "Cannot find add-in.... Delete from list?". Click "Yes"

For Excel 2007:

1. Find the add-in file you want to remove, rename the file or delete the file if you do not want it permanently
2. Open Excel, click the "Office Button" and then click the "Excel Options" button
3. Click the "Add-Ins" tab in the left pane and select the add-in you want to remove. Click "Go..." button at the bottom of the window
4. The "Add-Ins" dialog box appears. Uncheck the add-in file you want to remove from the "Add-Ins Available:" drop-down list
5. An alert dialog box appears "Cannot find add-in.... Delete from list?". Click "Yes"

For Excel 2003 or earlier versions:

1. Find the add-in file you want to remove, rename the file or delete the file if you do not want it permanently
2. Open Excel, under "Tools" menus, select "Add-Ins"
3. The "Add-Ins" dialog box appears. Uncheck the add-in file you want to remove from the "Add-Ins Available:" drop-down list
4. An alert dialog box appears "Cannot find add-in.... Delete from list?". Click "Yes"

1.3 Some Excel Tips

Q: How to determine whether I have 32-bit or 64-bit Excel?

A: For Excel 2013 or newer versions: Click the "File" menu, then click "Account", and click "About Excel". The version and bit-level of Excel will be displayed in the top line of the window.

For Excel 2010: Click the "File" menu and then click the "Help" button. The version and bit-level of Excel will appear under "About Microsoft Excel".

For Excel 2007 or earlier versions: It is 32-bit.

Q: How to set up manual calculation in Excel?

A: In Excel 2010: Open Excel, click the "File" menu and then click the "Options" button. Click the "Formulas" tab in the left pane and then select "Manual" for "Calculation options" as shown below.

For Excel 2007: Open Excel, click the "Office Button" and then click the "Excel Options" button. Click the "Formulas" tab in the left pane and then select "Manual" for "Calculation options" as shown below.

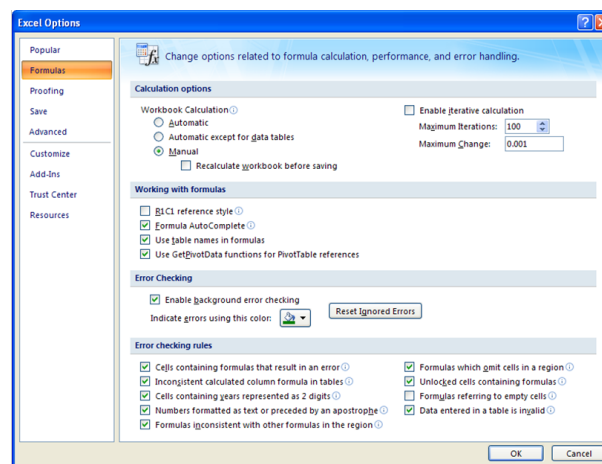


Figure 1.1: Excel Option

For Excel 2003 or earlier versions: Open Excel, under "Tools" menus, select "Options...". Select "Calculation" tab and then select "Manual".

Q: How to use functions in a cell?

A: Functions can be accessed via either "insert function", function wizard in formula bar or immediate prompt while entering in function names in cells. For example type "`=sqrt(2)`" in a cell.

Q: What is an array function?

A: An array function outputs more than one cell in spreadsheet. "`sqrt()`" function is not an array function, since it only outputs one number, the squared root of a given number. The Excel built-in function "`minverse()`" is an array function for matrix inverse. It outputs an inverse matrix in multiple cells. For a 3 by 3 input matrix, the output is 3 by 3 matrix.

Q: How to use array function?

A: For example, "`minverse()`" is an Excel built-in array function for matrix inverse and its output size depends on the input matrix.

1. First type the formula in a cell and complete all inputs. Hit "Enter" key. Now you have the output in

one cell.

2. Hold down the left-button of the mouse in the output cell and pull the mouse to right if you want to have more columns and pull the mouse down if you want to have more rows. You always hold down the left button of the mouse in this step. Release the left-button of the mouse. Now you have selected more than one cell.
3. You can finish step 2 above using keyboard without using mouse. Click the first cell in the output. Hold SHIFT key by the left hand and use the right hand to hit arrow keys "LEFT", "RIGHT", "UP", "DOWN" to select the cells you want to select.
4. Click in the formula bar and enter CTRL+SHIFT+ENTER to complete the command. Now you will see more output.
5. If you want to enlarge the output area, just select more cells as shown in the steps above.
6. You cannot shrink the output area. If you try to shrink the output by selecting less rows or columns, you will prompt the following alert dialog box. Hit "Esc" key to escape any trouble you may have.
7. If you do want to shrink the output area, delete the formula and redo. However, you can type **CTRL+Q** to expand or shrink the output area if you install DataMinerXL_Utility.xla.

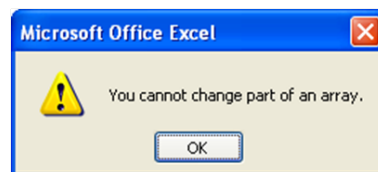


Figure 1.2: Error Prompt

Q: What are the most useful function keys?

A: The most useful function keys are:

- **Esc** When you have any troubles, just hit "Esc" key to escape the troubles.
- **CTRL+Q** Expands the array formula to the right size. You do not need to manually select the cells. It can expand or shrink the output area to the right size. You must install DataMinerXL_Utility.xla to have this hotkey.
- **CTRL+SHIFT+ENTER** When you run array formula, first click in any cell in formula cells, then click formula bar. Enter this command.
- **SHIFT+F9** Calculates the active worksheet. If SHIFT+F9 does not re-calculate the active worksheet, select the whole sheet and replace "=" with "=" as shown in the following dialogbox.
- **CTRL+ALT+F9** Calculates all worksheets in all open workbooks.
- **CTRL+‘** Shows formula in the active worksheet. Enter this command again to turn off.
- **CTRL+SHIFT+A** When you finish type formula, type CTRL+SHIFT+A to show all inputs.

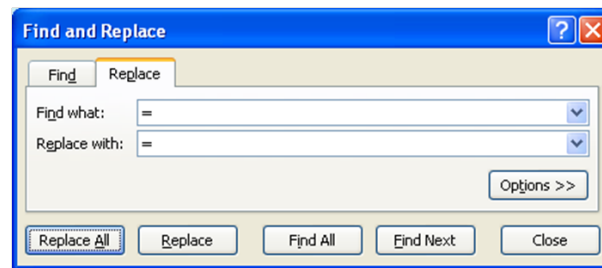


Figure 1.3: Replace All

Q: How to show all functions in an add-in?

A: Select an empty cell. Click fx in the formula bar and it will show "Insert Function" dialog box. From "Select a category" dropdown menu, select a category "DataMinerXL". Then you will see a list of all functions in this add-in. Alternatively, for DataMinerXL software, you can type the function "function_list()" to show all functions in this add-in: type "function_list()" in a cell, hit "ENTER" key, and type CTRL+Q.

1.4 Function List

1.4.1 Utility Functions

version Displays the version number and build date/time of DataMinerXL software

function_list Lists all functions in DataMinerXL software

1.4.2 Data Manipulation Functions

variable_list Lists the variable names in an input data file

subset Gets a subset of a data table

data_info Prints the number of records and variable names from an input data file

data_lookup Looks up data by matching multiple keys

data_save Saves a data table into a file

data_save_tex Saves a data table into a file in TEX format

data_load Loads a data table from a file

data_partition Gets random data partition

data_sort Sorts a data table given keys and orders

data_fill Fills missing data elements with a given value

sort_file Sorts a data file given keys and orders

merge_tables Merge two data tables by a single numerical key

rank_items Selects the items from the ranks by keys

split_str Splits a text string into a vector

1.4.3 Basic Statistical Functions

ranks Creates 1-based ranks of data points given a column of data

ranks_from_file Creates 1-based ranks of data points given a data file

freq Creates frequency tables given a data table

freq_from_file Creates frequency tables given a data file

freq_2d Creates a frequency cross-table for two variables given a data table

freq_2d_from_file Creates a frequency cross-table for two variables given a data file

means Generates basic statistics: sum, average, standard deviation, minimum, and maximum given a data table

means_from_file Generates basic statistics: sum, average, standard deviation, minimum, and maximum given a data file

univariate Generates univariate statistics given a data table

univariate_from_file Generates univariate statistics given a data file

percentiles Calculates p-th percentile of values in each subgroup

summary Generates descriptive statistics in classes given a data table

summary_from_file Generates descriptive statistics in classes given a data file

binning Creates equal interval binning given a column of data table

QQ_plot Tests normality of a univariate sample

variable_corr_select Selects variables by removing highly correlated variables

poly_roots Finds all roots given real coefficients of a polynomial

poly_prod Computes the coefficients of the product of two polynomials

Lagrange_interpolation Performs Lagrange polynomial interpolation given data points

three_moment_match_to_SLN Performs three moment match to shifted lognormal distribution

set Creates a set given a string/number matrix

set_union Creates a set from union of two sets

set_intersection Creates a set from intersection of two sets

set_difference Creates a set from difference of two sets

1.4.4 Modeling Functions for All Models

model_bin_eval Evaluates a binary target model given a column of actual values and a column of predicted values

model_bin_eval_from_file Evaluates a binary target model given a data file, a name of actual values, and a name of predicted values

model_cont_eval Evaluates a continuous target model given a column of actual values and a column of predicted values

model_cont_eval_from_file Evaluates a continuous target model given a data file, a name of actual values, and a name of predicted values

model_eval Evaluates model performance given a model and a data table

model_eval_from_file Evaluates model performance given a model and a data file

model_score Scores a population given a model and a data table

model_score_from_file Scores a population given a model and a data file

model_save_scoring_code Saves the scoring code of a given model to a file

1.4.5 Weight of Evidence Transformation Functions

woe_xcont_ybin Generates weight of evidence (WOE) of continuous independent variables and a binary dependent variable given a data table

woe_xcont_ybin_from_file Generates weight of evidence (WOE) of continuous independent variables and a binary dependent variable given a data file

woe_xcont_ycont Generates weight of evidence (WOE) of continuous independent variables and a continuous dependent variable given a data table

woe_xcont_ycont_from_file Generates weight of evidence (WOE) of continuous independent variables and a continuous dependent variable given a data file

woe_xcat_ybin Generates weight of evidence (WOE) of categorical independent variables and a binary dependent variable given a data table

woe_xcat_ybin_from_file Generates weight of evidence (WOE) of categorical independent variables and a binary dependent variable given a data file

woe_xcat_ycont Generates weight of evidence (WOE) of categorical independent variables and a continuous dependent variable given a data table

woe_xcat_ycont_from_file Generates weight of evidence (WOE) of categorical independent variables and a continuous dependent variable given a data file

woe_transform Performs weight of evidence (WOE) transformation given a WOE model and a data table

woe_transform_from_file Performs weight of evidence (WOE) transformation given a WOE model and a data file

1.4.6 Principal Component Analysis and Factor Analysis Functions

PCA Performs principal component analysis

factor_analysis Performs factor analysis

1.4.7 Linear Regression Functions

linear_reg Builds a linear regression model given a data table

linear_reg_from_file Builds a linear regression model given a data file

linear_reg_forward_select Builds a linear regression model by forward selection given a data table

linear_reg_forward_select_from_file Builds a linear regression model by forward selection given a data file

linear_reg_score_from_coefs Scores a population from the coefficients of a linear regression model given a data table

linear_reg_piecewise Builds a two-segment piecewise linear regression model for each variable given a data table

linear_reg_piecewise_from_file Builds a two-segment piecewise linear regression model for each variable given a data file

poly_reg Builds a polynomial regression model given a data table

1.4.8 Partial Least Square Regression Functions

pls_reg Builds a partial least square regression model given a data table

pls_reg_from_file Builds a partial least square regression model given a data file

1.4.9 Logistic Regression Functions

logistic_reg Builds a logistic regression model given a data table

logistic_reg_from_file Builds a logistic regression model given a data file

logistic_reg_forward_select Builds a logistic regression model by forward selection given a data table

logistic_reg_forward_select_from_file Builds a logistic regression model by forward selection given a data file

logistic_reg_score_from_coefs Scores a population from the coefficients of a logistic regression model given a data table

1.4.10 Time Series Analysis Functions

ts_acf Calculates the autocorrelation functions (ACF) given a data table

ts_pacf Calculates the partial autocorrelation functions (PACF) given a data table

ts_ccf Calculates the cross correlation functions (CCF) given two data tables

Box_white_noise_test Tests if a time series is a white noise by Box-Ljung or Box-Pierce test

Mann_Kendall_trend_test Tests if a time series has a trend

ADF_test Tests whether a unit root is in a time series using Augmented Dickey-Fuller (ADF) test

ts_diff Calculates the differences given lag and order

ts_sma Calculates the simple moving average (SMA) of a time series data

lowess Performs locally weighted scatterplot smoothing (lowess)

natural_cubic_spline Performs natural cubic spline

garch Estimates the parameters of GARCH(1, 1) model

stochastic_process Estimates the parameters of a stochastic process: normal, lognormal, or shifted log-normal

stochastic_process_simulate Simulates a stochastic process: normal, lognormal, or shifted lognormal

Holt_Winters Performs Holt-Winters exponential smoothing

Holt_Winters_forecast Performs forecast given a Holt-Winters exponential smoothing

HP_filter Performs the Hodrick-Prescott filter for a time-series data

arima Builds an ARIMA model

sarima Builds a seasonal ARIMA (SARIMA) model

arima_forecast Performs forecast given an ARIMA model

sarima_forecast Performs forecast given a seasonal ARIMA (SARIMA) model

arima_simulate Simulates an ARIMA process

sarima_simulate Simulates a seasonal ARIMA (SARIMA) process

arma_to_ma Converts an ARMA process to a pure MA process

arma_to_ar Converts an ARMA process to a pure AR process

acf_of_arma Calculates the autocorrelation functions (ACF) of an ARMA process

1.4.11 Linear and Quadratic Discriminant Analysis Functions

LDA Performs the linear discriminant analysis

QDA Performs the quadratic discriminant analysis

1.4.12 Survival Analysis Functions

Kaplan_Meier Performs Kaplan-Meier survival analysis

1.4.13 Correspondence Analysis Functions

corresp_analysis Performs simple correspondence analysis for a two-way cross table

1.4.14 Naive Bayes Classifier Functions

naive_bayes_classifier Builds a naive Bayes classification model given a data table

naive_bayes_classifier_from_file Builds a naive Bayes classification model given a data file

1.4.15 Tree-Based Model Functions

tree Builds a regression or classification tree model given a data table

tree_from_file Builds a regression or classification tree model given a data file

tree_boosting_logistic_reg Builds a logistic boosting tree model given a data table

tree_boosting_logistic_reg_from_file Builds a logistic boosting tree model given a data file

tree_boosting_ls_reg Builds a least square boosting tree model given a data table

tree_boosting_ls_reg_from_file Builds a least square boosting tree model given a data file

1.4.16 Clustering and Segmentation Functions

k_means Performs K-means clustering analysis given a data table

k_means_from_file Performs K-means clustering analysis given a data file

cmds Performs classical multi-dimensional scaling

mds Performs multi-dimensional scaling by Sammon's non-linear mapping

1.4.17 Neural Network Functions

neural_net Builds a neural network model given a data table

neural_net_from_file Builds a neural network model given a data file

1.4.18 Support Vector Machine Functions

svm Builds a support vector machine (SVM) model given a data table

svm_from_file Builds a support vector machine (SVM) model given a data file

1.4.19 Optimization Functions

linear_prog Solves a linear programming problem: $f(x) = c x$

quadratic_prog Solves a quadratic programming problem: $f(x) = c x + 0.5x^T H x$

lcp Solves a linear complementarity programming problem

nls_solver Solves a nonlinear least-square problem using the Levenberg-Marquardt algorithm

diff_evolver Solves a minimization problem given a function and lower/upper bounds of variables using differential evolution solver

diff_evolver_min Solves a minimization problem given a function, lower/upper bounds of variables, and data table using differential evolution solver

diff_evolver_nls Solves a nonlinear least squares problem given a function and lower/upper bounds of variables using differential evolution solver

transportation_solver Solves a transportation problem: find the number of units to ship from each source to each destination that minimizes or maximizes the total cost

assignment_solver Solves an assignment problem: find the optimal assignment that minimizes or maximizes the total cost

netflow_solver Solves a minimum or maximum cost network flow problem: to find optimal flows that minimize or maximize the total cost

maxflow_solver Solves a maximum flow problem: to find optimal flows that maximize the total flows from the start node to the end node

shortest_path_solver Solves the shortest path problem: to find the shortest path from the start node to the end node

1.4.20 Portfolio Optimization Functions

efficient_frontier Finds the efficient frontier for portfolios

Black_Litterman Finds posterior expected returns and covariance matrix using the Black-Litterman Model

1.4.21 Control Theory Functions

pole_placement Calculates the gains K for the pole placement

1.4.22 Matrix Operation Functions

matrix_random Generates a random matrix from a uniform distribution $U(0, 1)$ or a standard normal distribution $N(0, 1)$

matrix_cov Computes the covariance matrix given a data table

matrix_cov_from_file Computes the covariance matrix given a data file

matrix_corr Computes the correlation matrix given a data table

matrix_corr_from_file Computes the correlation matrix given a data file

matrix_corr_from_cov Computes the correlation matrix from a covariance matrix

matrix_cov_from_corr Computes the covariance matrix from a correlation matrix and a stdev vector

matrix_stdev_from_cov Computes the standard deviation vector from a covariance matrix

matrix_prod Computes the product of two matrices, one matrix could be a number

matrix_directprod Computes the direct product of two matrices

matrix_elementprod Computes the elementwise product of two matrices

matrix_plus Adds two matrices with the same dimension: $\text{matrix1} + \text{matrix2}$

matrix_minus Subtracts two matrices with the same dimension: $\text{matrix1} - \text{matrix2}$

matrix_I Creates an identity matrix

matrix_t Returns the transpose matrix of a matrix

matrix_diag Creates a diagonal matrix from a matrix or a vector

matrix_tr Returns the trace of a matrix

matrix_inv Computes the inverse of a square matrix

matrix_pinv Computes the pseudoinverse of a real matrix

matrix_complex_pinv Computes the pseudoinverse of a complex matrix

matrix_solver Solves a system of linear equations $Ax = B$

matrix_tridiagonal_solver Solves a system of tridiagonal linear equations $Ax = B$

matrix_pentadiagonal_solver Solves a system of pentadiagonal linear equations $Ax = B$

matrix_Sylvester_solver Solves a Sylvester equation $Ax + xB = C$

matrix_chol Computes the Cholesky decomposition of a symmetric positive semi-definite matrix

matrix_sym_eigen Computes the eigenvalue-eigenvector pairs of a symmetric real matrix

matrix_eigen Computes the eigenvalue-eigenvector pairs of a square real matrix

matrix_complex_eigen Computes the eigenvalue-eigenvector pairs of a square complex matrix

matrix_svd Computes the singular value decomposition (SVD) of a matrix

matrix_LU Computes the LU decomposition of a square matrix

matrix_QR Computes the QR decomposition of a square real matrix

matrix_complex_QR Computes the QR decomposition of a square complex matrix

matrix_Schur Computes the Schur decomposition a square real matrix

matrix_complex_Schur Computes the Schur decomposition a square complex matrix

matrix_sweep Sweeps a matrix given indexes

matrix_det Computes the determinant of a square matrix

matrix_exp Computes the matrix exponential of a square matrix

matrix_complex_exp Computes the matrix exponential of a square complex matrix

matrix_distance Computes the distance matrix given a data table

matrix_freq Creates a frequency table given a string matrix

matrix_from_vector Converts a matrix from a vector

matrix_to_vector Converts a matrix into a column vector

matrix_decimal_to_fraction Converts each decimal to a fraction for each element of a matrix if possible

1.4.23 Fast Fourier Transform Functions

FFT Performs fast Fourier transform

IFFT Performs inverse fast Fourier transform

1.4.24 Numerical Integration Functions

gauss_legendre Generates the abscissas and weights of the Gauss-Legendre n-point quadrature formula

gauss_laguerre Generates the abscissas and weights of the Gauss-Laguerre n-point quadrature formula

gauss_hermite Generates the abscissas and weights of the Gauss-Hermite n-point quadrature formula

integral Evaluates an 1-D integration of a function given lower and upper boundaries

function_eval Evaluates a function given arguments

prime_numbers Gets prime numbers

Halton_numbers Gets Halton numbers

Sobol_numbers Gets Sobol numbers

Latin_hypercube Gets Latin hypercube sampling

1.4.25 Probability Functions

prob_normal Computes the cumulative probability given z for the standard normal distribution: $N(z) = \text{Prob}(Z < z)$

prob_normal_inv Computes the percentile of a standard normal distribution: $\text{Prob}(Z < z) = p$

prob_normal_table Generates a table of the cumulative probabilities for the standard normal distribution: $N(z) = \text{Prob}(Z < z)$

prob_t Computes the cumulative probability given t and the degree of freedom for the Student's t distribution: $\text{Prob}(t_n < t)$

prob_t_inv Computes the percentile for the Student's t distribution: $\text{Prob}(t_n < t) = p$

prob_t_table Generates a table of the percentiles given a set of degrees of freedom and a set of probabilities for the Student's t distribution: $\text{Prob}(t_n < t) = P$

prob_chi Computes the cumulative probability given c and the degree of freedom for the Student's distribution: $\text{Prob}(X^2 < c)$

prob_chi_inv Computes the percentile for the Chi-Squared distribution: $\text{Prob}(X^2 < c) = p$

prob_chi_table Generates a table of the percentiles given a set of degrees of freedom and a set of probabilities for the Chi-Squared distribution: $\text{Prob}(X^2 < c) = P$

prob_f Computes the cumulative probability given f and the degree of freedom for the F-distribution: $\text{Prob}(F(df1, df2) < f)$

prob_f_inv Computes the percentile for the F-distribution: $\text{Prob}(F(df1, df2) < f) = p$

prob_f_table Generates a table of the percentiles given a set of degrees of freedom and a probability for the F-distribution: $\text{Prob}(F(df1, df2) < f) = p$

Cornish_Fisher_expansion Computes the percentile of a distribution with a skewness and an excess kurtosis by Cornish-Fisher expansion

1.4.26 Excel Built-in Statistical Distribution Functions

BETADIST Returns the beta cumulative distribution function

BETAINV Returns the inverse of the cumulative distribution function for a specified beta distribution

BINOMDIST Returns the individual term binomial distribution probability

CHIDIST Returns the one-tailed probability of the chi-squared distribution

CHIINV Returns the inverse of the one-tailed probability of the chi-squared distribution

CRITBINOM Returns the smallest value for which the cumulative binomial distribution is less than or equal to a criterion value

EXPONDIST Returns the exponential distribution

FDIST Returns the F probability distribution

FINV Returns the inverse of the F probability distribution

GAMMADIST Returns the gamma distribution

GAMMAINV Returns the inverse of the gamma cumulative distribution

HYPGEOMDIST Returns the hypergeometric distribution

LOGINV Returns the inverse of the lognormal distribution

LOGNORMDIST Returns the cumulative lognormal distribution

NEGBINOMDIST Returns the negative binomial distribution

NORMDIST Returns the normal cumulative distribution

NORMINV Returns the inverse of the normal cumulative distribution

NORMSDIST Returns the standard normal cumulative distribution

NORMSINV Returns the inverse of the standard normal cumulative distribution

POISSON Returns the Poisson distribution

TDIST Returns the Student's t-distribution

TINV Returns the inverse of the Student's t-distribution

WEIBULL Returns the Weibull distribution

1.5 Sample Spreadsheets

Here is a collection of sample spreadsheets showing how to use each function in DataMinerXL software. The spreadsheets are organized in terms of the following categories.

basic_stats.xlsx Spreadsheet for basic statistics

weight_of_evidence.xlsx Spreadsheet for weight of evidence transformation

pca_factor_analysis.xlsx Spreadsheet for principal component analysis and factor analysis

linear_reg.xlsx Spreadsheet for linear regression

pls_reg.xlsx Spreadsheet for partial least square regression

logistic_reg.xlsx Spreadsheet for logistic regression

time_series_analysis.xlsx Spreadsheet for time-series analysis

lda_qda.xlsx Spreadsheet for linear and quadratic discriminant analysis

correspondence_analysis.xlsx Spreadsheet for correspondence analysis

naive_bayes.xlsx Spreadsheet for naive-Bayes classification

decision_tree_based_model.xlsx Spreadsheet for decision tree-based model

clustering_segmentation.xlsx Spreadsheet for clustering and segmentation

neural_network_model.xlsx Spreadsheet for neural network model

support_vector_machine.xlsx Spreadsheet for support vector machine (SVM) model

optimization.xlsx Spreadsheet for optimization

portfolio_optimization.xlsx Spreadsheet for portfolio optimization

matrix_operations.xlsx Spreadsheet for matrix operations

fast_Fourier_transform.xlsx Spreadsheet for fast Fourier transform

numerical_integration.xlsx Spreadsheet for numerical integration by Gaussian quadrature

data_manipulation_functions.xlsx Spreadsheet for data manipulation functions

Chapter 2

Utility Functions

version Displays the version number and build date/time of DataMinerXL software

function_list Lists all functions in DataMinerXL software

2.1 version

Displays the version number and build date/time of DataMinerXL software

version()

Returns

The version number and build date/time of DataMinerXL software

Return to the [index](#)

2.2 function_list

Lists all functions in DataMinerXL software

function_list()

Returns

A list of all functions in DataMinerXL software

Return to the [index](#)

Chapter 3

Data Manipulation Functions

variable_list Lists the variable names in an input data file

subset Gets a subset of a data table

data_info Prints the number of records and variable names from an input data file

data_lookup Looks up data by matching multiple keys

data_save Saves a data table into a file

data_save_tex Saves a data table into a file in TEX format

data_load Loads a data table from a file

data_partition Gets random data partition

data_sort Sorts a data table given keys and orders

data_fill Fills missing data elements with a given value

sort_file Sorts a data file given keys and orders

merge_tables Merge two data tables by a single numerical key

rank_items Selects the items from the ranks by keys

split_str Splits a text string into a vector

3.1 variable_list

Lists the variable names in an input data file

`variable_list (filename, delimiter)`

Returns

The variable names in an input data file

Parameters

filename Input data file name. The first line of the file is the header line with variable names

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Return to the [index](#)

3.2 subset

Gets a subset of a data table

subset (inputData, indicator)

Returns

A subset of a data table

Parameters

inputData Input data table for subsetting

indicator Indicators in one row or column for subsetting, 1 for selecting and 0 for dropping. The order of the indicators is the same as the variables in the input data table

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

3.3 data_info

Prints the number of records and variable names from an input data file

data_info (filename, delimiter)

Returns

The number of records and variable names in an input data file

Parameters

filename Input data file name. The first line of the file is the header line with variable names

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Return to the [index](#)

3.4 data_lookup

Looks up data by matching multiple keys

data_lookup (keys, dataTable, columnIndexesForOutput)

Returns

Matched data by multiple keys

Parameters

keys The variables for lookup keys

dataTable The data table including the lookup keys. The keys in dataTable must be unique; otherwise the first matching row will be selected

columnIndexesForOutput Optional: 1-based column indexes in one row or one column for outputting columns in the dataTable. Default: output all columns in matching row if omitted

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

3.5 data_save

Saves a data table into a file

data_save (inputData, filename, delimiter)

Returns

A data file containing the data from the input data table

Parameters

inputData Input data

filename The file name the data saved to

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

3.6 data_save_tex

Saves a data table into a file in TEX format

`data_save_tex (inputData, filename)`

Returns

A data file containing the data from the input data table in TEX format

Parameters

inputData Input data

filename The file name the data saved to

Examples

`data_manipulation_functions.xlsx`

Return to the [index](#)

3.7 data_load

Loads a data table from a file

`data_load (filename, varNames, numRecords, delimiter)`

Returns

A table from a file

Parameters

filename The file name the data table loaded from. The first line of the file is the header line with variable names

varNames Optional: variable names to be loaded from the file. Default: load all variables when missing

numRecords Optional: number of records to be loaded from the file. Default: load all records when missing

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

`data_manipulation_functions.xlsx`

Return to the [index](#)

3.8 data_partition

Gets random data partition

data_partition (inputData, partition, part, seed)

Returns

A random data partition

Parameters

inputData Input data with headers in the first row

partition Partitioning percentages. For example, a three-part partitioning [0.5, 0.3, 0.2] generates three partitions with the first 50%, the second 30%, and the third 20%. The sum of partitioning percentages must be 1

part A part number (1-based) of partitioning returned. The first part is 1

seed A non-negative integer seed for generating random numbers. 0 is for using timer

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

3.9 data_sort

Sorts a data table given keys and orders

data_sort (inputData, keys)

Returns

A sorted data table

Parameters

inputData Input data table

keys Two column input with variable names in the 1st column and sorting order (1 for ascending, -1 for descending) in the 2nd column

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

3.10 data_fill

Fills missing data elements with a given value

`data_fill (inputData, value)`

Returns

A data table with missing values replaced

Parameters

inputData Input data table

value value to replace missing elements

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

3.11 sort_file

Sorts a data file given keys and orders

`sort_file (filename, keys, outfilename, delimiter)`

Returns

A sorted data file

Parameters

filename Input data file name. The first line of the file is the header line with variable names

keys Two column input with variable names in the 1st column and sorting order (1 for ascending, -1 for descending) in the 2nd column

outfilename Optional: output data file name. Default: overwrite the input data file

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

3.12 merge_tables

Merge two data tables by a single numerical key

merge_tables (table1, table2, key1, key2, output1, output2)

Returns

A merged data table

Parameters

table1 Input data table1

table2 Input data table2

key1 A column number (1-based) as a key in table1. The values of the merge key must be numbers and sorted

key2 A column number (1-based) as a key in table2. The values of the merge key must be numbers and sorted

output1 An array of column numbers (1-based) for output from table1

output2 An array of column numbers (1-based) for output from table2

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

3.13 rank_items

Selects the items from the ranks by keys

rank_items (keys, items, rankFrom, rankTo, order)

Returns

The items from the ranks by keys

Parameters

keys One column input for the keys. The keys must be numerical

items One column input for the items. The items must be categorical

rankFrom The rank number (1-based) of the first output item

rankTo Optional: the rank number (1-based) of the last output item. Default: rankFrom

order Optional: the order when sorting keys. 1 for descending, -1 for ascending. Default: 1 for descending

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

3.14 split_str

Splits a text string into a vector

split_str (text, delimiter)

Returns

A vector splitted from a text string

Parameters

text A text string

delimiter Optional: One character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma

Examples

data_manipulation_functions.xlsx

Return to the [index](#)

Chapter 4

Basic Statistical Functions

- ranks** Creates 1-based ranks of data points given a column of data
- ranks_from_file** Creates 1-based ranks of data points given a data file
- freq** Creates frequency tables given a data table
- freq_from_file** Creates frequency tables given a data file
- freq_2d** Creates a frequency cross-table for two variables given a data table
- freq_2d_from_file** Creates a frequency cross-table for two variables given a data file
- means** Generates basic statistics: sum, average, standard deviation, minimum, and maximum given a data table
- means_from_file** Generates basic statistics: sum, average, standard deviation, minimum, and maximum given a data file
- univariate** Generates univariate statistics given a data table
- univariate_from_file** Generates univariate statistics given a data file
- percentiles** Calculates p-th percentile of values in each subgroup
- summary** Generates descriptive statistics in classes given a data table
- summary_from_file** Generates descriptive statistics in classes given a data file
- binning** Creates equal interval binning given a column of data table
- QQ_plot** Tests normality of a univariate sample
- variable_corr_select** Selects variables by removing highly correlated variables
- poly_roots** Finds all roots given real coefficients of a polynomial
- poly_prod** Computes the coefficients of the product of two polynomials
- Lagrange_interpolation** Performs Lagrange polynomial interpolation given data points
- three_moment_match_to_SLN** Performs three moment match to shifted lognormal distribution
- set** Creates a set given a string/number matrix
- set_union** Creates a set from union of two sets
- set_intersection** Creates a set from intersection of two sets
- set_difference** Creates a set from difference of two sets

4.1 ranks

Creates 1-based ranks of data points given a column of data

`ranks (inputData, numBins, order)`

Returns

Ranks of data points

Parameters

inputData One column of numerical data with header in the first row

numBins Number of bins

order Optional: The order of ranking, 1 for ascending, -1 for descending. Default: 1 for ascending

Examples

basic_stats.xlsx

Return to the [index](#)

4.2 ranks_from_file

Creates 1-based ranks of data points given a data file

`ranks_from_file (varName, filename, rankVarName, outfilename, numBins, order, delimiter)`

Returns

Ranks of data points

Parameters

varName Variable name of a numerical variable for ranking

filename Input data file name. The first line of the file is the header line with variable names

rankVarName Rank variable name

outfilename Output data file name

numBins Number of bins

order Optional: The order of ranking, 1 for ascending, -1 for descending. Default: 1 for ascending

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

basic_stats.xlsx

Return to the [index](#)

4.3 freq

Creates frequency tables given a data table

freq (inputData, includeMissing)

Returns

Frequency tables for variables in a given data table

Parameters

inputData Input data with headers in the first row. Each variable can be either numerical or categorical

includeMissing Optional: binary flag 0 or 1 to indicate if the missings are included (when it is 1) or excluded (when it is 0) in frequency table. Default: 0

Examples

basic_stats.xlsx

Return to the [index](#)

4.4 freq_from_file

Creates frequency tables given a data file

freq_from_file (filename, varNames, delimiter, includeMissing)

Returns

Frequency tables for the variables selected

Parameters

filename Input data file name. The first line of the file is the header line with variable names

varNames Variable names in one row or one column. Each variable can be either numerical or categorical

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

includeMissing Optional: binary flag 0 or 1 to indicate if the missings are included (when it is 1) or excluded (when it is 0) in frequency table. Default: 0

Examples

basic_stats.xlsx

Return to the [index](#)

4.5 freq_2d

Creates a frequency cross-table for two variables given a data table

freq_2d (x1, x2, format, output)

Returns

A frequency cross-table for two variables

Parameters

x1 One column input for the 1st variable with header in the first row. The variable can be numerical or categorical

x2 One column input for the 2nd variable with header in the first row. The variable can be numerical or categorical

format Optional: format of output, TABLE or LIST. Default: TABLE

output Optional: control the output for Freq, Percent, RowPct, ColPct. Y/N for Yes/No. Default: YYYY for output all four variables

Examples

basic_stats.xlsx

Return to the [index](#)

4.6 freq_2d_from_file

Creates a frequency cross-table for two variables given a data file

freq_2d_from_file (filename, x1Name, x2Name, format, output, delimiter)

Returns

A frequency cross-table for two variables

Parameters

filename Input data file name. The first line of the file is the header line with variable names

x1Name 1st variable name. The variable can be numerical or categorical

x2Name 2nd variable name. The variable can be numerical or categorical

format Optional: format of output, TABLE or LIST. Default: TABLE

output Optional: control the output for Freq, Percent, RowPct, ColPct. Y/N for Yes/No. Default: YYYY for output all four variables

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

basic_stats.xlsx

Return to the [index](#)

4.7 means

Generates basic statistics: sum, average, standard deviation, minimum, and maximum given a data table
`means (inputData)`

Returns

Basic statistics: sum, average, standard deviation, minimum, and maximum

Parameters

inputData Input data of numerical variables with headers in the first row

Examples

basic_stats.xlsx

Return to the [index](#)

4.8 means_from_file

Generates basic statistics: sum, average, standard deviation, minimum, and maximum given a data file
`means_from_file (filename, varNames, delimiter)`

Returns

Basic statistics: sum, average, standard deviation, minimum, and maximum

Parameters

filename Input data file name. The first line of the file is the header line with variable names

varNames Variable names in one row or one column. All variables must be numerical

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

basic_stats.xlsx

Return to the [index](#)

4.9 univariate

Generates univariate statistics given a data table

`univariate (inputData)`

Returns

Univariate statistics given a data table

Parameters

inputData Input data of numerical variables with headers in the first row

Examples

basic_stats.xlsx

Return to the [index](#)

4.10 univariate_from_file

Generates univariate statistics given a data file

`univariate_from_file (filename, varNames, delimiter)`

Returns

Univariate statistics given a data file

Parameters

filename Input data file name. The first line of the file is the header line with variable names

varNames Variable names in one row or one column. All variables must be numerical

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

basic_stats.xlsx

Return to the [index](#)

4.11 percentiles

Calculates p-th percentiles of values in each subgroup

`percentiles (x, p, classVar)`

Returns

p-th percentiles of values in each subgroup

Parameters

x Input data in one column of numerical variable with header in the first row

p Percentiles which are numbers between 0 and 1, inclusive in one row or one column

classVar Optional: class variable used to form subgroups in one column. Default: missing classified as one group

Examples

basic_stats.xlsx

Return to the [index](#)

4.12 summary

Generates descriptive statistics in classes given a data table

summary (classVars, x, weight, nway, output)

Returns

Descriptive statistics

Parameters

classVars Class variables with headers in the first row. Each variable can be either numerical or categorical. They are used to form subgroups for descriptive analysis

x Input data of numerical variables with headers in the first row

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

nway Optional: binary flag 1 or 0. Default: 1. With flag 1 it outputs all combinations with all class variables, and with flag 0 it outputs all combinations with all subsets of class variables

output Optional: output for Sum, Avg, Stdev, Min, Max. Default: missing for outputting all five variables

Remarks

For example, the 1st class variable has 2 classes and the 2nd class variable has 3 classes. Setting nway as 1 generates 6 groups:

Type	Class variable 1	Class variable 2
3	1	1
3	1	2
3	1	3
3	2	1
3	2	2
3	2	3

Setting nway as 0 generates 12 groups:

Type	Class variable 1	Class variable 2
0	ALL	ALL
1	ALL	1
1	ALL	2
1	ALL	3
2	1	ALL
2	2	ALL
3	1	1
3	1	2
3	1	3
3	2	1
3	2	2
3	2	3

Examples

basic_stats.xlsx

Return to the [index](#)

4.13 summary_from_file

Generates descriptive statistics in classes given a data file

summary_from_file (filename, classVarNames, xNames, weightName, nway, delimiter, output)

Returns

Descriptive statistics

Parameters

filename Input data file name. The first line of the file is the header line with variable names

classVarNames Names of class variables used to form subgroups for descriptive analysis in one row or one column. Each variable can be either numerical or categorical

xNames Variable names in one row or one column. All variables must be numerical

weightName Optional: weight variable name. Default: 1 for all weights

nway Optional: binary flag 1 or 0. Default: 1. With flag 1 it outputs all combinations with all class variables, and with flag 0 it outputs all combinations with all subsets of class variables

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

output Optional: output for Sum, Avg, Stdev, Min, Max. Default: missing for outputting all five variables

Remarks

For example, the 1st class variable has 2 classes and the 2nd class variable has 3 classes. Setting nway as 1 generates 6 groups:

Type	Class variable 1	Class variable 2
3	1	1
3	1	2
3	1	3
3	2	1
3	2	2
3	2	3

Setting nway as 0 generates 12 groups:

Type	Class variable 1	Class variable 2
0	ALL	ALL
1	ALL	1
1	ALL	2
1	ALL	3
2	1	ALL
2	2	ALL
3	1	1
3	1	2
3	1	3
3	2	1
3	2	2
3	2	3

Examples

basic_stats.xlsx

Return to the [index](#)

4.14 binning

Creates equal interval binning given a column of data table

binning (inputData, lower, upper, numBins)

Returns

Equal interval binning

Parameters

inputData Input data of numerical variable with header in the first row in one column

lower Lower boundary for binning

upper Upper boundary for binning

numBins Number of bins

Examples

basic_stats.xlsx

Return to the [index](#)

4.15 QQ_plot

Tests normality of a univariate sample

QQ_plot (inputData)

Returns

Standard normal quantiles for QQ-plot

Parameters

inputData Input data of numerical variable with header in the first row in one column

Remarks

Let x_1, x_2, \dots, x_n be n data points. Sort the values to get $x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)}$. The probability levels are

$$p_{(j)} = \frac{j - 1/2}{n}, \quad j = 1, 2, \dots, n$$

The standard normal quantiles are

$$q_{(j)} = N^{-1} \left(\frac{j - 1/2}{n} \right), \quad j = 1, 2, \dots, n$$

where $N^{-1}()$ is the inverse function of the standard normal cumulative function. The Q-Q plot is the plot of the pairs $(q_{(j)}, x_{(j)})$, $j = 1, 2, \dots, n$.

Examples

basic_stats.xlsx

Return to the [index](#)

4.16 variable_corr_select

Selects variables by removing highly correlated variables

variable_corr_select (x, y, corrCutOff, weight)

Returns

A selected variable list and a dropped variable list

Parameters

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

corrCutOff The correlation cutoff value

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

Given a threshold of correlation, it generates a table with pair-wise correlations with their absolute values larger than the threshold. It selects the variable with the largest absolute value of correlation with the target variable, delete all variables directly correlated to the selected variable. Repeat this procedure until no correlation is larger than the threshold.

The more description can be found in Section 7.10 of the reference [2].

Examples

basic_stats.xlsx

Return to the [index](#)

4.17 poly_roots

Finds all roots given real coefficients of a polynomial

poly_roots (coefs)

Returns

All roots of a polynomial with real coefficients

Parameters

coefs Real coefficients of a polynomial. n+1 coefficients of $c_0 + c_1 x + c_2 x^2 + \dots + c_n x^n$

Remarks

A polynomial

$$c_0 + c_1 x + c_2 x^2 + \dots + c_n x^n = 0$$

where $c = [c_0, c_1, c_2, \dots, c_n]$ are real coefficients.

Examples

basic_stats.xlsx

Return to the [index](#)

4.18 poly_prod

Computes the coefficients of the product of two polynomials

poly_prod (a, b)

Returns

Coefficients of the product of two polynomials

Parameters

a Real coefficients of a polynomial. m+1 coefficients of $a_0 + a_1 x + a_2 x^2 + \dots + a_m x^m$

b Real coefficients of a polynomial. n+1 coefficients of $b_0 + b_1 x + b_2 x^2 + \dots + b_n x^n$

Remarks

A polynomial c is the product of two polynomials a and b :

$$c_0 + c_1 x + c_2 x^2 + \dots + c_{m+n} x^{m+n} = (a_0 + a_1 x + a_2 x^2 + \dots + a_m x^m)(b_0 + b_1 x + b_2 x^2 + \dots + b_n x^n)$$

where $a = [a_0, a_1, a_2, \dots, a_m]$, $b = [b_0, b_1, b_2, \dots, b_n]$, $c = [c_0, c_1, c_2, \dots, c_{m+n}]$ are real coefficients.

Examples

basic_stats.xlsx

Return to the [index](#)

4.19 Lagrange_interpolation

Performs Lagrange polynomial interpolation given data points

Lagrange_interpolation (x, y)

Returns

A polynomial from Lagrange polynomial interpolation

Parameters

x x-values

y y-values

Remarks

Given n data points (x_i, y_i) , $i = 1, 2, \dots, n$, the polynomial of order $n - 1$ from Lagrange polynomial interpolation is

$$p_{n-1}(x) = \sum_{i=1}^n y_i L_i(x) = \sum_{i=1}^n y_i \prod_{k=1, k \neq i}^n \frac{x - x_k}{x_i - x_k}$$

Examples

basic_stats.xlsx

Return to the [index](#)

4.20 three_moment_match_to_SLN

Performs three moment match to a shifted lognormal distribution

three_moment_match_to_SLN (m1, m2, m3)

Returns

Parameters of a shifted lognormal distribution

Parameters

m1 Non-centered first moment

m2 Non-centered second moment

m3 Non-centered third moment

Remarks

Given three non-centered moments m_1, m_2 and m_3 , to match a shifted lognormal distribution. A shifted lognormal distribution with three parameters, a mean μ , volatility σ , and shift s is

$$x \sim s + \mu e^{-1/2\sigma^2 + \sigma\epsilon}$$

where ϵ is a standard normal distribution. First calculate the centered moments as

$$\mu_2 = m_2 - m_1^2$$

$$\mu_3 = m_3 - 3m_2m_1 + 2m_1^3$$

Then find a real root from a cubic polynomial equation:

$$y^3 - 3y^2 = \frac{\mu_3^2}{\mu_2^3}$$

The solution is:

$$\sigma = \sqrt{\ln(y - 2)}$$

$$\mu = \sqrt{\frac{\mu_2}{y - 3}}$$

$$s = m_1 - \mu$$

Examples

basic_stats.xlsx

Return to the [index](#)

4.21 set

Creates a set given a string/number matrix

set (matrix)

Returns

A set

Parameters

matrix Input string/number matrix. Each element could be string or number

Examples

basic_stats.xlsx

Return to the [index](#)

4.22 set_union

Creates a set from union of two sets

set_union (set1, set2)

Returns

A union set

Parameters

set1 Input string/number matrix as set1. Each element could be string or number

set2 Input string/number matrix as set2. Each element could be string or number

Examples

basic_stats.xlsx

Return to the [index](#)

4.23 set_intersection

Creates a set from intersection of two sets

set_intersection (set1, set2)

Returns

An intersection set

Parameters

set1 Input string/number matrix as set1. Each element could be string or number

set2 Input string/number matrix as set2. Each element could be string or number

Examples

basic_stats.xlsx

Return to the [index](#)

4.24 set_difference

Creates a set from difference of two sets

set_difference (set1, set2)

Returns

A difference set

Parameters

set1 Input string/number matrix as set1. Each element could be string or number

set2 Input string/number matrix as set2. Each element could be string or number

Examples

basic_stats.xlsx

Return to the [index](#)

Chapter 5

Modeling Functions for All Models

model_bin_eval Evaluates a binary target model given a column of actual values and a column of predicted values

model_bin_eval_from_file Evaluates a binary target model given a data file, a name of actual values, and a name of predicted values

model_cont_eval Evaluates a continuous target model given a column of actual values and a column of predicted values

model_cont_eval_from_file Evaluates a continuous target model given a data file, a name of actual values, and a name of predicted values

model_eval Evaluates model performance given a model and a data table

model_eval_from_file Evaluates model performance given a model and a data file

model_score Scores a population given a model and a data table

model_score_from_file Scores a population given a model and a data file

model_save_scoring_code Saves the scoring code of a given model to a file

5.1 model_bin_eval

Evaluates a binary target model performance given a column of actual values and a column of predicted values

`model_bin_eval (yActual, yPredicted, numBins, weight)`

Returns

Binary target model performance

Parameters

yActual Actual values with header in the first row

yPredicted Predicted values with header in the first row

numBins Number of bins in gains chart

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Examples

logistic_reg.xlsx
decision_tree_based_model.xlsx
neural_network_model.xlsx

Return to the [index](#)

5.2 model_bin_eval_from_file

Evaluates a binary target model given a data file, a name of actual values, and a name of predicted values

model_bin_eval_from_file (filename, yActualName, yPredictedName, numBins, weightName, delimiter)

Returns

Binary target model performance

Parameters

filename Input data file name. The first line of the file is the header line with variable names

yActualName Actual target variable name

yPredictedName Predicted target variable name

numBins Number of bins in gains chart

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

logistic_reg.xlsx
decision_tree_based_model.xlsx
neural_network_model.xlsx

Return to the [index](#)

5.3 model_cont_eval

Evaluates a continuous target model given a column of actual values and a column of predicted values

model_cont_eval (yActual, yPredicted, numParams, numBins, weight)

Returns

Continuous target model performance

Parameters

- yActual* Actual values with header in the first row
yPredicted Predicted values with header in the first row
numParams Number of parameters estimated in model
numBins Number of bins in gains chart
weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Examples

linear_reg.xlsx
pls_reg.xlsx
decision_tree_based_model.xlsx
neural_network_model.xlsx

Return to the [index](#)

5.4 model_cont_eval_from_file

Evaluates a continuous target model given a data file, a name of actual values, and a name of predicted values

model_cont_eval_from_file (filename, yActualName, yPredictedName, numParams, numBins, weight-Name, delimiter)

Returns

Continuous target model performance

Parameters

- filename* Input data file name. The first line of the file is the header line with variable names
yActualName Actual target variable name
yPredictedName Predicted target variable name
numParams Number of parameters estimated in model
numBins Number of bins in gains chart
weightName Optional: weight variable name. Default: 1 for all weights
delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

linear_reg.xlsx
pls_reg.xlsx
decision_tree_based_model.xlsx
neural_network_model.xlsx

Return to the [index](#)

5.5 model_eval

Evaluates model performance given a model and a data table

`model_eval (model, x, y, numBins, weight)`

Returns

Model performance

Parameters

model A model. It supports linear regression, partial least square regression, logistic regression, classification and regression tree, logistic regression boosting tree, least square regression boosting tree, neural network, and SVM

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

numBins Number of bins in gains chart

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Examples

linear_reg.xlsx
pls_reg.xlsx
logistic_reg.xlsx
decision_tree_based_model.xlsx
neural_network_model.xlsx

See also

[linear_reg](#)
[linear_reg_from_file](#)
[linear_reg_forward_select](#)
[linear_reg_forward_select_from_file](#)
[poly_reg](#)
[pls_reg](#)
[pls_reg_from_file](#)
[logistic_reg](#)
[logistic_reg_from_file](#)
[logistic_reg_forward_select](#)
[logistic_reg_forward_select_from_file](#)
[tree](#)
[tree_from_file](#)
[tree_boosting_logistic_reg](#)
[tree_boosting_logistic_reg_from_file](#)
[tree_boosting_ls_reg](#)
[tree_boosting_ls_reg_from_file](#)
[neural_net](#)
[neural_net_from_file](#)
[svm](#)
[svm_from_file](#)

Return to the [index](#)

5.6 model_eval_from_file

Evaluates model performance given a model and a data file

model_eval_from_file (model, filename, yName, numBins, weightName, delimiter)

Returns

Model performance

Parameters

model A model. It supports linear regression, partial least square regression, logistic regression, classification and regression tree, logistic regression boosting tree, least square regression boosting tree, neural network, and SVM

filename Input data file name. The first line of the file is the header line with variable names

yName Dependent variable name

numBins Number of bins in gains chart

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

linear_reg.xlsx
pls_reg.xlsx
logistic_reg.xlsx
decision_tree_based_model.xlsx
neural_network_model.xlsx

See also

[linear_reg](#)
[linear_reg_from_file](#)
[linear_reg_forward_select](#)
[linear_reg_forward_select_from_file](#)
[poly_reg](#)
[pls_reg](#)
[pls_reg_from_file](#)
[logistic_reg](#)
[logistic_reg_from_file](#)
[logistic_reg_forward_select](#)
[logistic_reg_forward_select_from_file](#)
[tree](#)
[tree_from_file](#)
[tree_boosting_logistic_reg](#)
[tree_boosting_logistic_reg_from_file](#)
[tree_boosting_ls_reg](#)
[tree_boosting_ls_reg_from_file](#)
[neural_net](#)
[neural_net_from_file](#)
[svm](#)
[svm_from_file](#)

Return to the [index](#)

5.7 model_score

Scores a population given a model and a data table

model_score (model, x)

Returns

A column of scores of a population

Parameters

model A model of linear regression, PLS regression, logistic regression, classification and regression tree, logistic regression boosting tree, least square regression boosting tree, neural network, SVM, or naive Bayes classifier

x Input data for independent variables with headers in the first row

Examples

linear_reg.xlsx
pls_reg.xlsx
logistic_reg.xlsx
decision_tree_based_model.xlsx
neural_network_model.xlsx

See also

[linear_reg](#)
[linear_reg_from_file](#)
[linear_reg_forward_select](#)
[linear_reg_forward_select_from_file](#)
[poly_reg](#)
[pls_reg](#)
[pls_reg_from_file](#)
[logistic_reg](#)
[logistic_reg_from_file](#)
[logistic_reg_forward_select](#)
[logistic_reg_forward_select_from_file](#)
[LDA](#)
[QDA](#)
[tree](#)
[tree_from_file](#)
[tree_boosting_logistic_reg](#)
[tree_boosting_logistic_reg_from_file](#)
[tree_boosting_ls_reg](#)
[tree_boosting_ls_reg_from_file](#)
[neural_net](#)
[neural_net_from_file](#)
[svm](#)
[svm_from_file](#)
[naive_bayes_classifier](#)

Return to the [index](#)

5.8 model_score_node

Scores a population to get node string given a model and a data table

`model_score_node (model, x)`

Returns

Columns of strings of scoring nodes of a population

Parameters

model A model of logistic regression boosting tree

x Input data for independent variables with headers in the first row

Examples

`decision_tree_based_model.xlsx`

See also

[tree_boosting_logistic_reg](#)

Return to the [index](#)

5.9 model_score_from_file

Scores a population given a model and a data file

`model_score_from_file (model, infilename, scoreName, outfilename, delimiter)`

Returns

A file containing scores of a population

Parameters

model A model. It supports linear regression, partial least square regression, logistic regression, classification and regression tree, logistic regression boosting tree, least square regression boosting tree, neural network, SVM, and naive Bayes classifier

infilename Input data file name. The first line of the file is the header line with variable names

scoreName Score name

outfilename Output data file name. Output all fields in the input data file and append a column for scores

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

```
linear_reg.xlsx  
pls_reg.xlsx  
logistic_reg.xlsx  
decision_tree_based_model.xlsx  
neural_network_model.xlsx
```

See also

[linear_reg](#)
[linear_reg_from_file](#)
[linear_reg_forward_select](#)
[linear_reg_forward_select_from_file](#)
[poly_reg](#)
[pls_reg](#)
[pls_reg_from_file](#)
[logistic_reg](#)
[logistic_reg_from_file](#)
[logistic_reg_forward_select](#)
[logistic_reg_forward_select_from_file](#)
[tree](#)
[tree_from_file](#)
[tree_boosting_logistic_reg](#)
[tree_boosting_logistic_reg_from_file](#)
[tree_boosting_ls_reg](#)
[tree_boosting_ls_reg_from_file](#)
[neural_net](#)
[neural_net_from_file](#)
[svm](#)
[svm_from_file](#)
[naive_bayes_classifier](#)

Return to the [index](#)

5.10 model_save_scoring_code

Saves the scoring code of a given model to a file

`model_save_scoring_code (model, filename)`

Returns

A file containing a model's scoring code

Parameters

model A model its scoring code to be saved to a file. It supports linear regression, partial least square regression, logistic regression, classification and regression tree, logistic regression boosting tree, least square regression boosting tree, and neural network

filename A filename the scoring code is saved to. The scoring code is in C format if the filename has extension .h, .c, .cpp, or .java, otherwise it is in SAS format

Examples

linear_reg.xlsx
pls_reg.xlsx
logistic_reg.xlsx
decision_tree_based_model.xlsx
neural_network_model.xlsx

See also

[woe_xcont_ybin](#)
[woe_xcont_ybin_from_file](#)
[woe_xcont_ycont](#)
[woe_xcont_ycont_from_file](#)
[woe_xcat_ybin](#)
[woe_xcat_ybin_from_file](#)
[woe_xcat_ycont](#)
[woe_xcat_ycont_from_file](#)
[linear_reg](#)
[linear_reg_from_file](#)
[linear_reg_forward_select](#)
[linear_reg_forward_select_from_file](#)
[linear_reg_piecewise](#)
[linear_reg_piecewise_from_file](#)
[poly_reg](#)
[pls_reg](#)
[pls_reg_from_file](#)
[logistic_reg](#)
[logistic_reg_from_file](#)
[logistic_reg_forward_select](#)
[logistic_reg_forward_select_from_file](#)
[tree](#)
[tree_from_file](#)
[tree_boosting_logistic_reg](#)
[tree_boosting_logistic_reg_from_file](#)
[tree_boosting_ls_reg](#)
[tree_boosting_ls_reg_from_file](#)
[neural_net](#)
[neural_net_from_file](#)

Return to the [index](#)

Chapter 6

Weight of Evidence Transformation Functions

woe_xcont_ybin Generates weight of evidence (WOE) of continuous independent variables and a binary dependent variable given a data table

woe_xcont_ybin_from_file Generates weight of evidence (WOE) of continuous independent variables and a binary dependent variable given a data file

woe_xcont_ycont Generates weight of evidence (WOE) of continuous independent variables and a continuous dependent variable given a data table

woe_xcont_ycont_from_file Generates weight of evidence (WOE) of continuous independent variables and a continuous dependent variable given a data file

woe_xcat_ybin Generates weight of evidence (WOE) of categorical independent variables and a binary dependent variable given a data table

woe_xcat_ybin_from_file Generates weight of evidence (WOE) of categorical independent variables and a binary dependent variable given a data file

woe_xcat_ycont Generates weight of evidence (WOE) of categorical independent variables and a continuous dependent variable given a data table

woe_xcat_ycont_from_file Generates weight of evidence (WOE) of categorical independent variables and a continuous dependent variable given a data file

woe_transform Performs weight of evidence (WOE) transformation given a WOE model and a data table

woe_transform_from_file Performs weight of evidence (WOE) transformation given a WOE model and a data file

6.1 woe_xcont_ybin

Generates weight of evidence (WOE) of continuous independent variables and a binary dependent variable given a data table

`woe_xcont_ybin (x, y, initialNumBins, pvalue, maxNumBins, minNumRecords, weight)`

Returns

Weight of evidence (WOE) of continuous independent variables and binary dependent variable

Parameters

x Input data of numerical independent variables with headers in the first row
y Input data of binary dependent variable with header in the first row
initialNumBins Initial number of bins
pvalue Optional: p-value for the threshold of merging groups. Default: 1
maxNumBins Optional: maximum number of bins. Default: infinity
minNumRecords Optional: minimum number of records with missing x for classifying as a group. Default: 50
weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

First, bin the whole population into initialNumBins using equal population binning. Recursively merge the pair of neighboring bins using pvalue as threshold of Chi-square test until all neighboring bins are significantly different. If the number of records with missing x is larger than or equal to minNumRecords, all missing x are classified as a group in scoring code and transformation.

Examples

weight_of_evidence.xlsx

See also

[model_save_scoring_code](#)
[woe_transform](#)
[woe_transform_from_file](#)

Return to the [index](#)

6.2 woe_xcont_ybin_from_file

Generates weight of evidence (WOE) of continuous independent variables and a binary dependent variable given a data file

woe_xcont_ybin_from_file (filename, xNames, yName, initialNumBins, pvalue, maxNumBins, minNumRecords, weightName, delimiter)

Returns

Weight of evidence (WOE) of continuous independent variables and binary dependent variable

Parameters

filename Input data file name. The first line of the file is the header line with variable names
xNames Independent variable names in one row or one column

yName Dependent variable name

initialNumBins Initial number of bins

pvalue Optional: p-value for the threshold of merging groups. Default: 1

maxNumBins Optional: maximum number of bins. Default: infinity

minNumRecords Optional: minimum number of records with missing x for classifying as a group. Default: 50

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

First, bin the whole population into initialNumBins using equal population binning. Recursively merge the pair of neighboring bins using pvalue as threshold of Chi-square test until all neighboring bins are significantly different. If the number of records with missing x is larger than or equal to minNumRecords, all missing x are classified as a group in scoring code and transformation.

Examples

weight_of_evidence.xlsx

See also

[model_save_scoring_code](#)

[woe_transform](#)

[woe_transform_from_file](#)

Return to the [index](#)

6.3 woe_xcont_ycont

Generates weight of evidence (WOE) of continuous independent variables and a continuous dependent variable given a data table

woe_xcont_ycont (x, y, initialNumBins, pvalue, maxNumBins, minNumRecords, weight)

Returns

Weight of evidence (WOE) of continuous independent variables and continuous dependent variable

Parameters

x Input data of numerical independent variables with headers in the first row

y Input data of dependent variable with header in the first row

initialNumBins Initial number of bins

pvalue Optional: p-value for the threshold of merging groups. Default: 1

maxNumBins Optional: maximum number of bins. Default: infinity

minNumRecords Optional: minimum number of records with missing x for classifying as a group. Default: 50

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

First, bin the whole population into initialNumBins using equal population binning. Recursively merge the pair of neighboring bins using pvalue as threshold of t-test until all neighboring bins are significantly different. If the number of records with missing x is larger than or equal to minNumRecords, all missing x are classified as a group in scoring code and transformation.

Examples

weight_of_evidence.xlsx

See also

[model_save_scoring_code](#)
[woe_transform](#)
[woe_transform_from_file](#)

Return to the [index](#)

6.4 woe_xcont_ycont_from_file

Generates weight of evidence (WOE) of continuous independent variables and a continuous dependent variable given a data file

woe_xcont_ycont_from_file (filename, xNames, yName, initialNumBins, pvalue, maxNumBins, minNumRecords, weightName, delimiter)

Returns

Weight of evidence (WOE) of continuous independent variables and continuous dependent variable

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column

yName Dependent variable name

initialNumBins Initial number of bins

pvalue Optional: p-value for the threshold of merging groups. Default: 1

maxNumBins Optional: maximum number of bins. Default: infinity

minNumRecords Optional: minimum number of records with missing x for classifying as a group. Default: 50

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

First, bin the whole population into initialNumBins using equal population binning. Recursively merge the pair of neighboring bins using pvalue as threshold of t-test until all neighboring bins are significantly different. If the number of records with missing x is larger than or equal to minNumRecords, all missing x are classified as a group in scoring code and transformation.

Examples

weight_of_evidence.xlsx

See also

[model_save_scoring_code](#)
[woe_transform](#)
[woe_transform_from_file](#)

Return to the [index](#)

6.5 woe_xcat_ybin

Generates weight of evidence (WOE) of categorical independent variables and a binary dependent variable given a data table

`woe_xcat_ybin (x, y, pvalue, maxNumBins, weight)`

Returns

Weight of evidence (WOE) of categorical independent variables and binary dependent variable

Parameters

x Input data of categorical independent variables with headers in the first row

y Input data of dependent variable with header in the first row

pvalue Optional: p-value for the threshold of merging groups. Default: 1

maxNumBins Optional: maximum number of bins. Default: infinity

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

Recursively merge the pair of neighboring bins using pvalue as threshold of Chi-square test until all neighboring bins are significantly different

Examples

weight_of_evidence.xlsx

See also

[model_save_scoring_code](#)
[woe_transform](#)
[woe_transform_from_file](#)

Return to the [index](#)

6.6 woe_xcat_ybin_from_file

Generates weight of evidence (WOE) of categorical independent variables and a binary dependent variable given a data file

`woe_xcat_ybin_from_file (filename, xNames, yName, pvalue, maxNumBins, weightName, delimiter)`

Returns

Weight of evidence (WOE) of categorical independent variables and binary dependent variable

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column

yName Dependent variable name

pvalue Optional: p-value for the threshold of merging groups. Default: 1

maxNumBins Optional: maximum number of bins. Default: infinity

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

Recursively merge the pair of neighboring bins using pvalue as threshold of Chi-square test until all neighboring bins are significantly different

Examples

`weight_of_evidence.xlsx`

See also

[model_save_scoring_code](#)

[woe_transform](#)

[woe_transform_from_file](#)

Return to the [index](#)

6.7 woe_xcat_ycont

Generates weight of evidence (WOE) of categorical independent variables and a continuous dependent variable given a data table

`woe_xcat_ycont (x, y, pvalue, maxNumBins, weight)`

Returns

Weight of evidence (WOE) of categorical independent variables and continuous dependent variable

Parameters

- x* Input data of categorical independent variables with headers in the first row
y Input data of continuous dependent variable with header in the first row
pvalue Optional: p-value for the threshold of merging groups. Default: 1
maxNumBins Optional: maximum number of bins. Default: infinity
weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

Recursively merge the pair of neighboring bins using pvalue as threshold of t-test until all neighboring bins are significantly different

Examples

weight_of_evidence.xlsx

See also

[model_save_scoring_code](#)
[woe_transform](#)
[woe_transform_from_file](#)

Return to the [index](#)

6.8 woe_xcat_ycont_from_file

Generates weight of evidence (WOE) of categorical independent variables and a continuous dependent variable given a data file

woe_xcat_ycont_from_file (filename, xNames, yName, pvalue, maxNumBins, weightName, delimiter)

Returns

Weight of evidence (WOE) of categorical independent variables and continuous dependent variable

Parameters

- filename* Input data file name. The first line of the file is the header line with variable names
xNames Independent variable names in one row or one column
yName Dependent variable name
pvalue Optional: p-value for the threshold of merging groups. Default: 1
maxNumBins Optional: maximum number of bins. Default: infinity
weightName Optional: weight variable name. Default: 1 for all weights
delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

Recursively merge the pair of neighboring bins using pvalue as threshold of t-test until all neighboring bins are significantly different

Examples

weight_of_evidence.xlsx

See also

[model_save_scoring_code](#)
[woe_transform](#)
[woe_transform_from_file](#)

Return to the [index](#)

6.9 woe_transform

Performs weight of evidence (WOE) transformation given a WOE model and a data table

`woe_transform (woeModel, inputData)`

Returns

Weight of evidence (WOE) transformtion given a WOE model and a data table

Parameters

woeModel A WOE model

inputData Input data with headers in the first row

Examples

weight_of_evidence.xlsx

See also

[woe_xcont_ybin](#)
[woe_xcont_ybin_from_file](#)
[woe_xcont_ycont](#)
[woe_xcont_ycont_from_file](#)
[woe_xcat_ybin](#)
[woe_xcat_ybin_from_file](#)
[woe_xcat_ycont](#)
[woe_xcat_ycont_from_file](#)

Return to the [index](#)

6.10 woe_transform_from_file

Performs weight of evidence (WOE) transformation given a WOE model and a data file

`woe_transform_from_file (woeModel, xNames, infilename, outfilename, delimiter)`

Returns

Weight of evidence (WOE) transformation given a WOE model and data table

Parameters

woeModel A WOE model

xNames Independent variable names in one row or one column

infilename Input data file name. The first line of the file is the header line with variable names

outfilename Output data file name

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

weight_of_evidence.xlsx

See also

[woe_xcont_ybin](#)
[woe_xcont_ybin_from_file](#)
[woe_xcont_ycont](#)
[woe_xcont_ycont_from_file](#)
[woe_xcat_ybin](#)
[woe_xcat_ybin_from_file](#)
[woe_xcat_ycont](#)
[woe_xcat_ycont_from_file](#)

Return to the [index](#)

Chapter 7

Principal Component Analysis and Factor Analysis Functions

PCA Performs principal component analysis

factor_analysis Performs factor analysis

7.1 PCA

Performs principal component analysis

PCA (inputData, covOrCorr)

Returns

Principal components

Parameters

inputData Input data with or without headers

covOrCorr Optional: COV, CORR, or MATRIX: analysis based on covariance (COV) or correlation (CORR) matrix calculated from inputData, or direct input (MATRIX). Default: COV

Examples

pca_factor_analysis.xlsx

Return to the [index](#)

7.2 factor_analysis

Performs factor analysis

factor_analysis (inputData, numFactors, covOrCorr)

Returns

Factors

Parameters

inputData Input data with or without headers

numFactors Number of factors

covOrCorr Optional: COV, CORR, or MATRIX: analysis based on covariance (COV) or correlation (CORR) matrix calculated from inputData, or direct input (MATRIX). Default: COV

Examples

pca_factor_analysis.xlsx

Return to the [index](#)

Chapter 8

Linear Regression Functions

linear_reg Builds a linear regression model given a data table

linear_reg_from_file Builds a linear regression model given a data file

linear_reg_forward_select Builds a linear regression model by forward selection given a data table

linear_reg_forward_select_from_file Builds a linear regression model by forward selection given a data file

linear_reg_score_from_coefs Scores a population from the coefficients of a linear regression model given a data table

linear_reg_piecewise Builds a two-segment piecewise linear regression model for each variable given a data table

linear_reg_piecewise_from_file Builds a two-segment piecewise linear regression model for each variable given a data file

poly_reg Builds a polynomial regression model given a data table

8.1 linear_reg

Builds a linear regression model given a data table

`linear_reg (x, y, weight, lambda)`

Returns

A linear regression model

Parameters

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

lambda Optional: a number or vector of ridge constants for ridge regression; if given a vector, the size must match with x. Default: 0

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.
In ridge regression, the ridge constant is added to each diagonal element of the correlation matrix of the independent variables,

$$X^T X \rightarrow X^T X + \lambda$$

where $X^T X$ is the correlation matrix of the independent variables and λ is a diagonal matrix with ridge constants.

Examples

linear_reg.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

8.2 linear_reg_from_file

Builds a linear regression model given a data file

linear_reg_from_file (filename, xNames, yName, weightName, lambda, delimiter)

Returns

A linear regression model

Parameters

filename Input data file name. The first line of the file is the header line with variable names
xNames Independent variable names in one row or one column
yName Dependent variable name
weightName Optional: weight variable name. Default: 1 for all weights
lambda Optional: a number or vector of ridge constants for ridge regression; if given a vector, the size must match with x. Default: 0
delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.
In ridge regression, the ridge constant is added to each diagonal element of the correlation matrix of the independent variables,

$$X^T X \rightarrow X^T X + \lambda$$

where $X^T X$ is the correlation matrix of the independent variables and λ is a diagonal matrix with ridge constants.

Examples

linear_reg.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

8.3 linear_reg_forward_select

Builds a linear regression model by forward selection given a data table

linear_reg_forward_select (x, y, pvalue, steps, startsWith, weight)

Returns

A linear regression model by forward selection

Parameters

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

pvalue p-value for the criteria for forward selection

steps Maximum number of variables to be selected, excluding startsWith variables

startsWith Optional: the names of variables which must be included in variable selection at the beginning. Default: empty

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.

Examples

linear_reg.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

8.4 linear_reg_forward_select_from_file

Builds a linear regression model by forward selection given a data file

`linear_reg_forward_select_from_file (filename, xNames, yName, pvalue, steps, startsWith, weightName, delimiter)`

Returns

A linear regression model by forward selection

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column

yName Dependent variable name

pvalue p-value for forward selection

steps Maximum number of variables to be selected, excluding startsWith variables

startsWith Optional: the names of variables which must be included in variable selection at the beginning

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.

Examples

linear_reg.xlsx

See also

[model_save_scoring_code](#)

[model_score](#)

[model_score_from_file](#)

[model_eval](#)

[model_eval_from_file](#)

Return to the [index](#)

8.5 linear_reg_score_from_coefs

Scores a population from the coefficients of a linear regression model given a data table

`linear_reg_score_from_coefs (coefs, inputData)`

Returns

A column of scores of a population from a linear regression

Parameters

coefs Coefficients of linear regression model. Two column table with variable names in the 1st column and coefficients in the 2nd column

inputData Input data with headers in the first row

Examples

linear_reg.xlsx

Return to the [index](#)

8.6 linear_reg_piecewise

Builds a two-segment piecewise linear regression model for each variable given a data table

linear_reg_piecewise (x, y, weight)

Returns

Two-segment piecewise linear regression model for each variable

Parameters

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Examples

linear_reg.xlsx

See also

[model_save_scoring_code](#)

[model_score](#)

Return to the [index](#)

8.7 linear_reg_piecewise_from_file

Builds a two-segment piecewise linear regression model for each variable given a data file

linear_reg_piecewise_from_file (filename, xNames, yName, weightName, delimiter)

Returns

A linear regression model

Parameters

- filename** Input data file name. The first line of the file is the header line with variable names
- xNames** Independent variable names in one row or one column
- yName** Dependent variable name
- weightName** Optional: weight variable name. Default: 1 for all weights
- delimiter** Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

linear_reg.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)

Return to the [index](#)

8.8 poly_reg

Builds a polynomial regression model given a data table

poly_reg (x, y, numDegree, weight, lambda)

Returns

A polynomial regression model

Parameters

- x** Input data of independent variable with header in the first row
- y** Input data of dependent variable with header in the first row
- numDegree** Input data of the degree of the polynomial to fit
- weight** Optional: input data of weight variable with header in the first row. Default: 1 for all weights
- lambda** Optional: a number or vector of ridge constants for ridge regression; if given a vector, the size must match with numDegree. Default: 0

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.

In polynomial regression, the independent variables are derived from input variable x as $x, x^2, x^3, \dots, x^{numDegree}$.

In ridge regression, the ridge constant is added to each diagonal element of the correlation matrix of the independent variables,

$$X^T X \rightarrow X^T X + \lambda$$

where $X^T X$ is the correlation matrix of the independent variables and λ is a diagonal matrix with ridge constants.

Examples

[linear_reg.xlsx](#)

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

Chapter 9

Partial Least Square Regression Functions

[pls_reg](#) Builds a partial least square regression model given a data table

[pls_reg_from_file](#) Builds a partial least square regression model given a data file

9.1 `pls_reg`

Builds a partial least square regression model given a data table

`pls_reg (x, y, ncc, weight)`

Returns

A partial least square regression model

Parameters

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

ncc Number of cardinal components

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

All records with at least one missing variable of *x*, *y*, or *weight* are excluded from regression.

Examples

`pls_reg.xlsx`

See also

[model_save_scoring_code](#)

[model_score](#)

[model_score_from_file](#)

[model_eval](#)

[model_eval_from_file](#)

Return to the [index](#)

9.2 pls_reg_from_file

Builds a partial least square regression model given a data file

pls_reg_from_file (filename, xNames, yName, ncc, weightName, delimiter)

Returns

A partial least square regression model

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column

yName Dependent variable name

ncc Number of cardinal components

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.

Examples

pls_reg.xlsx

See also

[model_save_scoring_code](#)

[model_score](#)

[model_score_from_file](#)

[model_eval](#)

[model_eval_from_file](#)

Return to the [index](#)

Chapter 10

Logistic Regression Functions

logistic_reg Builds a logistic regression model given a data table

logistic_reg_from_file Builds a logistic regression model given a data file

logistic_reg_forward_select Builds a logistic regression model by forward selection given a data table

logistic_reg_forward_select_from_file Builds a logistic regression model by forward selection given a data file

logistic_reg_score_from_coefs Scores a population from the coefficients of a logistic regression model given a data table

10.1 logistic_reg

Builds a logistic regression model given a data table

`logistic_reg (x, y, weight)`

Returns

A logistic regression model

Parameters

x Input data of independent variables with headers in the first row

y Input data of binary dependent variable with header in the first row

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

All records with at least one missing variable of *x*, *y*, or *weight* are excluded from regression.

Examples

`logistic_reg.xlsx`

See also

[model_save_scoring_code](#)

[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

10.2 logistic_reg_from_file

Builds a logistic regression model given a data file

logistic_reg_from_file (filename, xNames, yName, weightName, delimiter)

Returns

A logistic regression model

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column

yName Binary dependent variable name

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.

Examples

logistic_reg.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

10.3 logistic_reg_forward_select

Builds a logistic regression model by forward selection given a data table

logistic_reg_forward_select (x, y, pvalue, steps, startsWith, weight)

Returns

A logistic regression model by forward selection

Parameters

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

pvalue p-value for forward selection

steps maximum number of variables to be selected, excluding startsWith variables

startsWith Optional: the names of variables which must be included in variable selection at the beginning

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.

Examples

logistic_reg.xlsx

See also

[model_save_scoring_code](#)

[model_score](#)

[model_score_from_file](#)

[model_eval](#)

[model_eval_from_file](#)

Return to the [index](#)

10.4 logistic_reg_forward_select_from_file

Builds a logistic regression model by forward selection given a data file

logistic_reg_forward_select_from_file (filename, xNames, yName, pvalue, steps, startsWith, weight-Name, delimiter)

Returns

A logistic regression model by forward selection

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column
yName Dependent variable name
pvalue p-value for forward selection
steps maximum number of variables to be selected, excluding startsWith variables
startsWith Optional: the names of variables which must be included in variable selection at the beginning
weightName Optional: weight variable name. Default: 1 for all weights
delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.

Examples

logistic_reg.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

10.5 logistic_reg_score_from_coefs

Scores a population from the coefficients of a logistic regression model given a data table

logistic_reg_score_from_coefs (coefs, inputData)

Returns

Scores of a population

Parameters

coefs Coefficients of a logistic regression model. Two column table with variable names in the 1st column and coefficients in the 2nd columns

inputData Input data with header in the first rows in the first row

Examples

logistic_reg.xlsx

Return to the [index](#)

Chapter 11

Time Series Analysis Functions

ts_acf Calculates the autocorrelation functions (ACF) given a data table

ts_pacf Calculates the partial autocorrelation functions (PACF) given a data table

ts_ccf Calculates the cross correlation functions (CCF) given two data tables

Box_white_noise_test Tests if a time series is a white noise by Box-Ljung or Box-Pierce test

Mann_Kendall_trend_test Tests if a time series has a trend

ADF_test Tests whether a unit root is in a time series using Augmented Dickey-Fuller (ADF) test

ts_diff Calculates the differences given lag and order

ts_sma Calculates the simple moving average (SMA) of a time series

lowess Performs locally weighted scatterplot smoothing (lowess)

natural_cubic_spline Performs natural cubic spline

garch Estimates the parameters of GARCH(1, 1) model

stochastic_process Estimates the parameters of a stochastic process: normal, lognormal, or shifted log-normal

stochastic_process_simulate Simulates a stochastic process: normal, lognormal, or shifted lognormal

Holt_Winters Performs Holt-Winters exponential smoothing

Holt_Winters_forecast Performs forecast given a Holt-Winters exponential smoothing

HP_filter Performs the Hodrick-Prescott filter for a time-series data

arima Builds an ARIMA model

sarima Builds a seasonal ARIMA (SARIMA) model

arima_forecast Performs forecast given an ARIMA model

sarima_forecast Performs forecast given a seasonal ARIMA (SARIMA) model

arima_simulate Simulates an ARIMA process

sarima_simulate Simulates a seasonal ARIMA (SARIMA) process

arma_to_ma Converts an ARMA process to a pure MA process

arma_to_ar Converts an ARMA process to a pure AR process

acf_of_arma Calculates the autocorrelation functions (ACF) of an ARMA process

11.1 ts_acf

Calculates the autocorrelation functions (ACF) given a data table

`ts_acf (x, maxLag)`

Returns

The autocorrelation functions (ACF)

Parameters

x Input data of univariate time series with header in the first row

maxLag Optional: maximum lag for ACF. Default: 10

Remarks

Let $x_i (i = 1, 2, \dots, n)$ be n data points. The autocorrelation for the lag $k \geq 0$, ρ_k , is

$$\rho_k = \frac{\sum_{t=1}^{n-k} (x_t - \mu_x) \cdot (x_{t+k} - \mu_x)}{\sum_{t=1}^n (x_t - \mu_x)^2}$$

where $\mu_x = \sum_{t=1}^n x_t / n$. The possible maximum lag is $n - 1$.

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.2 ts_pacf

Calculates the partial autocorrelation functions (PACF) given a data table

`ts_pacf (x, maxLag)`

Returns

The partial autocorrelation functions (PACF)

Parameters

x Input data of univariate time series with header in the first row

maxLag Optional: maximum lag for PACF. Default: 10

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.3 ts_ccf

Calculates the cross correlation functions (CCF) given two data tables

ts_ccf (x, y, maxLag)

Returns

The cross correlation functions (CCF)

Parameters

x Input data of univariate time series with header in the first row

y Input data of univariate time series with header in the first row

maxLag Optional: maximum lag for CCF. Default: 10

Remarks

Let $(x_i, y_i)(i = 1, 2, \dots, n)$ be n data points. The cross correlation for the lag k , $\rho_k(x, y)$, is

$$\rho_k(x, y) = \frac{\sum_{k=\max(1, 1-k)}^{\min(n-k, n)} (x_t - \mu_x) \cdot (y_{t+k} - \mu_y)}{\sqrt{\sum_{k=1}^n (x_t - \mu_x)^2} \cdot \sqrt{\sum_{k=1}^n (y_t - \mu_y)^2}}$$

where $\mu_x = \sum_{k=1}^n x_t/n$ and $\mu_y = \sum_{k=1}^n y_t/n$. The possible minimum lag is $-(n-1)$ and the possible maximum lag is $n-1$.

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.4 Box_white_noise_test

Tests if a time series is a white noise by Box-Ljung or Box-Pierce test

Box_white_noise_test (x, maxLag, method, numParams)

Returns

Chi-squared and p-value of Box-Ljung or Box-Pierce test

Parameters

x Input data of univariate time series with header in the first row

maxLag Optional: maximum lag for Box-Ljung or Box-Pierce test. Default: 1

method Optional: Box-Ljung or Box-Pierce. Default: Box-Ljung

numParams Optional: number of parameters. Default: 0 (without model)

Remarks

Let $x_i (i = 1, 2, \dots, n)$ be n data points. Its autocorrelations are $\hat{\rho}_k, k = 1, 2, \dots, K$. The Box-Ljung test statistic is

$$Q(K) = n(n+2) \sum_{k=1}^K \frac{\hat{\rho}_k^2}{n-k}$$

The Box-Pierce test statistic is

$$Q(K) = n \sum_{k=1}^K \hat{\rho}_k^2$$

$Q(K) \sim \chi_{K-m}^2$, where m is the number of parameters of a model or 0 without model.

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.5 Mann_Kendall_trend_test

Tests if a time series has a trend

Mann_Kendall_trend_test (x, frequency)

Returns

Mann-Kendall trend test statistic and p-value

Parameters

x Input data of univariate time series with header in the first row

frequency Optional: number of data points per period with seasonality. Default: 1

Remarks

Let $x_i (i = 1, 2, \dots, n)$ be n data points. The Mann-Kendall trend test statistic is

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \text{sign}(x_j - x_i)$$

where $sign(x)$ is the sign function which is 1 for positive x , -1 for negative x , and 0 for zero x . The variance of S is

$$\text{var}(S) = \frac{1}{18} \left[n(n-1)(2n+5) - \sum_{i=1}^g t_i(t_i-1)(2t_i+5) \right]$$

where g is the number of tied groups and t_i is the number of data points in the i th tied group.

$$D = \left[\frac{1}{2}n(n-1) - \frac{1}{2} \sum_{i=1}^g t_i(t_i-1) \right]^{1/2} \left[\frac{1}{2}n(n-1) \right]^{1/2}$$

The Kendall's τ is defined as

$$\tau = \frac{S}{D}$$

The normalized Mann-Kendall trend test statistic is

$$Z = \begin{cases} \frac{S-1}{\sqrt{\text{var}(S)}} & \text{if } S > 1 \\ 0 & \text{if } S = 0 \\ \frac{S+1}{\sqrt{\text{var}(S)}} & \text{if } S < 0 \end{cases}$$

Under the null hypothesis, $Z \sim N[0, 1]$. The p-value is

$$p\text{-value} = 2(1 - \Phi(|Z|))$$

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.6 ADF_test

Tests whether a unit root is in a time series using Augmented Dickey-Fuller (ADF) test

ADF_test (x, maxLag, timeTerms)

Returns

ADF test statistic and critical values

Parameters

x Input data of univariate time series with header

maxLag Optional: Number for the max lag p. Default: $\lfloor 12(n/100)^{1/4} \rfloor$; n is the number of points in x

timeTerms Optional: String indicating which time terms to use in test; 'nc' = no constant, 'c' = constant only, 'ct' = constant and trend, 'ctt' = constant, trend, and trend squared. Default: 'ct'

Remarks

The Augmented Dickey-Fuller (ADF) test performs multiple linear regression on the following model:

$$\Delta x_t = \alpha + \beta_1 t + \beta_2 t^2 + \gamma x_{t-1} + \phi_1 \Delta x_{t-1} + \dots + \phi_p \Delta x_{t-p} + \epsilon_t$$

where

- $\alpha, \beta_1, \beta_2, \gamma, \phi_1, \dots, \phi_p$: are the regression coefficients to estimate.
- $t, t^2, x_{t-1}, \Delta x_{t-1}, \dots, \Delta x_{t-p}$: are the variables.
- p : is the maximum lag given by the maxLag parameter.

The $\alpha, \beta_1 t$, and $\beta_2 t^2$ terms are included or excluded based on the timeTerms parameter. We construct a matrix X to represent the independent variables. Each row represents a point from the input data and each column represents a variable. The ADF test statistic (τ) is given by,

$$\tau = \frac{\hat{\gamma}}{SE(\hat{\gamma})}$$

where

- $\hat{\gamma}$: is the estimate of the coefficient γ .
- $SE(\hat{\gamma})$: is the standard error of this estimate.

$SE(\hat{\gamma})$ is given by,

$$SE(\hat{\gamma}) = \sqrt{\frac{\sum_t (\Delta x_t - \Delta \hat{x}_t)^2}{df} [(X^T X)^{-1}]_{kk}}$$

where the degree of freedom df is the number of points minus the number of variables in the linear regression and k is the column in X that x_{t-1} represents.

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.7 ts_diff

Calculates the differences given lag and order

ts_diff (x, lag, order)

Returns

The differences given lag and order

Parameters

x Input data of univariate time series with header in the first row

lag Optional: lag for differences. Default: 1

order Optional: order for differences. Default: 1

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.8 ts_sma

Calculates the simple moving average (SMA) of a time series

ts_sma (x, n)

Returns

The simple moving average (SMA) of a time series

Parameters

x Input data of univariate time series with header in the first row

n Number of data points for average

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.9 lowess

Performs locally weighted scatterplot smoothing (lowess)

lowess (x, y, xForSmoothing, fraction, degree)

Returns

Locally weighted scatterplot smoothing points for xForSmoothing

Parameters

x Input data of x with header in the first row

y Input data of y with header in the first row

xForSmoothing Optional: Input data of x for smoothing with header in the first row. Default: x (the first input)

fraction Optional: A fraction of data points used in local regression, typically between 0.1 and 0.8. Default: 2/3

degree Optional: degree of local polynomials, 0 - moving average, 1 - locally linear, 2 - locally quadratic, etc. Default: 1

Remarks

Let $x_i (i = 1, 2, \dots, n)$ be n points for the local regression. The weight for each data point is defined as the tricube weight function:

$$w(d_i) = \begin{cases} (1 - |d_i|^3)^3 & \text{if } |d_i| \leq 1 \\ 0 & \text{if } |d_i| > 1 \end{cases}$$

where d_i is defined as

$$d_i = \frac{|x - x_i|}{\max_{j=1,2,\dots,n} |x - x_j|}$$

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.10 natural_cubic_spline

Performs natural cubic spline

natural_cubic_spline (xKnots, yKnots, x)

Returns

The points for x from the natural cubic spline

Parameters

xKnots Input data of x of the knots with header in the first row

yKnots Input data of y of the knots with header in the first row

x Input data of x for calculating with header in the first row

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.11 garch

Estimates the parameters of GARCH(1, 1) model

garch (returns, initialOmega, initialAlpha, initialBeta)

Returns

Parameters of GARCH(1, 1) model from the maximum likelihood estimation

Parameters

returns Input data of returns with header in the first row

initialOmega Optional: initial omega value. Default: 0.00001

initialAlpha Optional: initial alpha value. Default: 0.1

initialBeta Optional: initial beta value. Default: 0.85

Remarks

The GARCH(1, 1) (generalized autoregressive conditional heteroscedasticity) model with three parameters, ω, α, β .

$$\sigma_{t+1}^2 = \omega + \alpha r_t^2 + \beta \sigma_t^2$$

The parameters are estimated from the maximum likelihood method.

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.12 stochastic_process

Estimates the parameters of a stochastic process: normal, lognormal, or shifted lognormal

stochastic_process (x, process)

Returns

Parameters of a stochastic process: normal, lognormal, or shifted lognormal

Parameters

x Input data of values from a process with header

process The process to be estimated, N (NORMAL), LN (LOGNORMAL), or SLN (SHIFTEDLOG-NORMAL)

Remarks

The parameters of a process are estimated from the maximum likelihood method. For a realization of a process, $\{x_1, x_2, \dots, x_n\}$, the likelihood function is

$$L = \prod_{t=2}^n P(x_t | \{x_1, x_2, \dots, x_{t-1}\})$$

where $P(x_t | \{x_1, x_2, \dots, x_{t-1}\})$ is the probability density function given the past observations. A normal process with two parameters, a drift μ and volatility σ is

$$x_t = x_{t-1} + \mu + \sigma \epsilon_t$$

The log-likelihood function is

$$-2 \ln L = \sum_{t=2}^n \left[\frac{(x_t - x_{t-1} - \mu)^2}{\sigma^2} + \ln(2\pi\sigma^2) \right]$$

The maximum likelihood estimation (MLE) is

$$\mu = \frac{1}{n-1} \sum_{t=2}^n (x_t - x_{t-1})$$

$$\sigma^2 = \frac{1}{n-1} \sum_{t=2}^n (x_t - x_{t-1} - \mu)^2$$

$$-2 \ln L = (n-1)(\ln(2\pi\sigma^2) + 1)$$

A lognormal process with two parameters, a drift μ and volatility σ is

$$\ln x_t = \ln x_{t-1} + (\mu - \sigma^2/2) + \sigma\epsilon_t$$

Let $\tilde{\mu} = \mu - \sigma^2/2$, the MLE is

$$\tilde{\mu} = \frac{1}{n-1} \sum_{t=2}^n (\ln x_t - \ln x_{t-1})$$

$$\sigma^2 = \frac{1}{n-1} \sum_{t=2}^n (\ln x_t - \ln x_{t-1} - \tilde{\mu})^2$$

$$-2 \ln L = (n-1)(\ln(2\pi\sigma^2) + 1) + 2 \sum_{t=2}^n \ln x_t$$

A shifted lognormal process with three parameters, a drift μ , volatility σ , and shift s is

$$\ln(x_t + s) = \ln(x_{t-1} + s) + (\mu - \sigma^2/2) + \sigma\epsilon_t$$

There is no closed-form formula to estimate these three parameters. Given a shift s , let $\tilde{\mu} = \mu - \sigma^2/2$, the MLE is

$$\tilde{\mu} = \frac{1}{n-1} \sum_{t=2}^n (\ln(x_t + s) - \ln(x_{t-1} + s))$$

$$\sigma^2 = \frac{1}{n-1} \sum_{t=2}^n (\ln(x_t + s) - \ln(x_{t-1} + s) - \tilde{\mu})^2$$

$$-2 \ln L = (n-1)(\ln(2\pi\sigma^2) + 1) + 2 \sum_{t=2}^n \ln(x_t + s)$$

The optimal shift is estimated by minimizing $-2 \ln L$ using any numerical optimization algorithm.

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.13 stochastic_process_simulate

Simulates a stochastic process: normal, lognormal, or shifted lognormal

stochastic_process_simulate (process, numPoints, initialValue, mu, sigma, shift, seed)

Returns

A stochastic process: normal, lognormal, or shifted lognormal

Parameters

- process** The name of a process, N (NORMAL), LN (LOGNORMAL), or SLN (SHIFTEDLOGNORMAL)
- numPoints** The number of points
- initialValue** The initial value of a process
- mu** Optional: drift of a process. Default: 0
- sigma** Optional: volatility of a process. Default: 1
- shift** Optional: shift for a shifted lognormal process. Default: 0
- seed** Optional: non-negative integer seed for generating random numbers. Default: 0 (use timer)

Remarks

A normal process with two parameters, a drift μ and volatility σ is

$$x_t = x_{t-1} + \mu + \sigma\epsilon_t$$

A lognormal process with two parameters, a drift μ and volatility σ is

$$\ln x_t = \ln x_{t-1} + (\mu - \sigma^2/2) + \sigma\epsilon_t$$

A shifted lognormal process with three parameters, a drift μ , volatility σ , and shift s is

$$\ln(x_t + s) = \ln(x_{t-1} + s) + (\mu - \sigma^2/2) + \sigma\epsilon_t$$

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.14 Holt_Winters

Performs Holt-Winters exponential smoothing

Holt_Winters (x, type, frequency, numForecast, excludeList, force)

Returns

Parameters of Holt-Winters model and forecast

Parameters

- x** Input data of univariate time series with header in the first row
- type** Optional: type of Holt-Winters exponential smoothing (1, 2, or 3). 1 for local smoothing, 2 for time series with trend, and 3 for time series with trend and seasonality. Default: 1
- frequency** Optional: number of data points per period with seasonality. Default: 1
- numForecast** Optional: number of future data points to predict. Default: 10
- excludeList** Optional: a list of numbers that are excluded and treated as missing in Holt_Winters model. Default: empty

force Optional: an indicator of whether to force model fitting (1) or not (0). If force is 1 and it cannot find a stable solution in model type 3, it fits a simpler model type 2, and so on. Default: 0

Remarks

For single exponential smoothing for a time series without trend and seasonality (type = 1), the updating rule is

$$S_t = \alpha x_{t-1} + (1 - \alpha)S_{t-1}, \quad 0 \leq \alpha \leq 1$$

The forecast is $F_{t+k} = \alpha x_t + (1 - \alpha)S_t, k > 0$.

For double exponential smoothing for a time series with trend (type = 2), the updating rule is

$$\begin{aligned} S_t &= \alpha x_t + (1 - \alpha)(S_{t-1} + T_{t-1}), \quad 0 \leq \alpha \leq 1 \\ T_t &= \beta(S_t - S_{t-1}) + (1 - \beta)T_{t-1}, \quad 0 \leq \beta \leq 1 \\ F_t &= S_{t-1} + T_{t-1} \end{aligned}$$

The forecast is $F_{t+k} = S_t + kT_t, k > 0$.

For triple exponential smoothing for a time series with trend and seasonality (type = 3), the updating rule is

$$\begin{aligned} S_t &= \alpha x_t / I_{t-L} + (1 - \alpha)(S_{t-1} + T_{t-1}), \quad 0 \leq \alpha \leq 1 \\ T_t &= \beta(S_t - S_{t-1}) + (1 - \beta)T_{t-1}, \quad 0 \leq \beta \leq 1 \\ I_t &= \gamma x_t / S_t + (1 - \gamma)I_{t-L}, \quad 0 \leq \gamma \leq 1 \\ F_t &= (S_{t-1} + T_{t-1})I_{t-L} \end{aligned}$$

where L is the frequency of a time series with seasonality. The forecast is $F_{t+k} = (S_t + kT_t)I_{t-L+k}, k > 0$.

- S_t is the local level
- T_t is the trend
- I_t is the seasonal indices
- F_t is the forecast

Examples

time_series_analysis.xlsx

See also

[Holt_Winters_forecast](#)

Return to the [index](#)

11.15 Holt_Winters_forecast

Performs forecast given a Holt-Winters exponential smoothing

Holt_Winters_forecast (x, params, initialTrend, initialSeasonalIndices, numForecast, excludeList)

Returns

The forecast from Holt-Winters model

Parameters

x Input data of univariate time series with header in the first row

params Model parameters in one row or one column. It can be either 1 number (alpha), 2 numbers (alpha and beta), or 3 numbers (alpha, beta, and gamma)

initialTrend Optional: initial trend for model type 2 or 3. Default: 0

initialSeasonalIndices Optional: initial seasonal indices in one row or one column for model type 3. Default: empty for no seasonal indices

numForecast Optional: number of future data points to predict. Default: 0

excludeList Optional: a list of numbers that are excluded and treated as missing in Holt_Winters model. Default: empty

Remarks

For single exponential smoothing for a time series without trend and seasonality (type = 1), the updating rule is

$$S_t = \alpha x_{t-1} + (1 - \alpha)S_{t-1}, \quad 0 \leq \alpha \leq 1$$

The forecast is $F_{t+k} = \alpha x_t + (1 - \alpha)S_t, k > 0$.

For double exponential smoothing for a time series with trend (type = 2), the updating rule is

$$\begin{aligned} S_t &= \alpha x_t + (1 - \alpha)(S_{t-1} + T_{t-1}), \quad 0 \leq \alpha \leq 1 \\ T_t &= \beta(S_t - S_{t-1}) + (1 - \beta)T_{t-1}, \quad 0 \leq \beta \leq 1 \\ F_t &= S_{t-1} + T_{t-1} \end{aligned}$$

The forecast is $F_{t+k} = S_t + kT_t, k > 0$.

For triple exponential smoothing for a time series with trend and seasonality (type = 3), the updating rule is

$$\begin{aligned} S_t &= \alpha x_t / I_{t-L} + (1 - \alpha)(S_{t-1} + T_{t-1}), \quad 0 \leq \alpha \leq 1 \\ T_t &= \beta(S_t - S_{t-1}) + (1 - \beta)T_{t-1}, \quad 0 \leq \beta \leq 1 \\ I_t &= \gamma x_t / S_t + (1 - \gamma)I_{t-L}, \quad 0 \leq \gamma \leq 1 \\ F_t &= (S_{t-1} + T_{t-1})I_{t-L} \end{aligned}$$

where L is the frequency of a time series with seasonality. The forecast is $F_{t+k} = (S_t + kT_t)I_{t-L+k}, k > 0$.

- S_t is the local level
- T_t is the trend
- I_t is the seasonal indices
- F_t is the forecast

Examples

time_series_analysis.xlsx

See also

[Holt_Winters](#)

Return to the [index](#)

11.16 HP_filter

Performs the Hodrick-Prescott filter for a time-series data

HP_filter (x, lambda)

Returns

Trend and cyclical components

Parameters

x Input data of univariate time series with header in the first row

lambda Non-negative smoothing parameter. Commonly suggested values are: 100 for annual data, 1600 for quarterly data, 14400 for monthly data

Remarks

The Hodrick-Prescott filter with a smoothing factor for a time-series data is to find trend, T , by minimizing the following function

$$f = \sum_{i=1}^n (x_i - T_i)^2 + \lambda \sum_{i=2}^{n-1} (T_{i-1} + T_{i+1} - 2T_i)^2$$

The cyclical component is $C_i = x_i - T_i, i = 1, 2, \dots, n$. In matrix form, the trend can be solved by

$$(I + \lambda A^T A)T = X$$

where A is a $(n - 2) \times n$ matrix:

$$A = \begin{bmatrix} 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & 1 & -2 & 1 \end{bmatrix}$$

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.17 arima

Builds an ARIMA model

arima (x, hasMean, p, d, q, numForecast, lowerAndUpperBound, maxNumGenerations, seed, showDetails)

Returns

An ARIMA model

Parameters

x Input data of univariate time series with header in the first row

hasMean Optional: an indicator (TRUE or FALSE) whether to build model with mean. Default: TRUE

p Optional: the order of autoregressive (AR) terms. Default: 0 (no AR terms)

d Optional: the order of the difference. Default: 0

q Optional: the order of moving averaging (MA) terms. Default: 0 (no MA terms)

numForecast Optional: number of forecast points in future. Default: 0

lowerAndUpperBound Optional: a table containing constraints for model parameters in the order of: mean (if hasMean), (AR1, AR2, ..., MA1, MA2, ...). Each row is for each parameter in 2 columns: lower bound and upper bound

maxNumGenerations Optional: maximum number of generations, a positive integer. Default: 200

seed Optional: a non-negative integer seed for generating random numbers. 0 is for using timer. Default: 100

showDetails Optional: an indicator (TRUE or FALSE) whether to show the details of parameter estimation through optimization. Default: FALSE

Remarks

The ARIMA (p, d, q) process is in the following form

$$\phi_p(B)(1 - B)^d(X_t - \mu) = \theta_q(B)a_t$$

where

$$\phi_p(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \dots - \varphi_p B^p$$

and

$$\theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + \dots + \theta_q B^q$$

Let $Y_t = (1 - B)^d(X_t - \mu)$, the process can be expressed explicitly,

$$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \dots + \varphi_p Y_{t-p} + a_t + \theta_1 a_{t-1} + \dots + \theta_q a_{t-q}$$

where

- B is the backshift operator, $BX_t = X_{t-1}$
- μ is the mean
- d is the order of difference
- $\varphi_i (i = 1, 2, \dots, p)$ are the coefficients of autoregressive (AR) terms
- $\theta_i (i = 1, 2, \dots, q)$ are the coefficients of moving averaging (MA) terms
- $a_i (i = t, t - 1, \dots) \in N[0, \sigma^2]$ is white noise

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.18 sarima

Builds a seasonal ARIMA (SARIMA) model

sarima (x, hasMean, p, d, q, seasonalPeriod, seasonalP, seasonalD, seasonalQ, numForecast, lowerAndUpperBound, maxNumGenerations, seed, showDetails)

Returns

A seasonal ARIMA (SARIMA) model

Parameters

x Input data of univariate time series with header in the first row

hasMean Optional: an indicator (TRUE or FALSE) whether to build model with mean. Default: TRUE

p Optional: the order of autoregressive (AR) terms. Default: 0 (no AR terms)

d Optional: the order of the difference. Default: 0

q Optional: the order of moving averaging (MA) terms. Default: 0 (no MA terms)

seasonalPeriod Optional: seasonal period. Default: 0

seasonalP Optional: the order of seasonal autoregressive (AR) terms. Default: 0 (no seasonal AR terms)

seasonalD Optional: the order of seasonal difference. Default: 0

seasonalQ Optional: the order of seasonal moving averaging (MA) terms. Default: 0 (no seasonal MA terms)

numForecast Optional: number of forecast points in future. Default: 0

lowerAndUpperBound Optional: a table containing constraints for model parameters in the order of: mean (if hasMean), (AR1, ..., MA1, ...), seasonal (AR1, ..., MA1, ...). Each row is for each parameter in 2 columns: lower bound and upper bound

maxNumGenerations Optional: maximum number of generations, a positive integer. Default: 200

seed Optional: a non-negative integer seed for generating random numbers. 0 is for using timer. Default: 100

showDetails Optional: an indicator (TRUE or FALSE) whether to show the details of parameter estimation through optimization. Default: FALSE

Remarks

The seasonal ARIMA (SARIMA) process is an ARIMA $(p, d, q) \times (P, D, Q)_s$ process and it is in the following form

$$\Phi_P(B^s)\phi_p(B)(1-B^s)^D(1-B)^d(X_t - \mu) = \Theta_Q(B^s)\theta_q(B)a_t$$

where

$$\phi_p(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \dots - \varphi_p B^p, \Phi_P(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \dots - \Phi_P B^{Ps}$$

and

$$\theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + \dots + \theta_q B^q, \Theta_Q(B^s) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \dots + \Theta_Q B^{Qs}$$

Here

- B is the backshift operator, $BX_t = X_{t-1}$

- μ is the mean
- d is the order of difference
- $\varphi_i (i = 1, 2, \dots, p)$ are the coefficients of autoregressive (AR) terms
- $\theta_i (i = 1, 2, \dots, q)$ are the coefficients of moving averaging (MA) terms
- s is the seasonal period
- D is the order of seasonal difference
- $\Phi_i (i = 1, 2, \dots, P)$ are the coefficients of seasonal autoregressive (AR) terms
- $\Theta_i (i = 1, 2, \dots, Q)$ are the coefficients of seasonal moving averaging (MA) terms
- $a_i (i = t, t - 1, \dots) \in N[0, \sigma^2]$ is white noise

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.19 arima_forecast

Performs forecast given an ARIMA model

arima_forecast (x, numForecast, mean, ar, d, ma, sigma)

Returns

The forecast from an ARIMA model

Parameters

x Input data of univariate time series with header in the first row

numForecast The number of forecast points

mean Optional: mean of the process. Default: 0

ar Optional: coefficients of autoregressive (AR) terms in one row or one column. Default: empty (no AR terms)

d Optional: order of the difference. Default: 0

ma Optional: coefficients of moving averaging (MA) terms in one row or one column. Default: empty (no MA terms)

sigma Optional: standard deviation of white noise term. Default: 1

Remarks

The ARIMA (p, d, q) process is in the following form

$$\phi_p(B)(1 - B)^d(X_t - \mu) = \theta_q(B)a_t$$

where

$$\phi_p(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \dots - \varphi_p B^p$$

and

$$\theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + \dots + \theta_q B^q$$

Let $Y_t = (1 - B)^d(X_t - \mu)$, the process can be expressed explicitly,

$$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \dots + \varphi_p Y_{t-p} + a_t + \theta_1 a_{t-1} + \dots + \theta_q a_{t-q}$$

where

- B is the backshift operator, $BX_t = X_{t-1}$
- μ is the mean
- d is the order of difference
- $\varphi_i (i = 1, 2, \dots, p)$ are the coefficients of autoregressive (AR) terms
- $\theta_i (i = 1, 2, \dots, q)$ are the coefficients of moving averaging (MA) terms
- $a_i (i = t, t-1, \dots) \in N[0, \sigma^2]$ is white noise

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.20 sarima_forecast

Performs forecast given a seasonal ARIMA (SARIMA) model

sarima_forecast (x, numForecast, mean, ar, d, ma, seasonalPeriod, seasonalAR, seasonalD, seasonalMA, sigma)

Returns

The forecast from a SARIMA model

Parameters

x Input data of univariate time series with header in the first row

numForecast The number of forecast points in future

mean Optional: mean of the process. Default: 0

ar Optional: coefficients of autoregressive (AR) terms in one row or one column. Default: empty (no AR terms)

d Optional: order of the difference. Default: 0

ma Optional: coefficients of moving averaging (MA) terms in one row or one column. Default: empty (no MA terms)

seasonalPeriod Optional: seasonal period. Default: 0

seasonalAR Optional: coefficients of seasonal autoregressive (AR) terms in one row or one column. Default: empty (no seasonal AR terms)

seasonalD Optional: order of the seasonal difference. Default: 0

seasonalMA Optional: coefficients of seasonal moving averaging (MA) terms in one row or one column. Default: empty (no seasonal MA terms)

sigma Optional: standard deviation of white noise term. Default: 1

Remarks

The seasonal ARIMA (SARIMA) process is an ARIMA $(p, d, q) \times (P, D, Q)_s$ process and it is in the following form

$$\Phi_P(B^s)\phi_p(B)(1-B^s)^D(1-B)^d(X_t - \mu) = \Theta_Q(B^s)\theta_q(B)a_t$$

where

$$\phi_p(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \dots - \varphi_p B^p, \Phi_P(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \dots - \Phi_P B^{Ps}$$

and

$$\theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + \dots + \theta_q B^q, \Theta_Q(B^s) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \dots + \Theta_Q B^{Qs}$$

Here

- B is the backshift operator, $BX_t = X_{t-1}$
- μ is the mean
- d is the order of difference
- $\varphi_i (i = 1, 2, \dots, p)$ are the coefficients of autoregressive (AR) terms
- $\theta_i (i = 1, 2, \dots, q)$ are the coefficients of moving averaging (MA) terms
- s is the seasonal period
- D is the order of seasonal difference
- $\Phi_i (i = 1, 2, \dots, P)$ are the coefficients of seasonal autoregressive (AR) terms
- $\Theta_i (i = 1, 2, \dots, Q)$ are the coefficients of seasonal moving averaging (MA) terms
- $a_i (i = t, t-1, \dots) \in N[0, \sigma^2]$ is white noise

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.21 arima_simulate

Simulates an ARIMA process

arima_simulate (numPoints, mean, ar, d, ma, sigma, seed)

Returns

A time series simulated from an ARIMA process

Parameters

numPoints The number of points

mean Optional: mean of the process. Default: 0

ar Optional: coefficients of autoregressive (AR) terms in one row or one column. Default: empty (no AR terms)

d Optional: order of the difference. Default: 0

ma Optional: coefficients of moving averaging (MA) terms in one row or one column. Default: empty (no MA terms)

sigma Optional: standard deviation of noise term. Default: 1

seed Optional: non-negative integer seed for generating random numbers. Default: 0 (use timer)

Remarks

The ARIMA (p, d, q) process is in the following form

$$\phi_p(B)(1 - B)^d(X_t - \mu) = \theta_q(B)a_t$$

where

$$\phi_p(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \dots - \varphi_p B^p$$

and

$$\theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + \dots + \theta_q B^q$$

Let $Y_t = (1 - B)^d(X_t - \mu)$, the process can be expressed explicitly,

$$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \dots + \varphi_p Y_{t-p} + a_t + \theta_1 a_{t-1} + \dots + \theta_q a_{t-q}$$

where

- B is the backshift operator, $BX_t = X_{t-1}$
- μ is the mean
- d is the order of difference
- $\varphi_i (i = 1, 2, \dots, p)$ are the coefficients of autoregressive (AR) terms
- $\theta_i (i = 1, 2, \dots, q)$ are the coefficients of moving averaging (MA) terms
- $a_i (i = t, t - 1, \dots) \in N[0, \sigma^2]$ is white noise

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.22 sarima_simulate

Simulates a seasonal ARIMA (SARIMA) process

sarima_simulate (numPoints, mean, ar, d, ma, seasonalPeriod, seasonalAR, seasonalD, seasonalMA, sigma, seed)

Returns

A time series simulated from a seasonal ARIMA (SARIMA) process

Parameters

numPoints The number of points

- mean** Optional: mean of the process. Default: 0
- ar** Optional: coefficients of autoregressive (AR) terms in one row or one column. Default: empty (no AR terms)
- d** Optional: order of the difference. Default: 0
- ma** Optional: coefficients of moving averaging (MA) terms in one row or one column. Default: empty (no MA terms)
- seasonalPeriod** Optional: seasonal period. Default: 0
- seasonalAR** Optional: coefficients of seasonal autoregressive (AR) terms in one row or one column. Default: empty (no seasonal AR terms)
- seasonalD** Optional: the order of seasonal difference. Default: 0
- seasonalMA** Optional: coefficients of seasonal moving averaging (MA) terms in one row or one column. Default: empty (no seasonal MA terms)
- sigma** Optional: standard deviation of noise term. Default: 1
- seed** Optional: non-negative integer seed for generating random numbers. Default: 0 (use timer)

Remarks

The seasonal ARIMA (SARIMA) process is an $ARIMA(p, d, q) \times (P, D, Q)_s$ process and it is in the following form

$$\Phi_P(B^s)\phi_p(B)(1-B^s)^D(1-B)^d(X_t - \mu) = \Theta_Q(B^s)\theta_q(B)a_t$$

where

$$\phi_p(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \dots - \varphi_p B^p, \Phi_P(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \dots - \Phi_P B^{Ps}$$

and

$$\theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + \dots + \theta_q B^q, \Theta_Q(B^s) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \dots + \Theta_Q B^{Qs}$$

Here

- B is the backshift operator, $BX_t = X_{t-1}$
- μ is the mean
- d is the order of difference
- $\varphi_i (i = 1, 2, \dots, p)$ are the coefficients of autoregressive (AR) terms
- $\theta_i (i = 1, 2, \dots, q)$ are the coefficients of moving averaging (MA) terms
- s is the seasonal period
- D is the order of seasonal difference
- $\Phi_i (i = 1, 2, \dots, P)$ are the coefficients of seasonal autoregressive (AR) terms
- $\Theta_i (i = 1, 2, \dots, Q)$ are the coefficients of seasonal moving averaging (MA) terms
- $a_i (i = t, t-1, \dots) \in N[0, \sigma^2]$ is white noise

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.23 arma_to_ma

Converts an ARMA process to a pure MA process

arma_to_ma (ar, ma, maxLag)

Returns

A pure MA process

Parameters

ar Optional: coefficients of autoregressive (AR) terms in one row or one column. Default: empty (no autoregressive AR terms)

ma Optional: coefficients of moving averaging (MA) terms in one row or one column. Default: empty (no moving averaging MA terms)

maxLag Optional: maximum lag for MA process. Default: 10

Remarks

The ARMA (p, q) process is in the following form

$$x_t = \varphi_1 x_{t-1} + \dots + \varphi_p x_{t-p} + a_t + \theta_1 a_{t-1} + \dots + \theta_q a_{t-q}$$

where

- $\varphi_i (i = 1, 2, \dots, p)$ are the coefficients of autoregressive (AR) terms
- $\theta_i (i = 1, 2, \dots, q)$ are the coefficients of moving averaging (MA) terms
- $a_i (i = t, t-1, \dots, t-q) \in N[0, \sigma^2]$ is white noise

It can be converted to a pure MA process

$$x_t = a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \dots$$

$\psi_i (i = 1, 2, \dots)$ can be found in terms of the following recursive relation

$$\begin{aligned} \psi_0 &= 1 \\ \psi_i &= \sum_{j=0}^{i-1} \psi_j \varphi_{i-j} + \theta_i, \quad i \geq 1 \end{aligned}$$

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.24 arma_to_ar

Converts an ARMA process to a pure AR process

arma_to_ar (ar, ma, maxLag)

Returns

A pure AR process

Parameters

ar Optional: coefficients of autoregressive terms in one row or one column. Default: empty (no autoregressive terms)

ma Optional: coefficients of moving averaging terms in one row or one column. Default: empty (no moving averaging terms)

maxLag Optional: maximum lag for AR process. Default: 10

Remarks

The ARMA (p, q) process is in the following form

$$x_t = \varphi_1 x_{t-1} + \dots + \varphi_p x_{t-p} + a_t + \theta_1 a_{t-1} + \dots + \theta_q a_{t-q}$$

where

- $\varphi_i (i = 1, 2, \dots, p)$ are the coefficients of autoregressive terms
- $\theta_i (i = 1, 2, \dots, q)$ are the coefficients of moving averaging terms
- $a_i (i = t, t-1, \dots, t-q) \in N[0, \sigma^2]$ is white noise

It can be converted to a pure AR process

$$x_t = a_t + \pi_1 x_{t-1} + \pi_2 x_{t-2} + \dots$$

$\pi_i (i = 1, 2, \dots)$ can be found in terms of the following recursive relation

$$\begin{aligned} \pi_0 &= -1 \\ \pi_i &= - \sum_{j=0}^{i-1} \pi_j \theta_{i-j} + \varphi_i, \quad i \geq 1 \end{aligned}$$

Examples

time_series_analysis.xlsx

Return to the [index](#)

11.25 acf_of_arma

Calculates the autocorrelation functions (ACF) of an ARMA process

acf_of_arma (ar, ma, sigma, maxLag)

Returns

The autocorrelation functions (ACF) of an ARMA process

Parameters

ar Optional: coefficients of autoregressive terms in one row or one column. Default: empty (no autoregressive terms)

ma Optional: coefficients of moving averaging terms in one row or one column. Default: empty (no moving averaging terms)

sigma Optional: the standard deviation of the white noise. Default: 1

maxLag Optional: maximum lag for ACF. Default: 10

Remarks

The ARMA (p, q) process is in the following form

$$x_t = \varphi_1 x_{t-1} + \dots + \varphi_p x_{t-p} + a_t + \theta_1 a_{t-1} + \dots + \theta_q a_{t-q}$$

where

- $\varphi_i (i = 1, 2, \dots, p)$ are the coefficients of autoregressive terms
- $\theta_i (i = 1, 2, \dots, q)$ are the coefficients of moving averaging terms
- $a_i (i = t, t-1, \dots, t-q) \in N[0, \sigma^2]$ is white noise

It can be converted to a pure MA process

$$x_t = a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \dots$$

Let $\gamma(k) = E[X_t X_{t-k}]$ and $\theta_0 = 1$, we have

$$\begin{aligned} \gamma(k) &= \varphi_1 \gamma(k-1) + \dots + \varphi_p \gamma(k-p) + \sigma^2 \sum_{j=k}^q \psi_{j-k} \theta_j, & k \leq q \\ \gamma(k) &= \varphi_1 \gamma(k-1) + \dots + \varphi_p \gamma(k-p), & k > q \end{aligned}$$

For $k = 0, 1, 2, \dots, p$, we have $(p+1)$ equations for $\gamma(0), \gamma(1), \dots, \gamma(p)$. Therefore we can solve the linear equations for $\gamma(0), \gamma(1), \dots, \gamma(p)$. For $\gamma(k), k > p$, we calculate them from the above recursive equation.

Examples

time_series_analysis.xlsx

Return to the [index](#)

Chapter 12

Linear and Quadratic Discriminant Analysis Functions

LDA Performs the linear discriminant analysis

QDA Performs the quadratic discriminant analysis

12.1 LDA

Performs the linear discriminant analysis

LDA (*x*, *y*, priors)

Returns

The coefficients of linear discriminant functions

Parameters

x Input data of independent variables with headers

y Input data of dependent variable with header

priors Optional: the prior probabilities. The sum of prior probabilities must be 1. Default: use proportion in each group if missing

Remarks

The linear discriminant function is

$$\begin{aligned} d_k(x) &= x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \ln \pi_k \\ &= x^T w_k + c_k \end{aligned}$$

where $w_i = \Sigma^{-1} \mu_k$, $c_k = -\frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \ln \pi_k$, and π_k is the prior probability, $k = 1, 2, \dots, K$. The pooled within-class covariance matrix, Σ , is calculated from the covariance matrices from each group,

$$\Sigma = \frac{1}{n - K} \sum_{k=1}^K (n_k - 1) \Sigma_k$$

If the covariance matrix in each group Σ_k is singular, it has been regularized by

$$\Sigma_k \rightarrow (1 - \lambda) \Sigma_k + \lambda \text{diag}(\Sigma_k)$$

where $\lambda = 10^{-6}$. The posterior probability of belonging to group k is

$$p(k|x) = \frac{e^{d_k(x)}}{\sum_{k=1}^K e^{d_k(x)}}$$

The misclassification error is

$$E = \sum_{k=1}^K e_k \pi_k$$

where e_k is the percentage of misclassified data points in the k th group.

Examples

lda_qda.xlsx

See also

[QDA](#)
[model_score](#)

Return to the [index](#)

12.2 QDA

Performs the quadratic discriminant analysis

QDA (x, y, priors)

Returns

The coefficients of quadratic discriminant functions

Parameters

x Input data of independent variables with headers

y Input data of dependent variable with header

priors Optional: the prior probabilities. The sum of prior probabilities must be 1. Default: use proportion in each group if missing

Remarks

The quadratic discriminant function is

$$\begin{aligned} d_k(x) &= -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \ln(\det \Sigma_k) + \ln \pi_k \\ &= -\frac{1}{2} x^T \Sigma_k^{-1} x + x^T w_k + c_k \end{aligned}$$

where $w_i = \Sigma_k^{-1} \mu_k$, $c_k = -\frac{1}{2} \mu_k^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \ln(\det \Sigma_k) + \ln \pi_k$, and π_k is the prior probability, $k = 1, 2, \dots, K$. If the covariance matrix in each group Σ_k is singular, it has been regularized by

$$\Sigma_k \rightarrow (1 - \lambda) \Sigma_k + \lambda \text{diag}(\Sigma_k)$$

where $\lambda = 10^{-6}$. The posterior probability of belonging to group k is

$$p(k|x) = \frac{e^{d_k(x)}}{\sum_{k=1}^K e^{d_k(x)}}$$

The misclassification error is

$$E = \sum_{k=1}^K e_k \pi_k$$

where e_k is the percentage of misclassified data points in the k th group.

Examples

lda_qda.xlsx

See also

[LDA](#)
[model_score](#)

Return to the [index](#)

Chapter 13

Survival Analysis Functions

Kaplan_Meier Performs Kaplan-Meier survival analysis

13.1 Kaplan_Meier

Performs Kaplan-Meier survival analysis

`Kaplan_Meier` (`time`, `status`, `classVar`, `labelForCensor`)

Returns

A Kaplan-Meier estimate of the survival function

Parameters

time Input time of event or censored in one column with a header in the first row

status Input status either an event (1) or censored (0) in one column with a header in the first row

classVar Optional: class variable used to form subgroups in one column. Default: missing classified as one group

labelForCensor Optional: label for censored points. Default: '+' if missing

Remarks

Sort the n time points in ascending order, $t_1 \leq t_2 \leq \dots \leq t_n$. Let n_i be the number of observations at risk prior to the time t_i and d_i is the number of events observed at time t_i , $i = 1, 2, \dots, n$. The Kaplan-Meier estimate of the survival function is

$$S(t) = \prod_{t_i \leq t} \frac{n_i - d_i}{n_i}$$

and its estimated variance in Greenwood's formula is

$$var(S(t)) = S^2(t) \sum_{t_i \leq t} \frac{d_i}{n_i(n_i - d_i)}$$

Examples

survival_analysis.xlsx

Return to the [index](#)

Chapter 14

Correspondence Analysis Functions

corresp_analysis Performs simple correspondence analysis for a two-way cross table

14.1 corresp_analysis

Performs simple correspondence analysis for a two-way cross table

`corresp_analysis (crossTable, numDim, supRows, supCols)`

Returns

The row and column profiles and the principal coordinates

Parameters

crossTable A two-way cross table with each entry for frequency and with row and column labels

numDim Optional: the number of dimensions of principal axes. Default: 2

supRows Optional: a two-way cross table for the supplementary rows with row labels and without column labels. It has the same number of columns as crossTable

supCols Optional: a two-way cross table for the supplementary columns with column labels and without row labels. It has the same number of rows as crossTable

Remarks

The Chi-Square statistic for a two-way cross table of frequencies is calculated by

$$\chi_P^2 = n \sum_{i=1}^I \sum_{j=1}^J E_{ij}^2$$

where

$$E_{ij} = \frac{p_{ij} - p_{i+}p_{+j}}{\sqrt{p_{i+}p_{+j}}}$$

The percents of frequency are calculated from frequency table n_{ij}

$$p_{ij} = n_{ij}/n, \quad p_{i+} = \sum_{j=1}^J n_{ij}/n, \quad p_{+j} = \sum_{i=1}^I n_{ij}/n, \quad i = 1, 2, \dots, I, \quad j = 1, 2, \dots, J$$

The singular value decomposition (SVD) of the matrix E is

$$E = U\Lambda V^T$$

The principal coordinates (F) and standard coordinates (X) of the row profiles are, respectively

$$F = D_r^{-1/2}U\Lambda, \quad X = D_r^{-1/2}U$$

The principal coordinates (G) and standard coordinates (Y) of the column profiles are, respectively

$$G = D_c^{-1/2}V\Lambda^T, \quad Y = D_c^{-1/2}V$$

The contribution of the j th principal axis to the inertia for the i th row is

$$Ctr(i, j) = \frac{p_{i+} F_{ij}^2}{\sum_{i=1}^I p_{i+} F_{ij}^2}$$

The squared correlation of the j th principal axis for the i th row is

$$Corr(i, j) = \frac{F_{ij}^2}{\sum_{i=1}^I F_{ij}^2}$$

For supplementary rows ($I_s \times J$), define the following table from the frequency table q_{ij}

$$Q_{ij} = \frac{q_{ij}/q_{i+} - p_{+j}}{\sqrt{p_{+j}}}, \quad i = 1, 2, \dots, I_s, \quad j = 1, 2, \dots, J$$

the principal coordinates (F) and standard coordinates (X) of the row profiles are, respectively

$$F = QV, \quad X = QV\Lambda^{-1}$$

For supplementary columns ($I \times J_s$), define the following table from the frequency table q_{ij}

$$Q_{ij} = \frac{q_{ij}/q_{+j} - p_{i+}}{\sqrt{p_{i+}}}, \quad i = 1, 2, \dots, I, \quad j = 1, 2, \dots, J_s$$

the principal coordinates (G) and standard coordinates (Y) of the column profiles are, respectively

$$G = Q^T U, \quad Y = Q^T U(\Lambda^{-1})^T$$

Examples

correspondence_analysis.xlsx

Return to the [index](#)

Chapter 15

Naive Bayes Classifier Functions

[naive_bayes_classifier](#) Builds a naive Bayes classification model given a data table

[naive_bayes_classifier_from_file](#) Builds a naive Bayes classification model given a data file

15.1 naive_bayes_classifier

Builds a naive Bayes classification model given a data table

`naive_bayes_classifier (x, y)`

Returns

Naive Bayes classification

Parameters

x Input data of independent variables with headers in the first row. Each variable must be either categorical variable or discretized numerical variable

y Input data of dependent variable with header in the first row. It can be binary or multi-class variable

Examples

`naive_bayes.xlsx`

See also

[model_score](#)

[model_score_from_file](#)

Return to the [index](#)

15.2 naive_bayes_classifier_from_file

Builds a naive Bayes classification model given a data file

`naive_bayes_classifier_from_file (filename, xNames, yName, delimiter)`

Returns

Naive Bayes classification

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column. Each variable must be either categorical variable or discretized numerical variable

yName Dependent variable name. It can be binary or multi-class variable

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

naive_bayes.xlsx

See also

[model_score](#)

[model_score_from_file](#)

Return to the [index](#)

Chapter 16

Tree-Based Model Functions

tree Builds a regression or classification tree model given a data table

tree_from_file Builds a regression or classification tree model given a data file

tree_boosting_logistic_reg Builds a logistic boosting tree model given a data table

tree_boosting_logistic_reg_from_file Builds a logistic boosting tree model given a data file

tree_boosting_ls_reg Builds a least square boosting tree model given a data table

tree_boosting_ls_reg_from_file Builds a least square boosting tree model given a data file

16.1 tree

Builds a regression or classification tree model given a data table

`tree (x, y, treeConfig, weight)`

Returns

A regression or classification tree

Parameters

x Input data of independent variables with headers in the first row

y Input data of binary dependent variable with header in the first row

treeConfig Configuration of tree. Two column input with names in the 1st column and values in the 2nd column. For example:

method	LS, GINI or ENTROPY
numTerminals	4
minSize	50
minChild	30
maxLevel	3

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

When the method is "LS", a regression tree is built using least square criteria. When the method is "GINI" or "ENTROPY", a classification tree is built using information gains from "GINI" or "ENTROPY", respectively. For detailed description of algorithms, please see the reference [2].

Examples

decision_tree_based_model.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

16.2 tree_from_file

Builds a regression or classification tree model given a data file

tree_from_file (filename, xNames, yNname, treeConfig, weightName, delimiter)

Returns

A regression or classification tree

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column

yName Dependent variable name

treeConfig Configuration of tree. Two column input with names in the 1st column and values in the 2nd column. For example:

method	LS, GINI or ENTROPY
numTerminals	4
minSize	50
minChild	30
maxLevel	3

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

When the method is "LS", a regression tree is built using least square criteria. When the method is "GINI" or "ENTROPY", a classification tree is built using information gains from "GINI" or "ENTROPY", respectively. For detailed description of algorithms, please see the reference [2].

Examples

decision_tree_based_model.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

16.3 tree_boosting_logistic_reg

Builds a logistic regression boosting tree model given a data table

tree_boosting_logistic_reg (x, y, boostingTreeConfig, weight)

Returns

A logistic regression boosting tree model

Parameters

x Input data of independent variables with headers in the first row

y Input data of binary dependent variable with header in the first row

boostingTreeConfig Configuration of boosting trees. Two column input with names in the 1st column and values in the 2nd column. For example:

learnRate	0.1
numTrees	20
numTerminals	4
minSize	50
minChild	30
maxLevel	3

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

A sequence of least square regression trees are built. Each tree is built based on the residual of the output of the model so far.

$$T(x) = T_0(x) + \gamma_1 T_1(x) + \gamma_2 T_2(x) + \dots + \gamma_M T_M(x)$$

where M is the number of trees and $\gamma_i (i = 1, 2, \dots, M)$ are learning rates. For detailed description of algorithms, please see the reference [2].

Examples

decision_tree_based_model.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

16.4 tree_boosting_logistic_reg_from_file

Builds a logistic regression boosting tree model given a data file

tree_boosting_logistic_reg_from_file (filename, xNames, yName, boostingTreeConfig, weightName, delimiter)

Returns

A logistic regression boosting tree

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column

yName Dependent variable name

boostingTreeConfig Configuration of boosting trees. Two column input with names in the 1st column and values in the 2nd column. For example:

learnRate	0.1
numTrees	20
numTerminals	4
minSize	50
minChild	30
maxLevel	3

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

A sequence of least square regression trees are built. Each tree is built based on the residual of the output of the model so far.

$$T(x) = T_0(x) + \gamma_1 T_1(x) + \gamma_2 T_2(x) + \dots + \gamma_M T_M(x)$$

where M is the number of trees and $\gamma_i (i = 1, 2, \dots, M)$ are learning rates. For detailed description of algorithms, please see the reference [2].

Examples

decision_tree_based_model.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

16.5 tree_boosting_ls_reg

Builds a least square boosting tree model given a data table

tree_boosting_ls_reg (x, y, boostingTreeConfig, weight)

Returns

A least square boosting tree

Parameters

x Input data of independent variables with headers in the first row

y Input data of binary dependent variable with header in the first row

boostingTreeConfig Configuration of boosting trees. Two column input with names in the 1st column and values in the 2nd column. For example:

learnRate	0.1
numTrees	20
numTerminals	4
minSize	50
minChild	30
maxLevel	3

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Remarks

A sequence of least square regression trees are built. Each tree is built based on the residual of the output of the model so far.

$$T(x) = T_0(x) + \gamma_1 T_1(x) + \gamma_2 T_2(x) + \dots + \gamma_M T_M(x)$$

where M is the number of trees and $\gamma_i (i = 1, 2, \dots, M)$ are learning rates. For detailed description of algorithms, please see the reference [2].

Examples

decision_tree_based_model.xlsx

See also

[model_save_scoring_code](#)

[model_score](#)

[model_score_from_file](#)

[model_eval](#)

[model_eval_from_file](#)

Return to the [index](#)

16.6 tree_boosting_ls_reg_from_file

Builds a least square boosting tree model given a data file

tree_boosting_ls_reg_from_file (filename, xNames, yName, boostingTreeConfig, weightName, delimiter)

Returns

A least square boosting tree

Parameters

filename Input data file name. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column

yName Dependent variable name

boostingTreeConfig Configuration of boosting trees. Two column input with names in the 1st column and values in the 2nd column. For example:

learnRate	0.1
numTrees	20
numTerminals	4
minSize	50
minChild	30
maxLevel	3

weightName Optional: weight variable name. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

A sequence of least square regression trees are built. Each tree is built based on the residual of the output of the model so far.

$$T(x) = T_0(x) + \gamma_1 T_1(x) + \gamma_2 T_2(x) + \dots + \gamma_M T_M(x)$$

where M is the number of trees and $\gamma_i (i = 1, 2, \dots, M)$ are learning rates. For detailed description of algorithms, please see the reference [2].

Examples

decision_tree_based_model.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

Chapter 17

Clustering and Segmentation Functions

k_means Performs K-means clustering analysis given a data table

k_means_from_file Performs K-means clustering analysis given a data file

cmds Performs classical multi-dimensional scaling

mds Performs multi-dimensional scaling by Sammon's non-linear mapping

17.1 k_means

Performs K-means clustering analysis given a data table

`k_means (x, numClusters, seed, showAssignments)`

Returns

A assignment

Parameters

x Input data of independent variables with headers in the first row

numClusters Number of clusters

seed Optional: seed for randomizing initial cluster assignment. Default: 100

showAssignments Optional: indicator (1 or 0) to show assignment of cluster for each record. 1 for yes, 0 for no . Default: 0 (no)

Examples

clustering_segmentation.xlsx

Return to the [index](#)

17.2 k_means_from_file

Performs K-means clustering analysis given a data file

k_means_from_file (filename, varNames, numClusters, seed, delimiter, showAssignments)

Returns

A assignment

Parameters

filename Input data file name. The first line of the file is the header line with variable names

varNames Variable names in one row or one column

numClusters Number of clusters

seed Optional: seed for the randomized initial cluster assignment. Default: 100

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

showAssignments Optional: indicator (1 or 0) to show assignment of cluster for each record. 1 for yes, 0 for no . Default: 0 (no)

Examples

clustering_segmentation.xlsx

Return to the [index](#)

17.3 cmds

Performs classical multi-dimensional scaling

cmds (distanceMatrix, dim)

Returns

A classical multi-dimensional scaling

Parameters

distanceMatrix A distance matrix

dim Dimensions to project to

Examples

clustering_segmentation.xlsx

Return to the [index](#)

17.4 mds

Performs multi-dimensional scaling by Sammon's non-linear mapping

mds (distanceMatrix, dim, maxIteration, seed)

Returns

A multi-dimensional scaling by Sammon's non-linear mapping

Parameters

distanceMatrix A distance matrix

dim Dimensions to project to

maxIteration Maximum iterations

seed Optional: non-negative integer seed for generating random numbers. Default: 0 (use timer)

Examples

clustering_segmentation.xlsx

Return to the [index](#)

Chapter 18

Neural Network Functions

neural_net Builds a neural network model given a data table

neural_net_from_file Builds a neural network model given a data file

18.1 neural_net

Builds a neural network model given a data table

`neural_net (x, y, model, numHiddenNodes, epochs, seed, weight, xTest, yTest, weightTest)`

Returns

A neural network model

Parameters

x Input data of independent variables with headers in the first row for training

y Input data of dependent variable with header in the first row for training. Either continuous target for LS objective or binary target for ML objective

model LS or ML. LS for least squares objective for continuous target, ML for maximum likelihood objective for binary target

numHiddenNodes Numbers of nodes in hidden layers input in one row or one column

epochs Optional: number of epochs. Default: 20

seed Optional: seed for generating random numbers. Default: 100

weight Optional: input data of weight variable with header in the first row for training. Default: 1 for all weights

xTest Optional: input data of independent variables with headers in the first row for testing. Default: use training set for testing

yTest Optional: input data of dependent variable with header in the first row for testing. Default: use training set for testing

weightTest Optional: input data of weight variable with header in the first row for testing. Default: 1 for all weights

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.

Examples

neural_network_model.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

18.2 neural_net_from_file

Builds a neural network model given a data file

neural_net_from_file (filename, xNames, yName, model, numHiddenNodes, epochs, seed, weightName, xTestNames, yTestName, weightTestName, delimiter)

Returns

A neural network model

Parameters

filename Input data file name for training. The first line of the file is the header line with variable names

xNames Independent variable names in one row or one column for training

yName Dependent variable name for training. Either continuous target for LS objective or binary target for ML objective

model LS or ML. LS for least squares objective for continuous target, ML for maximum likelihood objective for binary target

numHiddenNodes Numbers of nodes in hidden layers input in one row or one column

epochs Optional: number of epochs. Default: 20

seed Optional: seed for generating random numbers. Default: 100

weightTrainName Optional: weight variable name for training. Default: 1 for all weights

testFileName Optional: input data file name for testing

xTestNames Optional: independent variable names in one row or one column for testing. Default: use training set for testing

yTestName Optional: dependent variable name for testing. Default: use training set for testing

weightTestName Optional: weight variable name for testing. Default: 1 for all weights

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

All records with at least one missing variable of x, y, or weight are excluded from regression.

Examples

neural_network_model.xlsx

See also

[model_save_scoring_code](#)
[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

Chapter 19

Support Vector Machine Functions

svm Builds a support vector machine (SVM) model given a data table

svm_from_file Builds a support vector machine (SVM) model given a data file

19.1 svm

Builds a support vector machine (SVM) model given a data table

svm (directory, x, y, svmType, kernelType, C, epsilon, degree, gamma, coef0, seed, doScaling)

Returns

A support vector machine (SVM) model

Parameters

directory Working directory for temporary files

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

svmType SVM type: SVC or SVR. SVC for classification problem, SVR for regression problem

kernelType Kernel type: LINEAR, POLYNOMIAL, RBF, or SIGMOID

C Optional: Penalty parameter C in objective function. Default: 1

epsilon Optional: Epsilon in ε – SVR model. Default: 0.1

degree Optional: Degree in kernel function for POLYNOMIAL. Default: 3

gamma Optional: Gamma in kernel function for POLYNOMIAL/RBF/SIGMOID. Default: 1 / number of variables

coef0 Optional: Coefficient 0 in kernel function for POLYNOMIAL/SIGMOID. Default: 0

seed Optional: Seed for generating random numbers. Default: 100

doScaling Optional: An indicator (1 or 0) to do automatic scaling of the input data. 1 for yes, 0 for no. Default: 1 (yes)

Remarks

All records with at least one missing variable of x, y, or weight are excluded from modeling.

Given a set of data points $\{(x_i, y_i), i = 1, 2, \dots, m\}$, where x_i is an input and $y_i \in \{1, -1\}$ is a binary target output, C –Support Vector Classification ($C - SVC$) solves the following classification problem

$$\begin{aligned} & \underset{w, b, \xi}{\text{minimize}} && \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ & \text{subject to} && y_i(w^T x_i + b) + \xi_i \geq 1 \\ & && \xi_i \geq 0, \quad i = 1, 2, \dots, m \end{aligned}$$

Here C is a given constant.

Given a set of data points $\{(x_i, y_i), i = 1, 2, \dots, m\}$, where x_i is an input and $y_i \in R$ is a continuous target output, ε –Support Vector Regression ($\varepsilon - SVR$) solves the following regression problem

$$\begin{aligned} & \underset{w, b, \xi, \xi^*}{\text{minimize}} && \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m (\xi_i + \xi_i^*) \\ & \text{subject to} && -(\varepsilon + \xi_i) \leq y_i - (w^T x_i + b) \leq \varepsilon + \xi_i^* \\ & && \xi_i \geq 0, \quad \xi_i^* \geq 0, \quad i = 1, 2, \dots, m \end{aligned}$$

Here C and ε are given constants.

The four most common kernels are

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + c_0)^d$
- RBF (Radial Basis Function): $K(x_i, x_j) = e^{-\gamma |x_i - x_j|^2}$
- Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + c_0)$

Here d, γ, c_0 are kernel parameters.

The implementation is based on LIBSVM described in reference [3].

Examples

support_vector_machine.xlsx

See also

[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

19.2 svm_from_file

Builds a support vector machine (SVM) model given a data file

svm_from_file (directory, filename, xNames, yName, C, epsilon, degree, gamma, coef0, seed, doScaling, delimiter)

Returns

A support vector machine (SVM) model

Parameters

- directory** Working directory for temporary files
- filename** Input data file name for training. The first line of the file is the header line with variable names
- xNames** Independent variable names in one row or one column for training set
- yName** Dependent variable name for training set
- svmType** SVM type: SVC or SVR. SVC for classification problem, SVR for regression problem
- kernelType** Kernel type: LINEAR, POLYNOMIAL, RBF, or SIGMOID
- C** Optional: Penalty parameter C in objective function. Default: 1
- epsilon** Optional: Epsilon in ε – SVR model. Default: 0.1
- degree** Optional: Degree in kernel function for POLYNOMIAL. Default: 3
- gamma** Optional: Gamma in kernel function for POLYNOMIAL/RBF/SIGMOID. Default: 1 / number of variables
- coef0** Optional: Coefficient 0 in kernel function for POLYNOMIAL/SIGMOID. Default: 0
- seed** Optional: Seed for generating random numbers. Default: 100
- doScaling** Optional: An indicator (1 or 0) to do automatic scaling of the input data. 1 for yes, 0 for no. Default: 1 (yes)
- delimiter** Optional: One character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Remarks

All records with at least one missing variable of x, y, or weight are excluded from modeling.
 Given a set of data points $\{(x_i, y_i), i = 1, 2, \dots, m\}$, where x_i is an input and $y_i \in \{1, -1\}$ is a binary target output, C–Support Vector Classification (C – SVC) solves the following classification problem

$$\begin{aligned} &\underset{w, b, \xi}{\text{minimize}} && \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ &\text{subject to} && y_i(w^T x_i + b) + \xi_i \geq 1 \\ &&& \xi_i \geq 0, \quad i = 1, 2, \dots, m \end{aligned}$$

Here C is a given constant.

Given a set of data points $\{(x_i, y_i), i = 1, 2, \dots, m\}$, where x_i is an input and $y_i \in R$ is a continuous target output, ε –Support Vector Regression (ε – SVR) solves the following regression problem

$$\begin{aligned} &\underset{w, b, \xi, \xi^*}{\text{minimize}} && \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m (\xi_i + \xi_i^*) \\ &\text{subject to} && -(\varepsilon + \xi_i) \leq y_i - (w^T x_i + b) \leq \varepsilon + \xi_i^* \\ &&& \xi_i \geq 0, \xi_i^* \geq 0, \quad i = 1, 2, \dots, m \end{aligned}$$

Here C and ε are given constants.

The four most common kernels are

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + c_0)^d$
- RBF (Radial Basis Function): $K(x_i, x_j) = e^{-\gamma |x_i - x_j|^2}$
- Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + c_0)$

Here d, γ , c_0 are kernel parameters.

The implementation is based on LIBSVM described in reference [3].

Examples

support_vector_machine.xlsx

See also

[model_score](#)
[model_score_from_file](#)
[model_eval](#)
[model_eval_from_file](#)

Return to the [index](#)

Chapter 20

Optimization Functions

linear_prog Solves a linear programming problem: $f(x) = c x$

quadratic_prog Solves a quadratic programming problem: $f(x) = c x + 0.5x^T H x$

lcp Solves a linear complementarity programming problem

nls_solver Solves a nonlinear least-square problem using the Levenberg-Marquardt algorithm

diff_evol_solver Solves a minimization problem given a function and lower/upper bounds of variables using differential evolution solver

diff_evol_min_solver Solves a minimization problem given a function, lower/upper bounds of variables, and data table using differential evolution solver

diff_evol_nls_solver Solves a nonlinear least squares problem given a function and lower/upper bounds of variables using differential evolution solver

transportation_solver Solves a transportation problem: find the number of units to ship from each source to each destination that minimizes or maximizes the total cost

assignment_solver Solves an assignment problem: find the optimal assignment that minimizes or maximizes the total cost

netflow_solver Solves a minimum or maximum cost network flow problem: to find optimal flows that minimize or maximize the total cost

maxflow_solver Solves a maximum flow problem: to find optimal flows that maximize the total flows from the start node to the end node

shortest_path_solver Solves the shortest path problem: to find the shortest path from the start node to the end node

20.1 linear_prog

Solves a linear programming problem: $f(x) = c x$

`linear_prog (c, constraints, maxOrMin)`

Returns

A solution of a linear programming problem

Parameters

c The coefficients of x in the objective function in one row or one column

constraints The constraints in rows excluding primary constraints

maxOrMin Optional: the objective to seek. MAX for maximizing or MIN for minimizing the objective function. Default: MAX

Remarks

The linear programming with n primary constraints and $m(m = m_1 + m_2 + m_3)$ additional constraints is

$$\begin{array}{ll} \text{maximize} & z = c \cdot x \\ \text{subject to} & a_i \cdot x \leq b_i \quad i = 1, \dots, m_1 \\ & a_i \cdot x \geq b_i \quad i = m_1 + 1, \dots, m_1 + m_2 \\ & a_i \cdot x = b_i \quad i = m_1 + m_2 + 1, \dots, m_1 + m_2 + m_3 \\ \text{with} & x_j \geq 0, \quad j = 1, \dots, n \end{array}$$

The constraints can be in any order. The optional input, maxOrMin, controls the problem as a maximization (default) or minimization problem.

Examples

optimization.xlsx

Return to the [index](#)

20.2 quadratic_prog

Solves a quadratic programming problem: $f(x) = c \cdot x + 0.5x^T H x$

quadratic_prog (c, H, constraints, minOrMax)

Returns

A solution of a quadratic programming problem

Parameters

c The coefficients of the linear terms of x in the objective function in one row or one column

H The coefficients of the quadratic terms of x in the objective function

constraints The linear constraints in rows

minOrMax Optional: the objective to seek. MIN for minimizing or MAX for maximizing the objective function. Default: MIN

Remarks

The quadratic programming with $m(m = m_1 + m_2 + m_3)$ constraints is

$$\begin{array}{ll} \text{minimize} & f(x) = c^T x + \frac{1}{2} x^T H x \\ \text{subject to} & a_i \cdot x \leq b_i \quad i = 1, \dots, m_1 \\ & a_i \cdot x \geq b_i \quad i = m_1 + 1, \dots, m_1 + m_2 \\ & a_i \cdot x = b_i \quad i = m_1 + m_2 + 1, \dots, m_1 + m_2 + m_3 \end{array}$$

The constraints can be in any order. The optional input, minOrMax, controls the problem as a minimization (default) or maximization problem.

Examples

optimization.xlsx

Return to the [index](#)

20.3 lcp

Solves a linear complementarity programming problem

`lcp (m, q)`

Returns

A solution of a linear complementarity programming problem

Parameters

m an $n \times n$ matrix

q a column vector $n \times 1$

Remarks

The linear complementarity programming is

$$\begin{aligned} w &= m z + q \\ x^T z &= 0 \\ w, z &\geq 0 \end{aligned}$$

where m is an $n \times n$ matrix and w, z, q are $n \times 1$ vectors.

Examples

optimization.xlsx

Return to the [index](#)

20.4 nls_solver

Solves a nonlinear least squares problem using the Levenberg-Marquardt algorithm

`nls_solver (func, params, x, y, weight)`

Returns

The solution to a nonlinear least squares problem

Parameters

func An expression for a function in one column. Use semicolons to separate sub-expressions. For example, $z1 := x*x + y*y; z2 := x*x - y*y; \sin(z1) + \cos(z2)$

params The parameter names in the first column and the initial values in the second column, optional minimum values and maximum values in the third column and fourth column

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

Examples

optimization.xlsx

Return to the [index](#)

20.5 diff_evol_solver

Solves a minimization problem given a function and lower/upper bounds of variables using differential evolution solver

diff_evol_solver (func, lowerAndUpperBound, maxNumGenerations, seed)

Returns

A solution to an optimization problem

Parameters

func An expression for a function in one column. Use semicolons to separate sub-expressions. For example, $z1 := x*x + y*y$; $z2 := x*x - y*y$; $\sin(z1) + \cos(z2)$

lowerAndUpperBound A table that each row has a constraint for each variable in 3 columns: variable name, lower bound, and upper bound

maxNumGenerations Optional: maximum number of generations, a positive integer. Default: 200

seed Optional: a non-negative integer seed for generating random numbers. 0 is for using timer. Default: 100

Remarks

A minimization problem is to find c in any function $f(c)$ that minimizes the function value.

$$\begin{aligned} c = & \arg \min_c f(c) \\ \text{subject to } & c_{\min}(i) \leq c_i \leq c_{\max}(i), i = 1, 2, \dots, n \end{aligned}$$

where n is the dimension of the vector c .

Examples

optimization.xlsx

Return to the [index](#)

20.6 diff_evol_min_solver

Solves a minimization problem given a function, lower/upper bounds of variables, and data table using differential evolution solver

diff_evol_min_solver (func, lowerAndUpperBound, x, weight, maxNumGenerations, seed)

Returns

A solution to a minimization problem

Parameters

func An expression for a function in one column. Use semicolons to separate sub-expressions. For example, z1 := x*x + y*y; z2 := x*x - y*y; sin(z1) + cos(z2)

lowerAndUpperBound A table that each row has a constraint for each variable in 3 columns: variable name, lower bound, and upper bound

x Input data of variables with headers in the first row

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

maxNumGenerations Optional: maximum number of generations, a positive integer. Default: 200

seed Optional: a non-negative integer seed for generating random numbers. 0 is for using timer. Default: 100

Remarks

A minimization problem is to find c in any function $f(x, c)$ that minimizes the sum of the function values.

$$\begin{aligned} c = & \arg \min_c \sum_{i=1}^m w_i f(x_i, c) \\ \text{subject to } & c_{\min}(i) \leq c_i \leq c_{\max}(i), i = 1, 2, \dots, n \end{aligned}$$

where m is the number of the data observations of x and n is the dimension of the vector c .

Examples

optimization.xlsx

Return to the [index](#)

20.7 diff_evol_nls_solver

Solves a nonlinear least squares problem given a function and lower/upper bounds of variables using differential evolution solver

diff_evol_nls_solver (func, lowerAndUpperBound, x, y, weight, maxNumGenerations, seed)

Returns

A solution to a nonlinear least squares problem

Parameters

func An expression for a function in one column. Use semicolons to separate sub-expressions. For example, z1 := x*x + y*y; z2 := x*x - y*y; sin(z1) + cos(z2)

lowerAndUpperBound A table that each row has a constraint for each variable in 3 columns: variable name, lower bound, and upper bound

x Input data of independent variables with headers in the first row

y Input data of dependent variable with header in the first row

weight Optional: input data of weight variable with header in the first row. Default: 1 for all weights

maxNumGenerations Optional: maximum number of generations, a positive integer. Default: 200

seed Optional: a non-negative integer seed for generating random numbers. 0 is for using timer. Default: 100

Remarks

A nonlinear least squares problem is to find c in a nonlinear function $f(x, c)$ that minimizes the sum of the square error.

$$\begin{aligned} c = & \arg \min_c \sum_{i=1}^m w_i [f(x_i, c) - y_i]^2 \\ \text{subject to } & c_{\min}(i) \leq c_i \leq c_{\max}(i), i = 1, 2, \dots, n \end{aligned}$$

where m is the number of the data observations of x and n is the dimension of the vector c .

Examples

optimization.xlsx

Return to the [index](#)

20.8 transportation_solver

Solves a transportation problem: find the number of units to ship from each source to each destination that minimizes or maximizes the total cost

transportation_solver (sources, destinations, cost, minOrMax)

Returns

An optimal allocation

Parameters

sources The number of units of supply in each source

destinations The number of units demanding in each destination

cost The cost matrix with the labels for rows and columns. The entry is the ship cost for one unit from each source to each destination

minOrMax Optional: the objective to seek. MIN for minimizing or MAX for maximizing the total cost. Default: MIN

Remarks

A balanced transportation problem is to find an optimal allocation of shipments from m sources to n destinations that minimizes or maximizes the total cost.

$$\begin{aligned} \text{minimize } & f = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \\ \text{subject to } & \sum_{j=1}^n x_{ij} = s_i \quad i = 1, 2, \dots, m \\ & \sum_{i=1}^m x_{ij} = d_j \quad j = 1, 2, \dots, n \\ \text{with } & x_{ij} \geq 0, \quad i = 1, \dots, m, j = 1, \dots, n \end{aligned}$$

where c_{ij} ($i = 1, \dots, m$ and $j = 1, \dots, n$) is the cost from the source i to the destination j . The units in each source and each destination are non-negative, $s_i \geq 0$ ($i = 1, 2, \dots, m$) and $d_j \geq 0$ ($j = 1, 2, \dots, n$). The transportation problem is balanced when the total units in all sources ($s = \sum_{i=1}^m s_i$) equals to the total units in all destinations ($d = \sum_{j=1}^n d_j$). For unbalanced transportation problems, when $s > d$, the constraints are

$$\begin{aligned} \sum_{j=1}^n x_{ij} &\leq s_i & i = 1, 2, \dots, m \\ \sum_{i=1}^m x_{ij} &= d_j & j = 1, 2, \dots, n \end{aligned}$$

and when $s < d$, the constraints are

$$\begin{aligned} \sum_{j=1}^n x_{ij} &= s_i & i = 1, 2, \dots, m \\ \sum_{i=1}^m x_{ij} &\leq d_j & j = 1, 2, \dots, n \end{aligned}$$

The unbalanced transportation problem can be reformulated as a balanced problem by introducing a dummy destination (when $s > d$) or a dummy source (when $s < d$). The optional input, minOrMax, controls the problem as a minimization (default) or maximization problem. The transportation problem can be solved by the simplex method.

Examples

optimization.xlsx

Return to the [index](#)

20.9 assignment_solver

Solves an assignment problem: find the optimal assignment that minimizes or maximizes the total cost
assignment_solver (cost, minOrMax)

Returns

An optimal assignment

Parameters

cost The cost matrix with the labels for rows and columns

minOrMax Optional: the objective to seek. MIN for minimizing or MAX for maximizing the total cost. Default: MIN

Remarks

An assignment problem is to find an optimal assignment for assigning m tasks to n people that minimizes or maximizes the total cost. For the case of $m = n$, the assignment problem is

$$\begin{aligned} \text{minimize} \quad & f = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^n x_{ij} = 1 \quad i = 1, 2, \dots, m \\ & \sum_{i=1}^m x_{ij} = 1 \quad j = 1, 2, \dots, n \\ \text{with} \quad & x_{ij} = 0 \text{ or } 1 \quad i = 1, \dots, m, j = 1, \dots, n \end{aligned}$$

where c_{ij} ($i = 1, \dots, m$ and $j = 1, \dots, n$) is the cost for assigning the task i to the person j . When $m > n$, the constraints are

$$\begin{aligned} \sum_{j=1}^n x_{ij} &\leq 1 \quad i = 1, 2, \dots, m \\ \sum_{i=1}^m x_{ij} &= 1 \quad j = 1, 2, \dots, n \end{aligned}$$

and when $m < n$, the constraints are

$$\begin{aligned} \sum_{j=1}^n x_{ij} &= 1 & i = 1, 2, \dots, m \\ \sum_{i=1}^m x_{ij} &\leq 1 & j = 1, 2, \dots, n \end{aligned}$$

The optional input, `minOrMax`, controls the problem as a minimization (default) or maximization problem. An assignment problem is a transportation problem in which the unit of each source and each destination equal to 1. The assignment problem can be solved by the Hungarian method.

Examples

optimization.xlsx

Return to the [index](#)

20.10 netflow_solver

Solves a minimum or maximum cost network flow problem: to find optimal flows that minimize or maximize the total cost

`netflow_solver (arcCapacityAndCost, nodeNetSupplies, minOrMax)`

Returns

Optimal flows

Parameters

arcCapacityAndCost Each arc's capacity constraints and cost in 5 columns: from node, to node, lower bound, upper bound, cost

nodeNetSupplies Optional: each node's net supply (outflow - inflow) in 2 columns: node, net supply. The net supply is 0 for the omitted node. Default: empty

minOrMax Optional: the objective to seek. MIN for minimizing or MAX for maximizing the total cost. Default: MIN

Examples

optimization.xlsx

Return to the [index](#)

20.11 maxflow_solver

Solves a maximum flow problem: to find optimal flows that maximize the total flows from the start node to the end node

`maxflow_solver (arcCapacity, startNode, endNode)`

Returns

Optimal flows

Parameters

arcCapacity Each arc's capacity constraints in 4 columns: from node, to node, lower bound, upper bound

startNode The start node

endNode The end node

Examples

optimization.xlsx

Return to the [index](#)

20.12 shortest_path_solver

Solves the shortest path problem: to find the shortest path from the start node to the end node

shortest_path_solver (arcDistance, startNode, endNode)

Returns

The shortest path from the start node to the end node

Parameters

arcDistance Each arc's distance in 3 columns: from node, to node, distance

startNode The start node

endNode The end node

Examples

optimization.xlsx

Return to the [index](#)

Chapter 21

Portfolio Optimization Functions

efficient_frontier Finds the efficient frontier for portfolios

Black_Litterman Finds posterior expected returns and covariance matrix using the Black-Litterman Model

21.1 efficient_frontier

Finds the efficient frontier for portfolios

`efficient_frontier (assetReturns, assetCov, portfolioMinReturn, portfolioMaxReturn, numberOfPoints, allowNegativeWeights)`

Returns

Portfolio returns, standard deviations, and proportions to invest in each asset

Parameters

assetReturns Input vector asset returns

assetCov Input asset covariance matrix

portfolioMinReturn Optional: Input minimum return of portfolio. Default: min(assetReturns)

portfolioMaxReturn Optional: Input maximum return of portfolio. Default: max(assetReturns)

numberOfPoints Optional: Number of returns of portfolios to look at. Default: 20

allowNegativeWeights Optional: Boolean for whether weights can be negative. Default: TRUE

Remarks

The mean-variance portfolio optimization is to find weights $w_i, i = 1, 2, \dots, n$ for n assets to minimize the portfolio variance, σ_p^2 , for a given return r_p .

$$\begin{aligned} &\text{minimize} && \sigma_p^2 = \sum_{i=1}^n \sum_{j=1}^n w_i w_j \sigma_i \sigma_j \rho_{ij} \\ &\text{subject to} && \sum_{i=1}^n w_i = 1 \\ &&& \sum_{i=1}^n w_i r_i = r_p \end{aligned}$$

where r_i is the return and σ_i is the standard deviation for the i th asset. $\sigma_{ij} = \sigma_i \sigma_j \rho_{ij}$ is the element of the covariance matrix and ρ is the correlation matrix among n assets. Optionally, we can impose the condition to not allow negative weights. The efficient frontier is a set of points in standard deviation/return space where one can achieve the least risk with a given return.

Examples

portfolio_optimization.xlsx

See also

[matrix_cov_from_corr](#)

Return to the [index](#)

21.2 Black_Litterman

Finds posterior expected returns and covariance matrix using the Black-Litterman Model

Black_Litterman (*assetReturns*, *assetCov*, *viewAssetWeights*, *viewReturns*, *viewCov*, *viewConfidenceLevels*, *riskAversionCoef*, *tau*)

Returns

Posterior expected returns and covariance matrix incorporating investor's views

Parameters

assetReturns A vector of the prior returns of the assets

assetCov A covariance matrix of the assets

viewAssetWeights A matrix of asset weights within each view; Rows: specific views; Columns: assets

viewReturns A vector of expected returns for each view

viewCov Optional: direct input of diagonal covariance matrix of the views; if missing, it will be calculated from each view's weights and confidence level. Default: missing

viewConfidenceLevels Optional: a vector of confidence levels for each view between 0 and 1 inclusive. Default: 0.5

riskAversionCoef Optional: a risk aversion coefficient. Default: 2.5

tau Optional: a number indicating the uncertainty of the CAPM distribution between 0 and 1 inclusive. Default: 0.025

Remarks

In the Black-Litterman model with N assets and K views, the posterior expected return, a $N \times 1$ vector, is given by,

$$E(R) = [(\tau\Sigma)^{-1} + P^T\Omega^{-1}P]^{-1}[(\tau\Sigma)^{-1}\Pi + P^T\Omega^{-1}Q]$$

and the posterior covariance matrix, a $N \times N$ matrix, is given by,

$$\Sigma_p = \Sigma + [(\tau\Sigma)^{-1} + P^T\Omega^{-1}P]^{-1}$$

where the variables are defined as

- Π : $N \times 1$ vector of implied/prior returns of the assets.
- Σ : $N \times N$ prior covariance matrix of the assets.
- τ : number representing uncertainty of the CAPM distribution.

- $P : K \times N$ matrix of the asset weights within each view. Each row corresponds to a view. Each column corresponds to an asset.
- $Q : K \times 1$ vector of the expected returns for each view.
- $\Omega : K \times K$ diagonal covariance matrix of the views with entries of the uncertainty within each view.

If the view's covariance matrix (viewCov) is not input, it is calculated from each view's weights and confidence level,

$$\Omega_i = \alpha_i P_i^T (\tau \Sigma) P_i, \quad \alpha = (1 - \text{confidence level}) / \text{confidence level}$$

where P_i is the i th row of the matrix P representing the i th view's weights. The prior optimal portfolio weights in the absence of constraints are calculated by,

$$w = (\delta \Sigma)^{-1} \Pi$$

and the posterior optimal portfolio weights in the absence of constraints are calculated by,

$$w_p = (\delta \Sigma_p)^{-1} E(R)$$

where δ is the risk aversion coefficient.

Examples

portfolio_optimizer.xlsx

Return to the [index](#)

Chapter 22

Control Theory Functions

pole_placement Calculates the gains K for the pole placement

22.1 pole_placement

Calculates the gains K for the pole placement

pole_placement (A , B , poles)

Returns

A vector for the gains K

Parameters

A Input matrix A

B Input matrix B

poles Desired poles. The first column is for the real parts and the second column (optional) is for the imaginary parts of the poles

Remarks

Let $p(s)$ be a monic polynomial of degree n given by the desired poles *poles*. For a linear system

$$\dot{X} = AX + Bu$$

if it is controllable, then there is a unique state feedback law by a gain vector K

$$u = -Kx$$

such that the characteristic polynomial of $A - BK$ is $p(s)$, namely

$$p(s) = \det(sI - (A - BK))$$

where the dimensions of the matrices are $A = [n \times n]$, $X = [n \times 1]$, $B = [n \times 1]$, $u = [1 \times 1]$, and $K = [1 \times n]$.

Examples

Control_theory.xlsx

Return to the [index](#)

Chapter 23

Matrix Operation Functions

matrix_random Generates a random matrix from a uniform distribution $U(0, 1)$ or a standard normal distribution $N(0, 1)$

matrix_cov Computes the covariance matrix given a data table

matrix_cov_from_file Computes the covariance matrix given a data file

matrix_corr Computes the correlation matrix given a data table

matrix_corr_from_file Computes the correlation matrix given a data file

matrix_corr_from_cov Computes the correlation matrix from a covariance matrix

matrix_cov_from_corr Computes the covariance matrix from a correlation matrix and a stdev vector

matrix_stdev_from_cov Computes the standard deviation vector from a covariance matrix

matrix_prod Computes the product of two matrices, one matrix could be a number

matrix_directprod Computes the direct product of two matrices

matrix_elementprod Computes the elementwise product of two matrices

matrix_plus Adds two matrices with the same dimension: $\text{matrix1} + \text{matrix2}$

matrix_minus Subtracts two matrices with the same dimension: $\text{matrix1} - \text{matrix2}$

matrix_I Creates an identity matrix

matrix_t Returns the transpose matrix of a matrix

matrix_diag Creates a diagonal matrix from a matrix or a vector

matrix_tr Returns the trace of a matrix

matrix_inv Computes the inverse of a square matrix

matrix_pinv Computes the pseudoinverse of a real matrix

matrix_complex_pinv Computes the pseudoinverse of a complex matrix

matrix_solver Solves a system of linear equations $Ax = B$

matrix_tridiagonal_solver Solves a system of tridiagonal linear equations $Ax = B$

- matrix_pentadiagonal_solver** Solves a system of pentadiagonal linear equations $Ax = B$
- matrix_Sylvester_solver** Solves a Sylvester equation $Ax + xB = C$
- matrix_chol** Computes the Cholesky decomposition of a symmetric positive semi-definite matrix
- matrix_sym_eigen** Computes the eigenvalue-eigenvector pairs of a symmetric real matrix
- matrix_eigen** Computes the eigenvalue-eigenvector pairs of a square real matrix
- matrix_complex_eigen** Computes the eigenvalue-eigenvector pairs of a square complex matrix
- matrix_svd** Computes the singular value decomposition (SVD) of a matrix
- matrix_LU** Computes the LU decomposition of a square matrix
- matrix_QR** Computes the QR decomposition of a square real matrix
- matrix_complex_QR** Computes the QR decomposition of a square complex matrix
- matrix_Schur** Computes the Schur decomposition a square real matrix
- matrix_complex_Schur** Computes the Schur decomposition a square complex matrix
- matrix_sweep** Sweeps a matrix given indexes
- matrix_det** Computes the determinant of a square matrix
- matrix_exp** Computes the matrix exponential of a square matrix
- matrix_complex_exp** Computes the matrix exponential of a square complex matrix
- matrix_distance** Computes the distance matrix given a data table
- matrix_freq** Creates a frequency table given a string matrix
- matrix_from_vector** Converts a matrix from a vector
- matrix_to_vector** Converts a matrix into a column vector
- matrix_decimal_to_fraction** Converts each decimal to a fraction for each element of a matrix if possible

23.1 matrix_random

Generates a random matrix from a uniform distribution $U(0, 1)$, a standard normal distribution $N(0, 1)$, or a discrete uniform distribution

`matrix_random (nrows, ncols, dist, corr, lower, upper, seed)`

Returns

A random matrix

Parameters

nrows The number of rows

ncols The number of columns

dist Optional: the distribution name, UNIFORM, NORMAL (GAUSSIAN), or DISCRETE_UNIFORM. Default: UNIFORM

corr Optional: correlation matrix. Default: identity matrix

lower Optional: lower boundary of a discrete uniform distribution. Default: 0

upper Optional: upper boundary of a discrete uniform distribution. Default: 1

seed Optional: non-negative integer seed for generating random numbers. Default: 0 (use timer)

Examples

matrix_operations.xlsx

Return to the [index](#)

23.2 matrix_cov

Computes the covariance matrix given a data table

matrix_cov (inputData)

Returns

The covariance matrix

Parameters

inputData Input data with or without headers

Examples

matrix_operations.xlsx

Return to the [index](#)

23.3 matrix_cov_from_file

Computes the covariance matrix given a data file

matrix_cov_from_file (filename, varNames, delimiter)

Returns

A covariance matrix

Parameters

filename Input data file name. The first line of the file is the header line with variable names

varNames Variable names in one row or one column

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

matrix_operations.xlsx

Return to the [index](#)

23.4 matrix_corr

Computes the correlation matrix given a data table

`matrix_corr (inputData)`

Returns

A correlation matrix

Parameters

inputData Input data with or without headers

Examples

matrix_operations.xlsx

Return to the [index](#)

23.5 matrix_corr_from_file

Computes the correlation matrix given a data file

`matrix_corr_from_file (filename, varNames, delimiter)`

Returns

A correlation matrix

Parameters

filename Input data file name. The first line of the file is the header line with variable names

varNames Variable names in one row or one column

delimiter Optional: one character delimiter. 't' for a tab and 's' for a space. If the input is a string, its first character is used. Default: comma for comma-separated-value (.csv) file

Examples

matrix_operations.xlsx

Return to the [index](#)

23.6 matrix_corr_from_cov

Computes the correlation matrix from a covariance matrix

`matrix_corr_from_cov (matrix)`

Returns

A correlation matrix from a covariance matrix

Parameters

matrix Input covariance matrix

Examples

matrix_operations.xlsx

Return to the [index](#)

23.7 matrix_cov_from_corr

Computes the covariance matrix from a correlation matrix and a stdev vector

`matrix_cov_from_corr (matrix, stdevs)`

Returns

A covariance matrix from a correlation matrix

Parameters

matrix Input correlation matrix

stdevs Input standard deviation vector

Examples

matrix_operations.xlsx

Return to the [index](#)

23.8 matrix_stdev_from_cov

Computes the standard deviation vector from a covariance matrix

`matrix_stdev_from_cov (matrix)`

Returns

A standard deviation vector from a covariance matrix

Parameters

matrix Input covariance matrix

Examples

matrix_operations.xlsx

Return to the [index](#)

23.9 matrix_prod

Computes the product of two matrices, one matrix could be a number

matrix_prod (matrix1, matrix2)

Returns

The product of two input matrices

Parameters

matrix1 Input matrix1

matrix2 Input matrix2

Examples

matrix_operations.xlsx

Return to the [index](#)

23.10 matrix_directprod

Computes the direct product of two matrices

matrix_directprod (matrix1, matrix2)

Returns

The direct product of two input matrices

Parameters

matrix1 Input matrix1

matrix2 Input matrix2

Examples

matrix_operations.xlsx

Return to the [index](#)

23.11 matrix_elementprod

Computes the elementwise product of two matrices, one matrix could be a number

`matrix_elementprod (matrix1, matrix2)`

Returns

The elementwise product of two input matrices

Parameters

matrix1 Input matrix1

matrix2 Input matrix2

Examples

matrix_operations.xlsx

Return to the [index](#)

23.12 matrix_plus

Adds two matrices with the same dimension: `matrix1 + matrix2`

`matrix_plus (matrix1, matrix2)`

Returns

The addition of two input matrices

Parameters

matrix1 Input matrix1

matrix2 Input matrix2

Examples

matrix_operations.xlsx

Return to the [index](#)

23.13 **matrix_minus**

Subtracts two matrices with the same dimension: matrix1 - matrix2

`matrix_minus (matrix1, matrix2)`

Returns

The subtraction of two input matrices

Parameters

matrix1 Input matrix1

matrix2 Input matrix2

Examples

matrix_operations.xlsx

Return to the [index](#)

23.14 **matrix_I**

Creates an identity matrix

`matrix_I (dim)`

Returns

An identity matrix

Parameters

dim The dimension of an identity matrix

Examples

matrix_operations.xlsx

Return to the [index](#)

23.15 **matrix_t**

Returns the transpose matrix of a matrix

`matrix_t (matrix)`

Returns

The transpose matrix of an input matrix

Parameters

matrix Input matrix

Examples

matrix_operations.xlsx

Return to the [index](#)

23.16 matrix_diag

Creates a diagonal matrix from a matrix or a vector

matrix_diag (matrix)

Returns

The diagonal matrix of an input matrix or an input vector

Parameters

matrix Input matrix

Examples

matrix_operations.xlsx

Return to the [index](#)

23.17 matrix_tr

Returns the trace of a matrix

matrix_tr (matrix)

Returns

The trace (sum of diagonal elements) of an input matrix

Parameters

matrix Input matrix

Examples

matrix_operations.xlsx

Return to the [index](#)

23.18 matrix_inv

Computes the inverse of a square matrix

`matrix_inv (matrix)`

Returns

The inverse of a square matrix

Parameters

matrix Input square matrix

Remarks

For a square matrix A , its inverse matrix is A^{-1} such that

$$AA^{-1} = A^{-1}A = I$$

where I is an identity matrix.

Examples

matrix_operations.xlsx

Return to the [index](#)

23.19 matrix_pinv

Computes the pseudoinverse of a real matrix

`matrix_pinv (matrix)`

Returns

The pseudoinverse of a real matrix

Parameters

matrix Input matrix

Remarks

For a matrix A (not necessary a square matrix), its pseudo-inverse matrix is A^+ such that it satisfies the following four properties:

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. AA^+ is symmetric
4. A^+A is symmetric

Examples

matrix_operations.xlsx

Return to the [index](#)

23.20 matrix_complex_pinv

Computes the pseudoinverse of a complex matrix

matrix_complex_pinv (matrixReal, matrixImag)

Returns

The pseudoinverse of a complex matrix

Parameters

matrixReal Real part of an input complex matrix

matrixImag Imaginary part of an input complex matrix

Remarks

For a matrix A (not necessary a square matrix), its pseudo-inverse matrix is A^+ such that it satisfies the following four properties:

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. AA^+ is symmetric
4. A^+A is symmetric

Examples

matrix_operations.xlsx

Return to the [index](#)

23.21 matrix_solver

Solves a system of linear equations $Ax = B$

matrix_solver (A, B)

Returns

The solution of a system of linear equations

Parameters

A Input matrix A

B Input matrix B

Remarks

For an $m \times n$ matrix A (not necessary a square matrix), its singular value decomposition (SVD) is

$$A = UWV^T$$

where the dimensions of the matrices are $U = [m \times n]$, $W = [n \times n]$, and $V = [n \times n]$. Let A^+ be the pseudo-inverse matrix of A ,

$$A^+ = VW^{-1}U^T$$

then the solution of the system of linear equations is

$$x = A^+B$$

Examples

matrix_operations.xlsx

Return to the [index](#)

23.22 matrix_tridiagonal_solver

Solves a system of tridiagonal linear equations $Ax = B$

matrix_tridiagonal_solver (A, B)

Returns

The solution of a system of tridiagonal linear equations

Parameters

A Input tridiagonal matrix A . Lower diagonal, diagonal, and upper diagonal elements are in columns in the order of ADC : A is the lower diagonal, D is the main diagonal, and C is the upper diagonal

B Input matrix B

Examples

matrix_operations.xlsx

Return to the [index](#)

23.23 matrix_pentadiagonal_solver

Solves a system of pentadiagonal linear equations $Ax = B$

matrix_pentadiagonal_solver (A, B)

Returns

The solution of a system of pentadiagonal linear equations

Parameters

A Input pentadiagonal matrix A . Lower diagonal, diagonal, and upper diagonal elements are in columns in the order of $EADCF$: E and A are the lower diagonals, D is the main diagonal, and C and F are the upper diagonals

B Input matrix B

Examples

matrix_operations.xlsx

Return to the [index](#)

23.24 matrix_Sylvester_solver

Solves a Sylvester equation $AX + XB = C$

matrix_Sylvester_solver (A, B, C)

Returns

The solution of a Sylvester equation

Parameters

A Input matrix A

B Input matrix B

C Input matrix C

Remarks

For a Sylvester equation

$$AX + XB = C$$

where the dimensions of the matrices are $A = [m \times m]$, $B = [n \times n]$, $C = [m \times n]$, and $X = [m \times n]$. It can be solved by

$$(I_n \otimes A + B^T \otimes I_m) Vect(X) = Vect(C)$$

where I is the identity matrix, \otimes is the Kronecker product, and $Vect(X)$ is a column vector obtained by stacking the columns of the matrix X on top of one another.

Examples

matrix_operations.xlsx

Return to the [index](#)

23.25 matrix_chol

Computes the Cholesky decomposition of a symmetric positive semi-definite matrix

`matrix_chol (matrix)`

Returns

The Cholesky decomposition

Parameters

matrix Input symmetric positive semi-definite matrix

Remarks

For a symmetric positive semi-definite matrix A , its Cholesky decomposition is

$$A = UU^T$$

where U is lower triangular.

Examples

matrix_operations.xlsx

Return to the [index](#)

23.26 matrix_sym_eigen

Computes the eigenvalue-eigenvector pairs of a symmetric real matrix

`matrix_sym_eigen (matrix)`

Returns

The eigenvalue-eigenvector pairs

Parameters

matrix Input symmetric real matrix

Remarks

For a symmetric matrix A , let $p_i (i = 1, 2, \dots, n)$ be an eigenvector with an eigenvalue $\lambda_i (i = 1, 2, \dots, n)$, $Ap_i = \lambda_i p_i (i = 1, 2, \dots, n)$. Define a matrix $U = [p_1, p_2, \dots, p_n]$, whose columns are the eigenvectors, and a diagonal matrix composed of the eigenvalues, $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$.

$$A = U\Lambda U^T$$

Examples

matrix_operations.xlsx

Return to the [index](#)

23.27 matrix_eigen

Computes the eigenvalue-eigenvector pairs of a square real matrix

`matrix_eigen (matrix)`

Returns

The eigenvalue-eigenvector pairs

Parameters

matrix Input square real matrix

Remarks

For a square matrix A , let $p_i (i = 1, 2, \dots, n)$ be an eigenvector with an eigenvalue $\lambda_i (i = 1, 2, \dots, n)$, $Ap_i = \lambda_i p_i (i = 1, 2, \dots, n)$.

If A is symmetric, all eigenvalues are real. If A is not symmetric, then eigenvectors can be complex in general. If the i th eigenvalue is real, then column i of eigenvectors contains the corresponding real eigenvector. If the i th and $(i + 1)$ th eigenvalues are complex-conjugate pair of eigenvalues, $Re(\lambda) \pm iIm(\lambda)$, then columns i and $i + 1$ of eigenvectors contain the real, u , and imaginary, v , parts, respectively, of the two corresponding eigenvectors $u \pm iv$.

Examples

`matrix_operations.xlsx`

Return to the [index](#)

23.28 matrix_complex_eigen

Computes the eigenvalue-eigenvector pairs of a square complex matrix.

`matrix_complex_eigen (matrixReal, matrixImag)`

Returns

The eigenvalue-eigenvector pairs

Parameters

matrixReal Real part of an input complex matrix

matrixImag Imaginary part of an input complex matrix

Remarks

For a square matrix A , let $p_i (i = 1, 2, \dots, n)$ be an eigenvector with an eigenvalue $\lambda_i (i = 1, 2, \dots, n)$, $Ap_i = \lambda_i p_i (i = 1, 2, \dots, n)$.

If A is a Hermitian matrix, all eigenvalues are real. If A is not a Hermitian matrix, then eigenvectors can be complex in general.

Examples

matrix_operations.xlsx

Return to the [index](#)

23.29 matrix_svd

Computes the singular value decomposition (SVD) of a matrix

matrix_svd (matrix)

Returns

The singular value decomposition (SVD) of a matrix

Parameters

matrix Input matrix

Remarks

For any matrix $A = [m \times n]$, it can be decomposed in terms of three matrices

$$A = UWV^T$$

where the dimensions of the matrices are $U = [m \times n]$, $W = [n \times n]$, and $V = [n \times n]$.

Examples

matrix_operations.xlsx

Return to the [index](#)

23.30 matrix_LU

Computes the LU decomposition of a square matrix

matrix_LU (matrix)

Returns

The LU decomposition of a square matrix

Parameters

matrix Input matrix

Remarks

For any square matrix A , its rowwise permutation PA can be decomposed in terms of a lower triangular matrix, L , and an upper triangular matrix, U

$$PA = LU$$

Examples

matrix_operations.xlsx

Return to the [index](#)

23.31 matrix_QR

Computes the QR decomposition of a square real matrix

matrix_QR (matrix)

Returns

The QR decomposition of a square real matrix

Parameters

matrix Input square real matrix

Remarks

For any square real matrix A , it can be decomposed in terms of an orthogonal matrix, Q , and an upper triangular matrix, R

$$A = QR$$

Examples

matrix_operations.xlsx

Return to the [index](#)

23.32 matrix_complex_QR

Computes the QR decomposition of a square complex matrix

matrix_complex_QR (matrixReal, matrixImag)

Returns

The QR decomposition of a square complex matrix

Parameters

matrixReal Real part of an input complex matrix

matrixImag Imaginary part of an input complex matrix

Remarks

For any square complex matrix A , it can be decomposed in terms of a unitary matrix, Q , and an upper triangular matrix, R

$$A = QR$$

Examples

matrix_operations.xlsx

Return to the [index](#)

23.33 matrix_Schur

Computes the Schur decomposition of a square real matrix

matrix_Schur (matrix)

Returns

The Schur decomposition of a square real matrix

Parameters

matrix Input square real matrix

Remarks

For any square real matrix A , it can be decomposed in terms of a unitary matrix, Q , and an upper triangular matrix, R

$$A = QRQ^H$$

where Q^H is the Hermitian conjugate of Q .

Examples

matrix_operations.xlsx

Return to the [index](#)

23.34 matrix_complex_Schur

Computes the Schur decomposition of a square complex matrix

matrix_complex_Schur (matrixReal, matrixImag)

Returns

The Schur decomposition of a square complex matrix

Parameters

matrixReal Real part of an input complex matrix

matrixImag Imaginary part of an input complex matrix

Remarks

For any square matrix A , it can be decomposed in terms of a unitary matrix, Q , and a upper triangular matrix, R

$$A = QRQ^H$$

where Q^H is the Hermitian conjugate of Q .

Examples

matrix_operations.xlsx

Return to the [index](#)

23.35 matrix_sweep

Sweeps a matrix given indexes

matrix_sweep (matrix, pivotIndexes)

Returns

The swept matrix

Parameters

matrix Input matrix

pivotIndexes Optional: pivot indexes for sweep. Default: all possible indexes for a given matrix

Remarks

Let $A = [m \times n]$ be a general matrix (not necessarily square) with a partition denoted as

$$A = \begin{bmatrix} R & S \\ T & U \end{bmatrix}$$

where R is a square matrix. The sweep of matrix of A with respect to R is

$$\text{sweep}(A, R) = \begin{bmatrix} R^{-1} & R^{-1}S \\ -TR^{-1} & U - TR^{-1}S \end{bmatrix}$$

Examples

matrix_operations.xlsx

Return to the [index](#)

23.36 matrix_det

Computes the determinant of a square matrix

`matrix_det (matrix)`

Returns

The determinant of a square matrix

Parameters

matrix Input square matrix

Examples

matrix_operations.xlsx

Return to the [index](#)

23.37 matrix_exp

Computes the matrix exponential of a square matrix

`matrix_exp (matrix)`

Returns

The matrix exponential of a square matrix

Parameters

matrix Input square matrix

Remarks

Let A be a square matrix, its exponential is

$$e^A = \sum_{n=0}^{\infty} \frac{1}{n!} A^n = I + A + \frac{1}{2!} A^2 + \dots$$

Examples

matrix_operations.xlsx

Return to the [index](#)

23.38 matrix_complex_exp

Computes the matrix exponential of a square complex matrix

`matrix_complex_exp (matrixReal, matrixImag)`

Returns

The matrix exponential of a square complex matrix

Parameters

matrixReal Optional: Real part of an input complex square matrix. Default: zero matrix

matrixImag Optional: Imaginary part of an input complex square matrix. Default: zero matrix

Remarks

Let A be a square matrix, its exponential is

$$e^A = \sum_{n=0}^{\infty} \frac{1}{n!} A^n = I + A + \frac{1}{2!} A^2 + \dots$$

Examples

`matrix_operations.xlsx`

Return to the [index](#)

23.39 matrix_distance

Computes the distance matrix given a data table

`matrix_distance (inputData, p)`

Returns

The distance matrix

Parameters

inputData Input data with or without headers

p Optional: The power of the Minkowski distance or the name of distance ("Euclidean", "Manhattan", or "Chebyshev"). Default: $p = 2$ for Euclidean distance

Remarks

In an n -dimensional space, the distance between two points, x and y , is defined as p -norm:

$$d(x, y) = \left[\sum_{i=1}^n |x_i - y_i|^p \right]^{1/p}$$

where $p \geq 1$. The three special cases are

- $p = 1 : d(x, y) = \sum_{i=1}^n |x_i - y_i|$. Manhattan distance
- $p = 2 : d(x, y) = \left[\sum_{i=1}^n |x_i - y_i|^2 \right]^{1/2}$. Euclidean distance
- $p = \text{infinity} : d(x, y) = \max\{|x_i - y_i|\}$. Chebyshev distance

Examples

matrix_operations.xlsx

Return to the [index](#)

23.40 matrix_freq

Creates a frequency table given a matrix

matrix_freq (matrix, includeMissing)

Returns

A frequency table from a matrix

Parameters

matrix Input matrix. The elements could be strings, numbers, or missing. The missing values are counted or not depending on the input includeMissing

includeMissing Optional: binary flag 0 or 1. When the flag is 1 (0), the missings are included (not included) in frequency table. Default: 0

Examples

matrix_operations.xlsx

Return to the [index](#)

23.41 matrix_from_vector

Converts a matrix from a vector

matrix_from_vector (vector, numRows, numCols, byRowOrCol)

Returns

A matrix

Parameters

vector Input vector

numRows Optional: Number of rows. It can be omitted if byRowOrCol is ROW. Default: fill out the maximum number of rows if missing

numCols Optional: Number of columns. It can be omitted if byRowOrCol is COL. Default: fill out the maximum number of columns if missing

byRowOrCol Optional: COL for outputting by column, ROW for outputting by row. Default: COL

Examples

matrix_operations.xlsx

Return to the [index](#)

23.42 matrix_to_vector

Converts a matrix into a column vector

matrix_to_vector (matrix, byRowOrCol)

Returns

A column vector

Parameters

matrix Input matrix

byRowOrCol Optional: COL for inputting by column, ROW for inputting by row. Default: COL

Examples

matrix_operations.xlsx

Return to the [index](#)

23.43 matrix_decimal_to_fraction

Converts each decimal to a fraction for each element of a matrix if possible

matrix_decimal_to_fraction (matrix)

Returns

A matrix with fractions

Parameters

matrix Input matrix

Examples

matrix_operations.xlsx

Return to the [index](#)

Chapter 24

Fast Fourier Transform Functions

FFT Performs fast Fourier transform

IFFT Performs inverse fast Fourier transform

24.1 FFT

Performs fast Fourier transform

FFT (xReal, xImag)

Returns

The discrete Fourier transform

Parameters

xReal Real part of input data: 1-D vector or 2-D matrix

xImag Optional: Imaginary part of input data: 1-D vector or 2-D matrix. Default: missing

Remarks

When xImag is given, it must be the same size as xReal. When xReal and optional xImag are in one row or on one column, the discrete 1-D Fourier transform is

$$F_j = \sum_{k=0}^{N-1} f_k e^{-2\pi i \frac{jk}{N}}, j = 0, 1, \dots, N-1$$

The inverse discrete 1-D Fourier transform is

$$f_k = \frac{1}{N} \sum_{j=0}^{N-1} F_j e^{2\pi i \frac{jk}{N}}, k = 0, 1, \dots, N-1$$

When xReal and optional xImag are 2-D matrix with N_1 rows and N_2 columns, the discrete 2-D Fourier transform is

$$F_{j_1, j_2} = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} f_{k_1, k_2} e^{-2\pi i \left(\frac{j_1 k_1}{N_1} + \frac{j_2 k_2}{N_2} \right)}, j_1 = 0, 1, \dots, N_1-1, j_2 = 0, 1, \dots, N_2-1$$

The inverse discrete 2-D Fourier transform is

$$f_{k_1, k_2} = \frac{1}{N_1 N_2} \sum_{j_1=0}^{N_1-1} \sum_{j_2=0}^{N_2-1} F_{j_1, j_2} e^{2\pi i \left(\frac{j_1 k_1}{N_1} + \frac{j_2 k_2}{N_2} \right)}, k_1 = 0, 1, \dots, N_1 - 1, k_2 = 0, 1, \dots, N_2 - 1$$

Examples

fast_Fourier_transform.xlsx

See also

[IFFT](#)

Return to the [index](#)

24.2 IFFT

Performs inverse fast Fourier transform

IFFT (xReal, xImag)

Returns

The discrete inverse Fourier transform

Parameters

xReal Real part of input data: 1-D vector or 2-D matrix

xImag Optional: Imaginary part of input data: 1-D vector or 2-D matrix. Default: missing

Remarks

When xImag is given, it must be the same size as xReal. When xReal and optional xImag are in one row or on one column, the discrete 1-D Fourier transform is

$$F_j = \sum_{k=0}^{N-1} f_k e^{-2\pi i \frac{jk}{N}}, j = 0, 1, \dots, N - 1$$

The inverse discrete 1-D Fourier transform is

$$f_k = \frac{1}{N} \sum_{j=0}^{N-1} F_j e^{2\pi i \frac{jk}{N}}, k = 0, 1, \dots, N - 1$$

When xReal and optional xImag are 2-D matrix with N_1 rows and N_2 columns, the discrete 2-D Fourier transform is

$$F_{j_1, j_2} = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} f_{k_1, k_2} e^{-2\pi i \left(\frac{j_1 k_1}{N_1} + \frac{j_2 k_2}{N_2} \right)}, j_1 = 0, 1, \dots, N_1 - 1, j_2 = 0, 1, \dots, N_2 - 1$$

The inverse discrete 2-D Fourier transform is

$$f_{k_1, k_2} = \frac{1}{N_1 N_2} \sum_{j_1=0}^{N_1-1} \sum_{j_2=0}^{N_2-1} F_{j_1, j_2} e^{2\pi i \left(\frac{j_1 k_1}{N_1} + \frac{j_2 k_2}{N_2} \right)}, k_1 = 0, 1, \dots, N_1 - 1, k_2 = 0, 1, \dots, N_2 - 1$$

Examples

fast_Fourier_transform.xlsx

See also

[FFT](#)

Return to the [index](#)

Chapter 25

Numerical Integration Functions

gauss_legendre Generates the abscissas and weights of the Gauss-Legendre n-point quadrature formula

gauss_laguerre Generates the abscissas and weights of the Gauss-Laguerre n-point quadrature formula

gauss_hermite Generates the abscissas and weights of the Gauss-Hermite n-point quadrature formula

integral Evaluates an 1-D integration of a function given lower and upper bounds

function_eval Evaluates a function given arguments

prime_numbers Gets prime numbers

Halton_numbers Gets Halton numbers

Sobol_numbers Gets Sobol numbers

Latin_hypercube Gets Latin hypercube sampling

25.1 gauss_legendre

Generates the abscissas and weights of the Gauss-Legendre n-point quadrature formula

`gauss_legendre (numPoints, lower, upper)`

Returns

The abscissas and weights of the Gauss-Legendre n-point quadrature formula

Parameters

numPoints The number of points

lower Lower boundary

upper Upper boundary

Remarks

The Gauss-Legendre n -point quadrature formula is

$$\int_a^b f(x)dx \approx \sum_{i=1}^n w_i f(x_i)$$

where a is the lower boundary and b is the upper boundary of integration. x_i and w_i ($i = 1, 2, \dots, n$) are the abscissas and weights, respectively.

Examples

numerical_integration.xlsx

Return to the [index](#)

25.2 gauss_laguerre

Generates the abscissas and weights of the Gauss-Laguerre n-point quadrature formula

gauss_laguerre (numPoints)

Returns

The abscissas and weights of the Gauss-Laguerre n-point quadrature formula

Parameters

numPoints The number of points

Remarks

The Gauss-Laguerre n -point quadrature formula is

$$\int_0^{\infty} e^{-x} f(x) dx \approx \sum_{i=1}^n w_i f(x_i)$$

where x_i and w_i ($i = 1, 2, \dots, n$) are the abscissas and weights, respectively. The Gauss-Laguerre quadrature is suitable to evaluating the integral when

$$\lim_{x \rightarrow \infty} e^{-x} f(x) = 0$$

Examples

numerical_integration.xlsx

Return to the [index](#)

25.3 gauss_hermite

Generates the abscissas and weights of the Gauss-Hermite n-point quadrature formula

gauss_hermite (numPoints)

Returns

The abscissas and weights of the Gauss-Hermite n -point quadrature formula

Parameters

numPoints The number of points

Remarks

The Gauss-Hermite n -point quadrature formula is

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} f(x) dx \approx \sum_{i=1}^n w_i f(x_i)$$

where x_i and w_i ($i = 1, 2, \dots, n$) are the abscissas and weights, respectively. The Gauss-Hermite quadrature is suitable to evaluating the integral when

$$\lim_{x \rightarrow \pm\infty} e^{-x^2/2} f(x) = 0$$

Examples

numerical_integration.xlsx

Return to the [index](#)

25.4 integral

Evaluates an 1-D integration of a function given lower and upper bounds

integral (func, from, to)

Returns

Integration of an 1-D function

Parameters

func An expression for an 1-D function in one column

from Lower bound of the integration range

to Upper bound of the integration range

Examples

numerical_integration.xlsx

Return to the [index](#)

25.5 function_eval

Evaluates a function given arguments

`function_eval (func, x)`

Returns

The function values

Parameters

func An expression for a function in one column. Use semicolons to separate sub-expressions. For example, `z1 := x*x + y*y; z2 := x*x - y*y; sin(z1) + cos(z2)`

x A table of the arguments of a function with variable names in the first row

Examples

numerical_integration.xlsx

Return to the [index](#)

25.6 prime_numbers

Gets prime numbers

`prime_numbers (numPrimeNumbers)`

Returns

Prime numbers

Parameters

numPrimeNumbers The number of prime numbers

Examples

numerical_integration.xlsx

Return to the [index](#)

25.7 Halton_numbers

Gets Halton numbers

`Halton_numbers (numbers, dimension)`

Returns

Halton numbers

Parameters

numbers The number of Halton numbers

dimension Optional: The dimension of Halton numbers. Default: 1

Examples

numerical_integration.xlsx

Return to the [index](#)

25.8 Sobol_numbers

Gets Sobol numbers

Sobol_numbers (numbers, dimension)

Returns

Sobol numbers

Parameters

numbers The number of Sobol numbers

dimension Optional: The dimension of Sobol numbers. The maximum dimension is 1111. Default: 1

Examples

numerical_integration.xlsx

Return to the [index](#)

25.9 Latin_hypercube

Gets Latin hypercube sampling

Latin_hypercube (numbers, dimension, seed, symbol)

Returns

Latin hypercube sampling

Parameters

numbers The number of partitions

dimension The number of dimensions

seed Optional: a non-negative integer seed for generating random numbers. 0 is for using timer.
Default: 100

symbol Optional: a symbol printed in Lation squares. Default: X

Examples

numerical_integration.xlsx

Return to the [index](#)

Chapter 26

Probability Functions

prob_normal Computes the cumulative probability given z for the standard normal distribution: $N(z) = \text{Prob}(Z < z)$

prob_normal_inv Computes the percentile of a standard normal distribution: $\text{Prob}(Z < z) = p$

prob_normal_table Generates a table of the cumulative probabilities for the standard normal distribution: $N(z) = \text{Prob}(Z < z)$

prob_t Computes the cumulative probability given t and the degree of freedom for the Student's t distribution: $\text{Prob}(t_n < t)$

prob_t_inv Computes the percentile for the Student's t distribution: $\text{Prob}(t_n < t) = p$

prob_t_table Generates a table of the percentiles given a set of degrees of freedom and a set of probabilities for the Student's t distribution: $\text{Prob}(t_n < t) = P$

prob_chi Computes the cumulative probability given c and the degree of freedom for the Student's distribution: $\text{Prob}(X^2 < c)$

prob_chi_inv Computes the percentile for the Chi-Squared distribution: $\text{Prob}(X^2 < c) = p$

prob_chi_table Generates a table of the percentiles given a set of degrees of freedom and a set of probabilities for the Chi-Squared distribution: $\text{Prob}(X^2 < c) = P$

prob_f Computes the cumulative probability given f and the degree of freedom for the F-distribution: $\text{Prob}(F(df1, df2) < f)$

prob_f_inv Computes the percentile for the F-distribution: $\text{Prob}(F(df1, df2) < f) = p$

prob_f_table Generates a table of the percentiles given a set of degrees of freedom and a probability for the F-distribution: $\text{Prob}(F(df1, df2) < f) = p$

Cornish_Fisher_expansion Computes the percentile of a distribution with a skewness and an excess kurtosis by Cornish-Fisher expansion

26.1 prob_normal

Computes the cumulative probability given z for the standard normal distribution: $N(z) = \text{Prob}(Z < z)$

`prob_normal (z)`

Returns

The cumulative probability

Parameters

z Input value

Examples

probability_distribution.xlsx

Return to the [index](#)

26.2 prob_normal_inv

Computes the percentile of a standard normal distribution: $\text{Prob}(Z < z) = p$

prob_normal_inv (p)

Returns

The percentile

Parameters

p Probability

Examples

probability_distribution.xlsx

Return to the [index](#)

26.3 prob_normal_table

Generates a table of the cumulative probabilities for the standard normal distribution: $N(z) = \text{Prob}(Z < z)$

prob_normal_table ()

Returns

A table of the cumulative probabilities

Examples

probability_distribution.xlsx

Return to the [index](#)

26.4 prob_t

Computes the cumulative probability given t and the degree of freedom for the Student's t distribution:
 $\text{Prob}(t_n < t)$

`prob_t (t, df)`

Returns

The cumulative probability

Parameters

t Input value

df Degree of freedom

Examples

probability_distribution.xlsx

Return to the [index](#)

26.5 prob_t_inv

Computes the percentile for the Student's t distribution: $\text{Prob}(t_n < t) = p$

`prob_t_inv (p, df)`

Returns

The percentile

Parameters

p Probability

df Degree of freedom

Examples

probability_distribution.xlsx

Return to the [index](#)

26.6 prob_t_table

Generates a table of the percentiles given a set of degrees of freedom and a set of probabilities for the Student's t distribution: $\text{Prob}(t_n < t) = P$

`prob_t_table ()`

Returns

A table of the percentiles for the Student's t distribution

Examples

probability_distribution.xlsx

Return to the [index](#)

26.7 prob_chi

Computes the cumulative probability given c and the degree of freedom for the Chi-Squared distribution:
 $\text{Prob}(X^2 < c)$

prob_chi (c , df , nc)

Returns

The cumulative probability

Parameters

x Input value

df Degree of freedom

nc Optional: noncentrality parameter. Default: 0

Examples

probability_distribution.xlsx

Return to the [index](#)

26.8 prob_chi_inv

Computes the percentile for the Chi-Squared distribution: $\text{Prob_chi}(X^2 < x) = p$

prob_chi_inv (p , df , nc)

Returns

The percentile

Parameters

p Probability

df Degree of freedom

nc Optional: noncentrality parameter. Default: 0

Examples

probability_distribution.xlsx

Return to the [index](#)

26.9 prob_chi_table

Generates a table of the percentiles given a set of degrees of freedom and a set of probabilities for the Chi-Squared distribution: $\text{Prob}(X^2 < c) = P$

prob_chi_table (nc)

Returns

A table of the percentiles for the Chi-Squared distribution

Parameters

nc Optional: noncentrality parameter. Default: 0

Examples

probability_distribution.xlsx

Return to the [index](#)

26.10 prob_f

Computes the cumulative probability given *f* and the degree of freedom for the F-distribution: $\text{Prob}(F(df1, df2) < f)$

prob_f (f, df1, df2)

Returns

The cumulative probability

Parameters

f Input value

df1 Degree of freedom for the numerator

df2 Degree of freedom for the denominator

Examples

probability_distribution.xlsx

Return to the [index](#)

26.11 prob_f_inv

Computes the percentile for the F-distribution: $\text{Prob}(F(df1, df2) < f) = p$

`prob_f_inv (p, df1, df2)`

Returns

The percentile

Parameters

p Probability

df1 Degree of freedom for the numerator

df2 Degree of freedom for the denominator

Examples

probability_distribution.xlsx

Return to the [index](#)

26.12 prob_f_table

Generates a table of the percentiles given a set of degrees of freedom and a probability for the F-distribution:

$\text{Prob}(F(df1, df2) < f) = p$

`prob_f_table (p)`

Returns

A table of the percentiles for the F-distribution

Parameters

p A probability

Examples

probability_distribution.xlsx

Return to the [index](#)

26.13 Cornish_Fisher_expansion

Computes the percentile of a distribution with a skewness and an excess kurtosis by Cornish-Fisher expansion

`Cornish_Fisher_expansion (cl, skewness, kurtosis)`

Returns

A percentile

Parameters

cl A confidence level

skewness A skewness parameter

kurtosis An excess kurtosis parameter (in excess of 3, which corresponds to a Gaussian distribution)

Remarks

The Cornish-Fisher expansion up to the first three terms is

$$x = z + \frac{S}{6}(z^2 - 1) + \frac{K}{24}(z^3 - 3z) - \frac{S^2}{36}(2z^3 - 5z)$$

where z is the percentile of a standard Gaussian distribution for the given confidence level, S is the skewness parameter, and K is the excess kurtosis parameter (in excess of 3, which corresponds to a Gaussian distribution). For a standard Gaussian distribution, the skewness, $S = E[z^3] = 0$, the excess kurtosis, $K = E[z^4] - 3 = 0$.

Examples

probability_distribution.xlsx

Return to the [index](#)

Chapter 27

Excel Built-in Statistical Distribution Functions

BETADIST Returns the beta cumulative distribution function

BETAINV Returns the inverse of the cumulative distribution function for a specified beta distribution

BINOMDIST Returns the individual term binomial distribution probability

CHIDIST Returns the one-tailed probability of the chi-squared distribution

CHIINV Returns the inverse of the one-tailed probability of the chi-squared distribution

CRITBINOM Returns the smallest value for which the cumulative binomial distribution is less than or equal to a criterion value

EXPONDIST Returns the exponential distribution

FDIST Returns the F probability distribution

FINV Returns the inverse of the F probability distribution

GAMMADIST Returns the gamma distribution

GAMMAINV Returns the inverse of the gamma cumulative distribution

HYPGEOMDIST Returns the hypergeometric distribution

LOGINV Returns the inverse of the lognormal distribution

LOGNORMDIST Returns the cumulative lognormal distribution

NEGBINOMDIST Returns the negative binomial distribution

NORMDIST Returns the normal cumulative distribution

NORMINV Returns the inverse of the normal cumulative distribution

NORMSDIST Returns the standard normal cumulative distribution

NORMSINV Returns the inverse of the standard normal cumulative distribution

POISSON Returns the Poisson distribution

TDIST Returns the Student's t-distribution

TINV Returns the inverse of the Student's t-distribution

WEIBULL Returns the Weibull distribution

Return to the [index](#)

References

[1] <http://www.DataMinerXL.com> This website has more information about DataMinerXL software. You can download this software and sample spreadsheets at this website.

[2] Wu, J. and Coggeshall, S. (2012), *Foundations of Predictive Analytics*, Chapman & Hall/CRC.

http://www.amazon.com/Foundations-Predictive-Analytics-Knowledge-Discovery/dp/14398691_18?ie=UTF8&qid=1328999555&sr=8-18

[3] Chang, C.C. and Lin, C.J. (2011), LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>

Index

- acf_of_arma, [101](#)
- ADF_test, [83](#)
- arima, [92](#)
- arima_forecast, [95](#)
- arima_simulate, [97](#)
- arma_to_ar, [100](#)
- arma_to_ma, [100](#)
- assignment_solver, [137](#)

- Basic Statistical Functions, [27](#)
- basic_stats.xlsx, [14](#)
- BETADIST, [189](#)
- BETAINV, [189](#)
- binning, [35](#)
- BINOMDIST, [189](#)
- Black_Litterman, [142](#)
- Box_white_noise_test, [81](#)

- CHIDIST, [189](#)
- CHIINV, [189](#)
- Clustering and Segmentation Functions, [119](#)
- clustering_segmentation.xlsx, [14](#)
- cmds, [120](#)
- Control Theory Functions, [145](#)
- Cornish_Fisher_expansion, [186](#)
- corresp_analysis, [109](#)
- Correspondence Analysis Functions, [109](#)
- correspondence_analysis.xlsx, [14](#)
- CRITBINOM, [189](#)

- Data Manipulation Functions, [19](#)
- data_fill, [24](#)
- data_info, [20](#)
- data_load, [22](#)
- data_lookup, [21](#)
- data_manipulation_functions.xlsx, [14](#)
- data_partition, [23](#)
- data_save, [21](#)
- data_save_tex, [22](#)
- data_sort, [23](#)
- decision_tree_based_model.xlsx, [14](#)
- diff_evol_min_solver, [135](#)
- diff_evol_nls_solver, [135](#)
- diff_evol_solver, [134](#)

- efficient_frontier, [141](#)
- Excel Built-in Statistical Distribution Functions, [189](#)
- EXPONDIST, [189](#)

- factor_analysis, [63](#)
- Fast Fourier Transform Functions, [171](#)
- fast_Fourier_transform.xlsx, [14](#)
- FDIST, [189](#)
- FFT, [171](#)
- FINV, [189](#)
- freq, [29](#)
- freq_2d, [30](#)
- freq_2d_from_file, [30](#)
- freq_from_file, [29](#)
- function_eval, [178](#)
- function_list, [17](#)

- GAMMADIST, [189](#)
- GAMMAINV, [189](#)
- garch, [86](#)
- gauss_hermite, [176](#)
- gauss_laguerre, [176](#)
- gauss_legendre, [175](#)

- Halton_numbers, [178](#)
- Holt_Winters, [89](#)
- Holt_Winters_forecast, [90](#)
- HP_filter, [92](#)
- HYPGEOMDIST, [189](#)

- IFFT, [172](#)
- integral, [177](#)

- k_means, [119](#)
- k_means_from_file, [120](#)
- Kaplan_Meier, [107](#)

- Lagrange_interpolation, [38](#)
- Latin_hypercube, [179](#)
- lcp, [133](#)
- LDA, [103](#)
- lda_qda.xlsx, [14](#)
- Linear and Quadratic Discriminant Analysis Functions, [103](#)

- Linear Regression Functions, 65
- linear_prog, 131
- linear_reg, 65
- linear_reg.xlsx, 14
- linear_reg_forward_select, 67
- linear_reg_forward_select_from_file, 68
- linear_reg_from_file, 66
- linear_reg_piecewise, 69
- linear_reg_piecewise_from_file, 69
- linear_reg_score_from_coefs, 68
- LOGINV, 189
- Logistic Regression Functions, 75
- logistic_reg, 75
- logistic_reg.xlsx, 14
- logistic_reg_forward_select, 77
- logistic_reg_forward_select_from_file, 77
- logistic_reg_from_file, 76
- logistic_reg_score_from_coefs, 78
- LOGNORMDIST, 189
- lowess, 85
- Mann_Kendall_trend_test, 82
- Matrix Operation Functions, 147
- matrix_chol, 160
- matrix_complex_eigen, 161
- matrix_complex_exp, 167
- matrix_complex_pinv, 157
- matrix_complex_QR, 163
- matrix_complex_Schur, 164
- matrix_corr, 150
- matrix_corr_from_cov, 151
- matrix_corr_from_file, 150
- matrix_cov, 149
- matrix_cov_from_corr, 151
- matrix_cov_from_file, 149
- matrix_decimal_to_fraction, 169
- matrix_det, 166
- matrix_diag, 155
- matrix_directprod, 152
- matrix_distance, 167
- matrix_eigen, 161
- matrix_elementprod, 153
- matrix_exp, 166
- matrix_freq, 168
- matrix_from_vector, 168
- matrix_I, 154
- matrix_inv, 156
- matrix_LU, 162
- matrix_minus, 154
- matrix_operations.xlsx, 14
- matrix_pentadiagonal_solver, 158
- matrix_pinv, 156
- matrix_plus, 153
- matrix_prod, 152
- matrix_QR, 163
- matrix_random, 148
- matrix_Schur, 164
- matrix_solver, 157
- matrix_stdev_from_cov, 151
- matrix_svd, 162
- matrix_sweep, 165
- matrix_Sylvester_solver, 159
- matrix_sym_eigen, 160
- matrix_t, 154
- matrix_to_vector, 169
- matrix_tr, 155
- matrix_tridiagonal_solver, 158
- maxflow_solver, 138
- mds, 121
- means, 31
- means_from_file, 31
- merge_tables, 25
- model_bin_eval, 43
- model_bin_eval_from_file, 44
- model_cont_eval, 44
- model_cont_eval_from_file, 45
- model_eval, 46
- model_eval_from_file, 47
- model_save_scoring_code, 50
- model_score, 48
- model_score_from_file, 49
- model_score_node, 49
- Modeling Functions for All Models, 43
- Naive Bayes Classifier Functions, 111
- naive_bayes.xlsx, 14
- naive_bayes_classifier, 111
- naive_bayes_classifier_from_file, 111
- natural_cubic_spline, 86
- NEGBINOMDIST, 189
- netflow_solver, 138
- Neural Network Functions, 123
- neural_net, 123
- neural_net_from_file, 124
- neural_network_model.xlsx, 14
- nls_solver, 133
- NORMDIST, 189
- NORMINV, 189
- NORMSDIST, 189
- NORMSINV, 189
- Numerical Integration Functions, 175
- numerical_integration.xlsx, 14
- Optimization Functions, 131
- optimization.xlsx, 14
- Partial Least Square Regression Functions, 73
- PCA, 63

- pca_factor_analysis.xlsx, 14
- percentiles, 32
- pls_reg, 73
- pls_reg.xlsx, 14
- pls_reg_from_file, 74
- POISSON, 189
- pole_placement, 145
- poly_prod, 37
- poly_reg, 70
- poly_roots, 37
- Portfolio Optimization Functions, 141
- portfolio_optimization.xlsx, 14
- prime_numbers, 178
- Principal Component Analysis and Factor Analysis Functions, 63
- prob_chi, 184
- prob_chi_inv, 184
- prob_chi_table, 185
- prob_f, 185
- prob_f_inv, 186
- prob_f_table, 186
- prob_normal, 181
- prob_normal_inv, 182
- prob_normal_table, 182
- prob_t, 183
- prob_t_inv, 183
- prob_t_table, 183
- Probability Functions, 181
- QDA, 104
- QQ_plot, 36
- quadratic_prog, 132
- rank_items, 25
- ranks, 28
- ranks_from_file, 28
- sarima, 94
- sarima_forecast, 96
- sarima_simulate, 98
- set, 39
- set_difference, 41
- set_intersection, 40
- set_union, 40
- shortest_path_solver, 139
- Sobol_numbers, 179
- sort_file, 24
- split_str, 26
- stochastic_process, 87
- stochastic_process_simulate, 88
- subset, 20
- summary, 33
- summary_from_file, 34
- Support Vector Machine Functions, 127
- support_vector_machine.xlsx, 14
- Survival Analysis Functions, 107
- svm, 127
- svm_from_file, 128
- TDIST, 189
- three_moment_match_to_SLN, 39
- Time Series Analysis Functions, 79
- time_series_analysis.xlsx, 14
- TINV, 189
- transportation_solver, 136
- tree, 113
- Tree-Based Model Functions, 113
- tree_boosting_logistic_reg, 115
- tree_boosting_logistic_reg_from_file, 116
- tree_boosting_ls_reg, 117
- tree_boosting_ls_reg_from_file, 118
- tree_from_file, 114
- ts_acf, 80
- ts_ccf, 81
- ts_diff, 84
- ts_pacf, 80
- ts_sma, 85
- univariate, 32
- univariate_from_file, 32
- Utility Functions, 17
- variable_corr_select, 36
- variable_list, 19
- version, 17
- WEIBULL, 189
- Weight of Evidence Transformation Functions, 53
- weight_of_evidence.xlsx, 14
- woe_transform, 60
- woe_transform_from_file, 60
- woe_xcat_ybin, 57
- woe_xcat_ybin_from_file, 58
- woe_xcat_ycont, 58
- woe_xcat_ycont_from_file, 59
- woe_xcont_ybin, 53
- woe_xcont_ybin_from_file, 54
- woe_xcont_ycont, 55
- woe_xcont_ycont_from_file, 56