sasToolkit Macro Library Documentation

Luke W. Johnston

true

Contents

anova **2**

anova . . . 3

beta_glm **4**

beta_glm . . . 4

chisq **5**

chisq . . . 6

contents **7**

contents . . . 7

correlation **8**

correlation . . . 8

csvgz_import **9**

csvgz_import . . . 9

csvimport **10**

csvimport . . . 10

for **12**

for . . . 12

freq **13**

freq 14

gee **14**

gee 15

logreg **16**

oddsratio 17

macros **17**

means **17**

means 18

mergeMeansAnova **18**

mergeMeansAnova 19

output_data **19**

output_data 20

pca **20**

pca 21

anova

Analysis of Variance (ANOVA)

Author:

- Luke W. Johnston

Created:

- 2014-10-28

anova

```
%macro anova ( dsn= , category= , numerical= , adjust=tukey , by= , where= ,
outds=_NULL_ , outpdiff=_NULL_ , dcovar= , ccovar= );
```

Analysis of Variance (ANOVA) loop

Runs an ANOVA, using a loop for multiple variables, and prints out the pertinent information. An Analysis of Covariance (ANCOVA) can also be run if the `dcovar` or `ccovar` arguments are specified.

Examples:

Parameters:

- dsn - The dataset that contains the variables. This argument is positional and needs to be the first argument
- category - The discrete/categorical variable (for example, Sex - "M"/"F"
- numerical - The continuous variable (for example, weight)
- adjust - Post-hoc test (including `tukey`, `bon`) (default value: tukey)
- by - The by variable to split the analysis based on the `by` argument variable
- where - Argument to subset the dataset by before the analysis
- outds - The main output dataset, if specified (default value: _NULL_)
- outpdiff - The output dataset that contains the post-hoc p-values between the categories of the discrete variable (`category` argument) (default value: _NULL_)
- dcovar - The discrete covariate to adjust for
- ccovar - The continuous covariate to adjust for

Returns:

- Prints the results of the analysis, including the post-hoc test.

beta_glm

Simple or multiple linear regression

Author:

- Luke W. Johnston

Created:

- 2014-11-04

beta_glm

```
%macro beta_glm ( dsn= , y= , x= , dcovar= , ccovar= , interactvar= ,
outall=_NULL_ , outcore=_NULL_ , outObs=_NULL_ , outRSq=_NULL_ , outResid=tmp
, sigDigits=0.01 , by= );
```

Multiple (or simple) linear regression for predicting the outcome based on the exposures.

Runs linear regression, sending the results into output datasets that can be used in LaTeX as tables. In this macro, there are two loops going on. This allows any number of exposures and outcomes to be specified in the macro, running regressions on each outcome with each exposure. This is allows the code to be cleaner, leaner, more efficient, and more maintainable.

The loops work in a combinatoric fashion, starting with the `y`, or dependent variable, then going through each of the `x` variables. For instance, I want to run a regression on BMI and dietary fat with insulin resistance and blood lipids. The `y` variable would have insulin resistance and blood lipids, while the `x` would have BMI and dietary fat. The macro would run insulin resistance with BMI, then insulin resistance with dietary fat, then blood lipids and BMI and so on.

All output datasets are optional as each of the dataset variables are set to _NULL_. The results datasets that can be output are the beta and standard error (plus p-value), the `$R^2$`, and the sample size.

Each variable, except for the dataset variables, can have multiple variables included, each separated by a space, **not** a comma.

As a reminder (for using the below variables), the linear regression equation is: $$y = B_0 + B_1 x_1 + \ldots + B_n x_n + e$$

Examples:

```
    %beta_glm(sashelp.fish, y = Weight, x = Length1 Width, dcovar = Species,
outcore = regressionResults); proc print data=regressionResults; run;
```

Parameters:

- dsn - The dataset that contains the variables. This arg is positional and so needs to come first
- y - The outcome or dependent variable; this arg is positional and comes second. If more than one variable is included, the macro will loop through each variable
- x - The exposure or independent variable; this arg is positional and comes third. If more than one variable is included, the macro will loop through each variable
- dcovar - The discrete covariates
- ccovar - The continuous covariates
- interactvar - The interaction variable (if of interest)
- outall - Outputs the beta estimates for all the variables in the linear regression model, including the covariates (default value: _NULL_)
- outcore - Outputs only the beta estimates (default value: _NULL_)
- outObs - Outputs the observations read and used (default value: _NULL_)
- outRSq - Outputs the R-squared (default value: _NULL_)
- outResid - Outputs the regression residuals (default value: tmp)
- sigDigits - Specify the number of significant digits for the output datasets (default value: 0.01)
- by - Variable to run the analysis on individually (for example, to run a regression on each Sex)

Returns:

- Prints the regression model beta estimates, the observations used, as well as the R-squared

chisq

Chi-Square Statistical Analysis.

Author:

- Luke Johnston

Created:

- 2014-09-29

chisq

```
%macro chisq ( vars= , dsn= , tests=chisq , outFreq=NULL , outChi=tmp ,
order=freq , testOpt= , where= , by= );
```

Runs a Chi-Square analysis.

Chi square is a simple statistical test to calculate differences between proportions in a 2x2 and other contingency/frequency tables. This macro simply runs a chi-square test and outputs the frequency and probability of difference between groups.

Examples:

```
` Using the SAS bone marrow transplant dataset; %chisq(Group Status,
sashelp.bmt, where=T lt 2000, tests=chisq or, testOpt=relrisk, outChi=chi); `
```

Parameters:

- vars - Contains the variables to be analyzed. The argument is positional — the variables must be specified first before other arguments
- dsn - Dataset name which contains the variables. The argument is positional — the variable must be specified second
- tests - Specify which tests to run (default value: chisq)
- outFreq - Output dataset which contains the frequencies (default value: NULL)
- outChi - Output dataset which contains the chi-square statistics (default value: tmp)
- order - The order in which to display the variables in the `outFreq` dataset (default value: freq)
- testOpt - Options to pass to the tables `chisq` command
- where - To subset the data before analyzing the variables
- by - To run the analysis separately by categories of the by variable

Returns:

- Prints the frequencies and chi-square statistic results by default. Will output datasets if specified.

contents

Print dataset variable contents.

Author:

- Luke W Johnston

Created:

- 2014-10-08

contents

```
%macro contents ( datasets= , lib=work , outds=tmp );
```

Print the variable contents of one or more datasets.

This macro loops through multiple (or just one) dataset and prints off all the variable names (the header row) of each dataset. It also prints what type the variable is, either a character or a numeric.

Examples:

```
Two SAS help datasets;\
%contents(datasets=class heart, lib=sashelp);\
```

Parameters:

- datasets - One or more datasets that you want to see the contents within. It is a positional macro variable, so requires that the variable be given first
- lib - The libname. The `work` libname is the default environment SAS uses. Other examples may be `sashelp`; you can also create your own libname if you have a SAS dataset (which I don't recommend having) (default value: work)
- outds - Specify an output dataset name to output the results of the macro (default value: tmp)

Returns:

- Prints all the variables and their formats within the specified datasets.

correlation

Correlation coefficient statistics.

Author:

- Luke W. Johnston

Created:

- 2014-10-26

correlation

```
%macro correlation ( dsn= , topvar= , sidevar= , covar= , where= , outds=_NULL_
, coeff_test=Spearman );
```

Compute correlation coefficients.

Runs any type of correlation coefficients on either continuous or discrete variables. You can specify to use either Spearman, Pearson, Tau, and others for calculating the coefficients. Partial correlations, which are adjusted for other variables, can also be computed.

Examples:

```
`%correlation(dsn=sashelp.class, topvar=Height, sidevar=Weight,
coeff_test=Pearson);`
```

Parameters:

- dsn - The dataset with the variables. The macro argument is positional and needs to be specified first
- topvar - The variables on the top of the output (the header row; those that make up the **columns**). This macro argument is positional and needs to be specified second.
- sidevar - The variables on the side of the output (generally the first column; those that make up the **rows**). This variable is optional. If not specified, the correlations form a matrix.
- covar - Variables to adjust for
- where - A condition to subset the analysis by (for example, `where=Sex eq 'F'`)
- outds - The output dataset (default value: _NULL_)

- coeff_test - The correlation statistical test to run (`Spearman`, `Pearson`, `Tau`, etc.) (default value: Spearman)

Returns:

- Prints the correlation coefficients. The results are not sent to the output dataset by default.

csvgz_import

Import gzipped compressed `csv` into SAS

Author:

- Luke W. Johnston

Created:

- 2014-10-05

csvgz_import

```
%macro csvgz_import ( dataset= , outds=&ds , dir=/tmp );
```

Imports a gzipped (compressed) `csv` file into the SAS work space.

This macro is used to uncompress than read in a `csv.gz` (meaning gzipped) file. SAS temporarily uncompresses the file, so that the original file remains intact.

Requirements: `%csvimport()` macro and a Unix/Linux operating system (OS). I will need to make this macro more flexible for other OS.

Examples:

```
`%csvgz_import(dataset=/home/username/data/projectData.csv.gz, outds=working,
dir=/home/username/projects/researchprojects/diabetesObesity/data);`
```

Parameters:

- dataset - The importing dataset, with the full path to the dataset to be imported
- outds - The output dataset (default value: &ds)

- dir - The directory where the data will temporarily created. The recommended directory is where the subsetted data will be saved to in the research project folder structure (default value: /tmp)

Returns:

- Imports a compressed (`.csv.gz`) file to the specified directory.

csvimport

Import csv data files

Author:

- Luke W. Johnston

Created:

- 2014-11-21 (updated)

csvimport

```
%macro csvimport ( dataset= , outds=&dataset , dir=../data );
```

Import comma separated value files into the SAS workspace.

Imports a non-compressed csv (comma separated values) dataset and puts it in the SAS workspace (outds).

Examples:

```
` %csvimport(testdata, outds=working, dir=./data); `
```

Parameters:

- dataset - The name of the dataset file to import, **WITHOUT** the file extension (csv; for example a file called testdata.csv, the dataset name would be testdata). Is a positional argument and must come first
- outds - The name of the output dataset as it will be referenced by SAS in `proc` or `data` statements (default value: &dataset)
- dir - The location (file path) of the input dataset, for example ./data or /home/users/research/data (default value: ../data)

Returns:

- Imports a SAS work dataset into the workspace.

directory to your SAS AUTOCALL library. See the `README.pdf` file in the `sasToolkit folder.`

These SAS files are currently being maintained by Luke W. Johnston. Refer to the GitHub repository (github.com/lwjohnst86/sasToolkit) for an up-to-date version of these SAS macro files.

Preferably, when using these macros, either reference them in their original location (e.g. $HOME/sasToolkit, or the project directory. If the files are to be pasted into the diretory (i.e. for modularity of your research project), it would be best to change the permissions of the file to `read-only. This would ensure that the macro files outside of the master location would not be modified and thus maintaining your sanity (as multiple copies of the macro files is a nightmare to manage). The best bet is to use the AUTOCALL facility in SAS.`

A note on terminology:
directory = folder
$HOME = your home user folder (such as /home/joe/ for Linux, C:\Users\joe for Windows, /Users/joe for Mac)
AUTOCALL library = please see the `README.pdf in the sasToolkit folder.`

Files

anova

Analysis of Variance (ANOVA).

beta_glm

Simple or multiple linear regression.

chisq

Chi-Square Statistical Analysis.

contents

Print dataset variable contents.

correlation

Correlation coefficient statistics.

csvgz_import

Import gzipped compressed `csv` into SAS.

csvimport

Import csv data files.

for

For loop in SAS.

freq

Frequency of discrete variables.

gee

Generalized Estimating Equations Analysis.

logreg

Logistic regression – in development.

macros

means

Prints summary stats of continuous variables.

mergeMeansAnova

Merge means and ANOVA output datasets.

output_data

Output a dataset to a `csv` file.

pca

for

For loop in SAS

Author:

- Jim Anderson, UCSF, james.anderson@ucsf.edu
  "Please keep, use and pass on the %for macro with this authorship note. -Thanks"

Created:

- 2012-02-12 (as per last update on site)

for

```
%macro for ( macro_var_list= , in= , do= );
```

For loop able to operate in open code.

Taken from:

This macro performs a loop generating SAS code. It proceeds sequentially through one of 5 kinds of data objects: SAS dataset, value list, number range, dataset contents and directory contents. Data object values are assigned to macro variables specified in macro_var_list upon each loop iteration.

Please send improvements, fixes or comments to Jim Anderson.

The example below loops over the dataset "report_hosps", which has dataset variables "hospid" and "hospname". For each dataset observation, macro variables &hospid and &hospname are assigned values from the identically named dataset variables and the loop code is generated, which in the example prints a report.

Examples:

```
` %for(hospid hospname, in=report_hosps, do=%nrstr( title "&hospid &hospname
Patient Safety Indicators"; proc print data=psi_iqi(where=(hospid="&hospid"));
run; )) `
```

Parameters:

- macro_var_list - space-separated list of macro variable names to be assigned values upon each loop iteration. Is a positional argument, so needs to come first
- in - Data object to access. Object type is distinguished by choice of brackets (or no backets for a range):

freq

Frequency of discrete variables.

Author:

- Luke W. Johnston

Created:

- 2014-10-19

freq

```
%macro freq ( vars= , dsn=&ds , by= , where= , outds=_NULL_ );
```

Compute the frequencies of discrete variables.

Macro for determining the frequencies of categorical/discrete variables. Said another way, the macro determines the count of each category in a discrete variable (for example, the number of “Males” vs “Females”).

Examples:

```
`%freq(Group Status, dsn=sashelp.bmt, where=T gt 500, outds=test);`
```

Parameters:

- vars - Discrete variables to compute frequencies
- dsn - The (input) dataset containing the variables (default value: &ds)
- by - Variable to split the discrete variable up and then calculate the frequencies
- where - Expression that subsets the dataset before running the analysis
- outds - The output dataset (default value: _NULL_)

Returns:

- Prints the frequencies of the discrete variables.

gee

Generalized Estimating Equations Analysis

Author:

- Luke W. Johnston

Created:

- 2014-09-20

gee

```
%macro gee ( dsn= , x= , y= , time= , subject= , ccovar= , dcovar= , dist=normal
, link=identity , wcorr=exch , sigDigits=0.001 , outAll=_NULL_ , outCore=tmp
, outObs=tmp );
```

Generalized estimating equation (GEE) loop for exposures and outcomes.

This macro runs a longitudinal statistical test known as **generalized estimating equations**. As with my other macros (for example `beta_glm`), this macro has two loops: one loops through all of the exposure or `x` variables and another that will loop through all the outcome or `y` variables.

By default, the macro prints the main findings from the GEE analysis. However, output datasets can be specified (for instance, `outCore`), which can than be "massaged" and/or output into a `.csv` file. Many of the macro variables can have multiple variables specified, but each additional variable must be separated by a space **not a comma**.

While GEE in SAS has to have the distribution, link, and working correlation matrix specified, GEE is very robust (aka consistent, reliable) to misspecified assumptions.

Examples:

Parameters:

- dsn - Dataset that contains the variables of interest. This is a positional variable and must be declared first
- x - The independent or exposure variables. If more than one `x` is provided, the macro will loop through each `x` and run the GEE on each
- y - The dependent or outcome variables. As with the `x` variable, more than one outcome variable will be looped through the GEE
- time - The variable used to indicate time, for example `VisitNumber` or `Age`
- subject - The variable that specifies the identifier for the subject/participant, for example `SID` or `ID`
- ccovar - The continuous covariates or confounders
- dcovar - The discrete/categorical covariates or confounders
- dist - The distribution assumption, which is dependent on the type of data the `x` or the `y` is (for example, continuous or discrete). Other distributions include Normal, Poisson, Binomial, Multinomial, etc. (default value: normal)

- link - The link function to be used in conjunction with the `dist` variable. For example, when `dist=poisson`, the default link is `log`. Other links include Logit, Identity, Inverse, etc. (default value: identity)
- wcorr - The specified GEE working correlation matrix. The standard and commonly used is the Exchangeable (or `exch`), but others include Autoregressive and Independent. GEE is very robust (aka reliable/consistent) to a misspecified working correlation matrix (default value: exch)
- sigDigits - Significant digits to round the output results to (default value: 0.001)
- outAll - The name of the results dataset that contains all of the parameter estimates (that is, including the covariates) (default value: _NULL_)
- outCore - The name of the results dataset that contains the parameter estimates of **only** the `x` and `y` variables (default value: tmp)
- outObs - The output results dataset that contains the observations used in the analysis (default value: tmp)

Returns:

- Prints GEE model info, working correlations, and the parameter estimates. By default, no result datasets are output. However, datasets can be output to be massaged or output into a file.

logreg

Logistic regression – in development

Author:

- Luke W. Johnston

Created:

- 2014-11-21 (developing)

oddsratio

```
%macro oddsratio ( y=&dep , x=&indep , dcovar= , ccovar= , dsn=&ds , outall=_NULL_
, outcore=_NULL_ , outobs=_NULL_ );
```

Logistic regression in development

Examples:

Parameters:

- y - Dependent variable (default value: &dep)
- x - (default value: &indep)
- dcovar -
- ccovar -
- dsn - (default value: &ds)
- outall - (default value: _NULL_)
- outcore - (default value: _NULL_)
- outobs - (default value: _NULL_)

Returns:

- In progress

macros

means

Prints summary stats of continuous variables.

Author:

- Luke W Johnston

Created:

- 2014-10-14

means

```
%macro means ( vars= , dsn=&ds , by= , class= , where= , outds=_NULL_ );
```

Prints summary statistics such as means and median for continuous variables.

This macro outputs summary statistics from continuous variables. Means and standard deviations are output as a single column, as well as the median and interquartile range, for ease in putting into tables. It can be univariate or divariate.

Examples:

```
`%means(Height Weight Age, dsn=sashelp.class, where=Sex eq 'F');`
```

Parameters:

- vars - The continuous variables to summarize. It is a positional argument, and is placed first
- dsn - The dataset that contains the variables (default value: &ds)
- by - Summarizes the `vars` according to a discrete, **sorted**, variable
- class - Similar to `by`, except it does **not** need to be sorted
- where - A condition to subset the data by
- outds - The output dataset name (default value: _NULL_)

Returns:

- Prints the summary statistics.

mergeMeansAnova

Merge means and ANOVA output datasets

Author:

- Luke W. Johnston

Created:

- 2014-11-21 (updated)

mergeMeansAnova

```
%macro mergeMeansAnova ( meansds= , anovads= , byVar= , byVarNumLevels=_NULL_
, byVarCatLevels=_NULL_ , outds=_NULL_ );
```

Merge the output datasets generated from the means and anova macros.

This macro merges the results of the `means` and `anova` macro, which are datasets, into one dataset with the p-value included for difference between groups. The output for the means needs to have had an argument for the by variable (`by=`), so that the means dataset can be transformed from long to wide. The variables from both the `means` and the `anova` macro need to be in the same order/sequence.

Examples:

Parameters:

- meansds - Output dataset from the `means` macro, needs to be the first argument
- anovads - Output dataset from the `anova` macro, has to be the second argument
- byVar - The categorical (or discrete/binary number, such as for an order) variable that was used to group both the anova and the means datasets
- byVarNumLevels - The levels of the discrete/binary number `byVar`, such as 0 1, 0 1 2 3, etc (default value: _NULL_)
- byVarCatLevels - The levels of the categorical `byVar` such as Yes No or Female Male (default value: _NULL_)
- outds - Optional as the output dataset will be named after the `means` dataset (default value: _NULL_)

Returns:

- Outputs a dataset with the merged means and p-values for differences between groups

output_data

Output a dataset to a `csv` file.

Author:

- Luke W Johnston

Created:

- 2014-10-11

output_data

```
%macro output_data ( dataset= , dir=tmp );
```

Output a dataset into a `csv` file into a directory.

This macro takes a dataset and converts it into a "comma separated value" (or `csv`) file. The macro also removes double quotes from the output dataset.

Examples:

```
`proc means data=sashelp.class; ods output summary=meansData; run;
%output_data(meansData, dir=./output);`
```

Parameters:

- dataset - The dataset to output to the `csv` file. It is a positional argument and so needs to be first
- dir - The directory/folder path that the `csv` output will be saved to (default value: tmp)

Returns:

- Outputs a `csv` file.

pca

Principal Component Analysis

Author:

- Luke W. Johnston

Created:

- 2014-10-22

pca

```
%macro pca ( dsn= , vars= , numPC= , opt_rotate=none , by= , where= ,
outEig=tmp , outPattern=tmp1 , outRotPat=tmp2 , outVariance=tmp3 );
```

Run principal component analysis to reduce the number of dimensions.

This macro runs a principal component analysis (PCA) on the specified variables. PCA is a dimensionality reduction statistical technique, used to take a large number of variables and output a smaller number of variables that explain a large amount of variance within the data matrix. PCA is a subset of factor analysis, though it differs quite markedly from exploratory (EFA) or confirmatory factor analysis (CFA), among others. There is no assumption of an underlying factor or factors for the variables of interest, while EFA and CFA do make that assumption.

Examples:

```
`%pca(sashelp.fish, Weight Length1 Length2 Length3 Height Width, numPC=1,
opt_rotate=varimax);`
```

Parameters:

- dsn - The dataset that contains the variables. Is a positional variable, so needs to be specified first
- vars - The variables that are used to generate the principal components. It is a positional variable and needs to be specified second
- numPC - The number of principal components to output
- opt_rotate - The (optional) rotation applied to the PCA (default value: none)
- by - Variable to split the analysis up
- where - Condition to subset the analysis
- outEig - Output the eigenvalues from the PCA (default value: tmp)
- outPattern - Output the component patterns (default value: tmp1)
- outRotPat - Output the rotated component patterns (default value: tmp2)
- outVariance - Output the variance of the patterns (default value: tmp3)

Returns:

- Prints the eigenvalues, explained variance, and component patterns by default