

Package ‘procs’

September 5, 2023

Type Package

Title Recreates Some 'SAS®' Procedures in 'R'

Version 1.0.3

Maintainer David Bosak <dbosak01@gmail.com>

Description Contains functions to simulate the most commonly used 'SAS®' procedures. Specifically, the package aims to simulate the functionality of 'proc freq', 'proc means', 'proc transpose', 'proc sort', and 'proc print'. The simulation will include recreating all statistics with the highest fidelity possible.

License CC0

Encoding UTF-8

URL <https://procs.r-sassy.org>, <https://github.com/dbosak01/procs>

BugReports <https://github.com/dbosak01/procs/issues>

Depends R (>= 3.6.0), common

Suggests knitr, rmarkdown, testthat (>= 3.0.0), covr, logr

Imports utils, fmtr, reporter, stats, tibble

VignetteBuilder knitr

RoxygenNote 7.2.3

Config/testthat/edition 3

NeedsCompilation no

Author David Bosak [aut, cre],
Kevin Kramer [ctb],
Brian Varney [ctb],
Duong Tran [ctb],
Yifei Chen [ctb]

Repository CRAN

Date/Publication 2023-09-05 16:40:02 UTC

R topics documented:

proc_freq	2
proc_means	9
proc_print	14
proc_sort	15
proc_transpose	18

Index	22
--------------	-----------

proc_freq	<i>Generates Frequency Statistics</i>
-----------	---------------------------------------

Description

The `proc_freq` function generates frequency statistics. It is both an interactive function that can be used for data exploration, and can produce dataset output for further analysis. The function can perform one and two-way frequencies. Two-way frequencies are produced as a cross-tabulation by default. There are many options to control the generated tables. The function will return requested tables in a named list.

Usage

```
proc_freq(
  data,
  tables = NULL,
  output = NULL,
  by = NULL,
  weight = NULL,
  options = NULL,
  titles = NULL
)
```

Arguments

<code>data</code>	The input data frame to perform frequency calculations on. Input data as the first parameter makes this function pipe-friendly.
<code>tables</code>	The variable or variables to perform frequency counts on. The table specifications are passed as a vector of strings. For one-way frequencies, simply pass the variable name. For two-way tables, pass the desired combination of variables separated by a star (*) operator. The parameter does not accept SAS® style grouping syntax. All cross combinations should be listed explicitly. If the table request is named, the name will be used as the list item name on the return list of tables. See "Example 3" for an illustration on how to name an output table.
<code>output</code>	Whether or not to return datasets from the function. Valid values are "out", "none", and "report". Default is "out". This parameter also accepts the data shaping options "long", "stacked", and "wide". See the Data Shaping section for a description of these options. Multiple output keywords may be passed on

	a character vector. For example, to produce both a report dataset and a "long" output dataset, use the parameter <code>output = c("report", "out", "long")</code> .
<code>by</code>	An optional by group. Parameter accepts a vector of one or more variable names. When this parameter is set, data will be subset for each by group, and tables will be generated for each subset.
<code>weight</code>	An optional weight parameter. This parameter is passed as a variable name to use for the weight. If a weight variable is indicated, the weighted value will be summed to calculate the frequency counts.
<code>options</code>	The options desired for the function. Options are passed to the parameter as a vector of quoted strings. You may also use the <code>v()</code> function to pass unquoted strings. The following options are available: "chisq", "crosstab", "fisher", "list", "missing", "nlevels", "nocol", "nocum", "nofreq", "nopercent", "noprint", "nonobs", "norow", "nosparse", "notable", "outcum". See the Options section for a description of these options.
<code>titles</code>	A vector of titles to assign to the interactive report.

Details

The `proc_freq` function generates frequency statistics for one-way and two-way tables. Data is passed in on the `data` parameter. The desired frequencies are specified on the `tables` parameter.

Value

The function will return all requested datasets by default. This is equivalent to the `output = "out"` option. To return the datasets as created for the interactive report, pass the "report" output option. If no output datasets are desired, pass the "none" output option. If a single dataset is requested, the function will return a single dataset. If multiple datasets are requested, the function will return a list of datasets. The type of data frame returned will correspond to the type of data frame passed in on the `data` parameter. If the input data is a tibble, the output data will be a tibble. If the input data is a Base R data frame, the output data will be a Base R data frame.

Report Output

By default, `proc_freq` results will be immediately sent to the viewer as an HTML report. This functionality makes it easy to get a quick analysis of your data with very little effort. To turn off the interactive report, pass the "noprint" keyword to the `options` parameter or set `options("procs.print" = FALSE)`.

The `titles` parameter allows you to set one or more titles for your report. Pass these titles as a vector of strings.

If the frequency variables have a label assigned, that label will be used in the report output. This feature gives you some control over the column headers in the final report.

The exact datasets used for the interactive output can be returned as a list. To return these datasets as a list, pass the "report" keyword on the `output` parameter. This list may in turn be passed to `proc_print` to write the report to a file.

Data Frame Output

The `proc_freq` function returns output datasets. If you are requesting only one table, a single data frame will be returned. If you request multiple tables, a list of data frames will be returned.

By default, the list items are named according to the strings specified on the `tables` parameter. You may control the names of the returned results by using a named vector on the `tables` parameter.

The standard output datasets are optimized for data manipulation. Column names have been standardized, and additional variables may be present to help with data manipulation. For instance, the `by` variable will always be named "BY", and the frequency category will always be named "CAT". In addition, data values in the output datasets are not rounded or formatted to give you the most accurate statistical results.

Frequency Weight

Normally the `proc_freq` function counts each row in the input data equally. In some cases, however, each row in the data can represent multiple observations, and rows should not be treated equally. In these cases, use the `weight` parameter. The parameter accepts a variable/column name to use as the weighted value. If the `weight` parameter is used, the function will sum the weighted values instead of counting rows.

By Groups

You may request that frequencies be separated into by groups using the `by` parameter. The parameter accepts one or more variable names from the input dataset. When this parameter is assigned, the data will be subset by the "by" variable(s) before frequency counts are calculated. On the interactive report, the by groups will appear in separate tables. On the output dataset, the by groups will be identified by additional columns.

Options

The `options` parameter accepts a vector of options. Normally, these options must be quoted. But you may pass them unquoted using the `v()` function. For example, you can request the number of category levels and the Chi-Square statistic like this: `options = v(nlevels, chisq)`.

Below are all the available options and a description of each:

- **crosstab**: Two-way output tables are a list style by default. If you want a crosstab style, pass the "crosstab" option.
- **list**: Two-way interactive tables are a crosstab style by default. If you want a list style two-way table, pass the "list" option.
- **missing**: Normally, missing values are not counted and not shown on frequency tables. The "missing" option allows you to treat missing (NA) values as normal values, so that they are counted and shown on the frequency table. Missing levels will appear on the table as a single dot (".").
- **nlevels**: The "nlevels" option will display the number of unique values for each variable in the frequency table. These levels are generated as a separate table that appears on the report, and will also be output from the function as a separate dataset.
- **nocol**: Two-way cross tabulation tables include column percents by default. To turn them off, pass the "nocol" option.

- **nocum**: Whether to include the cumulative frequency and percent columns on one-way, interactive tables. These columns are included by default. To turn them off, pass the "nocum" option.
- **nofreq**: The "nofreq" option will remove the frequency column from one-way and two-way tables.
- **nopercent**: The "nopercent" option will remove the percent column from one-way and two-way tables.
- **noprint**: Whether to print the interactive report to the viewer. By default, the report is printed to the viewer. The "noprint" option will inhibit printing.
- **nonobs**: Whether to include the number of observations "N" column on the output and interactive tables. By default, the N column will be included. The "nonobs" option turns it off.
- **norow**: Whether to include the row percentages on two-way crosstab tables. The "norow" option will turn them off.
- **nosparse/sparse**: Whether to include categories for which there are no frequency counts. Zero-count categories will be included by default, which is the "sparse" option. If the "nosparse" option is present, zero-count categories will be removed.
- **notable**: Whether to include the frequency table in the output dataset list. Normally, the frequency table is included. You may want to exclude the frequency table in some cases, for instance, if you only want the Chi-Square statistic.
- **outcum**: Whether to include the cumulative frequency and percent on output frequency tables. By default, these columns are not included. The "outcum" option will include them.

Statistics Options

In addition to the above options, the options parameter accepts some statistics options. The following keywords will generate an additional tables of specialized statistics. These statistics options are only available on two-way tables:

- **chisq**: Requests that the Chi-square statistics be produced.
- **fisher**: Requests that the Fisher's exact statistics be produced.

Using Factors

There are some occasions when you may want to define the tables variable or by variables as a factor. One occasion is for sorting/ordering, and the other is for obtaining zero-counts on sparse data.

To order the frequency categories in the frequency output, define the tables variable as a factor in the desired order. The function will then retain that order for the frequency categories in the output dataset and report.

You may also wish to define the tables variable as a factor if you are dealing with sparse data and some of the frequency categories are not present in the data. To ensure these categories are displayed with zero-counts, define the tables variable or by variable as a factor and use the "sparse" option. Note that the "sparse" option is actually the default.

If you do not want to show the zero-count categories on a variable that is defined as a factor, pass the "nosparse" keyword on the options parameter.

Data Shaping

By default, the `proc_freq` function returns an output dataset of frequency results. If running interactively, the function also prints the frequency results to the viewer. As described above, the output dataset can be somewhat different than the dataset sent to the viewer. The output parameter allows you to choose which datasets to return. There are three choices: "out", "report", and "none". The "out" keyword returns the default output dataset. The "report" keyword returns the dataset(s) sent to the viewer. You may also pass "none" if you don't want any datasets returned from the function.

In addition, the output dataset produced by the "out" keyword can be shaped in different ways. These shaping options allow you to decide whether the data should be returned long and skinny, or short and wide. The shaping options can reduce the amount of data manipulation necessary to get the frequencies into the desired form. The shaping options are as follows:

- **long**: Transposes the output datasets so that statistics are in rows and frequency categories are in columns.
- **stacked**: Requests that output datasets be returned in "stacked" form, such that both statistics and frequency categories are in rows.
- **wide**: Requests that output datasets be returned in "wide" form, such that statistics are across the top in columns, and frequency categories are in rows. This shaping option is the default.

See Also

For summary statistics, see [proc_means](#). To pivot or transpose the data coming from `proc_freq`, see [proc_transpose](#).

Examples

```
library(procs)

# Turn off printing for CRAN checks
options("procs.print" = FALSE)

# Create sample data
df <- as.data.frame(HairEyeColor, stringsAsFactors = FALSE)

# Assign labels
labels(df) <- list(Hair = "Hair Color",
                  Eye = "Eye Color",
                  Sex = "Sex at Birth")

# Example #1: One way frequencies on Hair and Eye color with weight option.
res <- proc_freq(df,
                tables = v(Hair, Eye),
                options = outcum,
                weight = Freq)

# View result data
res
# $Hair
#   VAR  CAT  N CNT    PCT CUMSUM  CUMPCT
# 1 Hair Black 592 108 18.24324    108 18.24324
```

```

# 2 Hair Blond 592 127 21.45270    235 39.69595
# 3 Hair Brown 592 286 48.31081    521 88.00676
# 4 Hair Red 592 71 11.99324    592 100.00000
#
# $Eye
# VAR CAT N CNT PCT CUMSUM CUMPCT
# 1 Eye Blue 592 215 36.31757 215 36.31757
# 2 Eye Brown 592 220 37.16216 435 73.47973
# 3 Eye Green 592 64 10.81081 499 84.29054
# 4 Eye Hazel 592 93 15.70946 592 100.00000

# Example #2: 2 x 2 Crosstabulation table with Chi-Square statistic
res <- proc_freq(df, tables = Hair * Eye,
                weight = Freq,
                options = v(crosstab, chisq))

# View result data
res
# $`Hair * Eye`
# Category Statistic Blue Brown Green Hazel Total
#1 Black Frequency 20.000000 68.000000 5.0000000 15.000000 108.00000
#2 Black Percent 3.378378 11.486486 0.8445946 2.533784 18.24324
#3 Black Row Pct 18.518519 62.962963 4.6296296 13.888889 NA
#4 Black Col Pct 9.302326 30.909091 7.8125000 16.129032 NA
#5 Blond Frequency 94.000000 7.000000 16.000000 10.000000 127.00000
#6 Blond Percent 15.878378 1.182432 2.7027027 1.689189 21.45270
#7 Blond Row Pct 74.015748 5.511811 12.5984252 7.874016 NA
#8 Blond Col Pct 43.720930 3.181818 25.000000 10.752688 NA
#9 Brown Frequency 84.000000 119.000000 29.000000 54.000000 286.00000
#10 Brown Percent 14.189189 20.101351 4.8986486 9.121622 48.31081
#11 Brown Row Pct 29.370629 41.608392 10.1398601 18.881119 NA
#12 Brown Col Pct 39.069767 54.090909 45.3125000 58.064516 NA
#13 Red Frequency 17.000000 26.000000 14.000000 14.000000 71.00000
#14 Red Percent 2.871622 4.391892 2.3648649 2.364865 11.99324
#15 Red Row Pct 23.943662 36.619718 19.7183099 19.718310 NA
#16 Red Col Pct 7.906977 11.818182 21.8750000 15.053763 NA
#17 Total Frequency 215.000000 220.000000 64.000000 93.000000 592.00000
#18 Total Percent 36.317568 37.162162 10.8108108 15.709459 100.00000

# $`chisq:Hair * Eye`
# CHISQ CHISQ.DF CHISQ.P
# 1 138.2898 9 2.325287e-25

#' # Example #3: By variable with named table request
res <- proc_freq(df, tables = v(Hair, Eye, Cross = Hair * Eye),
                by = Sex,
                weight = Freq)

# View result data
res
# $Hair
# BY VAR CAT N CNT PCT
# 1 Female Hair Black 313 52 16.61342

```

```

# 2 Female Hair Blond 313 81 25.87859
# 3 Female Hair Brown 313 143 45.68690
# 4 Female Hair Red 313 37 11.82109
# 5 Male Hair Black 279 56 20.07168
# 6 Male Hair Blond 279 46 16.48746
# 7 Male Hair Brown 279 143 51.25448
# 8 Male Hair Red 279 34 12.18638
#
# $Eye
# BY VAR CAT N CNT PCT
# 1 Female Eye Blue 313 114 36.421725
# 2 Female Eye Brown 313 122 38.977636
# 3 Female Eye Green 313 31 9.904153
# 4 Female Eye Hazel 313 46 14.696486
# 5 Male Eye Blue 279 101 36.200717
# 6 Male Eye Brown 279 98 35.125448
# 7 Male Eye Green 279 33 11.827957
# 8 Male Eye Hazel 279 47 16.845878
#
# $Cross
# BY VAR1 VAR2 CAT1 CAT2 N CNT PCT
# 1 Female Hair Eye Black Blue 313 9 2.8753994
# 2 Female Hair Eye Black Brown 313 36 11.5015974
# 3 Female Hair Eye Black Green 313 2 0.6389776
# 4 Female Hair Eye Black Hazel 313 5 1.5974441
# 5 Female Hair Eye Blond Blue 313 64 20.4472843
# 6 Female Hair Eye Blond Brown 313 4 1.2779553
# 7 Female Hair Eye Blond Green 313 8 2.5559105
# 8 Female Hair Eye Blond Hazel 313 5 1.5974441
# 9 Female Hair Eye Brown Blue 313 34 10.8626198
# 10 Female Hair Eye Brown Brown 313 66 21.0862620
# 11 Female Hair Eye Brown Green 313 14 4.4728435
# 12 Female Hair Eye Brown Hazel 313 29 9.2651757
# 13 Female Hair Eye Red Blue 313 7 2.2364217
# 14 Female Hair Eye Red Brown 313 16 5.1118211
# 15 Female Hair Eye Red Green 313 7 2.2364217
# 16 Female Hair Eye Red Hazel 313 7 2.2364217
# 17 Male Hair Eye Black Blue 279 11 3.9426523
# 18 Male Hair Eye Black Brown 279 32 11.4695341
# 19 Male Hair Eye Black Green 279 3 1.0752688
# 20 Male Hair Eye Black Hazel 279 10 3.5842294
# 21 Male Hair Eye Blond Blue 279 30 10.7526882
# 22 Male Hair Eye Blond Brown 279 3 1.0752688
# 23 Male Hair Eye Blond Green 279 8 2.8673835
# 24 Male Hair Eye Blond Hazel 279 5 1.7921147
# 25 Male Hair Eye Brown Blue 279 50 17.9211470
# 26 Male Hair Eye Brown Brown 279 53 18.9964158
# 27 Male Hair Eye Brown Green 279 15 5.3763441
# 28 Male Hair Eye Brown Hazel 279 25 8.9605735
# 29 Male Hair Eye Red Blue 279 10 3.5842294
# 30 Male Hair Eye Red Brown 279 10 3.5842294
# 31 Male Hair Eye Red Green 279 7 2.5089606
# 32 Male Hair Eye Red Hazel 279 7 2.5089606

```

proc_means	<i>Calculates Summary Statistics</i>
------------	--------------------------------------

Description

The `proc_means` function generates summary statistics for selected variables on the input dataset. The variables are identified on the `var` parameter. The statistics to perform are identified on the `stats` parameter. Results are displayed in the viewer interactively and returned from the function.

Usage

```
proc_means(
  data,
  var = NULL,
  stats = c("n", "mean", "std", "min", "max"),
  output = NULL,
  by = NULL,
  class = NULL,
  options = NULL,
  titles = NULL
)
```

Arguments

<code>data</code>	The input data frame for which to calculate summary statistics. This parameter is required.
<code>var</code>	The variable(s) to calculate summary statistics for. If no variables are specified, summary statistics will be generated for all numeric variables on the input data frame.
<code>stats</code>	A vector of summary statistics keywords. Valid keywords are: "css", "clm", "cv", "kurt", "kurtosis", "lclm", "mean", "median", "mode", "min", "max", "n", "nmiss", "nobs", "p1", "p5", "p10", "p20", "p25", "p30", "p40", "p50", "p60", "p70", "p75", "p80", "p90", "p95", "p99", "q1", "q3", "qrange", "range", "skew", "skewness", "std", "stddev", "stderr", "sum", "uclm", "uss", and "vari". For hypothesis testing, the function supports "t", "prt", "probt", and "df". Default statistics are: "n", "mean", "std", "min", and "max".
<code>output</code>	Whether or not to return datasets from the function. Valid values are "out", "none", and "report". Default is "out", and will produce dataset output specifically designed for programmatic use. The "none" option will return a NULL instead of a dataset or list of datasets. The "report" keyword returns the datasets from the interactive report, which may be different from the standard output. The output parameter also accepts data shaping keywords "long", "stacked", and "wide". The keywords are shaping options that control the structure of the output data. See the Data Shaping section for additional details. Note that multiple output keywords may be passed on a character vector. For example, to produce both a report dataset and a "long" output dataset, use the parameter <code>output = c("report", "out", "long")</code> .

by	An optional by group. If you specify an by group, the input data will be subset on the by variable(s) prior to performing any statistics.
class	The class parameter is similar to the by parameter, but the output is different. By groups will create completely separate tables, while class groups will be continued in the same table. When a by and a class are both specified, the class will be nested in the by.
options	A vector of optional keywords. Valid values are: "alpha =", "completetypes", "maxdec =", "noprint", "notype", "nofreq", "nonobs". The "notype", "nofreq", and "nonobs" keywords will turn off columns on the output datasets. The "alpha =" option will set the alpha value for confidence limit statistics. The default is 95% (alpha = 0.05). The "maxdec =" option sets the maximum number of decimal places displayed on report output.
titles	A vector of one or more titles to use for the report output.

Details

The `proc_means` function is for analysis of continuous variables. Data is passed in on the data parameter. The desired statistics are specified using keywords on the `stats` parameter. The function can segregate data into groups using the `by` and `class` parameters. There are also options to determine whether and what results are returned.

Value

Normally, the requested summary statistics are shown interactively in the viewer, and output results are returned as a data frame. If the request produces multiple data frames, they will be returned in a list. You may then access individual datasets from the list. The interactive report can be turned off using the "noprint" option, and the output datasets can be turned off using the "none" keyword on the output parameter.

Interactive Output

By default, `proc_freq` results will be immediately sent to the viewer as an HTML report. This functionality makes it easy to get a quick analysis of your data. To turn off the interactive report, pass the "noprint" keyword to the `options` parameter.

The `titles` parameter allows you to set one or more titles for your report. Pass these titles as a vector of strings.

The exact datasets used for the interactive report can be returned as a list. To return these datasets as a list, pass the "report" keyword on the output parameter. This list may in turn be passed to [proc_print](#) to write the report to a file.

Dataset Output

Dataset results are also returned from the function by default. If the results are a single dataset, a single data frame will be returned. If there are multiple results, a list of data frames will be returned.

The output datasets generated are optimized for data manipulation. The column names have been standardized, and additional variables may be present to help with data manipulation. For example, the by variable will always be named "BY", and the class variable will always be named "CLASS".

In addition, data values in the output datasets are intentionally not rounded or formatted to give you the most accurate statistical results.

Statistics Keywords

The following statistics keywords can be passed on the `stats` parameter. Normally, each statistic will be contained in a separate column and the column name will be the same as the statistic keyword. You may pass statistic keywords as a quoted vector of strings, or an unquoted vector using the `v()` function.

- **css**: Corrected Sum of Squares.
- **clm, lclm, uclm**: Upper and lower confidence limits.
- **cv**: Coefficient of Variation.
- **kurt/kurtosis**: The Kurtosis is a description of the distribution tails. It requires at least 4 complete observations.
- **mean**: The arithmetic mean.
- **median**: The median.
- **mode**: The mode of the target variable.
- **min, max**: The minimum and maximum values of the target variable.
- **n**: The number of non-missing observations.
- **nmiss**: The number of missing observations.
- **nobs**: The number of observations, whether missing or non-missing.
- **p1 - p99**: Percentile ranges from p1 to p99, in increments of 5.
- **qrange, q1, q3**: Quantile ranges for the first and third quantiles.
- **range**: Difference between the minimum and maximum values.
- **skew/skewness**: A measure of distribution skewness. It requires at least 3 complete observations.
- **std/stddev**: Standard deviation.
- **stderr**: Standard error.
- **sum**: The sum of variable values.
- **uss**: Uncorrected sum of squares.
- **vari**: The variance.

The function supports the following keywords to perform hypothesis testing:

- **t**: Student's t statistic.
- **prrt/probt**: A two-tailed p-value for the Student's t statistic.
- **df**: Degrees of freedom for the Student's t statistic.

Options

The `proc_means` function recognizes the following options. Options may be passed as a quoted vector of strings, or an unquoted vector using the `v()` function.

- **alpha = :** The "alpha = " option will set the alpha value for confidence limit statistics. Set the alpha as a decimal value between 0 and 1. For example, you can set a 90% confidence limit as `alpha = 0.1`.
- **completetypes:** The "completetypes" option will generate all combinations of the class variable, even if there is no data present for a particular level. Combinations will be distinguished by the TYPE variable. To use the "completetypes" option, define the class variable(s) as a factor.
- **maxdec = :** The "maxdec = " option will set the maximum of decimal places displayed on report output. For example, you can set 4 decimal places as follows: `maxdec = 4`. Default is 7 decimal places. This option will not round any values on the output dataset.
- **nofreq, nonobs:** Turns off the FREQ column on the output datasets.
- **noprint:** Whether to print the interactive report to the viewer. By default, the report is printed to the viewer. The "noprint" option will inhibit printing.
- **notype:** Turns off the TYPE column on the output dataset.
- **nway:** Returns only the highest level TYPE combination. By default, the function return all TYPE combinations.

TYPE and FREQ Variables

The TYPE and FREQ variables appear on the output dataset by default.

The FREQ variable contains a count of the number of input rows/observations that were included in the statistics for that output row. The FREQ count can be different from the N statistic. The FREQ count is a count of the number of rows/observations, while the N statistic is a count of non-missing values. These counts can be different if you have missing values in your data. If you want to remove the FREQ column from the output dataset, use the "nofreq" option.

The TYPE variable identifies combinations of class categories, and produces summary statistics for each of those combinations. For example, the output dataset normally produces statistics for TYPE 0, which is all class categories, and a TYPE 1 which is each class category. If there are multiple classes, there will be multiple TYPE values for each level of class combinations. If you do not want to show the various type combinations, use the "nway" option. If you want to remove the TYPE column from the output dataset, use the "notype" option.

Using Factors

There are some occasions when you may want to define the class variable(s) as a factor. One occasion is for sorting/ordering, and the other is for obtaining zero-counts on sparse data.

To order the class categories in the means output, define the class variable as a factor in the desired order. The function will then retain that order for the class categories in the output dataset and report.

You may also wish to define the class variable as a factor if you are dealing with sparse data and some of the class categories are not present in the data. To ensure these categories are displayed with zero-counts, define the class variable as a factor and use the "completetypes" option.

Data Shaping

The output dataset produced by the "out" keyword can be shaped in different ways. These shaping options allow you to decide whether the data should be returned long and skinny, or short and wide. The shaping options can reduce the amount of data manipulation necessary to get the frequencies into the desired form. The shaping options are as follows:

- **long**: Transposes the output datasets so that statistics are in rows and variables are in columns.
- **stacked**: Requests that output datasets be returned in "stacked" form, such that both statistics and variables are in rows.
- **wide**: Requests that output datasets be returned in "wide" form, such that statistics are across the top in columns, and variables are in rows. This shaping option is the default.

Examples

```
# Turn off printing for CRAN checks
options("procs.print" = FALSE)

# Default statistics on iris
res1 <- proc_means(iris)

# View results
res1
#   TYPE FREQ          VAR  N    MEAN      STD MIN MAX
# 1    0   150 Sepal.Length 150 5.843333 0.8280661 4.3 7.9
# 2    0   150 Sepal.Width 150 3.057333 0.4358663 2.0 4.4
# 3    0   150 Petal.Length 150 3.758000 1.7652982 1.0 6.9
# 4    0   150 Petal.Width 150 1.199333 0.7622377 0.1 2.5

# Defaults statistics with by
res2 <- proc_means(iris,
                   by = Species)

# View results
res2
#   BY TYPE FREQ          VAR  N    MEAN      STD MIN MAX
# 1   setosa    0   50 Sepal.Length 50 5.006 0.3524897 4.3 5.8
# 2   setosa    0   50 Sepal.Width 50 3.428 0.3790644 2.3 4.4
# 3   setosa    0   50 Petal.Length 50 1.462 0.1736640 1.0 1.9
# 4   setosa    0   50 Petal.Width 50 0.246 0.1053856 0.1 0.6
# 5 versicolor 0   50 Sepal.Length 50 5.936 0.5161711 4.9 7.0
# 6 versicolor 0   50 Sepal.Width 50 2.770 0.3137983 2.0 3.4
# 7 versicolor 0   50 Petal.Length 50 4.260 0.4699110 3.0 5.1
# 8 versicolor 0   50 Petal.Width 50 1.326 0.1977527 1.0 1.8
# 9 virginica  0   50 Sepal.Length 50 6.588 0.6358796 4.9 7.9
# 10 virginica 0   50 Sepal.Width 50 2.974 0.3224966 2.2 3.8
# 11 virginica 0   50 Petal.Length 50 5.552 0.5518947 4.5 6.9
# 12 virginica 0   50 Petal.Width 50 2.026 0.2746501 1.4 2.5

# Specified variables, statistics, and options
res3 <- proc_means(iris,
                  var = v(Petal.Length, Petal.Width),
                  class = Species,
```

```

stats = v(n, mean, std, median, qrange, clm),
options = nofreq,
output = long)

# View results
res3
#      CLASS TYPE  STAT Petal.Length Petal.Width
# 1      <NA>  0     N    150.0000000 150.0000000
# 2      <NA>  0  MEAN     3.7580000  1.1993333
# 3      <NA>  0   STD     1.7652982  0.7622377
# 4      <NA>  0 MEDIAN     4.3500000  1.3000000
# 5      <NA>  0 QRANGE     3.5000000  1.5000000
# 6      <NA>  0  LCLM     3.4731854  1.0763533
# 7      <NA>  0  UCLM     4.0428146  1.3223134
# 8      setosa  1     N    50.0000000 50.0000000
# 9      setosa  1  MEAN     1.4620000  0.2460000
# 10     setosa  1   STD     0.1736640  0.1053856
# 11     setosa  1 MEDIAN     1.5000000  0.2000000
# 12     setosa  1 QRANGE     0.2000000  0.1000000
# 13     setosa  1  LCLM     1.4126452  0.2160497
# 14     setosa  1  UCLM     1.5113548  0.2759503
# 15 versicolor  1     N    50.0000000 50.0000000
# 16 versicolor  1  MEAN     4.2600000  1.3260000
# 17 versicolor  1   STD     0.4699110  0.1977527
# 18 versicolor  1 MEDIAN     4.3500000  1.3000000
# 19 versicolor  1 QRANGE     0.6000000  0.3000000
# 20 versicolor  1  LCLM     4.1264528  1.2697993
# 21 versicolor  1  UCLM     4.3935472  1.3822007
# 22 virginica  1     N    50.0000000 50.0000000
# 23 virginica  1  MEAN     5.5520000  2.0260000
# 24 virginica  1   STD     0.5518947  0.2746501
# 25 virginica  1 MEDIAN     5.5500000  2.0000000
# 26 virginica  1 QRANGE     0.8000000  0.5000000
# 27 virginica  1  LCLM     5.3951533  1.9479453
# 28 virginica  1  UCLM     5.7088467  2.1040547

```

```
proc_print
```

```
Prints a dataset
```

Description

The `proc_print` function is used to print a dataset or datasets. To print multiple datasets, pass them to the `data` parameter in a list. By default, the function prints to the viewer. It may also be used to print to the file system using the `output_type` and `file_path` parameters. This print function has limited options, and is meant to quickly view your data or dump it out to a file. For more reporting options, use the [reporter](#) package.

Usage

```
proc_print(
```

```

    data,
    file_path = NULL,
    output_type = "HTML",
    titles = NULL,
    style = NULL,
    view = TRUE
  )

```

Arguments

data	The data to print. Can be either a single dataset, or a list of datasets.
file_path	The path of the report to print.
output_type	The type of report to create. Valid values are "TXT", "RTF", "PDF", "HTML", and "DOCX". Default is "HTML".
titles	A vector of titles.
style	A style object, as defined by the reporter package. See that package for details.
view	Whether to send the print output to the viewer. Valid values are TRUE and FALSE. Default is TRUE.

Value

If a file report was produced, the full path of the report. Otherwise, a NULL. In either case, the value will be returned invisibly.

Examples

```

# Turn off printing to pass CRAN checks
options("procs.print" = FALSE)

# Print mtcars to the viewer
proc_print(mtcars)

# Print mtcars to an RTF
pth <- proc_print(mtcars,
  file_path = tempfile(fileext = ".rtf"),
  titles = "MTCARS Proc Print Example",
  output_type = "RTF", view = FALSE)

# View file
# file.show(pth)

```

Description

The `proc_sort` function sorts a dataset according to the variables passed on the `by` parameter. If no parameters are passed on the `by` parameter, it will sort by all variables. The direction of the sort is controlled with the `order` parameter. Use the `nodupkey` option to eliminate duplicate rows from the dataset, and the `keep` parameter to subset columns. The parameters will accept either quoted or unquoted values.

Usage

```
proc_sort(data, by = NULL, keep = NULL, order = "ascending", options = NULL)
```

Arguments

<code>data</code>	The input data to sort.
<code>by</code>	A vector of variables to sort by.
<code>keep</code>	A vector of variables on the output data to keep. All other variables will be dropped.
<code>order</code>	The sort order of the variables on the <code>by</code> parameter. Valid values are 'ascending' or 'descending'. These values may also be abbreviated to 'asc', 'desc', 'a', or 'd'. You may pass a vector of order values equal to the number of variables on the <code>by</code> parameter. Default is 'ascending' for all by variables.
<code>options</code>	Any options desired for the sort. Available options are 'dupkey' and 'nodupkey'. The 'nodupkey' option removes duplicate rows from the sorted dataset. The 'dupkey' option removes unique rows from the sorted dataset.

Value

The sorted dataset. If a data frame was input, a data frame will be output. If a tibble was input, a tibble will be output.

Options

Below are the available options for the `proc_sort` function:

- **dupkey**: This option keeps duplicate rows and discards unique rows. Duplicate rows will be identified by the key variables listed on the `by` parameter if passed. This option is the opposite of 'nodupkey'.
- **nodupkey**: Removes duplicate rows following the sort. Duplicate rows will be identified by the key variables listed on the `by` parameter if passed. Otherwise, the function will dedupe on all variables returned.

Examples

```
# Prepare data subset
dat <- data.frame(HairEyeColor, stringsAsFactors = FALSE)[1:32 %% 4 == 1, ]

# View data
dat
```



```

#   Hair   Eye   Sex Freq
# 1 Black Brown  Male  32
# 5 Black  Blue  Male  11
# 9 Black Hazel  Male  10
#13 Black Green  Male   3
#17 Black Brown Female 36
#21 Black  Blue Female  9
#25 Black Hazel Female  5
#29 Black Green Female  2

# Sort by Frequency
res1 <- proc_sort(dat, by = Freq)

# View results
res1
#   Hair   Eye   Sex Freq
#29 Black Green Female   2
#13 Black Green  Male   3
#25 Black Hazel Female   5
#21 Black  Blue Female   9
# 9 Black Hazel  Male  10
# 5 Black  Blue  Male  11
# 1 Black Brown  Male  32
#17 Black Brown Female  36

# Sort by Frequency descending
res2 <- proc_sort(dat, by = Freq, order = d)

# View results
res2
#   Hair   Eye   Sex Freq
#17 Black Brown Female  36
# 1 Black Brown  Male  32
# 5 Black  Blue  Male  11
# 9 Black Hazel  Male  10
#21 Black  Blue Female   9
#25 Black Hazel Female   5
#13 Black Green  Male   3
#29 Black Green Female   2

# Get unique combinations of Eye and Sex
res3 <- proc_sort(dat, keep = v(Eye, Sex), options = nodupkey)

# View results
res3
#   Eye   Sex
# 1 Brown  Male
#17 Brown Female
# 5 Blue   Male
#21 Blue  Female
# 9 Hazel  Male
#25 Hazel Female
#13 Green  Male

```

```
# 29 Green Female
```

```
proc_transpose      Transposes a Dataset
```

Description

A function to pivot or transpose a data frame. In the default usage, the variables identified by the parameter `var` are transposed to become rows. The variable values in the parameter `id` become the new columns. The function has several more parameters to control how variables are named in the transposed data set. Parameters will accept quoted or unquoted values.

Usage

```
proc_transpose(  
  data,  
  by = NULL,  
  var = NULL,  
  id = NULL,  
  idlabel = NULL,  
  copy = NULL,  
  name = "NAME",  
  namelabel = NULL,  
  prefix = NULL,  
  delimiter = ".",  
  suffix = NULL,  
  where = NULL,  
  log = TRUE  
)
```

Arguments

<code>data</code>	The input data to transpose.
<code>by</code>	An optional <code>by</code> group. Parameter accepts a vector of one or more quoted variable names. If the <code>by</code> group is requested, the data will be subset by that variable and the transpose function will transpose each group and stack them together in a single table.
<code>var</code>	The variable or variables to transpose. Parameter accepts a vector of variable names. By default, all numeric variables will be transposed.
<code>id</code>	The variable or variables to use for the transposed column names.
<code>idlabel</code>	The variable to use for the transposed column labels.
<code>copy</code>	A vector of variables to retain in the output data without transposition. Values will be truncated or recycled to fit the number of output rows.
<code>name</code>	Specifies the name of the variable to be used for the <code>var</code> values.
<code>namelabel</code>	The label to use for the name variable.

prefix	Contains a prefix to be used in the construction of column names.
delimiter	Specifies a delimiter to be used in the construction of column names.
suffix	Contains a suffix to be used in the construction of column names.
where	An expression to filter the rows after the transform is complete. Use the expression function to define the where clause.
log	Whether or not to log the procedure. Default is TRUE. This parameter is used internally.

Details

The `proc_transpose` function takes an input data frame or tibble and transposes the columns and rows. If no parameters are specified, the function will assign all numeric variables to the `var` parameter. These variables will become rows, and generic column names ("COL1", "COL2", etc.) will be generated. Other variables will be dropped.

There are several parameters to control how new column names are constructed. If the desired column names already exist in your data, identify them on the `id` parameter. The function will then use those data values unaltered. The label for these new columns can also be constructed from data values using the `idLabel` parameter.

The `name` and `nameLabel` parameter are used to control the name of the column created for the `var` values. If this parameter is not passed, the column will be called "NAME", and no label will be assigned.

You may group the transposed values using the `by` parameter. This parameter accepts one or more variable names to use for grouping. If this parameter is used, the function will first subset the data by the unique combination of `by` variables, transpose each subset, and then combine the result into a single output dataset. The `by` group variables will be named on the output dataset with a generic name ("BY1", "BY2", etc.).

The `copy` parameter is used to simply copy columns from the input dataset to the transposed dataset. If necessary, these values will be recycled or truncated to fit the number of output rows. Any input variables not included in the `var`, `id`, or `copy` parameter will be dropped.

Once the transpose is complete, you may wish to filter the output data. Filtering can be accomplished using the `where` parameter. This parameter takes an expression using the `expression` function. The expression is constructed using standard R logical operators. Variable names do not need to be quoted.

The `prefix`, `delimiter`, and `suffix` parameter are used to control how generic column names are constructed. These parameters are especially useful when there are multiple `var` variables.

Value

The transposed dataset. If a data frame is input, a data frame will be output. If a tibble is input, a tibble will be output.

Examples

```
# Prepare data
dat <- data.frame(CAT = rownames(USPersonalExpenditure),
                 USPersonalExpenditure, stringsAsFactors = FALSE,
```

```

row.names = NULL)[1:4, ]

# View data
dat
#
# CAT X1940 X1945 X1950 X1955 X1960
# 1 Food and Tobacco 22.20 44.50 59.60 73.2 86.8
# 2 Household Operation 10.50 15.50 29.00 36.5 46.2
# 3 Medical and Health 3.53 5.76 9.71 14.0 21.1
# 4 Personal Care 1.04 1.98 2.45 3.4 5.4

# Default transpose
tdat1 <- proc_transpose(dat)

# View results
tdat1
# NAME COL1 COL2 COL3 COL4
# 1 X1940 22.2 10.5 3.53 1.04
# 2 X1945 44.5 15.5 5.76 1.98
# 3 X1950 59.6 29.0 9.71 2.45
# 4 X1955 73.2 36.5 14.00 3.40
# 5 X1960 86.8 46.2 21.10 5.40

# Transpose with ID and Name
tdat2 <- proc_transpose(dat, id = CAT, name = Year)

# View results
tdat2
# Year Food and Tobacco Household Operation Medical and Health Personal Care
# 1 X1940 22.2 10.5 3.53 1.04
# 2 X1945 44.5 15.5 5.76 1.98
# 3 X1950 59.6 29.0 9.71 2.45
# 4 X1955 73.2 36.5 14.00 3.40
# 5 X1960 86.8 46.2 21.10 5.40

# Transpose only some of the variables
tdat3 <- proc_transpose(dat, var = v(X1940, X1950, X1960), id = CAT, name = Year)

# View results
tdat3
# Year Food and Tobacco Household Operation Medical and Health Personal Care
# 1 X1940 22.2 10.5 3.53 1.04
# 2 X1950 59.6 29.0 9.71 2.45
# 3 X1960 86.8 46.2 21.10 5.40

# By with a where clause
tdat4 <- proc_transpose(dat, by = CAT, name = Year,
  where = expression(Year %in% c("X1940", "X1950", "X1960")))

# View Results
tdat4
# CAT Year COL1
# 1 Food and Tobacco X1940 22.20
# 2 Food and Tobacco X1950 59.60

```

# 3	Food and Tobacco	X1960	86.80
# 4	Household Operation	X1940	10.50
# 5	Household Operation	X1950	29.00
# 6	Household Operation	X1960	46.20
# 7	Medical and Health	X1940	3.53
# 8	Medical and Health	X1950	9.71
# 9	Medical and Health	X1960	21.10
# 10	Personal Care	X1940	1.04
# 11	Personal Care	X1950	2.45
# 12	Personal Care	X1960	5.40

Index

expression, [19](#)

proc_freq, [2](#)

proc_means, [6](#), [9](#)

proc_print, [3](#), [10](#), [14](#)

proc_sort, [15](#)

proc_transpose, [6](#), [18](#)

reporter, [14](#), [15](#)