Package 'freqweights'

October 9, 2014

Type Package

Title Working with frequency tables

Version 1.0.1

Date 2014-08-04

Author Emilio Torres-Manzanera

Maintainer Emilio Torres-Manzanera <torres@uniovi.es>

Description The frequency of a particular data value is the number of times it occurs. A frequency table is a table of values with their corresponding frequencies. Frequency weights are integer numbers that indicate how many cases each case represents. This package provides some functions to work with such type of collected data.

License GPL-3

Imports plyr, dplyr (>= 0.3), data.table, biglm, fastcluster,FactoMineR, stats

Suggests MASS, hflights, cluster, ggplot2, testthat, RSQLite

NeedsCompilation no

Repository CRAN

Date/Publication 2014-10-09 14:39:04

R topics documented:

`freqweights-package` *Working with frequency tables*

Description

The frequency of a particular data value is the number of times it occurs. A frequency table is a table of values with their corresponding frequencies. Frequency weights are integer numbers that indicate how many cases each case represents. This package provides some functions to work with such type of collected data.

Details

Package:	freqweights
Type:	Package
Version:	0.1.0
Date:	2014-05-20
License:	GPL 3.0

Author(s)

Emilio Torres-Manzanera

Maintainer: Emilio Torres-Manzanera <torres@uniovi.es>

Examples

```
tablefreq(iris)
lmfreq(Sepal.Length ~ Petal.Length, tablefreq(iris))
hclustvfreq(tablefreq(iris[,1:4]))
```

`biglmfreq` *Estimates the coefficients of a linear model*

Description

Estimates the coefficients of a linear model following the guidelines of `biglm`

Usage

```
biglmfreq(formula, data, freq = NULL)

## S3 method for class 'biglmfreq'
coef(object, ...)

## S3 method for class 'biglmfreq'
predict(object, ...)

## S3 method for class 'biglmfreq'
print(x, ...)

## S3 method for class 'biglmfreq'
update(object, ...)
```

Arguments

formula	a model formula
data	data frame that must contain all variables in formula and freq
freq	a string of the variable specifying frequency weights
object	a biglmfreq object
...	See Details
x	a biglmfreq object

Details

Any variables in the formula are removed from the data set.

It only computes the coefficients of the linear model.

... should be a data frame when predict. See Examples

... should be a data frame when update. See Examples

Value

A biglmfreq object.

See Also

biglm, make.readchunk

Examples

```
mt <- biglmfreq(Sepal.Length ~ Sepal.Width, iris)
coef(mt)

chunk1 <- iris[1:30,]
chunk2 <- iris[-c(1:30),]
mf1 <- biglmfreq(Sepal.Length ~ Sepal.Width, chunk1)
```

```
mf2 <- update(mf1, chunk2)

predict(mf2, iris)
```

`clarachunk` *Clustering Large Chunks*

Description

Clustering data splitted in several chunks into k clusters.

Usage

```
clarasub(x, k, samples = 50)

claramerge(subclusters, k, samples = 50)
```

Arguments

`x`	data matrix or data frame, each row corresponds to an observation, and each column corresponds to a variable. All variables must be numeric. Missing values (NAs) are allowed.
`k`	integer, the number of clusters. It is required that $0 < k < n$ where n is the number of observations of each chunk (i.e., n = nrow(x)).
`samples`	integer, number of samples to be drawn from the dataset.
`subclusters`	list of objects returned by clarasub

Details

See `clara` for further details.

See Examples.

Value

A list with the following values (see `clara`):

`n`	number of rows of the data set.
`sample`	labels or case numbers of the observations in the best sample, that is, the sample used by the clara algorithm for the final partition.
`medoids`	the medoids or representative objects of the clusters. It is a matrix with in each row the coordinates of one medoid.
`tablefreq`	a table of frequency. It is an approximation to the number of cases in each group.

Note

This function is based on `clara` that it is not available on Windows. Therefore, this implementation does not run on Windows.

References

Antonio Piccolboni mclust.mr https://github.com/RevolutionAnalytics/rmr2/blob/master/pkg/examples/mclust.mr.R

See Also

clara, make.readchunk

Examples

```
if(require(cluster)){
  k <- 3

  chunk1 <- iris[1:30,1:4]
  clus1 <- clarasub(chunk1,k)

  chunk2 <- iris[-c(1:30),1:4]
  clus2 <- clarasub(chunk2,k)

  subclusters <- list(clus1, clus2)
  b <- claramerge(subclusters,k)
     print(b$medoids)

   print(nrow(b$tablefreq))
  print(b$tablefreq)
}
```

---

evaldp *Eval a manip function using a string*

Description

Eval a manip function using a string

Usage

```
evaldp(.data, .fun.name, ..., .envir = parent.frame())
```

Arguments

.data	a tbl
.fun.name	any manip function
...	string arguments
.envir	environment

Details

Useful for programming with dplyr

Note

It is possible that in the next release of dplyr these functionalities would appear. Then they will be removed from this package.

References

https://gist.github.com/skranz/9681509

Examples

```
library(dplyr)
iris %>% evaldp(arrange,"Sepal.Length") %>%
         evaldp(filter, "Sepal.Length > 5, Species=='virginica'")
```

`hclustvfreq` *Fast hierarchical, agglomerative clustering of frequency data*

Description

This function implements a version of the hierarchical, agglomerative clustering `hclust.vector` focused on table of frequencies.

Usage

```
hclustvfreq(data, freq = NULL, method = "single", metric = "euclidean",
  p = NULL)

.hclustvfreq(tfq, method = "single", metric = "euclidean", p = NULL)
```

Arguments

`data`	any object that can be coerced into a double matrix
`freq`	a one-sided, single term formula specifying frequency weights
`method`	the agglomeration method to be used. This must be (an unambiguous abbreviation of) one of `"single"`, `"ward"`, `"centroid"` or `"median"`.
`metric`	the distance measure to be used. This must be one of `"euclidean"`, `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"` or `"minkowski"`
`p`	parameter for the Minkowski metric.
`tfq`	a frequency table

Details

Any variables in the formula are removed from the data set.

This function is a wrapper of `hclust.vector` to be used with tables of frequencies. It use the frequency weights as parameter `members`.

See Also

hclust.vector, link{tablefreq}

Examples

```
library(dplyr)
library(fastcluster)

data <- iris[,1:3,drop=FALSE]
hc <- hclustvfreq(data, method="centroid",metric="euclidean")
cutree(hc,3) ## Different length than data

tfq <- tablefreq(iris[,1:3])
hc <- .hclustvfreq(tfq, method="centroid",metric="euclidean")
tfq$group <- cutree(hc,3)
```

---

lmfreq lmfreq *is used to fit linear models with frequency tables*

Description

To fit linear models with data grouped in frequency tables.

Usage

```
lmfreq(formula, data, freq = NULL)

.lmfreq(formula, tfq)

## S3 method for class 'lmfreq'
logLik(object, ...)

## S3 method for class 'lmfreq'
extractAIC(fit, scale = 0, k = 2, ...)

## S3 method for class 'lmfreq'
AIC(object, ..., k = 2)

## S3 method for class 'lmfreq'
nobs(object, ...)

## S3 method for class 'lmfreq'
summary(object, ...)

## S3 method for class 'lmfreq'
print(x, ...)
```

```
## S3 method for class 'summary.lmfreq'
print(x, digits = getOption("digits") - 3, ...)

## S3 method for class 'lmfreq'
predict(object, ...)
```

Arguments

formula	an object of class formula
data	a data frame that must contain all variables present in formula and freq
freq	a character string specifying the variable of frequency weights
tfq	a tablefreq object
object	a lmfreq object
...	See Details
fit	a lmfreq object
scale	not used
k	penalty parameter
x	a lmfreq object
digits	digits

Details

It computes the linear model of a frequency table. See lm for further details.

Any variables in the formula are removed from the data set.

The dot function are for programming purpose. It does not check the data.

Value

It returns an object of class lmfreq, very similar to lm

See Also

tablefreq

Examples

```
## Benchmark
if(require(hflights)){
  formula <-  ArrDelay ~ DepDelay
  print(system.time(a <- lm(formula, data=hflights)))  ## ~0.4 seconds
  print(system.time(b <- lmfreq(formula, data=hflights))) ## ~0.12 seconds. 4x faster
}

l0 <- lm(Sepal.Length ~ Sepal.Width,iris)
summary(l0)

tfq <- tablefreq(iris[,1:2])
```

```
lf <- lmfreq(Sepal.Length ~ Sepal.Width,tfq, freq="freq")
summary(lf)

all.equal(coef(lf),coef(l0))
all.equal(AIC(lf),AIC(l0))

newdata <- data.frame(Sepal.Width=c(1,NA,7))
predict(lf, newdata)

if(require(MASS)){
   stepAIC(lf)
}

system.time(lmfreq(Sepal.Length ~ Sepal.Width,tfq, freq="freq"))
system.time(.lmfreq(Sepal.Length ~ Sepal.Width,tfq)) # Fast

library(dplyr)
igrouped <- iris %>% group_by(Species)
models <- igrouped %>% do(model=lmfreq(Sepal.Length ~ Sepal.Width, .))
coefs <- models %>%
  do(cbind(as.data.frame(rbind(coef(.$model))),
           Species=.$Species))
coefs

## Not run:
## If data is too granular, benchmark is worst
n <- 10^6
data <- data.frame(y=rnorm(n),x=rnorm(n))
system.time(lm(y~x,data)) ## ~5 seconds
system.time(lmfreq(y~x,data)) ## ~ 15 seconds
system.time(tfq <- tablefreq(data)) ## ~ 5 seconds
nrow(tfq) # same number of rows than original data
system.time(.lmfreq(y~x,tfq)) ## ~ 10 seconds

## End(Not run)
```

`make.readchunk` *Fast and friendly chunk file finagler*

Description

Read a file chunk by chunk

Usage

```
make.readchunk(input, FUN = identity, chunksize = 5000L)
```

Arguments

input	a length 1 character string. See Details.
FUN	any function applicated to each chunk
chunksize	number of lines for each chunk

Details

It creates a function that reads sucesive chunks of the data referenced by input usings the fread function. The input is characterized in the help page of fread. The data contained in the input reference should not have any header.

This function is inspired by the bigglm example.

Value

A function with an logical argument, reset. If this argument is TRUE, it indicates that the data should be reread from the beginning by subsequent calls. When it reads all the data, it automatically resets the file. This function returns the value of FUN applied to the chunk. By default, the chunk is returned as a tbl_df object.

See Also

bigglm, fread, tbl_df

Examples

```
## Not run:
library(hflights)
nrow(hflights) # Number of rows

## We create a file with no header
input <- "hflights.csv"
write.table(hflights,file=input,sep=",",
            row.names=FALSE,col.names=FALSE)

## Get the number of rows of each chunk
readchunk <- make.readchunk(input,FUN=function(x){NROW(x)})

a <- NULL
while(!is.null(b <- readchunk())) {
  if(is.null(a)) {
    a <- b
  } else {
    a <- a+b
  }
}
all.equal(a, nrow(hflights))

## It resets automatically the file
a <- NULL
while(!is.null(b <- readchunk())) {
```

```
  if(is.null(a)) {
    a <- b
  } else {
    a <- a+b
  }
}
all.equal(a, nrow(hflights))

## End(Not run)
```

---

pcafreq *Principal Component Analysis*

Description

It computes a principal component analysis with sumplementary quantitative and qualitative variables. It is wrapper of PCA.

Usage

```
pcafreq(data, freq = NULL, scale.unit = TRUE, ncp = 5, quantisup = NULL,
  qualisup = NULL, colw = NULL, graph = TRUE, axes = c(1, 2))

.pcafreq(tfq, scale.unit = TRUE, ncp = 5, quantisup = NULL,
  qualisup = NULL, colw = NULL, graph = TRUE, axes = c(1, 2))
```

Arguments

data	a data frame
freq	a name of the variable specifying frequency weights
scale.unit	a boolean, if TRUE (value set by default) then data are scaled to unit variance
ncp	number of dimensions kept in the results
quantisup	a vector indicating the names of the quantitative supplementary variables
qualisup	a vector indicating the names of the categorical supplementary variables
colw	an optional column weights (by default, uniform column weights)
graph	boolean, if TRUE a graph is displayed
axes	a length 2 vector specifying the components to plot
tfq	a table of frequencies

Details

This function calls PCA with the the frequency weights as row.w. Any variable present in freq are removed from the data.

Value

It returns a list described in PCA.

See Also

PCA, link{tablefreq}

Examples

```
pcafreq(iris,  qualisup="Species", graph=TRUE)

tfq <- tablefreq(iris)
.pcafreq(tfq,  qualisup="Species", graph=TRUE)
```

---

preprocesshflights *Preprocess the* hflights *data.*

Description

Preprocess the hflights data to get a nice format.

Usage

```
preprocesshflights(hflights)
```

Arguments

hflights	hflights data set

Details

Preprocess this data set.

Value

A preprocessed data set.

See Also

hflights

Examples

```
if(require(hflights)) {
a <- preprocesshflights(hflights[1:10000,])
summary(a)
}
## Not run:
library(hflights)
## We create a file with no header
input <- "hflights.csv"
write.table(hflights,file=input,sep=",",
            row.names=FALSE,col.names=FALSE)
## Inefficient way to read the data. Just as example
lines <- readLines(input)
lines <- gsub("\"","",lines,fixed=TRUE )
x <- strsplit(lines,",")
x <- as.data.frame(do.call("rbind",x))
x <- preprocesshflights(x)
summary(x)

## End(Not run)
```

`quickround` *Round data sets*

Description

A wrapper of `round_any` to round data sets to multiple of any number.

Usage

```
quickround(x, accuracy)
```

Arguments

`x`	a `tbl` object or a numeric or POSIXct matrix
`accuracy`	number to round to; for POSIXct objects, a number of seconds

Details

The `x` may contain non numerical varibles (factor, character, logical). They will remain unchanged. The numerical or time variables will be rounded to a multiple fo the accuracy.

If `accuracy` is of length 1, then this value is applied to all the columns of the data set. Otherwise, its length must be the same as the number of columns of `x`, including non numerical variables. If any value if `NA`, the corresponding variable will remain unchanged.

Value

A `tbl` object.

See Also

round_any

Examples

```
quickround(iris,0.2)
quickround(iris[,1:3],c(0.2,0.5,1.0))

tfq <- tablefreq(iris, vars=c("Sepal.Length","Species"))
a <- quickround(tfq, c(0.3, NA, NA))
b <- tablefreq(a, freq="freq")
b
```

---

statsfreq *Descriptive statistics of a frequency table.*

Description

Computes the descriptive statistics of a frequency table.

Usage

```
meanfreq(data, freq = NULL)

.meanfreq(tfq)

quantilefreq(data, probs = c(0, 0.25, 0.5, 0.75, 1), freq = NULL)

.quantilefreq(tfq, probs = c(0, 0.25, 0.5, 0.75, 1))

covfreq(data, freq = NULL)

.covfreq(tfq)

sdfreq(data, freq = NULL)

.sdfreq(tfq)

scalefreq(data, freq = NULL)

.scalefreq(tfq)

corfreq(data, freq = NULL)

.corfreq(tfq)
```

Arguments

data	any object that can be processed by link{tablefreq}.
freq	a single name of the variable specifying frequency weights.
tfq	a tablefreq object, or a matrix, data frame with the last column being the frequency wweights
probs	A vector of quantiles to compute. Default is 0 (min), .25, .5, .75, 1 (max).

Details

These functions compute various weighted versions of standard estimators.

meanfreq, sdfreq, quantilefreq, covfreq, corfreq estimate the mean, standard desviation, quantiles, covariances and correlation matrix, respectively. In this last two cases, resulst are equals to the pairwise.complete.obs option of cov and cor of the desaggregated data, respectively.

Missing values or cases with non-positive frequency weights are automatically removed.

If freq is not null, the data set must contain a column with that name. These variable are removed from the data set in order to calculate the descriptive statistics.

The dot versions are intented to be used when programing. The tfq may be a tablefreq object or a matrix or a data frame with the last column being the frequency weights.

The algorithm of quantilefreq are based on wtd.quantile.

The intern functions are for programming purpose. It does not check the data.

Value

meanfreq and sdfreq return vectors. quantilefreq returns a vector or matrix. covfreq and corfreq the estimated covariance matrix and correlation matrix, respectively. scalefreq return a data frame or matrix

Note

The author would like to thank Prof. Frank E. Harrell Jr. who allowed the reutilisation of part of his code.

References

Andrews, Chris, https://stat.ethz.ch/pipermail/r-help/2014-March/368350.html

See Also

tablefreq, wtd.quantile

Examples

```
if(require(hflights)) {
  meanfreq(hflights[,c("ArrDelay","DepDelay")])
  sdfreq(hflights[,c("ArrDelay","DepDelay")])
  corfreq(hflights[,c("ArrDelay","DepDelay")])
}
```

```
tfq <- tablefreq(iris$Sepal.Length)
tfq

meanfreq(iris$Sepal.Length)
meanfreq(tfq,freq="freq")
.meanfreq(tfq)

dat <- iris[,1:4]
quantilefreq(dat)
corfreq(dat)

tfq <- tablefreq(dat)
.meanfreq(tfq)
.quantilefreq(tfq)
.corfreq(tfq)

## dplyr integration
library(dplyr)
tfq  %>%
  summarise( mean = .meanfreq(cbind(Sepal.Length,freq)),
             sd = .sdfreq(cbind(Sepal.Length,freq)))

tfq <- tablefreq(iris)
tfq %>% group_by(Species) %>%
  summarise( mean = .meanfreq(cbind(Sepal.Length,freq)),
             sd = .sdfreq(cbind(Sepal.Length,freq)))
```

`tablefreq` *Create a table of frequencies*

Description

Create a table of frequencies

Usage

```
tablefreq(tbl, vars = NULL, freq = NULL)

## S3 method for class 'tablefreq'
update(object, ...)
```

Arguments

`tbl`	an object that can be coerced to a `tbl`. It must contain all variables in `vars` and in `freq`
`vars`	variables to count unique values of. It may be a character vector
`freq`	a name of a variable of the tbl object specifying frequency weights. See Details
`object`	a `tablefreq` object
`...`	more data

Details

Based on the count function, it can also work with matrices or external data bases and the result may be updated.

It creates a frequency table of the data, or just of the columns specified in vars.

If you provide a freq formula, the cases are weighted by the result of the formula. Any variables in the formula are removed from the data set. If the data set is a matrix, the freq formula is a classic R formula. Otherwise, the expresion of freq is treated as a mathematical expression.

This function uses all the power of dplyr to create frequency tables. The main advantage of this function is that it works with on-disk data stored in data bases, whereas count only works with in-memory data sets.

In general, in order to use the functions of this package, the frequency table obtained by this function should fit in memory. Otherwise you must use the 'chunk' versions (link{clarachunk}, link{biglmfreq}).

The code of this function are adapted from a wish list of the devel page of dplyr (See references). Prof. Wickham also provides a nice introduction about how to use it with databases.

Value

A tbl object a with label and freq columns. When it is possible, the last column is named freq and it represents the frequency counts of the cases. This object of class tablefreq, has two attributes:

freq	the weighting variable used to create the frequency table
colweights	Name of the column with the weighting counts

Note

The author would like to thank Prof. Hadley Wickham who allowed the reutilisation of part of his code. When using the update function, be careful with non-integer weights: The precision of the final weights may be wrong due to the multiple sums.

References

Hadley Wickham. Count function https://github.com/hadley/dplyr/issues/358 Hadley Wickham. Databases http://cran.rstudio.com/web/packages/dplyr/vignettes/databases.html

See Also

count, tbl

Examples

```
tablefreq(iris)
tablefreq(iris, c("Sepal.Length","Species"))
a <- tablefreq(iris,freq="Sepal.Length")
tablefreq(a, freq="Sepal.Width")

library(dplyr)
iris %>% tablefreq("Species")
```

```
tfq <- tablefreq(iris[,c(1:2)])

chunk1 <- iris[1:10,c(1:2)]
chunk2 <- iris[c(11:20),]
chunk3 <- iris[-c(1:20),]
a <- tablefreq(chunk1)
a <- update(a,chunk2)
a <- update(a,chunk3)
a

## Not run:

## External databases
library(dplyr)
if(require(RSQLite)){
  hflights_sqlite <- tbl(hflights_sqlite(), "hflights")
  hflights_sqlite
  tbl_vars(hflights_sqlite)
  tablefreq(hflights_sqlite,vars=c("Year","Month"),freq="DayofMonth")
}

##
## Graphs
##
if(require(ggplot2) && require(hflights)){
  library(dplyr)

  ## One variable
  ## Bar plot
  tt <- as.data.frame(tablefreq(hflights[,"ArrDelay"]))
  p <- ggplot() + geom_bar(aes(x=x, y=freq), data=tt, stat="identity")
  print(p)

  ## Histogram
  p <- ggplot() + geom_histogram(aes(x=x, weight= freq), data = tt)
  print(p)

  ## Density
  tt <- tt[complete.cases(tt),] ## remove missing values
  tt$w <- tt$freq / sum(tt$freq) ## weights must sum 1
  p <- ggplot() + geom_density(aes(x=x, weight= w), data = tt)
  print(p)

  ##
  ## Two distributions
  ##
  ## A numeric and a factor variable
  td <- tablefreq(hflights[,c("TaxiIn","Origin")])
  td <- td[complete.cases(td),]

  ## Bar plot
  p <- ggplot() + geom_bar(aes(x=TaxiIn, weight= freq, colour = Origin),
```

```
                            data = td, position ="dodge")
  print(p)

  ## Density
  ## compute the relative frequencies for each group
  td <- td %.% group_by(Origin) %.%
              mutate( ngroup= sum(freq), wgroup= freq/ngroup)
  p <- ggplot() + geom_density(aes(x=TaxiIn, weight=wgroup, colour = Origin),
                               data = td)
  print(p)

  ## For each group, plot its values
  p <- ggplot() + geom_point(aes(x=Origin, y=TaxiIn, size=freq),
                             data = td, alpha= 0.6)
  print(p)

  ## Two numeric variables
  tc <- tablefreq(hflights[,c("TaxiIn","TaxiOut")])
  tc <- tc[complete.cases(tc),]
  p <- ggplot() + geom_point(aes(x=TaxiIn, y=TaxiOut, size=freq),
                             data = tc, color = "red", alpha=0.5)
  print(p)

  ## Two factors
  tf <- tablefreq(hflights[,c("UniqueCarrier","Origin")])
  tf <- tf[complete.cases(tf),]

  ## Bar plot
  p <- ggplot() + geom_bar(aes(x=Origin, fill=UniqueCarrier, weight= freq),
                           data = tf)
  print(p)
}

## End(Not run)
```

Index