Package ‘UBStats’

August 29, 2026

Type Package

Title Basic Statistics

Version 0.4.3

Date 2026-08-29

Description Basic statistical analyses. The package provides comprehensive functions and datasets for teaching introductory statistics courses. It has been developed to be used in undergraduate statistics courses at Bocconi University (Milan, Italy), and constitutes the core software tool used throughout the textbook by Piccarreta, R., Tonini, D., & Trentini, F. (2026) ``From Data to Decisions. An Introduction to Applied Statistics'', BUP, ISBN:9788823824096.

Maintainer Sergio Venturini <sergio.venturini@unicatt.it>

License GPL-3

NeedsCompilation no

Repository CRAN

LazyData true

Imports graphics, grDevices, stats

Depends R (>= 3.6.0), utils

BugReports https://github.com/raffaellapiccarreta/UBStats/issues

Encoding UTF-8

Config/roxygen2/version 8.1.0

Author Raffaella Piccarreta [aut],
Sergio Venturini [cre]

Date/Publication 2026-08-29 10:20:02 UTC

Contents

`Banner` *Data Banner: Banner Click-Through Rate Experiment*

Description

The data frame contains data collected by a company interested in comparing a *Redesigned* banner (featuring a brighter colour scheme and modified layout) with the *Original* version. A random sample of users from a social network is shown one of two banners, and the effectiveness of the two banners is assessed via the **click-through rate (CTR)** – the proportion of users who click on a banner after viewing it.

Usage

```
data(Banner)
```

Format

A data frame with n = 5976 observations and 3 variables:

User (int) user id

Banner (chr) banner shown to the user (`Original`, `Redesigned`)

Click (logical) logical vector indicating whether the user clicked on the banner (TRUE) or not

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

`BasketValue` *Data BasketValue: Supermarket Spending and Basket Variety*

Description

The data frame contains data on a supermarket customers living relatively close to the store. Based on the details provided when the loyalty card was registered and on recorded expenses, for each customer information is available on a composite index, *SpendMix*, which captures how supermarket clients differ not only in how much they spend but also in the range of products they purchase. We are interested to identify factors associated with the level of spending and basket variety. Some characteristics (e.g., income and age) are obvious candidates: richer individuals usually spend more, and spending habits often change over the course of a person's life. Yet behavioural differences can also play a role. For instance, shoppers who are particularly responsive to discounts may be more inclined to experiment with new items or brands.

Usage

```
data(BasketValue)
```

Format

A data frame with $n = 128$ observations and 5 variables:

Id (int) customer identification number

Age (num) age of the customers in years

Wealth (num) composite indicator of household economic well-being obtained by estimating household wealth from the average market price and rental value of dwellings located in the same street as each customer's declared address, and adjusting for the number of cohabitants

Promo (factor) variable indicating whether promotional products are frequently purchased (`No`, `Yes`)

SpendMix (num) composite index which reflects both spending level and basket variety

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

CallCentre_KPI *Data CallCentre_KPI: Call Centre Key Performance Indicators*

Description

This data frame contains data on the calls to the call centre of a telecommunication company. It contains information about the calls (day and hour of the call), about the calling customer's characteristics (private or corporate, or unknown), and about some key performance indicators (KPI). In particular, it contains information on the duration of the phases before the customer eventually reaches an operator. First, an automated system directs incoming calls to the appropriate department based on the reason for the call; then the customer waits in a queue until an operator becomes available. It also contains a variable indicating whether the caller hung up during a specific phase. For those customers whose call was finally handled by an operator, we also know whether their issue was solved and their level of satisfaction, expressed at the end of the call.

Usage

```
data(CallCentre_KPI)
```

Format

A data frame with $n = 5007$ observations and 11 variables:

ID_Call (int) call identification

Day (factor) weekday of the call (`Mon`, `Tue`, `Wed`,..., `Sat`)

Hour (num) hour of the call

Cust.Type (chr) customer's type (`Unknown`, `Private`, `Corporate`)

Time.Auto (num) time (in seconds) spent interacting with the automatic responder

Time.Queue (num) time (in seconds) spent waiting in the queue for an operator

Time.Talk (num) duration (in seconds) of the (eventual) conversation with an operator

Reason (chr) reason for contact (`Admin` – administrative issues; `Landline` – issues with fixed line; `Mobile`; and `Activ/Transf` – calls related to the activation or the transfer of a line)

Outcome (chr) outcome of the call (`Left.Aut_Resp` – the client hung up before entering the queue; `Left.Queue` – the client hung up while waiting for an operator; `Operator` – the call was handled by an operator)

Solved (chr) variable indicating whether the client's issue was solved (`No`, `Yes`) – available only for clients who interacted with an operator

Satisf (factor) level of satisfaction expressed by customers who interacted with an operator (`VLow`, `Low`, `Med`, `High`, `VHigh`)

“

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

CI.diffmean *Confidence intervals for the difference between means*

Description

CI.diffmean() builds confidence intervals for the difference between the means of two independent or paired populations.

Usage

```
CI.diffmean(
  x,
  y,
  type = "independent",
  sigma.x = NULL,
  sigma.y = NULL,
  conf.level = 0.95,
  by,
  sigma.by = NULL,
  sigma.d = NULL,
  var.test = FALSE,
  digits = 2,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

x, y
: Unquoted strings identifying two *numeric* variables with the same length whose means have to be compared. x and y can be the names of vectors in the workspace or the names of columns in the data frame specified in the data argument. It is possible to use a mixed specification (e.g, one vector and one column in data).

type
: A length-one character vector specifying the type of samples. Allowed values are "independent" or "paired".

sigma.x, sigma.y
: Optional numeric values specifying the possibly known populations' standard deviations (when x and y are specified). If NULL (default) standard deviations are estimated using the data.

conf.level
: Numeric value specifying the required confidence level; default to 0.95.

by
: Optional unquoted string, available only when type = "independent", identifying a variable (of any type), defined same way as x, taking only **two** values used to split x into two **independent samples**. Given the two *ordered* values taken by by (alphabetical or numerical order, or order of the levels for factors), say *by1* and *by2*, the confidence interval is built for the difference between the

populations means in the *by1*- and in the *by2*-group. Note that only **one** between y and by can be specified.

sigma.by	Optional numeric value specifying the possibly known standard deviations for the two *independent* samples identified via by (when x and by are specified). sigma.by can be a single value indicating the same standard deviation in the two by-groups, or a vector with two values, specifying the standard deviations in the two by-groups. To avoid errors, in the latter case the vector should be named, with names coinciding with the two levels of by.
sigma.d	Optional numeric value specifying the possibly known standard deviation of the difference when samples are **paired**.
var.test	Logical value indicating whether to run a test on the equality of variance for two (**independent**) samples or not (default).
digits	Integer value specifying the number of decimals used to round statistics; default to 2. If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument force.digits is set to TRUE.
force.digits	Logical value indicating whether reported values should be forcedly rounded to the number of decimals specified in digits even if non-zero values are rounded to zero (default to FALSE).
use.scientific	Logical value indicating whether numbers in tables should be displayed using scientific notation (TRUE); default to FALSE.
data	An optional data frame containing x and/or y. If not found in data, the variables are taken from the environment from which CI.diffmean() is called.
...	Additional arguments to be passed to low level functions.

Value

A table reporting the confidence intervals for the difference between the populations' means. For *independent* samples in the case of unknown variances, the intervals are built both under the assumption that the variances are equal and under the assumption that they differ, using percentiles from both the normal and the Student's t distribution. If

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

TEST.diffmean() to test hypotheses on the difference between two populations' means.

Examples

```
data(MktDATA, package = "UBStats")

# Independent samples (default type), UNKNOWN variances
#  CI for the difference between means of males and females
#  - Using x,y: build vectors with data on the two groups
```

```
AOV_M <- MktDATA$AOV[MktDATA$Gender == "M"]
AOV_F <- MktDATA$AOV[MktDATA$Gender == "F"]
CI.diffmean(x = AOV_M, y = AOV_F)
#  - Change confidence level
CI.diffmean(x = AOV_M, y = AOV_F, conf.level = 0.99)
#  - Using x,by: groups identified by ordered levels of by
CI.diffmean(x = AOV, by = Gender, conf.level = 0.99, data = MktDATA)
#    Since order is F, M, CI is for mean(F) - mean(M)
#    To get the interval for mean(M) - mean(F)
Gender.R <- factor(MktDATA$Gender, levels = c("M", "F"))
CI.diffmean(x = AOV, by = Gender.R, conf.level = 0.99,
            data = MktDATA)
#  - Testing hypotheses on equality of unknown variances
CI.diffmean(x = AOV_M, y = AOV_F, conf.level = 0.99,
            var.test = TRUE)

#  - Output results: only information on the CI
out.ci_diffM<-CI.diffmean(x = AOV_M, y = AOV_F)
#  - Output results: list with information on CI and test on var
out.ci_diffM.V<-CI.diffmean(x = AOV_M, y = AOV_F, var.test = TRUE)

# Independent samples (default type), KNOWN variances
#  CI for the difference between means of males and females
#  - Using x,y: build vectors with data on the two groups
AOV_M <- MktDATA$AOV[MktDATA$Gender == "M"]
AOV_F <- MktDATA$AOV[MktDATA$Gender == "F"]
CI.diffmean(x = AOV_M, y = AOV_F,
            sigma.x = 10, sigma.y = 20)
#  - Using x,by: groups identified by ordered levels of by
CI.diffmean(x = AOV, by = Gender,
            sigma.by = c("M" = 10, "F"=20), data = MktDATA)
#    To change the sign, order levels as desired
Gender.R <- factor(MktDATA$Gender, levels = c("M", "F"))
CI.diffmean(x = AOV, by = Gender.R,
            sigma.by = c("M" = 10, "F"=20), data = MktDATA)
#  - Output results
out.ci_diffM<-CI.diffmean(x = AOV_M, y = AOV_F,
                          sigma.x = 10, sigma.y = 20)

# Paired samples: UNKNOWN variances
# - Default settings
CI.diffmean(x = NStore_Purch, y = NWeb_Purch,
            type = "paired", data=MktDATA)
# - Change confidence level
CI.diffmean(x = NStore_Purch, y = NWeb_Purch,
            type = "paired", conf.level = 0.9, data = MktDATA)
# Paired: KNOWN variances
CI.diffmean(x = NStore_Purch, y = NWeb_Purch,
            type = "paired", conf.level = 0.9,
            sigma.d = 2, data = MktDATA)
#  - Output results
out.ci_diffM<-CI.diffmean(x = NStore_Purch, y = NWeb_Purch,
                          type = "paired", conf.level = 0.9,
```

```
                          sigma.d = 2, data = MktDATA)

# Arguments force.digits and use.scientific
#  An input variable taking very low values
SmallX<-MktDATA$AOV/5000
SmallX_M <- SmallX[MktDATA$Gender == "M"]
SmallX_F <- SmallX[MktDATA$Gender == "F"]
# - Default: manages possible excess of rounding
CI.diffmean(x = SmallX_M, y = SmallX_F)
# - Force to the requested nr of digits (default, 2)
CI.diffmean(x = SmallX_M, y = SmallX_F,
            force.digits = TRUE)
# - Allow scientific notation
CI.diffmean(x = SmallX_M, y = SmallX_F,
            use.scientific = TRUE)
```

`CI.diffprop` *Confidence intervals for the difference between proportions*

Description

`CI.diffprop()` builds confidence intervals for the difference between the proportion of successes in two independent populations.

Usage

```
CI.diffprop(
  x,
  y,
  success.x = NULL,
  success.y = NULL,
  conf.level = 0.95,
  by,
  digits = 2,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x, y`	Unquoted strings identifying the variables of interest. `x` and `y` can be the names of vectors or factors in the workspace or the names of columns in the data frame specified in the `data` argument. It is possible to use a mixed specification (e.g, one vector and one column in data).

success.x, success.y
: If x,y are factors, character vectors, or numeric non-binary vectors, success must be used to indicate the category/value corresponding to success in the populations. These arguments can be omitted (NULL, default) if x,y are binary numeric vectors (taking values 0 or 1 only; in this case success is assumed to correspond to 1) or a logical vector (in these cases success is assumed to correspond to TRUE).

conf.level
: Numeric value specifying the required confidence level; default to 0.95.

by
: Optional unquoted string identifying a variable (of any type), defined same way as x, taking only **two** values used to split x into two independent samples. Given the two *ordered* values taken by by (alphabetical or numerical order, or order of the levels for factors), say *by1* and *by2*, the confidence interval is built for the difference between the populations proportions in the *by1*- and in the *by2*-group. Note that only **one** between y and by can be specified.

digits
: Integer value specifying the number of decimals used to round statistics; default to 2. If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument force.digits is set to TRUE.

force.digits
: Logical value indicating whether reported values should be forcedly rounded to the number of decimals specified in digits even if non-zero values are rounded to zero (default to FALSE).

use.scientific
: Logical value indicating whether numbers in tables should be displayed using scientific notation (TRUE); default to FALSE.

data
: An optional data frame containing x and/or y. If not found in data, the variables are taken from the environment from which CI.diffprop() is called.

...
: Additional arguments to be passed to low level functions.

Value

A table reporting the confidence intervals for the difference between the proportions of successes in two independent populations.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

TEST.diffprop() to test hypotheses on the difference between the proportions of successes in two populations.

Examples

```
data(MktDATA, package = "UBStats")

# Proportions of success defined on non-binary and
#  non-logical vectors; 'success' coded same way
#  for both vectors
```

```
#  - Using x,y: build vectors with data on the two groups
WouldSuggest_F <- MktDATA$WouldSuggest[MktDATA$Gender == "F"]
WouldSuggest_M <- MktDATA$WouldSuggest[MktDATA$Gender == "M"]
CI.diffprop(x = WouldSuggest_M, y = WouldSuggest_F,
            success.x = "Yes")

PastCampaigns_F<-MktDATA$PastCampaigns[MktDATA$Gender=="F"]
PastCampaigns_M<-MktDATA$PastCampaigns[MktDATA$Gender=="M"]
CI.diffprop(x = PastCampaigns_M, y = PastCampaigns_F,
            success.x = 0, conf.level = 0.99)

#  - Using x,by: groups identified by ordered levels of by
CI.diffprop(x = PastCampaigns, by = Gender,
            success.x=0, conf.level = 0.99,
            data = MktDATA)
#    Since order is F, M, CI is for prop(F) - prop(M)
#    To get the interval for prop(M) - prop(F)
Gender.R <- factor(MktDATA$Gender, levels = c("M", "F"))
CI.diffprop(x = PastCampaigns, by = Gender.R,
            success.x=0, conf.level = 0.99, data = MktDATA)

# Proportions of success defined based on
#  binary or logical vectors; 'success'
#  coded same way for both vectors
#  - Binary variable (success=1): based on x,y
LastCampaign_F<-MktDATA$LastCampaign[MktDATA$Gender=="F"]
LastCampaign_M<-MktDATA$LastCampaign[MktDATA$Gender=="M"]
CI.diffprop(x = LastCampaign_M, y = LastCampaign_F)
#  - Binary variable (success=1): based on x,y
#    see above for recoding of levels of Gender
Gender.R <- factor(MktDATA$Gender, levels = c("M", "F"))
CI.diffprop(x = LastCampaign, by = Gender.R, data = MktDATA)
#  - Logical variable (success=TRUE): based on x,y
Deals_w_child <- MktDATA$Deals.ge50[MktDATA$Children>0]
Deals_no_child <- MktDATA$Deals.ge50[MktDATA$Children==0]
CI.diffprop(x = Deals_w_child, y = Deals_no_child, conf.level = 0.9)

# Proportions defined on
#  non-binary and non-logical vectors, with 'success'
#  coded differently (only specification x,y is reasonable here)
WouldSuggest_Other<-c(rep("OK",310),rep("KO",650-310))
CI.diffprop(x = WouldSuggest, y = WouldSuggest_Other,
            success.x = "Yes", success.y = "OK",
            data = MktDATA)

# Proportions based on combined conditions
# - Build logical vector/s indicating whether a condition
#   is satisfied
IsTop<-MktDATA$AOV>80
IsTop_OK<-IsTop[MktDATA$WouldSuggest == "Yes"]
IsTop_KO<-IsTop[MktDATA$WouldSuggest == "No"]
CI.diffprop(x = IsTop_OK, y = IsTop_KO, conf.level = 0.9)
```

```
Deals<-MktDATA$NDeals>=5
Deals_Married <- Deals[MktDATA$Marital_Status=="Married" &
                         MktDATA$Children==0]
Deals_Single <- Deals[MktDATA$Marital_Status=="Single"]
CI.diffprop(x = Deals_Married, y = Deals_Single, conf.level = 0.9)

# Output results
Gender.R <- factor(MktDATA$Gender, levels = c("M", "F"))
out.ci_diffP<-CI.diffprop(x = PastCampaigns, by = Gender.R,
                          success.x=0, conf.level = 0.99,
                          data = MktDATA)

# Arguments force.digits and use.scientific
#  An input variable taking very low values
HighAOV <- MktDATA$AOV>150
# - Default: manages possible excess of rounding
CI.diffprop(x = HighAOV[MktDATA$Gender=="M"],
            y = HighAOV[MktDATA$Gender=="F"])
#  - Force to the exact number of digits (default, 2)
CI.diffprop(x = HighAOV[MktDATA$Gender=="M"],
            y = HighAOV[MktDATA$Gender=="F"],
            force.digits = TRUE)
#  - Allow scientific notation
CI.diffprop(x = HighAOV[MktDATA$Gender=="M"],
            y = HighAOV[MktDATA$Gender=="F"],
            use.scientific = TRUE)
```

`CI.mean` *Confidence intervals for the mean*

Description

`CI.mean()` builds confidence intervals for the mean of a population.

Usage

```
CI.mean(
  x,
  sigma = NULL,
  conf.level = 0.95,
  digits = 2,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x`	An unquoted string identifying the *numeric* variable whose mean is of interest. `x` can be the name of a vector in the workspace or the name of one of the columns in the data frame specified in the `data` argument.
`sigma`	An optional numeric value specifying the population standard deviation. If `NULL` (default) the population standard deviation is estimated using the data.
`conf.level`	Numeric value specifying the required confidence level; default to 0.95.
`digits`	Integer value specifying the number of decimals used to round statistics; default to 2. If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument `force.digits` is set to `TRUE`.
`force.digits`	Logical value indicating whether reported values should be forcedly rounded to the number of decimals specified in `digits` even if non-zero values are rounded to zero (default to `FALSE`).
`use.scientific`	Logical value indicating whether numbers in tables should be displayed using scientific notation (`TRUE`); default to `FALSE`.
`data`	An optional data frame containing `x`. If not found in `data`, `x` is taken from the environment from which `CI.mean()` is called.
`...`	Additional arguments to be passed to low level functions.

Value

A table reporting the confidence interval for the population mean. If the variance is unknown, the interval is built using percentiles from both the normal and the Student's t distribution.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

`TEST.mean()` to test hypotheses on a population mean.

Examples

```
data(MktDATA, package = "UBStats")

# CI for the mean with KNOWN variance; default options
CI.mean(AOV, sigma = 30, data = MktDATA)

# CI for the mean with UNKNOWN variance;
# - change digits and confidence level 0.99
CI.mean(AOV, conf.level = 0.99, digits = 3, data = MktDATA)

# Arguments force.digits and use.scientific
#  A variable taking very small values
SmallX<-MktDATA$AOV/5000
#  - Default: manages possible excess of rounding
```

```
CI.mean(SmallX)
#  - Forcing digits to the default values (2)
CI.mean(SmallX, force.digits = TRUE)
#  - Allow scientific notation
CI.mean(SmallX, use.scientific = TRUE)

# Output the table with the requested interval
out.ci_mean<-CI.mean(AOV, data = MktDATA)
```

`CI.prop` *Confidence intervals for the proportion*

Description

`CI.prop()` builds confidence intervals for the proportion of successes in a population.

Usage

```
CI.prop(
  x,
  success = NULL,
  conf.level = 0.95,
  digits = 2,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x`	An unquoted string identifying the variable of interest. `x` can be the name of a vector or a factor in the workspace or the name of one of the columns in the data frame specified in the `data` argument.
`success`	If `x` is a factor, a character vector, or a numeric non-binary vector, `success` must be used to indicate the category/value corresponding to success. The argument can be omitted (`NULL`, default) if `x` is a binary numeric vector (takes values 0 or 1 only; in this case success is assumed to be 1) or a logical vector (in these cases success is assumed to be `TRUE`).
`conf.level`	Numeric value specifying the required confidence level; default to 0.95.
`digits`	Integer value specifying the number of decimals used to round statistics; default to 2. If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument `force.digits` is set to `TRUE`.

force.digits
: Logical value indicating whether reported values should be forcedly rounded to the number of decimals specified in `digits` even if non-zero values are rounded to zero (default to `FALSE`).

use.scientific
: Logical value indicating whether numbers in tables should be displayed using scientific notation (`TRUE`); default to `FALSE`.

data
: An optional data frame containing `x`. If not found in `data`, `x` is taken from the environment from which `CI.prop()` is called.

...
: Additional arguments to be passed to low level functions.

Value

A table reporting the confidence intervals for the population proportion of successes.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

`TEST.prop()` to test hypotheses on the proportion of successes in a population.

Examples

```
data(MktDATA, package = "UBStats")

# Success = one value of a character vector or factor
CI.prop(WouldSuggest, success = "Yes", data = MktDATA)

# - change confidence level and rounding
CI.prop(Education, success = "Post-Grad",
        conf.level = 0.9, digits = 4,
        data = MktDATA)

# Success = numeric value
CI.prop(Children, success = 2, data = MktDATA)

# Binary variable ('success' is 1 by default)
CI.prop(LastCampaign, digits = 3, data = MktDATA)

# Logical variable ('success' is TRUE by default)
CI.prop(RespCampaign, conf.level = 0.9, digits = 3, data = MktDATA)

# Success based on combined conditions
# - Build a (logical) vector indicating whether a condition is satisfied
IsTop <- MktDATA$CustClass == "Gold" | MktDATA$CustClass == "Platinum"
CI.prop(IsTop, conf.level = 0.9)
# - A very rare event
HighAOV <- MktDATA$AOV>150
CI.prop(HighAOV, conf.level = 0.9)

# Arguments force.digits, use.scientific
```

```
# - Default: manages possible excess of rounding
CI.prop(HighAOV)
# - Forcing digits to the default values (2)
CI.prop(HighAOV, force.digits = TRUE)
# - Allow scientific notation
CI.prop(HighAOV, use.scientific = TRUE)

# Output results
out_ci_prop<-CI.prop(HighAOV)
```

---

distr.plot.x *Analysis of a univariate distribution using plots*

Description

distr.plot.x() generates plots of a univariate distribution.

Usage

```
distr.plot.x(
  x,
  freq = "counts",
  plot.type,
  ord.freq = "none",
  breaks,
  adj.breaks = TRUE,
  interval = FALSE,
  bw = FALSE,
  color = NULL,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

x	An unquoted string identifying the variable whose distribution has to be analysed. x can be the name of a vector or a factor in the workspace or the name of one of the columns in the data frame specified in the data argument.
freq	A single character specifying the frequencies to be displayed. Allowed options (possibly abbreviated) are "counts", "percentages", "proportions", "densities" (for histograms and density plots).
plot.type	A single character specifying the type of plot to build. Allowed options are "pie", "bars", "spike", "histogram", "density", "boxplot", and "cumulative".

ord.freq	A single character vector that can be specified when plot.type = "pie" or plot.type = "bars". It indicates whether the levels of x should be displayed in a standard order (ord.freq = "none", the default) or in an increasing or decreasing order (ord.freq = "increasing" or ord.freq = "decreasing").
breaks	Allows to classify a *numerical* variable x into intervals. It can be an integer indicating the number of intervals of equal width used to classify x, or a vector of increasing numeric values defining the endpoints of intervals (closed on the left and open on the right; the last interval is closed on the right too). To cover the entire range of values the maximum and the minimum values should be included between the first and the last break. It is possible to specify a set of breaks covering only a portion of the x range.
adj.breaks	Logical value indicating whether the endpoints of intervals of a numerical variable x when classified into intervals should be displayed avoiding scientific notation; default to TRUE.
interval	Logical value indicating whether x is a variable measured in intervals (TRUE). If the detected intervals are not consistent (e.g. overlapping intervals, or intervals with upper endpoint higher than the lower one), the variable is analyzed as it is, even if results are not necessarily consistent; default to FALSE.
bw	Logical value indicating whether plots should be colored in scale of greys (TRUE) rather than using a standard palette (FALSE, default).
color	Optional string vector allowing to specify colors to use in the plot rather than a standard palette (NULL, default).
use.scientific	Logical value indicating whether numbers on axes should be displayed using scientific notation (TRUE); default to FALSE.
data	An optional data frame containing x. If not found in data, x is taken from the environment from which distr.plot.x() is called.
...	Additional arguments to be passed to low level functions.

Value

No return value, called for side effects.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

distr.table.x() for tabulating a univariate distribution.

distr.table.xy() for tabulating a bivariate distribution.

distr.plot.xy() for plotting a bivariate distribution.

Examples

```
data(MktDATA, package = "UBStats")
```

```
# Pie charts
# - A character variable: grey scale
distr.plot.x(x = LikeMost, plot.type = "pie", bw = TRUE, data = MktDATA)
# - A discrete numeric variable: user-defined palette
distr.plot.x(x = Children, plot.type = "pie",
             color=c("red","gold","green","forestgreen"),
             data = MktDATA)

# Bar charts
# - A factor: standard order of levels
distr.plot.x(x = Education, plot.type = "bars",
             freq = "percentage", data = MktDATA)
# - A factor: levels arranged by decreasing percentage
distr.plot.x(x = Education, plot.type = "bars",
             freq = "perc", ord.freq = "dec", data = MktDATA)
# - A discrete variable (note: distance between values
#   not taken into account)
distr.plot.x(x = NPickUp_Purch, plot.type = "bars",
             freq = "percentage", data = MktDATA)

# Spike plots
# - A discrete variable
distr.plot.x(x = NPickUp_Purch, plot.type = "spike",
             freq = "percent", data = MktDATA)
# - A factor (levels placed at the same distance)
distr.plot.x(x = Education, plot.type = "spike",
             freq = "prop",data = MktDATA)
# - A variable measured in classes (levels placed at the
#   same distance)
distr.plot.x(x = Income.S, interval = TRUE,
             plot.type = "spike",
             freq = "prop",data = MktDATA)
# - A numeric variable classified into intervals
#   (levels placed at the same distance)
distr.plot.x(x = AOV, breaks = 5, plot.type = "spike",
             data = MktDATA)

# Cumulative distribution plots
# - A discrete variable
distr.plot.x(x = Children, plot.type = "cum", data = MktDATA)
# - A continuous numerical variable
distr.plot.x(x = AOV, plot.type = "cum",
             freq = "perc", data = MktDATA)
# - A numeric variable classified into intervals
distr.plot.x(AOV, plot.type = "cum",
             breaks = c(0,20,40,60,80,100,180), data = MktDATA)
# - A variable measured in classes
distr.plot.x(Income, plot.type = "cum", interval = TRUE,
             freq = "percent", data = MktDATA)
# - A factor
distr.plot.x(x = Education, plot.type = "cum",
             freq = "prop",data = MktDATA)
```

```
# Histograms
# - A continuous numerical variable: no breaks provided
#    default classes built by R
distr.plot.x(x = AOV, plot.type = "histogram", data = MktDATA)
# - A continuous numerical variable: equal width intervals
distr.plot.x(x = AOV, plot.type = "histogram",
             breaks = 10, data = MktDATA)
# - A continuous numerical variable: specified breaks
distr.plot.x(AOV, plot.type = "histogram",
             breaks = c(0,20,40,60,80,100,180),
             data = MktDATA)
# - A variable measured in classes
distr.plot.x(Income, plot.type = "histogram",
             interval = TRUE, data = MktDATA)

# Density plots
# - A  numerical variable
distr.plot.x(x = AOV, plot.type = "density", data = MktDATA)
# - A  numerical variable: breaks are ignored
distr.plot.x(AOV, plot.type = "density",
             breaks = c(0,20,40,60,80,100,180),
             data = MktDATA)
# - A variable measured in classes
distr.plot.x(Income, plot.type = "density",
             interval = TRUE, data = MktDATA)

# Boxplots (only for numerical unclassified variables)
# - A  numerical variable
distr.plot.x(x = TotVal, plot.type = "boxplot", data = MktDATA)
# - A  numerical variable: with specified breaks
#   the plot is not built
# distr.plot.x(AOV, plot.type = "boxplot",
#              breaks = c(0,20,40,60,80,100,180),
#              data = MktDATA)

# Arguments adj.breaks, use.scientific
#  A variable with a very wide range (very small densities)
LargeX<-MktDATA$AOV*5000000
#  - Default formatting for intervals' endpoints
distr.plot.x(LargeX, breaks = 5, plot.type = "spike")
#  - Scientific notation for intervals' endpoints
distr.plot.x(LargeX, breaks = 5,plot.type = "spike",
             adj.breaks = FALSE)
#  - Default formatting for axes
distr.plot.x(LargeX, breaks = 5,plot.type = "histogram",
             freq = "densities")
#  - Scientific notation for axes
distr.plot.x(LargeX, breaks = 5,plot.type = "histogram",
             freq = "densities",use.scientific = TRUE)
```

distr.plot.xy *Analysis of a bivariate distribution using plots*

Description

`distr.plot.xy()` generates plots of a bivariate distribution.

Usage

```
distr.plot.xy(
  x,
  y,
  plot.type,
  bar.type = "stacked",
  freq = "counts",
  freq.type = "joint",
  breaks.x,
  breaks.y,
  interval.x = FALSE,
  interval.y = FALSE,
  bw = FALSE,
  color = NULL,
  var.c,
  breaks.c,
  interval.c = FALSE,
  adj.breaks = TRUE,
  fitline = FALSE,
  legend = TRUE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x, y` Unquoted strings identifying the variables whose distribution has to be graphically displayed. `x` and `y` can be the name of a vector or a factor in the workspace or the name of one of the columns in the data frame specified in the `data` argument. Note that in the plot `x` is reported on the *horizontal* axis while `y` is reported on the *vertical* axis.

`plot.type` A single character specifying the type of plot to build. Allowed options are `"bars"`, `"scatter"`, and `"boxplot"`. If both `x` and `y` are character vectors or factors and `bar.type = "scatter"` a bubble plot is built, with dots having a size proportional to the joint frequency of each pair of observed values. If `bar.type = "boxplot"`, at least one input variable must be numeric; when both the variables are numeric the conditional distributions of `y|x` are displayed, unless otherwise specified using `freq.type="x|y"`.

bar.type
: A single character indicating whether in a bar plot stacked (bar.type = "stacked", default) or side-by-side (bar.type = "beside") bars should be displayed.

freq
: A single character specifying the frequencies to be displayed when a bar plot is requested (plot.type="bars"). Allowed options (possibly abbreviated) are "counts", "percentages" and "proportions".

freq.type
: A single character specifying the type of frequencies to be displayed when a bar plot is requested (plot.type="bars"). Allowed options are joint (default) for joint frequencies, x|y for the distributions of x conditioned to y, and y|x for the distributions of y conditioned to x. The option x|y can also be used when plot.type="boxplot".

breaks.x, breaks.y
: Allow to classify the variables x and/or y, if *numerical*, into intervals. They can be integers indicating the number of intervals of equal width used to classify x and/or y, or vectors of increasing numeric values defining the endpoints of the intervals (closed on the left and open on the right; the last interval is closed on the right too). To cover the entire range of values taken by one variable, the maximum and the minimum values should be included between the first and the last break. It is possible to specify a set of breaks covering only a portion of the variable's range.

interval.x, interval.y
: Logical values indicating whether x and/or y are variables measured in classes (TRUE). If the detected intervals are not consistent (e.g. overlapping intervals, or intervals with upper endpoint higher than the lower one), the variable is analyzed as it is, even if results are not necessarily consistent; default to FALSE.

bw
: Logical value indicating whether plots should be colored in scale of greys (TRUE) rather than using a standard palette (FALSE, default).

color
: Optional string vector allowing to specify colors to use in the plot rather than a standard palette (NULL, default).

var.c
: An optional unquoted string identifying one variable used to color points in a scatter plot (plot.type="scatter"), that can be defined same way as x. This is allowed only when at least one of the input variables x and y is numeric.

breaks.c
: Allows to classify the variable var.c, if *numerical*, into intervals. It can be defined as breaks.x.

interval.c
: Logical value indicating whether var.c is a variable measured in intervals (TRUE) or not, as described for interval.x; default to FALSE.

adj.breaks
: Logical value indicating whether the endpoints of intervals of a numerical variable (x, or y, or var.c) when classified into intervals should be displayed avoiding scientific notation; default to TRUE.

fitline
: Logical value indicating whether the line of best fit (also called trend line or regression line) should be added to a scatter plot (fitline = TRUE) or not (fitline = FALSE; default).

legend
: Logical value indicating whether a legend should be displayed in the plot (legend = TRUE; default) or not (legend = FALSE).

use.scientific
: Logical value indicating whether numbers on axes should be displayed using scientific notation (TRUE); default to FALSE.

data	An optional data frame containing x and/or y and/or var.c (the variable used to color points in scatter plots). If not found in data, the variables are taken from the environment from which distr.plot.xy() is called.
...	Additional arguments to be passed to low level functions.

Value

No return value, called for side effects.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

distr.table.xy() for tabulating a bivariate distribution.

distr.table.x() for tabulating a univariate distribution.

distr.plot.x() for plotting a univariate distribution.

Examples

```
data(MktDATA, package = "UBStats")

# Bivariate bar plots
# - Two discrete variables (factor or vector with few levels)
#   Joint counts
distr.plot.xy(CustClass, Children,plot.type = "bars",
              freq = "Counts", freq.type = "joint",
              data = MktDATA)
# - Two discrete variables (factor or vector with few levels)
#   Joint percentages, side-by-side bars
#   User-defined colors
distr.plot.xy(Children,CustClass, plot.type = "bars",
              bar.type = "beside",
              freq = "percent", freq.type = "joint",
              color = c("red","gold","green","forestgreen"),
              data = MktDATA)
# - One numeric variable classified into intervals
#   and one variable measured in classes
#   Conditional percentages of x|y
distr.plot.xy(TotPurch, Income, plot.type = "bars",
              freq = "percent",freq.type = "x|y",
              breaks.x = c(0,5,10,15,20,35),
              interval.y = TRUE, data = MktDATA)
#   Conditional percentages of y|x
distr.plot.xy(TotPurch, Income, plot.type = "bars",
              freq = "percent",freq.type = "y|x",
              breaks.x = c(0,5,10,15,20,35),
              interval.y = TRUE, data = MktDATA)

# Side-by-side boxplots
```

```
# - A continuous variable conditioned to a factor,
#   a character, or a classified variable
#   The distributions of the numeric variable conditioned
#   to the factor (or character) are displayed
distr.plot.xy(x = AOV, y = Education, plot.type = "boxplot",
              data = MktDATA)
distr.plot.xy(x = Income.S, y = AOV, plot.type = "boxplot",
              interval.x = TRUE, data = MktDATA)
distr.plot.xy(x = Baseline, y = TotPurch, plot.type = "boxplot",
              breaks.y = c(0,5,10,15,20,35),
              data = MktDATA)
# - Two numerical variables. By default distributions
#   of y|x are displayed unless differently
#   specified in freq.type
distr.plot.xy(x = NPickUp_Purch, y = NWeb_Purch,
              plot.type = "boxplot", data = MktDATA)
distr.plot.xy(x = NPickUp_Purch, y = NWeb_Purch,
              plot.type = "boxplot",freq.type = "x|y",
              data = MktDATA)

# Scatter plots
# - Two numerical variables: default options
distr.plot.xy(Baseline, TotVal, plot.type = "scatter",
              fitline = TRUE, data = MktDATA)
# - Two numerical variables: colors based on discrete var
distr.plot.xy(Baseline, TotVal, plot.type = "scatter",
              var.c = Marital_Status,
              fitline = TRUE, data = MktDATA)
distr.plot.xy(Baseline, TotVal, plot.type = "scatter",
              var.c = Income, interval.c = TRUE,
              fitline = TRUE, data = MktDATA)
distr.plot.xy(Baseline, TotVal, plot.type = "scatter",
              var.c = TotPurch, breaks.c = 10,
              fitline = TRUE, data = MktDATA)
# - Two numerical variables: colors based
#   on a continuous numerical variable
distr.plot.xy(Baseline, TotVal, plot.type = "scatter",
              var.c = AOV, fitline = TRUE, data = MktDATA)

# - One numerical variable and one factor or character
distr.plot.xy(Baseline, Marital_Status, plot.type = "scatter",
              fitline = TRUE, data = MktDATA)
distr.plot.xy(Income.S, Baseline, plot.type = "scatter",
              interval.x = TRUE,
              fitline = TRUE, data = MktDATA)
#   color based on a third variable
distr.plot.xy(TotPurch, TotVal, plot.type = "scatter",
              breaks.x = c(0,5,10,15,20,35),
              var.c = AOV,
              fitline = TRUE, data = MktDATA)

# - Two factors or character vectors: bubble plots
distr.plot.xy(Education, LikeMost, plot.type = "scatter",
```

```
              data = MktDATA)
# - Two classified variables (i.e. not properly numerical):
#   bubble plots, changed color
distr.plot.xy(Income.S, TotPurch, plot.type = "scatter",
              interval.x = TRUE,
              breaks.y = c(0,5,10,15,20,35),
              color = "orchid", data = MktDATA)

# Arguments adj.breaks and use.scientific
#  Variable with very wide ranges
LargeC<-MktDATA$AOV*5000000
LargeX<-MktDATA$Baseline*1000000
LargeY<-MktDATA$TotVal*1000000
#  - Default: no scientific notation
distr.plot.xy(LargeX, LargeY, plot.type = "scatter",
              var.c = LargeC, data = MktDATA)
distr.plot.xy(LargeX, LargeY, plot.type = "scatter",
              breaks.x = 10, var.c = LargeC,
              data = MktDATA)
#  - Scientific notation for axes
distr.plot.xy(LargeX, LargeY, plot.type = "scatter",
              breaks.x = 10, var.c = LargeC,
              use.scientific = TRUE,
              data = MktDATA)
#  - Scientific notation for intervals' endpoints
distr.plot.xy(LargeX, LargeY, plot.type = "scatter",
              breaks.x = 10, var.c = LargeC,
              adj.breaks = FALSE,
              data = MktDATA)
#  - Scientific notation for intervals endpoints and axes
distr.plot.xy(LargeX, LargeY, plot.type = "scatter",
              var.c = LargeC, fitline = TRUE,
              adj.breaks = FALSE, use.scientific = TRUE,
              data = MktDATA)
distr.plot.xy(LargeX, LargeY, plot.type = "scatter",
              breaks.x = 10, var.c = LargeC,
              adj.breaks = FALSE, use.scientific = TRUE,
              data = MktDATA)
```

`distr.summary.x` *Summary statistics for a single variable*

Description

`distr.summary.x()` computes summary statistics of a vector or a factor.

Usage

```
distr.summary.x(
```

```
  x,
  stats = c("summary"),
  by1,
  by2,
  breaks.by1,
  interval.by1 = FALSE,
  breaks.by2,
  interval.by2 = FALSE,
  adj.breaks = TRUE,
  digits = 2,
  f.digits = 4,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x` An unquoted string identifying the variable whose distribution has to be summarized. `x` can be the name of a vector or a factor in the workspace or the name of one of the columns in the data frame specified in the `data` argument.

`stats` A character vector specifying the summary statistics to compute (more summaries can be specified). Specific types of summaries can be requested with the following options:

- `"summary"`: min, q1, median, mean, q3, max, sd, var;
- `"central"`: central tendency measures;
- `"dispersion"`: measures of dispersion;
- `"fivenumbers"`: five-number summary;
- `"quartiles"`, `"quintiles"`, `"deciles"`, `"percentiles"`: set of quantiles.

It is also possible to request the following statistics: `"q1"`, `"q2"`, `"q3"`, `"mean"`, `"median"`, `"mode"` (which returns the mode, the number of modes and the proportion of cases with modal value respectively), `"min"`, `"max"`, `"sd"`, `"var"`, `"cv"` (coefficient of variation), `"range"`, `"IQrange"` (interquartile range), and `"p1"`, `"p2"`,..., `"p100"` (i.e. specific percentiles).

`by1, by2` Unquoted strings identifying optional variables (typically taking few values/levels) used to build conditional summaries, that can be defined same way as `x`.

`breaks.by1, breaks.by2` Allow classifying the variables `by1` and/or `by2`, if *numerical*, into intervals. They can be integers indicating the number of intervals of equal width used to classify `by1` and/or `by2`, or vectors of increasing numeric values defining the endpoints of intervals (closed on the left and open on the right; the last interval is closed on the right too). To cover the entire range of values the maximum and the minimum values should be included between the first and the last break. It is possible to specify a set of breaks covering only a portion of the range of `by1` and/or `by2`.

interval.by1, interval.by2
: Logical values indicating whether by1 and/or by2 are variables measured in classes (TRUE). If the intervals for one variable are not consistent (e.g. overlapping intervals, or intervals with upper endpoint higher than the lower one), the variable is analysed as it is, even if results are not necessarily consistent; default to FALSE.

adj.breaks
: Logical value indicating whether the endpoints of intervals of the numerical variables by1 or by2, when classified into intervals, should be displayed avoiding scientific notation; default to TRUE.

digits, f.digits
: Integer values specifying the number of decimals used to round respectively summary statistics (default: digits=4) and proportions percentages (default: f.digits=2). If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument force.digits is set to TRUE.

force.digits
: Logical value indicating whether the requested summaries should be forcedly rounded to the number of decimals specified in digits and f.digits even if non-zero values are rounded to zero (default to FALSE).

use.scientific
: Logical value indicating whether numbers in tables should be displayed using scientific notation (TRUE); default to FALSE.

data
: An optional data frame containing x and/or the variables specifying the layers, by1 and by2. If not found in data, the variables are taken from the environment from which distr.summary.x() is called.

...
: Additional arguments to be passed to low level functions.

Value

A list whose elements are tables (converted to dataframes) with the requested summaries, possibly conditioned to by1 and/or by2. The values taken by the conditioning variables are arranged in standard order (logical, alphabetical or numerical order for vectors, order of levels for factors, ordered intervals for classified variables or for variables measured in classes).

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

summaries.plot.x() to graphically display conditioned tendency summaries of a univariate distribution.

distr.table.x() for tabulating a univariate distribution.

distr.plot.x() for plotting a univariate distribution.

Examples

```
data(MktDATA, package = "UBStats")
```

```
# Marginal summaries
# - Numerical variable: Default summaries
distr.summary.x(x = AOV, data = MktDATA)
# - Numerical variable: More summaries
distr.summary.x(x = AOV,
                stats = c("central","dispersion","fivenum"),
                data = MktDATA)
distr.summary.x(x = AOV, stats = c("mode","mean","sd","cv","fivenum"),
                data = MktDATA)
# - Character or factor (only proper statistics calculated)
distr.summary.x(x = LikeMost, stats = c("mode","mean","sd","cv","fivenum"),
                data = MktDATA)
distr.summary.x(x = Education, stats = c("mode","mean","sd","cv","fivenum"),
                data = MktDATA)

# Measures conditioned to a single variable
# - Numerical variable by a character vector
distr.summary.x(x = TotVal,
                stats = c("p5","p10","p25","p50","p75","p90","p95"),
                by1 = Gender, digits = 1, data = MktDATA)
# - Numerical variable by a numerical variable
#   classified into intervals
distr.summary.x(x = TotVal,
                stats = c("central","dispersion"),
                by1 = AOV, breaks.by1 = 5,
                digits = 1, data = MktDATA)
# - Numerical variable by a variable measured in classes
distr.summary.x(x = TotVal,
                stats = c("central","dispersion"),
                by1 = Income.S,
                interval.by1 = TRUE,
                digits = 1, data = MktDATA)

# Measures conditioned to two variables
distr.summary.x(x = TotVal, stats = "fivenumbers",
                by1 = Gender, by2 = Kids, data = MktDATA)
distr.summary.x(x = TotVal, stats = "fivenumbers",
                by1 = Income.S, by2 = Gender,
                interval.by1 = TRUE, data = MktDATA)
distr.summary.x(x = TotVal, stats = "fivenumbers",
                by1 = Gender, by2 = AOV,
                breaks.by2 = 5, data = MktDATA)

# Arguments adj.breaks and use.scientific
#  Variables with a very wide range
LargeX<-MktDATA$TotVal*1000000
LargeBY<-MktDATA$AOV*5000000
#  - Default: no scientific notation
distr.summary.x(LargeX, by1=LargeBY, breaks.by1 = 5,
                data = MktDATA)
#  - Scientific notation for summaries
distr.summary.x(LargeX, by1=LargeBY, breaks.by1 = 5,
                use.scientific = TRUE, data = MktDATA)
```

```
#  - Scientific notation for intervals endpoints
distr.summary.x(LargeX, by1=LargeBY, breaks.by1 = 5,
                adj.breaks = FALSE, data = MktDATA)
#  - Scientific notation for intervals endpoints and summaries
distr.summary.x(LargeX, by1=LargeBY, breaks.by1 = 5,
                adj.breaks = FALSE, use.scientific = TRUE,
                data = MktDATA)

# Output the list with the requested summaries
Out_TotVal<-distr.summary.x(x = TotVal,
                            by1 = Income.S, by2 = Gender,
                            interval.by1 = TRUE,
                            stats = c("central","fivenum","dispersion"),
                            data = MktDATA)
```

---

distr.table.x — *Analysis of a univariate distribution using frequency tables*

Description

`distr.table.x()` computes the frequency table of a vector or a factor.

Usage

```
distr.table.x(
  x,
  freq = c("counts", "proportions"),
  total = TRUE,
  breaks,
  adj.breaks = TRUE,
  interval = FALSE,
  f.digits = 2,
  p.digits = 0,
  d.digits = 5,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x` An unquoted string identifying the variable whose distribution has to be analysed. `x` can be the name of a vector or a factor in the workspace or the name of one of the columns in the data frame specified in the `data` argument.

freq
: A character vector specifying the set of frequencies to be displayed (more options are allowed). Allowed options (possibly abbreviated) are "counts", "percentages", "proportions", "densities" (only for variables classified into intervals or measured in classes), and "cumulative". If no frequency is specified, "counts" and "proportions" are displayed by default. If only "cumulative" is specified, counts and proportions are displayed too, with their respective cumulative frequencies.

total
: Logical value indicating whether the sum of the requested frequencies should be added to the table; default to TRUE.

breaks
: Allows to classify a *numerical* variable x into intervals. It can be an integer indicating the number of intervals of equal width used to classify x, or a vector of increasing numeric values defining the endpoints of intervals (closed on the left and open on the right; the last interval is closed on the right too). To cover the entire range of values the maximum and the minimum values should be included between the first and the last break. It is possible to specify a set of breaks covering only a portion of the x range.

adj.breaks
: Logical value indicating whether the endpoints of intervals of a numerical variable x when classified into intervals should be displayed avoiding scientific notation; default to TRUE.

interval
: Logical value indicating whether x is a variable measured in intervals (TRUE). If the detected intervals are not consistent (e.g. overlapping intervals, or intervals with upper endpoint higher than the lower one), the variable is tabulated as it is, even if results are not necessarily consistent; default to FALSE.

f.digits, p.digits, d.digits
: Integer values specifying the number of decimals used to round respectively proportions (default: f.digits=2), percentages (default: p.digits=0), and densities (default: d.digits=5). If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument force.digits is set to TRUE.

force.digits
: Logical value indicating whether frequencies and densities should be forcedly rounded to the number of decimals specified in f.digits, p.digits, and d.digits even if non-zero values are rounded to zero (default to FALSE).

use.scientific
: Logical value indicating whether numbers in tables (typically densities) should be displayed using scientific notation (TRUE); default to FALSE.

data
: An optional data frame containing x. If not found in data, x is taken from the environment from which distr.table.x() is called.

...
: Additional arguments to be passed to low level functions.

Value

A table (converted to dataframe) listing the values taken by the variable, arranged in standard order (logical, alphabetical or numerical order for vectors, order of levels for factors, ordered intervals for classified variables or for variables measured in classes), and the requested set of frequencies.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

distr.plot.x() for plotting a univariate distribution.

distr.table.xy() for tabulating a bivariate distribution.

distr.plot.xy() for plotting a bivariate distribution.

Examples

```
data(MktDATA, package = "UBStats")

# Character vectors, factors, and discrete numeric vectors
distr.table.x(Education, data = MktDATA)

distr.table.x(Children, freq = c("count","prop","cum"),
              data = MktDATA)

# Numerical variable classified into intervals
# - Classes of equal width
distr.table.x(AOV, breaks = 6, freq = c("Count","Prop","Perc","Cum"),
              p.digits = 2, data = MktDATA)
# - Classes with specified endpoints
distr.table.x(AOV, breaks = c(0,20,30,50,100,180),
              freq = c("Count","Perc","Cum","Densities"),
              p.digits = 2, data = MktDATA)
# Numerical variable measured in classes
# - Variable measured in classes
distr.table.x(Income, freq = c("count","prop","cum","dens"),
              interval = TRUE, data = MktDATA)
# - An example of non-consistent intervals.
#   Densities are not calculated
x.inconsistent <- c(rep("0;10",30),rep("10;20",25),rep("25;8",25),
                    rep("15;31",15),rep("20;45",16),rep("30;40",18))
distr.table.x(x.inconsistent, freq = c("count","prop","cum","dens"),
              interval = TRUE)

# Arguments adj.breaks, use.scientific, and force.digits
#  A variable with a very wide range (very small densities)
LargeX <- MktDATA$AOV*5000000
# - Default: manages possible excess of rounding
distr.table.x(LargeX, breaks = 5,
              freq = c("count","percent","densities"))
# - Forcing digits to the default values
distr.table.x(LargeX, breaks = 5,
              freq=c("count","percent","dens"),
              force.digits = TRUE)
#  - Scientific notation for frequencies/densities
distr.table.x(LargeX, breaks = 5,
              freq = c("count","percent","dens"),
              use.scientific = TRUE)
#  - Scientific notation both for intervals' endpoints
#    and for frequencies/densities
distr.table.x(LargeX, breaks = 5, adj.breaks = FALSE,
```

```
              freq = c("count","percent","dens"),
              use.scientific = TRUE)

# Output a dataframe with the table
table.AOV<-distr.table.x(AOV, breaks = c(0,20,30,50,100,180),
                         freq = c("Count","Perc","Cum","Dens"),
                         data = MktDATA)
```

---

distr.table.xy *Analysis of a bivariate distribution using cross-tables*

Description

`distr.table.xy()` displays tables of joint or conditional distributions.

Usage

```
distr.table.xy(
  x,
  y,
  freq = "counts",
  freq.type = "joint",
  total = TRUE,
  breaks.x,
  breaks.y,
  adj.breaks = TRUE,
  interval.x = FALSE,
  interval.y = FALSE,
  f.digits = 2,
  p.digits = 0,
  force.digits = FALSE,
  data,
  ...
)
```

Arguments

`x, y`	Unquoted strings identifying the variables whose joint distribution has to be analysed. `x` and `y` can be the name of a vector or a factor in the workspace or the name of one of the columns in the data frame specified in the `data` argument. Note that in the table `x` is displayed on the *rows* and `y` on the *columns*.
`freq`	A character vector specifying the set of frequencies to be displayed (more options are allowed). Allowed options (possibly abbreviated) are `"counts"`, `"percentages"` and `"proportions"`.

freq.type — A character vector specifying the types of frequencies to be displayed (more types are allowed). Allowed options are `joint` (default) for joint frequencies, `x|y` (or `column`) for the distributions of `x` conditioned to `y`, and `y|x` (or `row`) for the distributions of `y` conditioned to `x`.

total — Logical value indicating whether the sum of the requested frequencies should be added to the table; default to `TRUE`.

breaks.x, breaks.y — Allow to classify the variables `x` and/or `y`, if *numerical*, into intervals. They can be integers indicating the number of intervals of equal width used to classify `x` and/or `y`, or vectors of increasing numeric values defining the endpoints of the intervals (closed on the left and open on the right; the last interval is closed on the right too). To cover the entire range of values taken by one variable, the maximum and the minimum values should be included between the first and the last break. It is possible to specify a set of breaks covering only a portion of the variable's range.

adj.breaks — Logical value indicating whether the endpoints of intervals of a numerical variable (`x` or `y`) when classified into intervals should be displayed avoiding scientific notation; default to `TRUE`.

interval.x, interval.y — Logical values indicating whether `x` and/or `y` are variables measured in classes (`TRUE`). If the detected intervals are not consistent (e.g. overlapping intervals, or intervals with upper endpoint higher than the lower one), the variable is tabulated as it is, even if results are not necessarily consistent; default to `FALSE`.

f.digits, p.digits — Integer values specifying the number of decimals used to round respectively proportions (default: `f.digits=2`) and percentages (default: `p.digits=0`). If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument `force.digits` is set to `TRUE`.

force.digits — Logical value indicating whether proportions and percentages should be forcedly rounded to the number of decimals specified in `f.digits` and `p.digits` even if non-zero values are rounded to zero (default to `FALSE`).

data — An optional data frame containing `x` and/or `y`. If not found in `data`, the variables are taken from the environment from which `distr.table.xy()` is called.

... — Additional arguments to be passed to low level functions.

Value

A list whose elements are the requested tables (converted to dataframes) listing the values taken by the two variables arranged in standard order (logical, alphabetical or numerical order for vectors, order of levels for factors, ordered intervals for classified variables or for variables measured in classes) and the specified joint or conditional types of frequencies.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

distr.plot.xy() for plotting a bivariate distribution.

distr.table.x() for tabulating a univariate distribution.

distr.plot.x() for plotting a univariate distribution.

Examples

```
data(MktDATA, package = "UBStats")

# Character vectors, factors, and discrete numeric vectors
# - Default: joint counts
distr.table.xy(LikeMost, Children, data = MktDATA)

# - Joint and conditional distribution of x|y
#   counts and proportions, no totals
distr.table.xy(LikeMost, Education, freq = c("counts","Prop"),
               freq.type = c("joint","x|y"), total = FALSE,
               data = MktDATA)
# - Joint and conditional row and column distributions (%)
distr.table.xy(CustClass, Children, freq = "Percentages",
               freq.type = c("joint","row","column"),
               data = MktDATA)

# Numerical variables classified or measured in classes
# - A numerical variable classified into intervals
#   and a factor
distr.table.xy(CustClass, TotPurch,
               breaks.y = c(0,5,10,15,20,35),
               freq = c("Counts","Prop"), freq.type = "y|x",
               data = MktDATA)

# - Two numerical variables, one measured in classes
#   and the other classified into intervals
distr.table.xy(Income.S, TotPurch, interval.x = TRUE,
               breaks.y = c(0,5,10,15,20,35),
               freq = c("Counts","Prop"),
               freq.type = c("row","col"), data = MktDATA)

# Argument force.digits
# - Default: manages possible excess of rounding
distr.table.xy(CustClass, Children, freq = "Percentages",
               freq.type = c("x|y"),data = MktDATA)
# - Force to the required rounding
distr.table.xy(CustClass, Children, freq = "Percentages",
               freq.type = c("x|y"),
               force.digits = TRUE, data = MktDATA)

# Output the list with the requested tables
tables.xy<-distr.table.xy(Income.S, TotPurch,
                          interval.x = TRUE,
                          breaks.y = c(0,5,10,15,20,35),
```

```
freq = c("Counts","Prop"),
freq.type = c("joint","row","col"),
data = MktDATA)
```

ESG_Returns — *Data ESG_Returns: Simulated ESG Performance and Financial Returns*

Description

This dataset was generated with the assistance of AI. Its structure and parameter values were designed to reproduce stylised empirical patterns discussed in academic studies on the relationship between environmental, social, and governance (ESG) performance and financial returns during crisis and normal periods. However, this dataset is entirely synthetic and created for **teaching purposes only**; it does not represent real firms, markets, or observed behaviours and should not be interpreted as empirical evidence.

Usage

```
data(ESG_Returns)
```

Format

A data frame with n = 18000 observations and 10 variables:

Firm (num) Firm identification number

Week (date) weekly time period to which each firm-level observation refers, covering 60 consecutive weeks beginning on 7 June 2020

Industry (chr) firm's sector classification (`Consumer`, `Energy`, or `Technology`)

Return (num) realised stock Return (in percentage points), with positive values indicating firm-specific gains and negative values indicating losses. These returns are risk-adjusted: the common market component has been removed so that they reflect the firm's idiosyncratic performance rather than general market movements

ESG (num) score, measured at the beginning of each month, which ranges between 0 and 100 and captures the firm's monthly sustainability performance

Crisis (int) dummy variable identifying a market stress period within the sample, approximately corresponding to October-December 2020. It is intended to mimic a phase of renewed financial uncertainty linked to the COVID-19 pandemic, when equity markets came under renewed pressure because of the second wave of infections and the reintroduction of restrictions, before the first positive vaccine announcements improved sentiment

Beta (num) the firm's sensitivity to market movements estimated from a 12-month pre-sample window

Size (num) a proxy for firm scale

Volatility (num) the firm-specific volatility of returns, measured as the standard deviation of daily returns in the month immediately preceding the first week of the sample

Momentum (num) the cumulative return over the previous 6 to 12 months (excluding the most recent month)

Details

The data frame contains (simulated) weekly observations for 300 firms over 60 weeks (starting from 7 June 2020) across three industry sectors. For each observation (firm-week), information is available on several groups of variables. Its structural variables identify the firm, the observation week, the firm's size, and its industry sector. The main variables of interest are the firm's weekly risk-adjusted return, its monthly ESG score, and the Crisis indicator for the market-stress period. The dataset also includes market-related characteristics, such as Beta and Momentum, together with a risk measure represented by firm-specific Volatility.

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

`Grocery_NE` *Data Grocery_NE: Grocery Chain Customer Survey (North-East Italy)*

Description

This dataset contains the variables from a survey on a sample of customers from a grocery chain operating in Italy. Specifically, data refer to the sub-sample of customers who regularly shop in stores located in north-eastern Italian region. Information is available on the activity observed in the last year (number of visits and transactions, amount spent), on customers' satisfaction with the retailer, and on the perceived weakness of the most frequently visited store.

Usage

```
data(Grocery_NE)
```

Format

A data frame with $n = 3114$ observations and 26 variables:

Id (int) customer identification

Sex (chr) customer's sex at birth (`F` or `M`)

Age (num) customer's age (in years)

Tenure (factor) Customer tenure in years (coded in classes: `[0,1)`, `[1,3)`, `[3,6)`, `[6,10)`, `[10,15)`, `[15,25)`, `[25,35)`)

Status (chr) customers' status (`Active`, `Silent`)

FavShop (chr) store customers visit the most (`NE.01`, `NE.02`, ..., `NE.07`)

FavShop_Region (chr) macro region where the favourite shop is located (here, only `North-East`)

TotShops (num) number of stores visited by the customer

WeekDay (chr) preferred shopping day (`1:Mon`, `2:Tue`, ..., `7:Sun`)

TimeSlot (chr) preferred shopping time slot (08-12, 12-14, 14-17, 17-23)

Satisf (chr) Customer's overall declared satisfaction with the retailer (VLow, Low, QLow – that is, moderately low; Med, QHigh – that is, moderately high; High, VHigh)

Complaint (factor) major weakness of the typically visited store (Quality&Variety, Prices, Resupply, Staff, Crowded, Checkout)

NMonths (num) number of months in which the customers visited a store at least once

MonthExp (chr) amount spent per month, in classes ([0,50), [50,100), [100,150), [150,200), [200,300), [300,400), [400,600), [600,800])

Transact_M (num) transactions per month

TBP (num) time between purchases

TotExp (num) amount spent in the last 12 months

TotVisits (num) total number of visits in the last year

Receipt (num) average receipt (transaction value)

Visits_Regular (num) score (ranging from 0 to 100) indicating the regularity of customer's shopping trips

Spending_Regular (num) score (ranging from 0 to 100) indicating the regularity of customer's spending

Discount (num) average discount on purchased products

CrossSelling (num) index measuring how diverse a customer's purchases are across different product categories

RecencyScore (num) index reflecting how recently a customer last interacted with a company (higher scores reflecting more recent interactions)

MonetaryScore (num) index reflecting the overall value of the customer to the business

FrequencyScore (num) index reflecting the regularity or repetition of customer transactions, with higher scores indicating more frequent purchases

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

JobEngage *Data JobEngage: Employee Job Engagement Survey*

Description

The data frame contains the results of a survey conducted to analyse job engagement (the emotional commitment of employees to their organisation) within a large company, focusing on factors previously identified through qualitative methods (including focus groups and exploratory interviews). *Constructs* related to the possible determinants of engagement were measured using Likert scale items, where respondents were asked to indicate their level of agreement with a series of statements pertaining to each factor. For example, perceived compensation fairness was measured through items such as 'My pay reflects the responsibilities of my role' and 'I am fairly compensated relative to others in similar jobs'. The responses to different items referring to the same construct were then averaged to form a composite score.

Usage

```
data(JobEngage)
```

Format

A data frame with n = 256 observations and 11 variables:

Id (int) respondent id

Engagement (num) score reflecting the respondent's level of engagement

Leadership (num) respondent evaluation of managers, as well as the quality of their relationship with them

Team (num) respondent evaluation of co-workers, as well as the quality of their relationship with them

Environment (num) respondent evaluation of the quality of the work environment

Growth (num) respondent perceived evaluation of the opportunity of professional development and advancement; it captures the degree to which employees see clear paths to promotion within the company and feel they are provided with opportunities to learn and develop skills

Compensation (num) respondent perceived fairness of compensation

WorkPlace (num) respondent evaluation of the physical work environment, including aspects such as lighting, noise level, the amount of space they have access to, and ergonomics

RoleConflict (num) extent to which employees feel that the expectations connected with their role are inconsistent or incompatible with each other (for example, when they receive conflicting instructions from different supervisors or face demands that are difficult to reconcile in day-to-day tasks)

Flex (num) degree to which employees experience freedom, comfort, and control in how and where they work (it reflects work-life balance, autonomy in tasks and time management)

RemoteWork (factor) remote-work arrangements (`No`: the employee works entirely on-site; `Structured`: remote work on predefined days or under fixed rules; and `Flexible`: remote work without constraints)

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

`LM.output` *Extract Model Residuals and other Regression Diagnostics*

Description

`LM.output()` Provides fitted values, residuals and other basic quantities used to check the quality of regression fits.

Usage

```
LM.output(object, data)
```

Arguments

object	An object returned by function lm.
data	An optional data frame containing the data frame possibly specified in the call of function lm.

Value

A dataframe containing the variables in the model and the model's fitted values, residuals and influence statistics, merged with the dataframe specified in the call of function lm, or with the dataframe possibly specified in data (if it is consistent with the model's output)

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

Examples

```
data(MktDATA, package = "UBStats")

# Model and output based on a given dataframe
mod1 <- lm(TotVal ~ Baseline + Kids + Age, data = MktDATA)
# Equivalent calls (since data is specified in lm()
mod1_out <- LM.output(mod1, data = MktDATA)
dim(mod1_out)
mod1_out <- LM.output(mod1)
dim(mod1_out) # same as above

# Model based on a dataframe's columns
mod2 <- lm(MktDATA$TotVal ~ MktDATA$Baseline +
           MktDATA$Kids + MktDATA$Age)
mod2_out <- LM.output(mod2)
# note: colnames in mod2_out
colnames(mod2_out)
# note that the dataframe in 'data' is not considered
# as compatible, because the names of columns differ
mod2_out <- LM.output(mod2, data = MktDATA)
```

Marketing_Mix — *Data Marketing_Mix: Marketing Investments and Digital Performance*

Description

This data frame contains data on marketing campaigns conducted across municipalities in Italy. It provides information on marketing investments in each area, including spending on online **paid** *search* (paying to have a website or ad appear at the top of a search engine results page), online *display* (paying for visual ads on websites, apps, or social media platforms) and *retargeting* (targeting users who previously interacted with the brand online – e.g. visiting a website, using an app, or engaging with content – but did not complete a desired action, like making a purchase).

Usage

```
data(Marketing_Mix)
```

Format

A data frame with $n = 247$ observations and 13 variables:

Municipality (num) municipality id

MacroArea (factor) macro geographical region (A, B, C, D, E)

LocalPromo1 (num) binary variable indicating whether Campaign 1 is active (1) or not (0)

LocalPromo2 (num) binary variable indicating whether Campaign 1 is active (1) or not (0)

Search_Spend (num) marketing expenses on online search advertisements (e.g., Google Ads)

Display_Spend (num) marketing expenses on online display advertisements (e.g., banner ads)

Retargeting_Spend (num) marketing expenses on retargeting (e.g., ads targeting previous visitors)

KPI_Interact (num) number of interactions (e.g., clicks, engagements) across all channels

KPI_Paid (num) number of sessions or visits generated through paid traffic campaigns

KPI_Organic (num) number of sessions or visits generated through organic (non-paid) traffic

Sales (num) Total sales units generated during the period

NewUsers (num) number of new customers acquired during the period

Arpu (num) Average Revenue Per User, calculated as total revenue divided by active users (then rounded)

Details

These investments affect key digital performance indicators (KPIs), such as customer *interactions* and *paid* or *organic* website sessions (visits resulting from sponsored or non-paid links), which represent intermediate outcomes that may influence *sales* and new *customer acquisition* – the campaign's key final targets. The data frame also includes information about two types of *local promotions* possibly running in each municipality. All variables in the data frame – investments, KPIs and outcomes – are adjusted to account for structural differences between municipalities.

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

MktDATA *Data MktDATA:*

Description

This dataset is a modification of the original MktDATA.Orig dataset and it is provided for user convenience.

Usage

```
data(MktDATA)
```

Format

A data frame with n = 2224 observations and 26 variables.

MktDATA.Orig *Data: MktDATA.Orig*

Description

This dataset contains the variables from a survey on a set of customers of a company operating in the retail food sector. The company sells products from 3 major categories (referred to as A, B, C) The customers can order and acquire products in the company physical stores, or through the company's website (in this case, they can order on the website and pick up the order in one store). Information is collected on customers' activity in the last two years (observation period), as well as some information retrieved through questionnaires or fidelity cards. During such period different marketing strategies were adopted to improve customers' fidelization, and 5 marketing campaigns were launched; a last campaign was launched at the end of the observation period.

Usage

```
data(MktDATA.Orig)
```

Format

A data frame with n = 2224 observations and the following 19 variables (levels of the variables listed in alphabetical order):

CustId (num) customer's identification label

Gender (chr) customer's gender (`F`, `M`)

Age (num) customer's age (in years)

Education (chr) customer's level of education(`College`, `Graduate`, `HighSchool`, `Post-Grad`)

Marital_Status (chr) customer's marital status (`Divorced`, `Married`, `Single`, `Together`, `Widow`)

Children (num) number of children in the household

Kids (num) number of kids aged less than 12 in the household

Income (chr) customer's income (measured in classes)

Baseline (num) index (from 0 to 1) assigned by the marketing dept indicating how promising the customer was judged at the beginning of the observation period

LikeMost (chr) Most frequently bought category in the last two years (`P.A`, `P.B`, `P.C`)

TotVal (num) amount spent in the last 2 years

NPickUp_Purch (num) number of purchases made through company's website and picked up in physical store

NWeb_Purch (num) number of purchases made through company's website and delivered at home

NStore_Purch (num) number of purchases made in a physical store

NDeals (num) number of products purchases with discount

CustClass (chr) customer's classification (assigned by the marketing dept) based on past profitability (`Bronze`, `Gold`, `Platinum`, `Silver`)

PastCampaigns (num) number of offers accepted by the customer in the last 2 years' marketing campaigns

LastCampaign (num) binary variable (0/1) indicating whether (1) or not (0) the customer accepted the offer in the campaign launched at the end of the observation period

WouldSuggest (chr) variable signalling whether (`Yes`) or not (`No`) the customer declared they would suggest the company's products to friends and family

Source

The data set has been adapted from https://www.kaggle.com/code/dmitryuarov/customers-clustering-eda.

`Spending` *Data Spending: Household Income and Spending Patterns*

Description

The data frame contains information on a sample of households within a specific demographic segment; it is used to explore the relationship between monthly spending on *Essentials* and *Discretionary* goods and monthly disposable *Income*. All variables in the data frame are normalised to ensure comparability across households of different sizes. Income is expressed in thousands of pounds per month per equivalised household to avoid scale artefacts.

Usage

```
data(Spending)
```

Format

A data frame with $n = 751$ observations and 4 variables:

Id (int) household identification number

Essentials (num) household monthly expenditures in essential goods (e.g., groceries, utilities, basic transport); score normalised to ensure comparability across household of different sizes

Discretionary (num) household monthly expenditures in discretionary goods (e.g., dining out, travel, electronics); score normalised to ensure comparability across household of different sizes

Income (num) monthly disposable income (in thousands of pounds) per equivalised household (meaning adjusted to allow meaningful comparisons across households with different size and composition)

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

`summaries.plot.x` *Plot of central and non-central conditional tendency measures for a single numeric variable*

Description

`summaries.plot.x()` plots location statistics for a numeric vector conditioned to the levels of one or more variables.

Usage

```
summaries.plot.x(
  x,
  stats = "mean",
  plot.type = "bars",
  conf.level = 0.95,
  by1,
  by2,
  breaks.by1,
  interval.by1 = FALSE,
  breaks.by2,
  interval.by2 = FALSE,
  adj.breaks = TRUE,
  bw = FALSE,
  color = NULL,
  legend = TRUE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x` An unquoted string identifying a *numerical* variable whose tendency measures have to be graphically displayed. `x` can be the name of a vector in the workspace or the name of one of the columns in the data frame specified in the `data` argument.

`stats` A single character specifying the conditioned tendency measure/s to display in the plot. The available options are `"mean"`, `"median"`, `"ci.mean"` (to plot the means and the confidence intervals for the means), and specific sets of quantiles, namely `"quartiles"`, `"quintiles"`, `"deciles"`, and `"percentiles"` (note that for quantiles only one single layer can be specified).

`plot.type` A single character specifying the type of plot used to compare the requested measures conditioned to the levels of one variable, `by1`, possibly broken down by the levels of a second variable, `by2`, if specified. The available options are:

- `"bars"`: Available only when `stats` is `"mean"`, `"median"`, or `"ci.mean"` and one single layer (`by1`) is specified. For each level of `by1` a bar is built whose height coincides with the conditional mean or median. Confidence intervals for the means are reported when `stats = "ci.mean"`.
- `"points"`: Available only when `stats` is `"mean"`, `"median"`, and `"ci.mean"`. Confidence intervals for the means are reported when `stats = "ci.mean"` and one single layer is specified.
- `"lines"`: Points joined by lines; this is the unique option available for quantiles.

`conf.level` A number between 0 and 1 indicating the confidence level of the intervals for the conditional means when `stats = "ci.mean"` is specified (default to 0.95).

`by1, by2` Unquoted strings identifying variables (typically taking few values/levels) used to build conditional summaries, that can be defined same way as `x`. At least one layer has to be specified. The conditional measures are plotted against the values of `by1`, broken down by the levels of `by2`, if specified.

`breaks.by1, breaks.by2`
Allow classifying the variables `by1` and/or `by2`, if *numerical*, into intervals. They can be integers indicating the number of intervals of equal width used to classify `by1` and/or `by2`, or vectors of increasing numeric values defining the endpoints of intervals (closed on the left and open on the right; the last interval is closed on the right too). To cover the entire range of values the maximum and the minimum values should be included between the first and the last break. It is possible to specify a set of breaks covering only a portion of the range of `by1` and/or `by2`.

`interval.by1, interval.by2`
Logical values indicating whether `by1` and/or `by2` are variables measured in classes (`TRUE`). If the intervals for one variable are not consistent (e.g. overlapping intervals, or intervals with upper endpoint higher than the lower one), the variable is analysed as it is, even if results are not necessarily consistent; default to `FALSE`.

`adj.breaks` Logical value indicating whether the endpoints of intervals of the numerical variables `by1` or `by2`, when classified into intervals, should be displayed avoiding scientific notation; default to `TRUE`.

bw	Logical value indicating whether plots should be colored in scale of greys (TRUE) rather than using a standard palette (FALSE, default).
color	Optional string vector to specify colors to use in the plot rather than a standard palette (NULL, default).
legend	Logical value indicating whether a legend should be displayed in the plot (legend = TRUE; default) or not (legend = FALSE).
use.scientific	Logical value indicating whether numbers on axes should be displayed using scientific notation (TRUE); default to FALSE.
data	An optional data frame containing x and/or the variables specifying the layers, by1 and by2. If not found in data, the variables are taken from the environment from which distr.summary.x() is called.
...	Additional arguments to be passed to low level functions.

Value

A table (converted to dataframe) reporting the requested statistics conditioned to the levels of the specified layers.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

distr.summary.x() for tabulating summary measures of a univariate distribution.

distr.plot.x() for plotting a univariate distribution.

distr.table.x() for tabulating a univariate distribution.

Examples

```
data(MktDATA, package = "UBStats")

# Means (and their CI) or medians by a single variable
# - Barplot of means (default) by a character
summaries.plot.x(x = TotVal, stats = "mean",
               by1 = Gender, data = MktDATA)
# - Barplot of medians by a numerical variable
#   classified into intervals: user-defined color
summaries.plot.x(x = TotVal, stats = "median",
                 by1 = AOV, breaks.by1 = 5,
                 color = "purple", data = MktDATA)
# - Lineplot of means and their CI by a variable
#   measured in classes
summaries.plot.x(x = TotVal,
                 stats = "ci.mean", plot.type = "lines",
                 by1 = Income.S, interval.by1 = TRUE,
                 data = MktDATA)
# - Barplot of means and their CI by a
#   numerical variable; change the confidence level
```

```
summaries.plot.x(x = TotVal,
                 stats = "ci.mean", conf.level = 0.90,
                 plot.type = "bars",
                 by1 = NWeb_Purch, data = MktDATA)
# - Note: no plot built for a variable with
#   too many levels (>20)
# summaries.plot.x(x = TotVal,
#                  stats = "ci.mean", plot.type = "lines",
#                  by1 = AOV, data = MktDATA)

# Quantiles by a single variable
# - Only lines plots allowed for quantiles
summaries.plot.x(x = Baseline,
                 stats = "deciles", plot.type = "lines",
                 by1 = NDeals, data = MktDATA)
summaries.plot.x(x = Baseline,
                 stats = "quartiles", plot.type = "lines",
                 by1 = Marital_Status, data = MktDATA)

# Means and medians by two variables
# - Default: only lines allowed
summaries.plot.x(x = TotVal, stats = "mean",
                 by1 = Education, by2 = Kids,
                 data = MktDATA)
summaries.plot.x(x = TotVal, stats = "median",
                 by1 = Income.S, by2 = Gender,
                 interval.by1 = TRUE,
                 data = MktDATA)
summaries.plot.x(x = Baseline, stats = "mean",
                 by1 = CustClass, by2 = AOV,
                 breaks.by2 = 5, data = MktDATA)
# - "ci.mean" not allowed with two layers
CustClass_Kids<-paste0(MktDATA$CustClass,"-",MktDATA$Kids)
summaries.plot.x(x = Baseline, stats = "ci.mean",
                 conf.level = 0.99, by1 = CustClass_Kids,
                 color = "gold", data = MktDATA)

# Arguments adj.breaks and use.scientific
#  Variables with a very wide range
LargeX<-MktDATA$TotVal*1000000
LargeBY<-MktDATA$AOV*5000000
#  - Default: no scientific notation
summaries.plot.x(LargeX, plot.type = "bars",
                 by1=LargeBY, breaks.by1 = 5, data = MktDATA)
#  - Scientific notation for summaries (axes)
summaries.plot.x(LargeX, plot.type = "lines",
                 by1=LargeBY, breaks.by1 = 5,
                 use.scientific = TRUE, data = MktDATA)
#  - Scientific notation for intervals endpoints
summaries.plot.x(LargeX, stats = "ci.mean",
                 plot.type = "lines",
                 by1=LargeBY, breaks.by1 = 5,
                 adj.breaks = FALSE, data = MktDATA)
```

```
#  - Scientific notation for intervals endpoints and summaries
summaries.plot.x(LargeX, stats = "quartiles",
                 plot.type = "lines",
                 by1=LargeBY, breaks.by1 = 5,
                 adj.breaks = FALSE, use.scientific = TRUE,
                 data = MktDATA)

# Output the table with the requested summaries
Out_TotVal<-summaries.plot.x(x = TotVal, stats = "ci.mean",
                             by1 = Education, data = MktDATA)
```

`TEST.diffmean` *Tests on the difference between means*

Description

`TEST.diffmean()` tests hypotheses on the difference between the means of two independent or paired populations.

Usage

```
TEST.diffmean(
  x,
  y,
  type = "independent",
  mdiff0 = 0,
  alternative = "two.sided",
  sigma.x = NULL,
  sigma.y = NULL,
  by,
  sigma.by = NULL,
  sigma.d = NULL,
  var.test = FALSE,
  digits = 2,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x, y` Unquoted strings identifying the *numeric* variables with the same length whose means have to be compared. `x` and `y` can be the names of vectors in the workspace or the names of columns in the data frame specified in the `data` argument. It is possible to use a mixed specification (e.g, one vector and one column in data).

`type`	A length-one character vector specifying the type of samples. Allowed values are `"independent"` or `"paired"`.
`mdiff0`	Numeric value that specifies the null hypothesis to test for (default is 0).
`alternative`	A length-one character vector specifying the direction of the alternative hypothesis. Allowed values are `"two.sided"` (difference between populations' means differs from `mdiff0`; default), or `"less"` (difference between populations' means is lower than `mdiff0`), or `"greater"` (difference between populations' means is higher than `mdiff0`).
`sigma.x`, `sigma.y`	Optional numeric values specifying the possibly known populations' standard deviations (when `x` and `y` are specified). If `NULL` (default) standard deviations are estimated using the data.
`by`	Optional unquoted string, available only when `type = "independent"`, identifying a variable (of any type), defined same way as `x`, taking only **two** values used to split `x` into two **independent samples**. Given the two *ordered* values taken by `by` (alphabetical or numerical order, or order of the levels for factors), say *by1* and *by2*, hypotheses are tested on the difference between the populations means in the *by1*- and in the *by2*-group. Note that only **one** between `y` and `by` can be specified.
`sigma.by`	Optional numeric value specifying the possibly known standard deviations for the two *independent* samples identified via `by` (when `x` and `by` are specified). `sigma.by` can be a single value indicating the same standard deviation in the two by-groups, or a vector with two values, specifying the standard deviations in the two by-groups. To avoid errors, in the latter case the vector should be named, with names coinciding with the two levels of `by`.
`sigma.d`	Optional numeric value specifying the possibly known standard deviation of the difference when samples are **paired**.
`var.test`	Logical value indicating whether to run a test on the equality of variance for two (**independent**) samples or not (default).
`digits`	Integer value specifying the number of decimals used to round statistics; default to 2. If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument `force.digits` is set to `TRUE`.
`force.digits`	Logical value indicating whether reported values should be forcedly rounded to the number of decimals specified in `digits` even if non-zero values are rounded to zero (default to `FALSE`).
`use.scientific`	Logical value indicating whether numbers in tables should be displayed using scientific notation (`TRUE`); default to `FALSE`.
`data`	An optional data frame containing `x` and/or `y` or `by`. If not found in `data`, the variables are taken from the environment from which `TEST.diffmean()` is called.
`...`	Additional arguments to be passed to low level functions.

Value

A table reporting the results of the test on the difference between the populations' means. For independent samples in the case of unknown variances the test is run both under the assumption that the variances are equal and under the assumption that they differ, using percentiles from both the normal and the Student's t distribution.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

CI.diffmean() to build confidence intervals for the difference between two populations' means.

Examples

```
data(MktDATA, package = "UBStats")

# Independent samples (default type), UNKNOWN variances
#  Bilateral test on difference between means of males and females
#  - Using x,y: build vectors with data on the two groups
AOV_M <- MktDATA$AOV[MktDATA$Gender == "M"]
AOV_F <- MktDATA$AOV[MktDATA$Gender == "F"]
TEST.diffmean(x = AOV_M, y = AOV_F, mdiff0 = 0)
#  - Using x,by: groups identified by ordered levels of by
TEST.diffmean(x = AOV, by = Gender, mdiff0 = 0, data = MktDATA)
#    Since order is F, M, hypothesis are on mean(F) - mean(M)
#    To test hypotheses on mean(M) - mean(F)
Gender.R <- factor(MktDATA$Gender, levels = c("M", "F"))
TEST.diffmean(x = AOV, by = Gender.R , mdiff0 = 0,
              data = MktDATA)
#  - Testing also hypotheses on equality of unknown variances
TEST.diffmean(x = AOV_M, y = AOV_F, mdiff0 = 0,
              var.test = TRUE)

#  - Output results: test on differences
out.test_diffM<-TEST.diffmean(x = AOV_M, y = AOV_F)
#  - Output results: list with both test on means and variances
out.test_diffM.V<-TEST.diffmean(x = AOV_M, y = AOV_F, var.test = TRUE)

# Independent samples (default type), KNOWN variances
#  Test hypotheses on the difference between means of males and females
#  - Using x,y: build vectors with data on the two groups
AOV_M <- MktDATA$AOV[MktDATA$Gender == "M"]
AOV_F <- MktDATA$AOV[MktDATA$Gender == "F"]
TEST.diffmean(x = AOV_M, y = AOV_F, mdiff0 = 10,
              alternative = "greater", sigma.x = 10, sigma.y = 20)
#  - Using x,by: groups identified by ordered levels of by
#    Adjust considering the ordering of levels
TEST.diffmean(x = AOV, by = Gender, mdiff0 = -10,
              alternative = "less",
              sigma.by = c("M" = 10, "F"=20), data = MktDATA)
```

```
#    To change the sign, order levels as desired
Gender.R <- factor(MktDATA$Gender, levels = c("M", "F"))
TEST.diffmean(x = AOV, by = Gender.R, mdiff0 = 10,
              alternative = "greater",
              sigma.by = c("M" = 10, "F"=20), data = MktDATA)
#  - Output results
out.test_diffM<-TEST.diffmean(x = AOV_M, y = AOV_F, mdiff0 = 10,
                              alternative = "greater",
                              sigma.x = 10, sigma.y = 20)

# Paired samples: UNKNOWN variances
# - Default settings
TEST.diffmean(x = NStore_Purch, y = NWeb_Purch,
              type = "paired",
              mdiff0 = 1.5, alternative = "greater", data=MktDATA)
# Paired: KNOWN variances
TEST.diffmean(x = NStore_Purch, y = NWeb_Purch,
              type = "paired", mdiff0 = 1.5, alternative = "greater",
              sigma.d = 2, data = MktDATA)
#  - Output results
out.test_diffM<-TEST.diffmean(x = NStore_Purch,
                              y = NWeb_Purch,
                              type = "paired", mdiff0 = 1.5, alternative = "greater",
                              sigma.d = 2, data = MktDATA)

# Arguments force.digits and use.scientific
#  An input variable taking very low values
SmallX<-MktDATA$AOV/50000
SmallX_M <- SmallX[MktDATA$Gender == "M"]
SmallX_F <- SmallX[MktDATA$Gender == "F"]
#  - Default output
TEST.diffmean(x = SmallX_M, y = SmallX_F)
#  - Request to use the exact number of digits (default, 2)
TEST.diffmean(x = SmallX_M, y = SmallX_F,
              force.digits = TRUE)
#  - Request to allow scientific notation
TEST.diffmean(x = SmallX_M, y = SmallX_F,
              use.scientific = TRUE)
```

TEST.diffprop *Tests on the difference between proportions*

Description

TEST.diffprop() tests hypotheses on the difference between the proportion of successes in two independent populations.

Usage

```
TEST.diffprop(
  x,
  y,
  success.x = NULL,
  success.y = NULL,
  pdiff0 = 0,
  alternative = "two.sided",
  by,
  digits = 2,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x, y` Unquoted strings identifying the variables of interest. `x` and `y` can be the names of vectors or factors in the workspace or the names of columns in the data frame specified in the `data` argument. It is possible to use a mixed specification (e.g, one vector and one column in data).

`success.x, success.y` If `x,y` are factors, character vectors, or numeric non-binary vectors, success must be used to indicate the category/value corresponding to success in the populations. These arguments can be omitted (`NULL`, default) if `x,y` are binary numeric vectors (taking values 0 or 1 only; in this case success is assumed to correspond to 1) or a logical vector (in these cases success is assumed to correspond to `TRUE`).

`pdiff0` Numeric value that specifies the null hypothesis to test for (default is 0).

`alternative` A length-one character vector specifying the direction of the alternative hypothesis. Allowed values are `"two.sided"` (difference between populations' proportions differs from `pdiff0`; default), or `"less"` (difference between populations' proportions is lower than `pdiff0`), or `"greater"` (difference between populations' proportions is higher than `pdiff0`).

`by` Optional unquoted string identifying a variable (of any type), defined same way as `x`, taking only **two** values used to split `x` into two independent samples. Given the two *ordered* values taken by `by` (alphabetical or numerical order, or order of the levels for factors), say *by1* and *by2*, hypotheses are tested on the difference between the populations proportions in the *by1*- and in the *by2*-group. Note that only **one** between `y` and `by` can be specified.

`digits` Integer value specifying the number of decimals used to round statistics; default to 2. If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument `force.digits` is set to `TRUE`.

`force.digits` Logical value indicating whether reported values should be forcedly rounded to the number of decimals specified in `digits` even if non-zero values are rounded

	to zero (default to FALSE).
use.scientific	Logical value indicating whether numbers in tables should be displayed using scientific notation (TRUE); default to FALSE.
data	An optional data frame containing x and/or y or by. If not found in data, the variables are taken from the environment from which TEST.diffprop() is called.
...	Additional arguments to be passed to low level functions.

Value

A table reporting the results of the test on the difference between the proportions of successes in two independent populations.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

CI.diffprop() to build confidence intervals for the difference between two populations' proportions of successes.

Examples

```
data(MktDATA, package = "UBStats")

# Proportions of success defined on non-binary and
#  non-logical vectors; 'success' coded same way
#  for both vectors
#  - Using x,y: build vectors with data on the two groups
WouldSuggest_F <- MktDATA$WouldSuggest[MktDATA$Gender == "F"]
WouldSuggest_M <- MktDATA$WouldSuggest[MktDATA$Gender == "M"]
TEST.diffprop(x = WouldSuggest_M, y = WouldSuggest_F,
              success.x = "Yes", pdiff0 = 0.1, alternative = "less")

PastCampaigns_F<-MktDATA$PastCampaigns[MktDATA$Gender=="F"]
PastCampaigns_M<-MktDATA$PastCampaigns[MktDATA$Gender=="M"]
TEST.diffprop(x = PastCampaigns_M, y = PastCampaigns_F,
              success.x = 0, pdiff0 = 0.2)

#  - Using x,by: groups identified by ordered levels of by
TEST.diffprop(x = PastCampaigns, by = Gender,
              success.x=0, pdiff0 = 0.2, data = MktDATA)
#    Since order is F, M, test is on prop(F) - prop(M)
#    To get the interval for prop(M) - prop(F)
Gender.R <- factor(MktDATA$Gender, levels = c("M", "F"))
TEST.diffprop(x = PastCampaigns, by = Gender.R,
              success.x=0, pdiff0 = 0.2, data = MktDATA)

# Proportions of success defined based on
#  binary or logical vectors; 'success'
```

```
#  coded same way for both vectors
#  - Binary variable (success=1): based on x,y
LastCampaign_F<-MktDATA$LastCampaign[MktDATA$Gender=="F"]
LastCampaign_M<-MktDATA$LastCampaign[MktDATA$Gender=="M"]
TEST.diffprop(x = LastCampaign_M, y = LastCampaign_F)
#  - Binary variable (success=1): based on x,y
#    see above for recoding of levels of Gender
TEST.diffprop(x = LastCampaign, by = Gender, data = MktDATA)
Gender.R <- factor(MktDATA$Gender, levels = c("M", "F"))
TEST.diffprop(x = LastCampaign, by = Gender.R, data = MktDATA)
#  - Logical variable (success=TRUE): based on x,y
Deals_w_child <- MktDATA$Deals.ge50[MktDATA$Children>0]
Deals_no_child <- MktDATA$Deals.ge50[MktDATA$Children==0]
TEST.diffprop(x = Deals_w_child, y = Deals_no_child,
              pdiff0 = 0.2, alternative = "less",)
# Proportions defined on
#  non-binary and non-logical vectors, with 'success'
#  coded differently (only specification x,y is reasonable here)
WouldSuggest_Other<-c(rep("OK",310),rep("KO",650-310))
TEST.diffprop(x = WouldSuggest, y = WouldSuggest_Other,
              success.x = "Yes", success.y = "OK",
              pdiff0 = 0.1, alternative = "greater",
              data = MktDATA)

# Proportions based on combined conditions
# - Build logical vector/s indicating whether a condition
#   is satisfied
IsTop<-MktDATA$AOV>80
IsTop_OK<-IsTop[MktDATA$WouldSuggest == "Yes"]
IsTop_KO<-IsTop[MktDATA$WouldSuggest == "No"]
TEST.diffprop(x = IsTop_OK, y = IsTop_KO, pdiff0 = 0.05,
              alternative = "greater")

Deals<-MktDATA$NDeals>=5
Deals_Married <- Deals[MktDATA$Marital_Status=="Married" &
                         MktDATA$Children==0]
Deals_Single <- Deals[MktDATA$Marital_Status=="Single"]
TEST.diffprop(x = Deals_Married, y = Deals_Single,
              alternative = "less")

# Output results
out.test_diffP<-TEST.diffprop(x = Deals_Married, y = Deals_Single,
                              alternative = "less")

# Arguments force.digits and use.scientific
#  An input variable taking very low values
HighAOV <- MktDATA$AOV>150
# - Default: manages possible excess of rounding
TEST.diffprop(x = HighAOV[MktDATA$Gender=="M"],
              y = HighAOV[MktDATA$Gender=="F"])
#  - Force to the exact number of digits (default, 2)
TEST.diffprop(x = HighAOV[MktDATA$Gender=="M"],
              y = HighAOV[MktDATA$Gender=="F"],
```

```
              force.digits = TRUE)
#  - Allow scientific notation
TEST.diffprop(x = HighAOV[MktDATA$Gender=="M"],
              y = HighAOV[MktDATA$Gender=="F"],
              use.scientific = TRUE)
```

TEST.diffvar *Tests on variances*

Description

TEST.diffvar() tests the hypothesis of equality between the variances of two independent populations.

Usage

```
TEST.diffvar(
  x,
  y,
  by,
  digits = 2,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

x, y	Unquoted strings identifying the *numeric* variables with the same length whose variances have to be compared. x and y can be the names of vectors in the workspace or the names of columns in the data frame specified in the data argument. It is possible to use a mixed specification (e.g, one vector and one column in data).
by	Optional unquoted string identifying a variable (of any type), defined same way as x, taking only **two** values used to split x into two independent samples. Since the null hypothesis of equal variances is tested against the bilateral alternative only, the order of the levels of by is irrelevant (differently from what holds for functions building confidence intervals or testing hypotheses on the differences between means or proportions). Note that only **one** between y and by can be specified.
digits	Integer value specifying the number of decimals used to round statistics; default to 2. If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument force.digits is set to TRUE.

`force.digits`	Logical value indicating whether reported values should be forcedly rounded to the number of decimals specified in `digits` even if non-zero values are rounded to zero (default to `FALSE`).
`use.scientific`	Logical value indicating whether numbers in tables should be displayed using scientific notation (`TRUE`); default to `FALSE`.
`data`	An optional data frame containing `x` and/or `y`. If not found in `data`, the variables are taken from the environment from which `TEST.diffvar()` is called.
`...`	Additional arguments to be passed to low level functions.

Value

A table reporting the results of the test on the difference between the variances of two independent populations.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

`CI.diffmean()` to build confidence intervals for the difference between two populations' means.

`TEST.diffmean()` to test hypotheses on the difference between two populations' means.

Examples

```
data(MktDATA, package = "UBStats")

# Using x,y: build vectors with data on the two groups
AOV_M <- MktDATA$AOV[MktDATA$Gender == "M"]
AOV_F <- MktDATA$AOV[MktDATA$Gender == "F"]
TEST.diffvar(x = AOV_M, y = AOV_F)
TEST.diffvar(x = AOV_F, y = AOV_M) # same

# Using x,by: groups identified by ordered levels of by
TEST.diffvar(x = AOV, by = Gender, data=MktDATA)

# Output results
out_test.diffV<-TEST.diffvar(x = AOV_M, y = AOV_F)

# Arguments force.digits and use.scientific
#  An input variable taking very low values
SmallX<-MktDATA$AOV/50000
SmallX_M <- SmallX[MktDATA$Gender == "M"]
SmallX_F <- SmallX[MktDATA$Gender == "F"]
#  - Default output
TEST.diffvar(x = SmallX_M, y = SmallX_F)
#  - Request to use the exact number of digits (default, 2)
TEST.diffvar(x = SmallX_M, y = SmallX_F,
             force.digits = TRUE)
#  - Request to allow scientific notation
```

```
TEST.diffvar(x = SmallX_M, y = SmallX_F,
             use.scientific = TRUE)
```

TEST.mean	*Test on the mean*

Description

`TEST.mean()` tests hypotheses on the mean of a population.

Usage

```
TEST.mean(
  x,
  sigma = NULL,
  mu0 = 0,
  alternative = "two.sided",
  digits = 2,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x`	An unquoted string identifying the *numeric* variable whose mean is of interest. `x` can be the name of a vector in the workspace or the name of one of the columns in the data frame specified in the `data` argument.
`sigma`	An optional numeric value specifying the population standard deviation. If `NULL` (default) the population standard deviation is estimated using the data.
`mu0`	Numeric value that specifies the null hypothesis to test for (default is 0).
`alternative`	A length-one character vector specifying the direction of the alternative hypothesis. Allowed values are `"two.sided"` (population mean differs from `mu0`; default), or `"less"` (population mean is lower than `mu0`), or `"greater"` (population mean is higher than `mu0`).
`digits`	Integer value specifying the number of decimals used to round statistics; default to 2. If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument `force.digits` is set to `TRUE`.
`force.digits`	Logical value indicating whether reported values should be forcedly rounded to the number of decimals specified in `digits` even if non-zero values are rounded to zero (default to `FALSE`).
`use.scientific`	Logical value indicating whether numbers in tables should be displayed using scientific notation (`TRUE`); default to `FALSE`.

`data`	An optional data frame containing `x`. If not found in `data`, `x` is taken from the environment from which `TEST.mean()` is called.
`...`	Additional arguments to be passed to low level functions.

Value

A table reporting the results of the test on the population mean. If the variance is unknown, the test is run using percentiles from both the normal and the Student's t distribution.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

`CI.mean()` to build confidence intervals for the population mean.

Examples

```
data(MktDATA, package = "UBStats")

# Test on the mean; KNOWN variance
# - Bilateral test
TEST.mean(NStore_Purch, sigma = 9, mu0 = 5,
          alternative = "two.sided", data = MktDATA)
# - Unilateral test
TEST.mean(NStore_Purch, sigma = 9,mu0 = 5,
          alternative = "greater", data = MktDATA)

# Test on the mean; UNKNOWN variance;
# - Unilateral test
TEST.mean(TotVal, mu0 = 600, alternative = "less",
          data = MktDATA)

# Arguments force.digits and use.scientific
# An input variable taking very low values
SmallX<-MktDATA$AOV/500
# Default output
TEST.mean(SmallX, mu0 = 0.1)
# Request to use the exact number of digits (default, 2)
TEST.mean(SmallX, mu0 = 0.1,force.digits=TRUE)
# Request to allow scientific notation
TEST.mean(SmallX, mu0 = 0.1,use.scientific=TRUE)

# Output results
out.test_mean<-TEST.mean(TotVal, mu0 = 600, alternative = "less",
                         data = MktDATA)
```

TEST.prop *Test on the proportion*

Description

`TEST.prop()` tests hypotheses on the proportion of successes in a population.

Usage

```
TEST.prop(
  x,
  success = NULL,
  p0 = 0.5,
  alternative = "two.sided",
  digits = 2,
  force.digits = FALSE,
  use.scientific = FALSE,
  data,
  ...
)
```

Arguments

`x`	An unquoted string identifying the variable of interest. `x` can be the name of a vector or a factor in the workspace or the name of one of the columns in the data frame specified in the `data` argument.
`success`	If `x` is a factor, a character vector, or a numeric non-binary vector, `success` must be used to indicate the category/value corresponding to success. The argument can be omitted (`NULL`, default) if `x` is a binary numeric vector (takes values 0 or 1 only; in this case success is assumed to be 1) or a logical vector (in these cases success is assumed to be `TRUE`).
`p0`	Numeric value that specifies the null hypothesis to test for (default is 0).
`alternative`	A length-one character vector specifying the direction of the alternative hypothesis. Allowed values are `"two.sided"` (population proportion differs from `p0`; default), or `"less"` (population proportion is lower than `p0`), or `"greater"` (population proportion is higher than `p0`).
`digits`	Integer value specifying the number of decimals used to round statistics; default to 2. If the chosen rounding formats some non-zero values as zero, the number of decimals is increased so that all values have at least one significant digit, unless the argument `force.digits` is set to `TRUE`.
`force.digits`	Logical value indicating whether reported values should be forcedly rounded to the number of decimals specified in `digits` even if non-zero values are rounded to zero (default to `FALSE`).
`use.scientific`	Logical value indicating whether numbers in tables should be displayed using scientific notation (`TRUE`); default to `FALSE`.

`data`	An optional data frame containing `x`. If not found in `data`, `x` is taken from the environment from which `TEST.prop()` is called.
`...`	Additional arguments to be passed to low level functions.

Value

A table reporting the results of the test on the population proportion of successes.

Author(s)

Raffaella Piccarreta <raffaella.piccarreta@unibocconi.it>

See Also

`CI.prop()` to build confidence intervals for the population proportion of successes.

Examples

```
data(MktDATA, package = "UBStats")

# Success = one value of a character vector or factor
# - Bilateral test
TEST.prop(WouldSuggest, success = "Yes", p0 = 0.7,
         data = MktDATA)
# - Unilateral test, change digits
TEST.prop(Education, success = "Post-Grad", p0 = 0.3,
          alternative = "less", digits = 4,data = MktDATA)

# Success = numeric value; bilateral test
TEST.prop(Children, success = 2, p0 = 0.3, data = MktDATA)

# Binary variable (success = 1 by default); unilateral
TEST.prop(LastCampaign, p0 = 0.1, alternative = "greater",
          digits = 3, data = MktDATA)

# Logical variable (success = TRUE by default); unilateral test
TEST.prop(Deals.ge50, p0 = 0.13, alternative = "greater",
          digits = 3, data = MktDATA)

# Success based on combined conditions
# - Build a (logical) vector
IsTop <- MktDATA$CustClass == "Gold" |
  MktDATA$CustClass == "Platinum"
TEST.prop(IsTop, p0 = 0.2, data = MktDATA)

HighAOV <- MktDATA$AOV>150
TEST.prop(HighAOV, p0 = 0.1)
TEST.prop(HighAOV, p0 = 0.1, force.digits = TRUE)
TEST.prop(HighAOV, p0 = 0.1, use.scientific = TRUE)

# Output results
out_test_prop<-TEST.prop(IsTop, p0 = 0.2, data = MktDATA)
```

Time_Social — *Data Time_Social: Time Spent on a Social Media Platform*

Description

The data frame contains information on the time spent on a social media platform, after the algorithm that selects displayed content to users was modified. Specifically, attention is limited to a sample of active users within a specific socio-demographic segment, who live in two distinct areas.

Usage

```
data(Time_Social)
```

Format

A data frame with $n = 5976$ observations and 4 variables:

User (int) user id

Time (num) Time (in minutes) spent on the platform during a specific day

Area (chr) geographic area where the user lives (`A`, `B`)

Push (chr) variable indicating whether the user activated push notifications (`No`, `Yes`)

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

Transition — *Data Transition: Employee Productivity After Software Transition*

Description

The data frame refers to data on the employees of a company that adopted a new management software system to replace its legacy tools. To facilitate a smooth transition, all administrative staff will participate in a structured training program designed to build proficiency with the new system. As part of the rollout, employees from two departments have been selected to complete the training first. The dataset contains information on each (trained) employee's productivity using the old and the new software, recorded after a defined period of use.

Usage

```
data(Transition)
```

Format

A data frame with n = 130 observations and 4 variables:

Employee (int) employee id

Pre (num) employee's productivity with the old software

Post (num) employee's productivity with the new software (after the training program)

Department (chr) employee's department (`Dept1`, `Dept2`)

References

Piccarreta, R., Tonini, D., & Trentini, F. (2026). *From Data to Decisions. An Introduction to Applied Statistics*. BUP. ISBN 9788823824096.

Index