Package: ExploreTheData (via r-universe)

May 22, 2026

Title A Set of Tools for Exploratory Data Analysis

Version 0.1.0

Description Functions to profile a dataset, identify anomalies (special values, outliers, and inliers, defined as data values that are repeated unusually often), and compare data subsets with respect to either numerical or categorical variable distributions.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports data.table, graphics, PropCIs

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author Ronald Pearson [aut, cre]

Maintainer Ronald Pearson <ronald.k.pearson@gmail.com>

Repository https://ronaldkpearson.r-universe.dev

Date/Publication 2026-02-20 09:02:10 UTC

RemoteUrl https://github.com/cran/ExploreTheData

RemoteRef HEAD

RemoteSha cb67bfa27f7e5c8124d29ae94441a09ab1828f54

Contents

`AccountingExample` *Synthetic accounting dataset example, from Excel*

Description

Small dataset illustrating various unexpected data formats arising from the accounting data format in an Excel spreadsheet. Variables that appear to be numeric based on the name are represented as character strings with embedded commas, dollar signs, percent signs, and parentheses to indicate negative numbers

Usage

```
AccountingExample
```

Format

`AccountingExample`**:**

data frame with 8 rows and 6 columns:

Year Four-digit integer year, with missing values coded NA

Quarter Two-character quarter designation, Q1 through Q4

CurrentTotal Dollar amount with dollar signs, commas, and decimal points

PriorYearTotal Dollar amount with dollar signs, commas, and decimal points

YOYchange Dollar amount with dollar signs, commas, decimal points and parentheses to indicate negative values

PctChange Ratio of YOYchange to CurrentTotal, converted to a percentage, with percent sign

BinomialCIsByCategorical *Compute binomial probabilities over categorical variable levels*

Description

Compute binomial probabilities over categorical variable levels

Usage

```
BinomialCIsByCategorical(
  DF,
  binVar,
  catVar,
  targetLevel,
  keepNA = "ifany",
  keepLevels = NULL,
  cLevel = 0.95
)
```

Arguments

DF	A data frame containing binVar and catVar
binVar	Binary variable for binomial probabilities
catVar	Categorical variable over which binomial probabilities are computed
targetLevel	Positive response level for binVar
keepNA	Missing data handling option: ifany (the default), no, or always
keepLevels	Optional subset of catVar levels to be used in the analysis; default NULL retains all levels
cLevel	Confidence level for binomial probabilities (default 0.95)

Value

Data frame with one row for each catVar level in the analysis and these 6 columns:

- Level the catVar level
- nWith the number of records with catVar equal to Level and binVar equal to targetLevel
- nTotal the total number of records with catVar equal to Level
- pEst the estimated probability that binVar equals targetLevel
- loCI the lower cLevel confidence limit for pEst
- upCI the upper cLevel confidence limit for pEst

Examples

```
catVar <- c(rep("A", 100), rep("B", 100), rep("C", 100))
binVar <- c(rep(0,80),rep(1,20), rep(0,50),rep(1,50), rep(0,20),rep(1,80))
DF <- data.frame(catVar = catVar, binVar = binVar)
BinomialCIsByCategorical(DF, "binVar", "catVar", 1)
```

`CompareCategoricalLevels` *Compare categorical level distribution between subsets*

Description

Given two data subsets, defined by `indexA` and `indexB`, and a categorical variable `catVar`, compute the probability that each level of `catVar` appears in each subset and the Agresti-Caffo confidence interval for the difference in these probabilities, based on the PropCIs::wald2ci function.

Usage

```
CompareCategoricalLevels(
  DF,
  catVar,
  indexA,
  indexB = NULL,
  cLevel = 0.95,
  includeNA = "ifany"
)
```

Arguments

`DF`	A data frame containing `catVar`
`catVar`	Categorical variable whose distribution is compared between two subsets
`indexA`	Defines records in the first subset
`indexB`	Defines records in the second subset; default NULL uses all records not in the first subset
`cLevel`	Confidence level for estimated probability differences
`includeNA`	Missing data handling option: `ifany` (the default), `no` or `always`

Value

Data frame with one row for each `catVar` level and these 10 columns:

- `Level` the `catVar` level
- `xA` the number of times `Level` appears in the first subset
- `nA` the total records in the first subset
- `xB` the number of times `Level` appears in the second subset

- nB the total records in the second subset
- pA the estimated probability that Level appears in the first subset
- pB the estimated probability that Level appears in the second subset
- loCI the lower confidence limit on the difference pA - pB
- upCI the upper confidence limit on the difference pA - pB
- signif a logical indicator of whether pA - pB is significantly different from zero

Examples

```
catVar <- c(rep("a", 100), rep("b", 100), rep("c", 100))
auxVar <- c(rep("Set1", 30), rep("Set2", 70),
           rep("Set1", 50), rep("Set2", 50),
           rep("Set1", 90), rep("Set2", 10))
DF <- data.frame(catVar = catVar, auxVar = auxVar)
indexA <- which(DF$auxVar == "Set1")
CompareCategoricalLevels(DF, "catVar", indexA)
```

CompareNumericSets *Compare distributions of numerical variables between subsets*

Description

Sets up and calls WelchRankTest to compare the distributions of a set of numerical variables between two record subsets. If the set of numerical variables contains a single element, this function effectively reduces to WelchRankTest.

Usage

```
CompareNumericSets(DF, IndexA, numVars, IndexB = NULL, cLevel = 0.95)
```

Arguments

DF	data frame containing all variables in numVars
IndexA	record index defining the first record subset to be compared
numVars	vector of numerical variable names from DF
IndexB	record index defining the second record subset to be compared (default NULL means the second set contains all records not included in the first)
cLevel	confidence level for the Welch rank test (default = 0.95)

Value

data frame with one row for each element of numVars and columns containing the numVars element name and all columns from WelchRankTest for that variable

Examples

```
x <- seq(-1, 1, length = 200)
a <- rep(c("a", "b"), 100)
offset <- rep(c(0, 0.2), 100)
xMod <- x + offset
DF <- data.frame(numVar = x, numVar2 = xMod, setVar = a)
indexA <- which(DF$setVar == "a")
CompareNumericSets(DF, indexA, c("numVar", "numVar2"))
```

ComputeOutlierLimits *Compuute outlier limits by three methods*

Description

Compute upper and lower outlier limits by three detection rules: the 3-sigma edit rule, the Hampel identifier, or the boxplot rule

Usage

```
ComputeOutlierLimits(x, method, t = NULL)
```

Arguments

x	numerical vector in which outliers are to be detected
method	single character specifying the outlier rule (T, H, or B)
t	threshold parameter (default NULL, gives 3 for T and H rules, 1.5 for B rule)

Value

named numerical vector with these 4 elements:

- nRec the number of elements in x
- nonMiss the number of non-missing elements in x
- loLim the lower outlier threshold for x elements
- upLim the upper outlier threshold for x elements

Examples

```
x <- seq(-1, 1, length = 100)
x[1:10] <- 10
ComputeOutlierLimits(x, "T")
ComputeOutlierLimits(x, "H")
ComputeOutlierLimits(x, "B")
```

FindInliers *Detect inliers based on unusual frequency of ocurrance*

Description

Returns an index to elements of a numerical vector whose frequency is unusually large relative to most elements, applying the three-sigma edit rule to counts of individual values. Inliers often represent data values that are incorrect but consistent with the overall data distribution, as in the case of numerically-coded disguised missing data

Usage

```
FindInliers(x, t = 3)
```

Arguments

x	numerical vector in which inliers are to be detected
t	threshold parameter for detecting outlying counts (default value 3)

Value

index to elements of x that occur unusually often, if any

Examples

```
x <- seq(-1, 1, length = 100)
x[45:54] <- 0
FindInliers(x)
```

FindOutliers *Find outliers by three methods*

Description

Returns an index into outlying points, if any, identified by one of three outlier detection rules: the three-sigma edit rule, the Hampel identifier, or the boxplot rule

Usage

```
FindOutliers(x, method, t = NULL)
```

Arguments

x	numerical vector in which outliers are to be detected
method	single character specifying the outlier rule (T, H, or B)
t	threshold parameter (default NULL, gives 3 for T and H rules, 1.5 for B rule)

Value

index into elements of x identified as outliers

Examples

```
x <- seq(-1, 1, length = 100)
x[1:10] <- 10
Tindex <- FindOutliers(x, "T")
x[Tindex]  # Example where the three-sigma rule fails
Hindex <- FindOutliers(x, "H")
x[Hindex]
Bindex <- FindOutliers(x, "B")
x[Bindex]
```

FirstAnomalyDataFrame *Dataset with missing values and other special cases*

Description

Small dataset illustrating standard missing values (NA and NaN), blanks, spaces, and other values sometimes used to represent missing data (e.g., blanks, spaces, and zeros)

Usage

```
FirstAnomalyDataFrame
```

Format

FirstAnomalyDataFrame**:**

a data frame with 5 rows and 6 columns:

NumVar1 numerical variable with positive, zero, negative and missing (NA) values

NumVar2 numerical variable with positive, zero, and missing (NA) values

NumVar3 the ratio of NumVar1 to NumVar2

CatVar1 categorical variable with missing data represented as NA

CatVar2 categorical variable with missing data represented with blanks or spaces

CatVar3 categorical variable with missing data represented with multiple spaces

plot.BinomCIframe *Plot binomial confidence intervals*

Description

Plot method for the S3 object class BinomCIframe generated by the `BinomialCIsByCategorical()` function

Usage

```
## S3 method for class 'BinomCIframe'
plot(x, ..., CIrange = NULL, addRef = TRUE)
```

Arguments

x	an S3 object of class BinomCIframe
...	optional named parameters to be passed to `plot()`
CIrange	two-element vector giving the minimum and maximum y-axis values to plot (default NULL uses minimum lower confidence limit and maximum upper confidence limit from x)
addRef	logical: add a reference line at the average probability of a positive response? (default = TRUE)

Value

None: this method generates a plot from x

Examples

```
catVar <- c(rep("A", 100), rep("B", 100), rep("C", 100))
binVar <- c(rep(0,80),rep(1,20), rep(0,50),rep(1,50), rep(0,20),rep(1,80))
DF <- data.frame(catVar = catVar, binVar = binVar)
CIframe <- BinomialCIsByCategorical(DF, "binVar", "catVar", 1)
plot(CIframe)
```

plot.CatDiffs *Plot significant categorical level differences between data subsets*

Description

Plot method for S3 objects of class CatDiffs generated by the `CompareCategoricalLevels()` function, creating a horizontal barplot of categorical variable level frequencies that differ significantly between two data subsets

Usage

```
## S3 method for class 'CatDiffs'
plot(x, ..., labelA, labelB, nMax = 20, levelFrac = 0.5, xLims = NULL)
```

Arguments

x	an S3 object of class CatDiffs
...	optional named parameters to be passed to plot()
labelA	plot label identifying the first data subset
labelB	plot label identifying the second data subset
nMax	maximum number of levels to include in the barplot (default = 20)
levelFrac	relative position of the level labels on the barplot (default = 0.5)
xLims	two-element vector of x-axis limits for the barplot (default sets the range from 0 to 1.2 times the length of the longest bar on the plot)

Value

None: this method generates a plot from x

Examples

```
catVar <- c(rep("a", 100), rep("b", 100), rep("c", 100))
auxVar <- c(rep("Set1", 30), rep("Set2", 70),
            rep("Set1", 50), rep("Set2", 50),
            rep("Set1", 90), rep("Set2", 10))
DF <- data.frame(catVar = catVar, auxVar = auxVar)
indexA <- which(DF$auxVar == "Set1")
CatDiffObj <- CompareCategoricalLevels(DF, "catVar", indexA)
plot(CatDiffObj, labelA = "Set1", labelB = "Set2")
```

ProfileDataFrame *Profile a data frame*

Description

Given the data frame DF, create a new data frame with one row for each column of DF that characterizes that column in terms of the number and fraction of missing values, the most frequent value and its frequency and other characteristics like the Shannon homogeneity measure computed by the ShannonHomogeneity() function.

Usage

```
ProfileDataFrame(DF, dgts = 3, charMax = 20)
```

Arguments

DF	data frame to be characterized
dgts	digits retained for numerical characterizations like fractions (default = 3)
charMax	maximum number of characters retained in representing the most frequent value for a variable (default = 20)

Value

data frame with one row for each column of DF and these columns:

- Variable the name of the column from DF being characterized
- Type the class of Variable (e.g., numeric, integer, character, etc.)
- nMiss the number of missing (NA) or blank Variable records
- fracMiss the fraction of total records represented by nMiss
- nLevels the number of distinct values Variable exhibits
- topValue the most frequently occurring Variable value, truncated to charMax characters
- topChars the actual number of characters required to represent topValue
- topFreq the number of times topValue occurs
- topFrac the fraction of total records represented by topFreq
- Homog the Shannon homogeneity measure for Variable

Examples

```
ProfileDataFrame(ChickWeight)
```

ShannonHomogeneity *Compute the Shannon homogeneity for a vector*

Description

Computes the Shannon homogeneity (normalized Shannon entropy) for a vector, typically categorical but the procedure also works with numerical vectors. Returns a value in the range from 0 (for a highly inhomogeneous vector, concentrated entirely on one of L > 1 levels) to 1 (for a completely homogeneous vector). By convention, vectors of length 0 or 1 return homogeneity values of 1.

Usage

```
ShannonHomogeneity(x, dgts = 3)
```

Arguments

x	the vector to be characterized
dgts	number of digits in the return value (default = 3)

Value

a numerical homogeneity measure between 0 and 1

Examples

```
x <- rep(c("a", "b", "c", "d", "e"), 200)
y <- c(rep("a", 497), rep("b", 497), rep("c", 2), rep("d", 2), rep("e", 2))
z <- c(rep("a", 996), "b", "c", "d", "e")
ShannonHomogeneity(x)
ShannonHomogeneity(y)
ShannonHomogeneity(z)
```

SummarizeInliers *Summarize elements of a numerical vector that occur unusually often*

Description

Applies the three-sigma edit rule to the frequencies of distinct values of a numerical vector, finding those that occur unusually often and identifying them either by record number or an associated identifying characteristic specified by `label`. Inliers often represent data values that are incorrect but consistent with the overall data distribution, as in the case of numerically-coded disguised missing data

Usage

```
SummarizeInliers(x, label = NULL, labelName = NULL, t = 3)
```

Arguments

`x`	numerical vector in which inliers are to be detected
`label`	optional identifying tag for inliers (default NULL gives an index into the elements of `x` declared inliers)
`labelName`	optional name for the `label` column, if specified (default NULL labels this column as Case)
`t`	detection threshold for the three-sigma edit rule applied to record counts (default value 3)

Value

Data frame with one row for each inlier detected and two columns:

- Record (or `labelName` value) identifying or characterizing each inlier
- Value the numerical value that occurs unusually often

Note that this data frame is empty (0 rows) if no inliers are detected

Examples

```
x <- seq(-1, 1, length = 100)
x[45:54] <- 0
SummarizeInliers(x)
```

`SummarizeOutliers` *Summarize outliers detected by three methods*

Description

Generates a summary of outliers detected by the three-sigma edit rule, the Hampel identifier, and the boxplot rule, including an optional label to identify the outlying points

Usage

```
SummarizeOutliers(x, label = NULL, labelName = NULL, thresh = c(3, 3, 1.5))
```

Arguments

`x`	numerical vector in which outliers are to be detected
`label`	optional identifying tag for outliers (default NULL gives an index into the elements of `x` declared outliers)
`labelName`	optional name for the `label` column, if specified (default NULL labels this column as Case)
`thresh`	vector of threshold values for each outlier detection rule (default = c(3, 3, 1.5))

Value

Data frame with one row for each outlier detected by any of the three methods and these 5 columns:

- Record (or `labelName`) giving the location or label for each outlier
- Value the value detected as an outlier by at least one method
- ThreeSigma 1 if the outlier is detected by the three-sigma rule, 0 otherwise
- Hampel 1 if the outlier is detected by the Hampel identifier, 0 otherwise
- Boxplot 1 if the outlier is detected by the boxplot rule, 0 otherwise

Note that this data frame is empty (0 rows) if no outliers are detected by any method

Examples

```
x <- seq(-1, 1, length = 100)
x[1:10] <- 10
SummarizeOutliers(x)
```

TabulateSpecialValues *Tabulate special values often representing missing data*

Description

Generates a summary of counts and fractions of records from the variables listed in xVars that are missing (using the standard R designation NA), blank (0 length, common in character data), spaces (one or more, also common in character data), and zeros or negative values in numerical data (sometimes indicative of range errors or disguised missing data)

Usage

```
TabulateSpecialValues(DF, xVars = NULL, subsetIndex = NULL, dgts = 3)
```

Arguments

DF	data frame containing all variables in the xVars list
xVars	character vector of the names of variables to be examined (default NULL means characterize all variables in data frame DF)
subsetIndex	index into record subset in DF to be examined (default NULL means characterize all records in DF)
dgts	number of digits in frequency results (default = 3)

Value

data frame with one row for each variable in xVars list and these columns:

- Variable an element of the xVars list
- nMiss number of records exhibiting the missing value NA (or NaN)
- fracMiss fraction of records represented by nMiss
- nBlank number of records listing the value blank (0 length character string)
- fracBlank fraction of records represented by nBlank
- nSpaces number of records consisting only of one or more spaces
- fracSpaces fraction of records represented by nSpaces
- nZero number of records listing the numerical value zero
- fracZero fraction of records represented by 'nZero'
- nNeg number of records listing a negative numerical value
- fracNeg fraction of records represented by nNeg

Examples

```
FirstAnomalyDataFrame
TabulateSpecialValues(FirstAnomalyDataFrame)
```

`WelchRankTest` *Compare two numerical data subsets*

Description

Uses the Welch rank-test (a robust alternative to the classical t-test, with better resistance to outliers and asymmetry) to compare the distributions of two subsets of the same numerical variable. The result characterizes the subsets in terms of their median values, and a small p-value (traditionally less than 0.05) implies significant distributional differences between the two subsets.

Usage

```
WelchRankTest(DF, xVar, indexA, indexB = NULL, cLevel = 0.95)
```

Arguments

`DF`	data frame containing `xVar`
`xVar`	numerical variable whose subsets are to be compared
`indexA`	record index defining the first subset of `xVar` values
`indexB`	record index defining the second subset of `xVar` values (default NULL means the second subset is all records not contained in the first)
`cLevel`	confidence level for the test (default = 0.95)

Value

a named vector with these 5 elements:

- nA the number of records in the first `xVar` subset
- nB the number of records in the second `xVar` subset
- medianA the median `xVar` value in the first subset
- medianB the median `xVar` value in the second subset
- pValue the p-value returned by the Welch rank test

Examples

```
x <- seq(-1, 1, length = 200)
a <- rep(c("a", "b"), 100)
DF <- data.frame(numVar = x, setVar = a)
indexA <- which(DF$setVar == "a")
WelchRankTest(DF, "numVar", indexA)  # No difference in distribution
offset <- rep(c(0, 0.2), 100)
DF$numVar2 <- x + offset
WelchRankTest(DF, "numVar2", indexA) # Significant difference
xMod <- x
xMod[indexA[1:4]] <- x[indexA[1:4]] + 10
DF$numVar3 <- xMod
WelchRankTest(DF, "numVar3", indexA) # No difference even with outliers
stats::t.test(DF[indexA, "numVar3"], DF[-indexA, "numVar3"]) # Compare t-test
```

Index