Package ‘smallstuff’

October 14, 2022

Type    Package
Title   Dr. Small's Functions
Version 1.0.1
Date    2022-05-29
Description Functions used in courses taught by Dr. Small at Drew University.
License GPL-3
Encoding UTF-8
Imports Matrix (>= 1.4-1), matlib (>= 0.9.5), pryr (>= 0.1.5), class
          (>= 7.3-20), igraph (>= 1.3.1), ROCR (>= 1.0-11), data.table
          (>= 1.14.2)
Suggests leaps (>= 3.1), faraway (>= 1.0.7)
RoxygenNote 7.2.0
NeedsCompilation no
Author    Ellie Small [aut, cre](<https://orcid.org/0000-0003-1313-115X>)
Maintainer Ellie Small <esmall1@drew.edu>
Repository CRAN
Date/Publication 2022-06-02 09:50:02 UTC

R topics documented:

smallstuff-package .................................................... 2
allspan3D ............................................................... 3
allvectors3D ............................................................ 4
as_adj_def ............................................................... 4
CI ............................................................... 5
coord2D ............................................................... 6
coord3D ............................................................... 6
crossing2 ............................................................. 7
CError ............................................................... 7
CErrorrknn ............................................................ 8
dataSet .............................................................. 9
smallstuff-package

A short title line describing what the package does

Description

A more detailed description of what the package does. A length of about one to five lines is recommended.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Your Name, email optional.

Maintainer: Your Name <your@email.com>
allspan3D

Plot Span and Vectors in 3D

Description
Plot the span of a matrix plus any vectors in a 3D plot at one or more angles. A plot is produced for each entry of `th`.

Usage
```r
allspan3D(M, V = NULL, th = c(-90, -45, 0, 45, 90, 135), V2 = NULL, col = NULL)
```

Arguments
- `M`: Matrix for which the span should be shown.
- `V`: Either `NULL`, a vector of length 3, or a matrix with each column a vector of length 3.
- `th`: A vector indicating the horizontal angle at which the plot should be shown.
- `V2`: A matrix or vector of the same dimensions as `M` indicating the starting points of the vectors in `M` (default is the origin for all).
- `col`: Vector colors; if entered, must have a value for each vector.

Value
No return value, called for side effects

Examples
```r
M <- matrix(c(1,2,4,3,0,2),3)
oldpar <- par(mfrow=c(3,2))
allspan3D(M, cbind(M, M[1]-M[2]), V2 = matrix(c(rep(0,6),M[2]),3), col=c(2,2,1))
par(oldpar)
```
allvectors3D  

Plot Vectors in 3D

Description
Plot one or more vectors in a 3D plot at one or more angles. A plot is produced for each entry of th.

Usage
allvectors3D(V, th = c(0, 30, 60, 90, 120, 150), V2 = NULL, col = NULL)

Arguments
V  Either a vector of length 3 or a matrix with each column a vector of length 3.
th  A vector indicating the angles at which the plot should be shown.
V2  A matrix or vector of the same dimensions as V indicating the starting points of the vectors in V (default is the origin for all).
col  Vector colors; if entered, must have a value for each vector.

Value
No return value, called for side effects

Examples
a=c(2,4,8)
b=c(6,0,4)
oldpar <- par(mfrow=c(3,2))
allvectors3D(cbind(a,b,a-b),V2=matrix(c(rep(0,6),b),3))
par(oldpar)

as_adj_def
Create an adjacency matrix from a multigraph according to the definition

Description
Create an adjacency matrix using the definition, i.e. an entry equals 1 if there is an edge from the vertex in the column to the vertex in the row, and cycles are counted twice.

Usage
as_adj_def(g, ...)

as_adj_def
**CI**

**Arguments**

- `g`: the graph (an igraph object)
- `...`: additional arguments to be passed to the igraph function `as_adj`

**Value**

Adjacency matrix for graph `g`

**Examples**

```r
g = igraph::graph_from_literal(1-2, 2-3:4, 3-5, 5-1, 6-6, simplify=FALSE)
as_adj_def(g)
```

---

**CI**

*Normal Confidence Interval*

**Description**

Confidence interval for a normally distributed sample mean

**Usage**

`CI(x = 0, s = 1, n = 1, level = 0.95)`

**Arguments**

- `x`: sample mean
- `s`: standard deviation
- `n`: sample size
- `level`: confidence level

**Value**

vector with two values containing the confidence interval for the sample mean

**Examples**

```r
CI()
CI(150, 5, 30, .9)
```
coord2D  
*Plot a 2D Coordinate System*

**Description**
Plot a coordinate system in 2D with the origin in the center.

**Usage**
coord2D(x = 5, y = 5)

**Arguments**
- **x**  
  Distance from the origin to the maximum x-value.
- **y**  
  Distance from the origin to the maximum y-value.

**Value**
No return value, called for side effects

**Examples**
coord2D()

---

coord3D  
*Plot a 3D Coordinate System*

**Description**
Plot a coordinate system in 3D with the origin bottom left.

**Usage**
coord3D(th = 0, x = 10, y = 10, z = 10)

**Arguments**
- **th**  
  The angle at which the 3D plot should be displayed.
- **x**  
  Distance from the origin to the maximum x-value.
- **y**  
  Distance from the origin to the maximum y-value.
- **z**  
  Distance from the origin to the maximum z-value.

**Value**
A matrix containing the plot coordinates (used when adding features).
crossing2

Examples
coord3D()

---

**crossing2**

*Find Edge Crossings*

**Description**

Determine if edges in a graph cross groups or stay within groups. This is similar to the crossings function in igraph, but uses a vector for the split rather than a communities object.

**Usage**
crossing2(split, g)

**Arguments**

- **split**: a vector with a value for each vertex in g, indicating the group each vertex belongs to.
- **g**: an igraph object

**Value**

A logical vector indicating for each edge if it crosses groups or not. For each edge that crosses, it is TRUE, otherwise it is FALSE.

**Examples**
g=igraph::graph_from_literal(1-2,2-3:4,3-4:5:6,5-1)
split=c("A","A","B","B","A","B")
igraph::V(g);split
igraph::E(g);crossing2(split,g)

---

**CError**

*k-Fold Cross Validation Error Rate*

**Description**

Given a logistic regression model (via glm), or an LDA or QDA model, and a number of folds k, the k-Fold CV error rate is calculated.

**Usage**

CError(mod, k = nrow(stats::model.frame(mod)))
**Arguments**

- **mod**: A logistic regression, LDA, or QDA model
- **k**: Number of folds; by default LOOCV will be returned

**Value**

The k-fold CV error rate if k is entered, otherwise the LOOCV error rate.

**Examples**

```r
mtcars$am = as.factor(mtcars$am)
gmod = glm(am ~ mpg, binomial, mtcars)
CVerror(gmod)
```

---

**CVerrorknn**

**k-Fold Cross Validation Error Rate for KNN**

**Description**

Given a dataset with predictors and a vector with responses, a number of neighbors K, and a number of folds k, the k-fold CV error rate for KNN is calculated.

**Usage**

```r
CVerrorknn(pred, resp, K = 1, k = nrow(pred))
```

**Arguments**

- **pred**: A dataset with predictors
- **resp**: A vector with responses
- **K**: The number of neighborhoods to consider when performing KNN
- **k**: The number of folds

**Value**

The k-fold CV error rate if k is entered, otherwise the LOOCV error rate.

**Examples**

```r
mtcars$am = as.factor(mtcars$am)
CVerrorknn(mtcars[, c("mpg", "hp")], mtcars$am)
```
dataSet

Obtain a Dataset from a Formula

Description

Given a formula, a dataset and a subset, retrieve the dataset that fulfills the formula and subset.

Usage

dataSet(formula, data, subset = NULL)

Arguments

formula A formula
data A dataset
subset Either a logical vector or a vector of indices of the rows to be returned. If NULL (default), all rows are returned.

Value

The dataset in data as a data table with variables as specified in formula and rows as specified by subset.

Examples

dataSet(mpg~.-disp, mtcars, 10:20)

dCohen

Cohen’s d

Description

Calculate Cohen’s d for one-sample t tests or two-sample independent tests or two-sample paired t-tests

Usage

dCohen(x, y = NULL, mu0 = 0, paired = FALSE)

Arguments

x vector with (numeric) data
y for two-sample tests, a vector with (numeric) data for group 2
mu0 for one-sample tests, the number to test against
paired TRUE for a paired two-sample t-test, FALSE for an independent sample t-test
get_subgraphs

Value
value of Cohen’s d

Examples

# one-sample
x = c(1:10, 5, 6, 3:8)
dCohen(x, mu0 = 7)

# two-sample independent
y = 1:15
dCohen(x, y)

# two-sample paired
dCohen(x, 1:18, paired = TRUE)

d_subgraphs

Split a Graph into Subgraphs

Description
Split a graph into subgraphs using the values in a vector to indicate which vertices belong together.

Usage
g_subgraphs(g, split)

Arguments

\textbf{g}
the graph (an igraph object)

\textbf{split}
a vector with a value for each vertex in \textbf{g}

Value
A list of graphs, where each graph is a subgraph of \textbf{g} containing the vertices with the same value in \textbf{split}.

Examples

g = igraph::graph_from_literal(1-2, 2-3:4, 3-4:5:6, 5-1)
split = c("A", "A", "B", "B", "A", "B")
igraph::V(g); split
igraph::V(g_subgraphs(g, split)[[1]])
igraph::V(g_subgraphs(g, split)[[2]])
**graph_attr_from_df**  
Add Graph Attributes to a Graph from a Data Frame

**Description**
Add graph attributes to a graph from a data frame where each column represents an attribute. Note that only the first row of the data frame is used.

**Usage**
```
graph_attr_from_df(g, df)
```

**Arguments**
- `g`: the graph (an igraph object) to which the graph attributes should be added
- `df`: data frame, or an object that can be converted to a data frame, where the first row contains a graph attribute in each column

**Value**
Graph `g` with the graph attributes in `df` added.

**Examples**
```
g = igraph::graph_from_literal(1-2, 2-3:4, 3-4:5:6, 5-1)
df = data.frame(name = "Test Graph", descr = "A graph")
graph_attr_from_df(g, df)
```

---

**impNA**  
Impute Missing Values

**Description**
Replace missing values in a vector using a function (by default the mean) on this vector.

**Usage**
```
impNA(x, fn = mean, ...)
```

**Arguments**
- `x`: A numeric vector
- `fn`: A function to apply to all values in the vector `x`
- `...`: Additional arguments to be passed to function `fn`
**isInt**

Determine if the Input contains Integers

**Description**

Determine if numbers in a vector are integers (not just of integer type)

**Usage**

```r
isInt(x, inf = TRUE)
```

**Arguments**

- `x` integer or numeric type vector
- `inf` logical field answering whether an infinite value should be considered an integer (default TRUE)

**Value**

TRUE for each value in `x` that is an integer, FALSE otherwise

**Examples**

```r
isInt(c(3,3.23,Inf))
```
**laCrossProd**  
*Cross Product (Linear Algebra)*

**Description**

Calculate the cross product as defined in linear algebra; note that this differs from the cross product as defined by R.

**Usage**

```r
laCrossProd(x, y)
```

**Arguments**

- `x` vector of length 3.
- `y` vector of length 3.

**Value**

Cross product of `x` and `y`.

**Examples**

```r
x = c(1, 2, 1)
y = 1:3
laCrossProd(x, y)
```

**lines3D**  
*Lines in 3D*

**Description**

Plot a line in a 3D plot through a set of points.

**Usage**

```r
lines3D(pl, x, y, z, ...)
```

**Arguments**

- `pl` Matrix containing the current plot coordinates.
- `x` Vector with x-coordinates.
- `y` Vector with y-coordinates.
- `z` Vector with z-coordinates.
- `...` additional graphical parameters (see `lines()`).
Value

No return value, called for side effects

Examples

pl=coord3D(30)
lines3D(pl,0:10,0:10,rep(0,11))
lines3D(pl,0:10,0:10,c(0,2,1,3:8,7,5),col=2)

lmPartReg             Partial Regression Plot

Description

Plot the partial regression plot for one of the predictors of a linear model

Usage

lmPartReg(mod, pred, ...)

Arguments

mod        A linear model object (obtained via the lm function)
pred       The name (in quotes) of the predictor for which the plot should be produced
...        Any other arguments to be passed to the plot

Value

A partial regression plot for pred in the linear model mod

Examples

lmod=lm(mpg~.,mtcars)
lmPartReg(lmod,"wt")
**lmSub**

*Best Linear Model in Subset Selection*

**Description**

Produces the best linear model for a specific number of predictors in a subset selection.

**Usage**

`lmSub(object, d)`

**Arguments**

- **object**
  - An object of type "regsubsets"
- **d**
  - Number of data predictors

**Value**

The best linear model with d predictors

**Examples**

```r
subs = leaps::regsubsets(mpg ~ ., mtcars)
summary(lmSub(subs, 3))
```

---

**logistErrorRate**

*Calculate the Error Rate and Results Table for Logistic Regression Models*

**Description**

Calculate the testing error rate for a dataset on a logistic regression model (or the training error rate if no dataset is entered), and a results table with responses versus predicted responses.

**Usage**

`logistErrorRate(gmod, nw = NULL, p = 0.5)`

**Arguments**

- **gmod**
  - A logistic regression model
- **nw**
  - A dataset for which a testing error rate should be calculated using the model in gmod. Note that it must contain the predictors as well as the responses. If this argument is NULL (the default) the training error rate will be calculated.
- **p**
  - Probability (default .5) above which the observation is assigned to the second level of the response.
Value
List with training error rate if nw is NULL, testing error rate otherwise, and a results table with responses versus predicted responses.

Examples

```r
gmod=glm(state~.,binomial,Puromycin)
logistErrorRate(gmod)
```

---

**outliers**  
*Find Outliers*

Description
Find the outliers in a vector of values.

Usage
```r
outliers(x)
```

Arguments

- `x` vector

Value
A list with a variable `idx` containing the indices of the outliers and a variable `values` containing the values of the outliers.

Examples

```r
x=c(100,30:40,101,25:28)
outliers(x)
```

---

**plotCol**  
*Plot Colors*

Description
Plot one or more colors

Usage
```r
plotCol(col)
```
pop.sd

Arguments

  col  vector with colors

Value

A plot showing the colors in col

Examples

  plotCol("maroon")

pop.sd

Calculate the Population Standard Deviation

Description

  Calculate the standard deviation of a numeric vector if the data constitutes the whole population. Note that missing values are excluded.

Usage

  pop.sd(x)

Arguments

  x  numeric vector

Value

  The population standard deviation of the entries in x

Examples

  pop.sd(c(1:6,NA,7:10))
pop.var  

*Calculate the Population Variance*

**Description**

Calculate the variance of a numeric vector if the data constitutes the whole population. Note that missing values are excluded.

**Usage**

`pop.var(x)`

**Arguments**

- `x`  
  numeric vector

**Value**

The population variance of the entries in `x`

**Examples**

```r
pop.var(c(1:6,NA,7:10))
```

---

**predict.regsubsets  

*Obtain Predictions using Subset Selection*

**Description**

Predict responses for the best model in a subset selection with a specific number of predictors.

**Usage**

```r
## S3 method for class 'regsubsets'
predict(object, d, newdata, ...)
```

**Arguments**

- `object`  
  An object of type "regsubsets"
- `d`  
  Number of data predictors
- `newdata`  
  Dataset for which to predict responses
- `...`  
  Additional arguments

**Value**

A set of predicted responses for `newdata`
projMatrix

Create the Projection Matrix of a Matrix

Description
Calculates the projection matrix for a full-rank matrix X with its number of rows greater than or equal to its number of columns.

Usage
projMatrix(X)

Arguments
X
npx Matrix; must be full-rank and have n >= p

Value
Projection matrix of X.

Examples
projMatrix(matrix(c(3,4,-1,2,1,1),3))

qqlineHalf

Line through a Half-Normal Plot

Description
Plot a line through the first and third quantile of a halfnormal line.

Usage
qqlineHalf(x)

Arguments
x
numeric vector

Value
No return value, called for side effects.
Examples

```r
z = rnorm(100)
faraway::halfnorm(z)
qqlineHalf(z)
```

---

**rcpp_hello_world**  
*Simple function using Rcpp*

---

**Description**

Simple function using Rcpp

**Usage**

```r
rcpp_hello_world()
```

**Examples**

```r
## Not run:
rcpp_hello_world()
## End(Not run)
```

---

**ROCcurve**  
*Plot the ROC curve*

---

**Description**

Plot the ROC curve for logistic regression, LDA, or QDA models.

**Usage**

```r
ROCcurve(mod, nw = NULL)
```

**Arguments**

- `mod`: A logistic regression, LDA, or QDA model
- `nw`: A dataset for which a testing ROC curve should be plotted using the model in `mod`. Note that it must contain the predictors as well as the responses. If this argument is NULL (the default) the training ROC curve will be plotted.

**Value**

A plot with the ROC curve will be produced, nothing is returned.

**Examples**

```r
gmod = glm(state~., binomial, Puromycin)
ROCcurve(gmod)
```
**ROCKnn**

**Description**

Plot the ROC curve for a KNN model. Note that it can only be used when the response is dichotomous.

**Usage**

ROCKnn(mod, response)

**Arguments**

- `mod` The output of the knn function, run with prob=TRUE
- `response` A vector with responses for the testing dataset used to run the knn function.

**Value**

A plot with the ROC curve will be produced, nothing is returned.

**Examples**

```r
yhat = class::knn(Puromycin[,c("conc","rate")],Puromycin[,c("conc","rate")],
                  Puromycin$state,10,prob=TRUE)
ROCKnn(yhat,Puromycin$state)
```

**round2**

**Round to the Nearest Number**

**Description**

Round to the nearest number with the number of digits as indicated. NOTE: Unlike the base round function it rounds a 5 to the higher number, rather than the nearest even number.

**Usage**

round2(x, digits = 0)

**Arguments**

- `x` number to be rounded
- `digits` number of digits to round to

**Value**

Number rounded to the number of digits indicated
span3D  
*Span of a Matrix*

**Description**
Displays a perspective plot showing the plane that is the span of a matrix.

**Usage**
```r
span3D(M, th = 0, ph = 15)
```

**Arguments**
- `M`: Matrix for which the span should be shown.
- `th`: A vector indicating the horizontal angle at which the plot should be shown.
- `ph`: A vector indicating the vertical angle at which the plot should be shown.

**Value**
A matrix containing the plot coordinates (used when adding features).

**Examples**
```r
span3D(matrix(c(1,0,0,1,1,1),3))
```

---

systemEq  
*Solve a System of Equations*

**Description**
Solve a system of equations if it has a unique solution; output an error message otherwise.

**Usage**
```r
systemEq(A, y)
```

**Arguments**
- `A`: matrix A in Ax=y
- `y`: output vector in Ax=y
vector2D

Value

the unique solution $x$ to $Ax=y$

Examples

```r
systemEq(matrix(c(1:3,2,4,4),3),c(3,6,7))
```

---

**vector2D**  
*Add a Vector to a 2D Coordinate System*

**Description**

Add a Vector to a 2D Coordinate System

**Usage**

```r
vector2D(v, fr = c(0, 0), col = 2)
```

**Arguments**

- `v`: A vector with 2 entries.
- `fr`: Vector containing the point at which the vector should start (defaults to the origin).
- `col`: Color of the vector (defaults to red).

**Value**

No return value, called for side effects

**Examples**

```r
a=c(2,4)
b=c(0,3)
coord2D()
vector2D(a)
vector2D(b)
vector2D(a-b,b,"blue")
```
vector3D  Add a Vector to a 3D Coordinate System

Description
Add a Vector to a 3D Coordinate System

Usage
vector3D(pl, v, fr = rep(0, 3), col = "red")

Arguments
- pl  Matrix containing the current plot coordinates.
- v   A vector with 3 entries.
- fr  The point at which the vector should start (defaults to the origin).
- col Color of the vector (defaults to red).

Value
No return value, called for side effects

Examples
a=c(2,4,8)
b=c(6,0,4)
pl=coord3D()
vector3D(pl,a)
vector3D(pl,b)
vector3D(pl,a-b,b,3)

weight_distribution  Weight Distribution of a Graph

Description
Obtain the weight distribution of a graph, indicating for each strength from zero to the maximum strength of any vertex, the proportion of vertices with such a strength. This assumes positive integer weights.

Usage
weight_distribution(g, cumulative = FALSE, ...)

withinPC

Arguments

- **g**
  - the graph (an igraph object)
- **cumulative**
  - TRUE if cumulative weights are to be used; default is FALSE
- **...**
  - additional parameters to be passed to the igraph function `strength`

Value

A vector with the weighted degree distribution for the graph `g`.

Examples

```r
g <- igraph::graph_from_literal(1-2,2-3:4,3-4:5:6,5-1)
igraph::E(g)$weight <- c(1,2,1,4,2,1,1)
table(igraph::strength(g))/6
weight_distribution(g)
```

withinPC

*Calculate Row or Column Percentages*

Description

Calculate percentages of values in a matrix or table with respect to the row or column totals.

Usage

```r
withinPC(X, rows = TRUE, rnd = 1)
```

Arguments

- **X**
  - matrix or table
- **rows**
  - TRUE (default) to calculate by rows, or FALSE to calculate by columns
- **rnd**
  - numbers of digits to round the result to

Value

A matrix or table with percentages

Examples

```r
(X <- matrix(c(1:12), 3))
withinPC(X)
```
# Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>package</em></td>
<td>smallstuff-package, 2</td>
</tr>
<tr>
<td>allspan3D</td>
<td>3</td>
</tr>
<tr>
<td>allvectors3D</td>
<td>4</td>
</tr>
<tr>
<td>as_adj_def</td>
<td>4</td>
</tr>
<tr>
<td>CI</td>
<td>5</td>
</tr>
<tr>
<td>coord2D</td>
<td>6</td>
</tr>
<tr>
<td>coord3D</td>
<td>6</td>
</tr>
<tr>
<td>crossing2</td>
<td>7</td>
</tr>
<tr>
<td>CVerror</td>
<td>7</td>
</tr>
<tr>
<td>CVerrorknn</td>
<td>8</td>
</tr>
<tr>
<td>dataSet</td>
<td>9</td>
</tr>
<tr>
<td>dCohen</td>
<td>9</td>
</tr>
<tr>
<td>get_subgraphs</td>
<td>10</td>
</tr>
<tr>
<td>graph_attr_from_df</td>
<td>11</td>
</tr>
<tr>
<td>impNA</td>
<td>11</td>
</tr>
<tr>
<td>isInt</td>
<td>12</td>
</tr>
<tr>
<td>laCrossProd</td>
<td>13</td>
</tr>
<tr>
<td>lines3D</td>
<td>13</td>
</tr>
<tr>
<td>lmPartReg</td>
<td>14</td>
</tr>
<tr>
<td>lmSub</td>
<td>15</td>
</tr>
<tr>
<td>logistErrorRate</td>
<td>15</td>
</tr>
<tr>
<td>outliers</td>
<td>16</td>
</tr>
<tr>
<td>plotCol</td>
<td>16</td>
</tr>
<tr>
<td>pop.sd</td>
<td>17</td>
</tr>
<tr>
<td>pop.var</td>
<td>18</td>
</tr>
<tr>
<td>predict.regsubsets</td>
<td>18</td>
</tr>
<tr>
<td>projMatrix</td>
<td>19</td>
</tr>
<tr>
<td>qqlineHalf</td>
<td>19</td>
</tr>
<tr>
<td>rcpp_hello_world</td>
<td>20</td>
</tr>
<tr>
<td>ROCcurve</td>
<td>20</td>
</tr>
<tr>
<td>ROCknn</td>
<td>21</td>
</tr>
<tr>
<td>round2</td>
<td>21</td>
</tr>
<tr>
<td>smallstuff (smallstuff-package)</td>
<td>2</td>
</tr>
<tr>
<td>smallstuff-package</td>
<td>2</td>
</tr>
<tr>
<td>span3D</td>
<td>22</td>
</tr>
<tr>
<td>systemEq</td>
<td>22</td>
</tr>
<tr>
<td>vector2D</td>
<td>23</td>
</tr>
<tr>
<td>vector3D</td>
<td>24</td>
</tr>
<tr>
<td>weight_distribution</td>
<td>24</td>
</tr>
<tr>
<td>withinPC</td>
<td>25</td>
</tr>
</tbody>
</table>