Package ‘nullabor’

February 20, 2015

Version 0.3.1

Description  Tools for visual inference. Generate null data sets
              and null plots using permutation and simulation. Calculate distance metrics
              for a lineup, and examine the distributions of metrics.

Title  Tools for Graphical Inference

Author  Hadley Wickham <h.wickham@gmail.com>, Niladri Roy Chowdhury
        <niladri.ia@gmail.com>, Di Cook <dicook@iastate.edu>

Maintainer  Di Cook <dicook@iastate.edu>

License  GPL

Depends  ggplot2

Imports  MASS, plyr, dplyr, moments, fpc

Suggests  reshape2, knitr

LazyData  true

Type  Package

LazyLoad  false

VignetteBuilder  knitr

NeedsCompilation  no

Repository  CRAN

Date/Publication  2014-12-17 17:57:32

R topics documented:

  bin_dist  ................................................................. 2
  box_dist  ............................................................... 3
  calc_diff  ............................................................... 3
  calc_mean_dist  ....................................................... 4
  decrypt  ................................................................. 5
  distmet  ................................................................. 5
  distplot  ............................................................... 7
  lal  ................................................................. 8
bin_dist

Description

Data X is binned into X.bin bins in x-direction and Y.bins in y-direction. The number of points in each cell is then counted. Same is done for data PX. An euclidean distance is calculated between the number of points in each cell between X and PX.

Usage

```r
bin_dist(X, PX, lineup.dat = lineup.dat, X.bin = 5, Y.bin = 5)
```

Arguments

- `X`: a data.frame with two variables, the first two columns are used
- `PX`: another data.frame with two variables, the first two columns are used
- `lineup.dat`: lineup data so that the binning is done based on the lineup data and not the individual plots, by default lineup.dat = lineup.dat; if one wishes to calculate the binned distance between two plots, one should use lineup.dat = NULL
- `X.bin`: number of bins on the x-direction, by default X.bin = 5
- `Y.bin`: number of bins on the y-direction, by default Y.bin = 5

Value

distance between X and PX

Examples

```r
with(mtcars, bin_dist(data.frame(wt, mpg), data.frame(sample(wt), mpg), lineup.dat = NULL))
```
**box_dist**

Distance based on side by side Boxplots for two levels

**Description**

Assuming there are only two groups, the first quartile, median and third quartile is calculated for each group of data X. The absolute difference between these statistics between the two groups are then calculated. Same is done for data PX. Finally an euclidean distance is calculated between the absolute differences of X and PX.

**Usage**

```r
box_dist(X, PX)
```

**Arguments**

- `X`: a data.frame with one factor variable and one continuous variable
- `PX`: a data.frame with one factor variable and one continuous variable

**Value**

distance between X and PX

**Examples**

```r
if(require('dplyr')) {with(mtcars, box_dist(data.frame(as.factor(am), mpg),
data.frame(as.factor(sample(am)), mpg)))}
```

**calc_diff**

Calculating the difference between true plot and the null plot with the maximum distance.

**Description**

Distance metric is used to calculate the mean distance between the true plot and all the null plots in a lineup. The difference between the mean distance of the true plot and the maximum mean distance of the null plots is calculated.

**Usage**

```r
calc_diff(lineup.dat, var, met, pos, dist.arg = NULL, m = 20)
```
**Arguments**

- `lineupNdat`: lineup data to get the lineup
- `var`: a vector of names of the variables to be used to calculate the difference
- `met`: distance metric needed to calculate the distance as a character
- `pos`: position of the true plot in the lineup
- `distNarg`: a list or vector of inputs for the distance metric `met`; NULL by default
- `m`: number of plots in the lineup, by default `m = 20`

**Value**

difference between the mean distance of the true plot and the maximum mean distance of the null plots

**Examples**

```r
if(require('dplyr')){
  lineup.dat <- lineup(null_permute('mpg'), mtcars, pos = 1)
  calc_diff(lineup.dat, var = c('mpg', 'wt'), met = 'bin_dist',
            dist.arg = list(lineup.dat = lineup.dat, X.bin = 5, Y.bin = 5), pos = 1, m = 8))
}
if(require('dplyr')){
  calc_diff(lineup(null_permute('mpg'), mtcars, pos = 1), var = c('mpg', 'wt'), met = 'reg_dist',
            dist.arg = NULL, pos = 1, m = 8))
```

---

**calc_mean_dist**: Calculating the mean distances of each plot in the lineup.

**Description**

Distance metric is used to calculate the mean distance between the true plot and all the null plots in a lineup. The mean distances of each null plot to all the other null plots are calculated. The mean distances are returned for all the plots in the lineup.

**Usage**

`calc_mean_dist(lineup.dat, var, met, pos, dist.arg = NULL, m = 20)`

**Arguments**

- `lineup.dat`: lineup data of the lineup
- `var`: a vector of names of the variables to be used to calculate the mean distances
- `met`: distance metric needed to calculate the distance as a character
- `pos`: position of the true plot in the lineup
- `dist.arg`: a list or vector of inputs for the distance metric `met`; NULL by default
- `m`: number of plots in the lineup, by default `m = 20`
decrypt

**Value**
the mean distances of each plot in the lineup

**Examples**

```r
if(require('dplyr')){
calc_mean_dist(lineup(null_permute('mpg'), mtcars, pos = 1), var = c('mpg', 'wt'),
met = 'reg_dist', pos = 1))
```

---

**decrypt**  
*Use decrypt to reveal the position of the real data.*

**Description**
The real data position is encrypted by the lineup function, and writes this out as a text string. Decrypt, decrypts this text string to reveal which where the real data is.

**Usage**
decrypt(...)

**Arguments**

...  
character vector to decrypt

**Examples**
decrypt('0uXR2p rut L202')

---

**distmet**  
*Empirical distribution of the distance*

**Description**
The empirical distribution of the distance measures is calculated based on the mean distance of each of the null plots from the other null plots in a lineup. At this moment this method works only for `null_permute` method. This function helps get some assessment of whether the actual data plot is very different from the null plots.

**Usage**
distmet(lineup.dat, var, met, method, pos, repl = 1000, dist.arg = NULL,  
m = 20)
Arguments

- **lineup.dat**: lineup data
- **var**: a vector of names of the variables to be used
- **met**: distance metric needed to calculate the distance as a character
- **method**: method for generating null data sets
- **pos**: position of the observed data in the lineup
- **repl**: number of sets of null plots selected to obtain the distribution; 1000 by default
- **dist.arg**: a list or vector of inputs for the distance metric met; NULL by default
- **m**: the number of plots in the lineup; m = 20 by default

Value

- **lineup** has the data used for the calculations
- **null_values** contains new null samples from which to compare nulls in lineup
- **diff** difference in distance between nulls and actual data and that of the null that is most different from other nulls. A negative value means that the actual data plot is similar to the null plots.
- **closest** list of the five closest nulls to the actual data plot
- **pos** position of the actual data plot in the lineup

Examples

```r
if (require("dplyr")) {
  d <- lineup(null_permute("mpg"), mtcars, pos = 1)
  dd <- distmet(d, var = c("mpg", "wt"),
                'reg_dist', null_permute("mpg"), pos = 1, repl = 100, m = 8)
  distplot(dd, m=8)
}

d <- lineup(null_permute("mpg"), mtcars, pos=4, n=8)
qplot(mpg, wt, data = d, geom = 'point') + facet_wrap(~ .sample, ncol=4)
if (require("dplyr")) {
  dd <- distmet(d, var = c("mpg", "wt"), 'bin_dist', null_permute("mpg"),
                pos = 4, repl = 100, dist.arg = list(lineup.dat = d, X.bin = 5,
                Y.bin = 5), m = 8)
  distplot(dd, m=8)
}

# Example using bin_dist
## Not run:
if (require("dplyr")) {
  d <- lineup(null_permute("mpg"), mtcars, pos = 1)
  qplot(mpg, wt, data=d, geom='point') + facet_wrap(~ .sample, ncol=5)
  dd <- distmet(d, var = c("mpg", "wt"),
                'bin_dist', null_permute("mpg"), pos = 1, repl = 500,
```
distplot

Plotting the distribution of the distance measure

Description

The permutation distribution of the distance measure is plotted with the distances for the null plots. Distance measure values for the null plots and the true plot are overlaid.

Usage

distplot(dat, m = 20)

Arguments

dat: output from distmet

m: the number of plots in the lineup; m = 20 by default

Examples

if (require('dplyr')) {
  d <- lineup(null_permute('mpg', mtcars, pos = 1)
  qplot(mpg, wt, data=d) + facet_wrap(~.sample)
  distplot(distmet(d, var = c('mpg', 'wt'), 'reg_dist', null_permute('mpg'),
                 pos = 1, repl = 100, m = 8), m = 8)
}
Los Angeles Lakers play-by-play data.

Description

Play by play data from all games played by the Los Angeles Lakers in the 2008/2009 season.

The line-up protocol.

Description

In this protocol the plot of the real data is embedded amongst a field of plots of data generated to be consistent with some null hypothesis. If the observer can pick the real data as different from the others, this lends weight to the statistical significance of the structure in the plot. The protocol is described in Buja, Cook, Hofmann, Lawrence, Lee, Swayne, Wickham (2009) Statistical inference for exploratory data analysis and model diagnostics, Phil. Trans. R. Soc. A, 367, 4361-4383.

Usage

```r
lineup(method, true = NULL, n = 20, pos = sample(n, 1), samples = NULL)
```

Arguments

- `method`: method for generating null data sets
- `true`: true data set. If NULL, `find_plot_data` will attempt to extract it from the current ggplot2 plot.
- `n`: total number of samples to generate (including true data)
- `pos`: position of true data. Leave missing to pick position at random. Encrypted position will be printed on the command line, `decrypt` to understand.
- `samples`: samples generated under the null hypothesis. Only specify this if you don’t want `lineup` to generate the data for you.

Details

Generate n - 1 null datasets and randomly position the true data. If you pick the real data as being noticeably different, then you have formally established that it is different to with p-value 1/n.

Examples

```r
if (require('ggplot2')) {
  qplot(mpg, wt, data = mtcars) %+% 
  lineup(null_permute('mpg'), mtcars) + 
  facet_wrap(~ .sample)
  qplot(mpg, .sample, data = lineup(null_permute('cyl'), mtcars), 
    colour = factor(cyl))
}
```
null_dist

Generate null data with a specific distribution.

Description

Null hypothesis: variable has specified distribution

Usage

null_dist(var, dist, params = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variable name</td>
</tr>
<tr>
<td>dist</td>
<td>distribution name. One of: beta, cauchy, chi-squared, exponential, f, gamma, geometric, log-normal, lognormal, logistic, negative binomial, normal, poisson, t, weibull</td>
</tr>
<tr>
<td>params</td>
<td>list of parameters of distribution. If NULL, will use fitdistr to estimate them.</td>
</tr>
</tbody>
</table>

Value

a function that given data generates a null data set. For use with lineup or rorschach

null_lm

Generate null data with null residuals from a model.

Description

Null hypothesis: variable is linear combination of predictors

Usage

null_lm(f, method = "rotate", ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>model specification formula, as defined by lm</td>
</tr>
<tr>
<td>method</td>
<td>method for generating null residuals. Built in methods 'rotate', 'pboot' and 'boot' are defined by resid_rotate, resid_pboot and resid_boot respectively</td>
</tr>
<tr>
<td>...</td>
<td>other arguments passed onto method.</td>
</tr>
</tbody>
</table>

Value

a function that given data generates a null data set. For use with lineup or rorschach
Examples

```r
if (requireNamespace('reshape2', quietly = TRUE)) {
  data("tips", package = "reshape2")
  x <- lm(tip ~ total_bill, data = tips)
  tips.reg <- data.frame(tips, resid = residuals(x), fitted = fitted(x))
  qplot(total_bill, resid, data = tips.reg) %>%
    lineup(null_lm(tip ~ total_bill, method = 'rotate'), tips.reg) +
    facet_wrap(~ .sample)
}
```

null_permute  

Generate null data by permuting a variable.

Description

Null hypothesis: variable is independent of others

Usage

```r
null_permute(var)
```

Arguments

- `var` name of variable to permute

Value

a function that given data generates a null data set. For use with `lineup` or `rorschach`

opt_bin_diff  

Finds the number of bins in x and y direction which gives the maximum binned distance.

Description

This function finds the optimal number of bins in both x and y direction which should be used to calculate the binned distance. The binned distance is calculated for each combination of provided choices of number of bins in x and y direction and finds the difference using `calc_diff` for each combination. The combination for which the difference is maximum should be used.

Usage

```r
opt_bin_diff(lineup.dat, var, xlow, xhigh, ylow, yhigh, pos, plot = FALSE,
             m = 20)
```
**Arguments**

- `lineup.dat` lineup data to get the lineup
- `var` a list of names of the variables to be used to calculate the difference
- `xlow` the lowest value of number of bins on the x-direction
- `xhigh` the highest value of number of bins on the x-direction
- `ylow` the lowest value of number of bins on the y-direction
- `yhigh` the highest value of number of bins on the y-direction
- `pos` position of the true plot in the lineup
- `plot` LOGICAL; if true, returns a tile plot for the combinations of number of bins with the differences as weights
- `m` number of plots in the lineup, by default m = 20

**Value**

a dataframe with the number of bins and differences the maximum mean distance of the null plots

**Examples**

```r
if(require('dplyr')){
  opt_bin_diff(lineup(null_permute('mpg', mtcars, pos = 1), var = c('mpg', 'wt'),
2, 5, 4, 8, pos = 1, plot = TRUE, m = 8))
}
```

---

**reg_dist**

*Distance based on the regression parameters*

**Description**

Dataset X is binned into 5 bins in x-direction. A regression line is fitted to the data in each bin and the regression coefficients are noted. Same is done for dataset PX. An euclidean distance is calculated between the two sets of regression parameters. If the relationship between X and PX looks linear, number of bins should be equal to 1.

**Usage**

```r
reg_dist(X, PX, nbins = 1)
```

**Arguments**

- `X` a data.frame with two variables, the first column giving the explanatory variable and the second column giving the response variable
- `PX` another data.frame with two variables, the first column giving the explanatory variable and the second column giving the response variable
- `nbins` number of bins on the x-direction, by default nbins = 1
Value

distance between X and PX

Examples

```r
with(mtcars, reg_dist(data.frame(wt, mpg), data.frame(sample(wt, mpg))))
```

---

**resid_boot**

*Bootstrap residuals.*

Description

For use with `null_lm`

Usage

```r
resid_boot(model, data)
```

Arguments

- `model` to extract residuals from
- `data` used to fit model

---

**resid_pboot**

*Parametric bootstrap residuals.*

Description

For use with `null_lm`

Usage

```r
resid_pboot(model, data)
```

Arguments

- `model` to extract residuals from
- `data` used to fit model
resid_rotate

Rotation residuals.

Description

For use with `null_lm`

Usage

`resid_rotate(model, data)`

Arguments

- `model`: to extract residuals from
- `data`: used to fit model

resid_sigma

Residuals simulated by a normal model, with specified sigma

Description

For use with `null_lm`

Usage

`resid_sigma(model, data, sigma = 1)`

Arguments

- `model`: to extract residuals from
- `data`: used to fit model
- `sigma`: a specific sigma to model
rorschach

**Description**

This protocol is used to calibrate the eyes for variation due to sampling. All plots are typically null data sets, data that is consistent with a null hypothesis. The protocol is described in Buja, Cook, Hofmann, Lawrence, Lee, Swayne, Wickham (2009) Statistical inference for exploratory data analysis and model diagnostics, Phil. Trans. R. Soc. A, 367, 4361-4383.

**Usage**

rorschach(method, true = NULL, n = R, p = 0)

**Arguments**

- **method**: method for generating null data sets
- **true**: true data set. If NULL, find_plot_data will attempt to extract it from the current ggplot2 plot.
- **n**: total number of samples to generate (including true data)
- **p**: probability of including true data with null data.

**sep_dist**

**Description**

The separation between clusters is defined by the minimum distances of a point in the cluster to a point in another cluster. The number of clusters are provided. If not, the hierarchical clustering method is used to obtain the clusters. The separation between the clusters for dataset X is calculated. Same is done for dataset PX. An euclidean distance is then calculated between these separation for X and PX.

**Usage**

sep_dist(X, PX, clustering = FALSE, nclust = 3)

**Arguments**

- **X**: a data.frame with two or three columns, the first two columns providing the dataset
- **PX**: a data.frame with two or three columns, the first two columns providing the dataset
- **clustering**: LOGICAL; if TRUE, the third column is used as the clustering variable, by default FALSE
- **nclust**: the number of clusters to be obtained by hierarchical clustering, by default nclust = 3
uni_dist

Value

distance between X and PX

Examples

if(require('fpc')) { with(mtcars, sep_dist(data.frame(wt, mpg, as.numeric(as.factor(mtcars$cyl))), data.frame(sample(wt), mpg, as.numeric(as.factor(mtcars$cyl))), clustering = TRUE))}

uni_dist Distance for univariate data

Description

The first four moments is calculated for data X and data PX. An euclidean distance is calculated between these moments for X and PX.

Usage

uni_dist(X, PX)

Arguments

X a data.frame where the first column is only used
PX another data.frame where the first column is only used

Value
distance between X and PX

Examples

if(require('moments')) { uni_dist(rnorm(100), rpois(100, 2))}
Index

bin_dist, 2
box_dist, 3

calc_diff, 3
calc_mean_dist, 4

decrypt, 5, 8
distmet, 5, 7
distplot, 7

find_plot_data, 8, 14
fitdistr, 9

lal, 8
lineup, 8, 9, 10
lm, 9

null_dist, 9
null_lm, 9, 12, 13
null_permute, 5, 10

opt_bin_diff, 10
reg_dist, 11
resid_boot, 9, 12
resid_pboot, 9, 12
resid_rotate, 9, 13
resid_sigma, 13
rorschach, 9, 10, 14

sep_dist, 14
uni_dist, 15