Package ‘reverseR’

April 24, 2019

Type Package

LazyLoad no

LazyData yes

Title Linear Regression Stability to Significance Reversal

Version 0.1

Date 2019-04-18

Author Andrej-Nikolai Spiess <a.spiess@uke.uni-hamburg.de>
Michal Burdukiewicz <michalburdukiewicz@gmail.com>
Stefan Roediger <stefan.roediger@b-tu.de>

Maintainer Andrej-Nikolai Spiess <a.spiess@uke.uni-hamburg.de>

Description Tests linear regressions for significance reversal through leave-one(multiple)-out and shifting/addition of response values. The paradigm of the package is loosely based on the somewhat forgotten ``dfstat'' criterion (Belsley, Kuh & Welsch 1980 <doi:10.1002/0471725153.ch2>), which tests influential values in linear models from their effect on statistical inference, i.e. changes in p-value.

License GPL (>= 2)

Depends R (>= 2.13.0), shiny, markdown, knitr

Imports DT

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-24 15:40:03 UTC

R topics documented:

`Influence plots` *Several diagnostic plots for checking p-value influencers*

Description

Seven different plot types that visualize *p*-value influencers.

1. `lmPlot`: plots the linear regression, marks the influencer(s) in red and displays trend lines for the full and leave-one-out (LOO) data set (black and red, respectively).
2. `pvalPlot`: plots the *p*-values for each LOO data point and displays the values as a full model/LOO model plot, together with the `alpha` border as defined in `lmInfl`.
3. `inflPlot`: plots `dfbeta` for slope, `dffits`, `covratio`, `cooks.distance`, leverage (`hatvalues`) and studentized residuals (`rstudent`) against the Δp-value. Herewith, changes in these six parameters can be compared to the effect on the corresponding drop/rise in *p*-value. The plots include vertical boundaries for threshold values as defined in the literature under 'References'.
4. `slsePlot`: plots all LOO-slopes and their standard errors together with the corresponding original model values and a t-value border as calculated by $Q_t(1 - \frac{\alpha}{2}, n - 2)$. LOO of points on the right of this border result in a significant model, and *vice versa*.
5. `threshPlot`: plots the output of `lmThresh`, i.e. the regression plot including confidence/prediction intervals, as well as for each response value y_i the region in which the model is significant (green). This is tested for either i) y_i that are shifted into this region (`newobs = FALSE` in `lmThresh`) or ii) when a new observation $y2_i$ is added (`newobs = TRUE` in `lmThresh`). In the latter case, it is informative if this region resides within the prediction interval (dashed line), indicating that a future additional measurement at x_i might reverse the significance statement.
6. `multPlot`: plots the output of `lmMult` as a point cloud of *p*-values for each 1...`max` sample removals and `n` combinations. All combinations for which the sample removal resulted in a significance reversal are colored in red, the percentages of these are given on top of the plot.
7. `stabPlot`: for single (to be selected) response values from the output of `lmThresh`, this function displays the region of significance reversal within the surrounding prediction interval. The probability of a either shifting the response value (if `lmThresh(..., newobs = FALSE)`) or of including a future (measurement) point (if `lmThresh(..., newobs = TRUE)`) to reverse the significance is shown as the integral between the "end of significance region" (eosr) and the nearest prediction interval boundary.

NOTE: The visual display should always be supplemented with the corresponding `stability` analysis.

Usage

```
lmPlot(infl, ...)
pvalPlot(infl, ...)
inflPlot(infl, ...)
slsePlot(infl, ...)
threshPlot(thresh, bands = FALSE, ...)
multPlot(mult, log = FALSE, ...)
stabPlot(stab, which = NULL, ...)
```

Arguments

`infl`	an object obtained from `lmInfl`.
`thresh`	an object obtained from `lmThresh`.
`stab`	an object obtained from using `stability` on an `lmThresh` output.
`bands`	logical. If `TRUE`, plots the confidence and prediction bands.
`mult`	an object obtained from `lmMult`.
`log`	should the *p*-values be displayed on a logarithmic y-axis?
`which`	which response value should be shown in `stabPlot`?
`...`	other plotting parameters.

Value

The corresponding plot.

Note

Cut-off values for the different influence measures are those defined in Belsley, Kuh E & Welsch (1980):

dfbeta slope: $|\Delta\beta1_i| > 2/\sqrt{n}$
dffits: $|\mathrm{dffits}_i| > 2\sqrt{2/n}$
covratio: $|\mathrm{covr}_i - 1| > 3k/n$
Cook's D: $D_i > Q_F(0.5, k, n-k)$
leverage: $h_{ii} > 2k/n$
studentized residual: $t_i > Q_t(0.975, n-k-1)$

Author(s)

Andrej-Nikolai Spiess

References

Regression diagnostics: Identifying influential data and sources of collinearity.
Belsley DA, Kuh E, Welsch RE.
John Wiley, New York (1980).

Applied Regression Analysis: A Research Tool.
Rawlings JO, Pantula SG, Dickey DA.
Springer; 2nd Corrected ed. 1998. Corr. 2nd printing 2001.

Applied Regression Analysis and Generalized Linear Models.
Fox J.
SAGE Publishing, 3rd ed, 2016.

Examples

```
## See Examples in 'lmInfl', 'lmThresh' and 'lmMult'.
```

`lmExact`	*Create random values that deliver linear regressions with exact parameters*

Description

Takes self-supplied x/y values or x/random values and transforms these as to deliver linear regressions $y = \beta_0 + \beta_1 x + \varepsilon$ (with potential replicates) with either

1) exact slope β_1 and intercept β_0,
2) exact *p*-value and intercept β_0, or
3) exact R^2 and intercept β_0.

Intended for testing and education, not for cheating ! ;-)

Usage

```
lmExact(x = 1:20, y = NULL, ny = 1, intercept = 0, slope = 0.1, error = 0.1,
        seed = 123, pval = NULL, rsq = NULL, plot = TRUE, verbose = FALSE, ...)
```

Arguments

`x`	the predictor values.
`y`	`NULL`. A possible vector of y values with `length(x)`.
`ny`	the number of replicate response values per predictor value.
`intercept`	the desired intercept β_0.
`slope`	the desired slope β_1.
`error`	if a single value, the standard deviation σ for sampling from a normal distribution, or a user-supplied vector of length `x` with random deviates.
`seed`	the random generator seed for reproducibility.
`pval`	the desired *p*-value of the slope.
`rsq`	the desired R^2.
`plot`	logical. If `TRUE`, the linear regression is plotted.
`verbose`	logical. If `TRUE`, a summary is printed to the console.
`...`	other arguments to `lm` or `plot`.

Details

For case **1)**, the `error` values are added to the exact $(x_i, \beta_0 + \beta_1 x_i)$ values, the linear model $y_i = \beta_0 + \beta_1 x_i + \varepsilon$ is fit, and the residuals $y_i - \hat{y}_i$ are re-added to $(x_i, \beta_0 + \beta_1 x_i)$.
For case **2)**, the same as in **1)** is conducted, however the slope delivering the desired *p*-value is found by an optimizing algorithm.
Finally, for case **3)**, a QR reconstruction, rescaling and refitting is conducted, using the code found

under 'References'.

If y is supplied, changes in slope, intercept and *p*-value will deliver the sames residuals as the linear regression through x and y. A different R^2 will change the response value structure, however.

Value

A list with the following items:

`lm`	the linear model of class `lm`.
`x`	the predictor values.
`y`	the (random) response values.
`summary`	the model summary for quick checking of obtained parameters.

Using both x and y will give a linear regression with the desired parameter values when refitted.

Author(s)

Andrej-Nikolai Spiess

References

For method **3**):
http://stats.stackexchange.com/questions/15011/generate-a-random-variable-with-a-defined-correlation-to-an-existing-variable.

Examples

```
## No replicates, intercept = 3, slope = 0.2, sigma = 2, n = 20.
res1 <- lmExact(x = 1:20, ny = 1, intercept = 3, slope = 2, error = 2)

## Same as above, but with 3 replicates, sigma = 1,  n = 20.
res2 <- lmExact(x = 1:20, ny = 3, intercept = 3, slope = 2, error = 1)

## No replicates, intercept = 2 and p-value = 0.025, sigma = 3, n = 50.
## => slope = 0.063
res3 <- lmExact(x = 1:50, ny = 1, intercept = 2, pval = 0.025, error = 3)

## 5 replicates, intercept = 1, R-square = 0.85, sigma = 2, n = 10.
## => slope = 0.117
res4 <- lmExact(x = 1:10, ny = 5, intercept = 1, rsq = 0.85, error = 2)

## Heteroscedastic (magnitude-dependent) noise.
error <- sapply(1:20, function(x) rnorm(3, 0, x/10))
res5 <- lmExact(x = 1:20, ny = 3, intercept = 1, slope = 0.2,
                error = error)

## Supply own x/y values, residuals are similar to an
## initial linear regression.
X <- c(1.05, 3, 5.2, 7.5, 10.2, 11.7)
```

```
set.seed(123)
Y <- 0.5 + 2 * X + rnorm(6, 0, 2)
res6 <- lmExact(x = X, y = Y, intercept = 1, slope = 0.2)
all.equal(residuals(lm(Y ~ X)), residuals(res6$lm))
```

`lmInfl` *Checks and analyzes leave-one-out (LOO) p-values in linear regression*

Description

This function calculates leave-one-out (LOO) *p*-values for all data points and identifies those resulting in "significance reversal", i.e. in the *p*-value of the model's slope traversing the user-defined α-level.

Usage

```
lmInfl(model, alpha = 0.05, method = c("pearson", "spearman"), verbose = TRUE, ...)
```

Arguments

`model`	the linear model of class `lm`.
`alpha`	the α-level to use as the threshold border.
`method`	select either parametric (`"pearson"`) or rank-based (`"spearman"`) statistics.
`verbose`	logical. If `TRUE`, results are displayed on the console.
`...`	other arguments to `lm`.

Details

The algorithm
1) calculates the *p*-value of the full model (all points),
2) calculates a LOO-*p*-value for each point removed,
3) checks for significance reversal in all data points and
4) returns all models as well as classical `influence.measures` with LOO-*p*-values, Δp-values, slopes and standard errors attached.
If `method = "spearman"`, *p*-values are based on Spearman Rank correlation, and the values given in the last column of the result matrix are Spearman's ρ.

The idea of *p*-value influencers was first introduced by Belsley, Kuh & Welsch, and described as an influence measure pertaining directly to the change in *t*-statistics, that will "show whether the conclusions of hypothesis testing would be affected", termed **dfstat** in [1, 2, 3] or **dfstud** in [4]:

$$\mathrm{dfstat_{ij}} \equiv \frac{\hat{\beta}_\mathrm{j}}{\mathrm{s}\sqrt{(\mathrm{X'X})^{-1}_\mathrm{jj}}} - \frac{\hat{\beta}_\mathrm{j(i)}}{\mathrm{s_{(i)}}\sqrt{(\mathrm{X'_{(i)}X_{(i)}})^{-1}_\mathrm{jj}}}$$

where $\hat{\beta}_j$ is the *j*-th estimate, *s* is the residual standard error, *X* is the design matrix and (*i*) denotes the *i*-th observation deleted.
dfstat, which for the regression's slope β_1 is the difference of *t*-statistics

$$\Delta t = t_{\beta 1} - t_{\beta 1(i)} = \frac{\beta_1}{\text{s.e.}(\beta_1)} - \frac{\beta_1(i)}{\text{s.e.}(\beta_1(\text{i}))}$$

is inextricably linked to the changes in *p*-value Δp, calculated from

$$\Delta p = p_{\beta 1} - p_{\beta 1(i)} = 2\,(1 - P_t(t_{\beta 1}, \nu)) - 2\left(1 - P_t(t_{\beta 1(i)}, \nu - 1)\right)$$

where P_t is the Student's *t* cumulative distribution function with ν degrees of freedom, and where significance reversal is attained when $\alpha \in [p_{\beta 1}, p_{\beta 1(i)}]$. Interestingly, in linear regression the seemingly mandatory check of the influence of single data points on statistical inference is living in oblivion: apart from [1-4], there is, to the best of our knowledge, no reference to **dfstat** or Δp in current literature on influence measures.

The influence output also includes the more recent Hadi's measure (column "hadi"):

$$H_i = \frac{p_{ii}}{1 - p_{ii}} + \frac{k}{1 - p_{ii}} \frac{d_i^2}{(1 - d_i^2)}$$

where p_{ii} are the diagonals of the hat matrix (leverages), $k = 2$ in univariate linear regression and $d_i = e_i / \sqrt{\text{SSE}}$.

Value

A list with the following items:

`origModel`	the original model with all data points.
`finalModels`	a list of final models with the influencer(s) removed.
`infl`	a matrix with the original data, classical `influence.measures`, studentized residuals, leverages, LOO-*p*-values, LOO-slopes/intercepts and their Δ's, LOO-standard errors and R^2s.
`sel`	a vector with the influencers' indices.
`alpha`	the selected α-level.
`origP`	the original model's *p*-value.
`stab`	the stability measure, see `stability`.

Author(s)

Andrej-Nikolai Spiess

References

For dfstat / dfstud :
1. Regression diagnostics: Identifying influential data and sources of collinearity.
Belsley DA, Kuh E, Welsch RE.

John Wiley, New York, USA (2004).

2. Econometrics, 5ed.
Baltagi B.
Springer-Verlag Berlin, Germany (2011).

3. Growth regressions and what the textbooks don't tell you.
Temple J.
Bull Econom Res, **52**, 2000, 181-205.

4. Robust Regression and Outlier Detection.
Rousseeuw PJ & Leroy AM.
John Wiley & Sons, New York, NY (1987).

Hadi's measure:
A new measure of overall potential influence in linear regression.
Hadi AS.
Comp Stat & Data Anal, **14**, 1992, 1-27.

Examples

```
## Example #1 with single influencers and insignificant model (p = 0.115).
## Removal of #18 results in p = 0.0227!
set.seed(123)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(20, 0, 1)
LM1 <- lm(b ~ a)
res1 <- lmInfl(LM1)
lmPlot(res1)
pvalPlot(res1)
inflPlot(res1)
slsePlot(res1)
stability(res1)

## Example #2 with multiple influencers and significant model (p = 0.0269).
## Removal of #2, #17, #18 or #20 result in crossing p = 0.05!
set.seed(125)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(20, 0, 1)
LM2 <- lm(b ~ a)
res2 <- lmInfl(LM2)
lmPlot(res2)
pvalPlot(res2)
inflPlot(res2)
slsePlot(res2)
stability(res2)

## Large Example #3 with top 10 influencers and significant model (p = 6.72E-8).
## Not possible to achieve a crossing of alpha with any point despite strong noise.
set.seed(123)
```

```
a <- 1:100
b <- 5 + 0.08 * a + rnorm(100, 0, 5)
LM3 <- lm(b ~ a)
res3 <- lmInfl(LM3)
lmPlot(res3)
stability(res3)

## Example #4 with replicates and single influencer (p = 0.114).
## Removal of #58 results in p = 0.039.
set.seed(123)
a <- rep(1:20, each = 3)
b <- 5 + 0.08 * a + rnorm(20, 0, 2)
LM4 <- lm(b ~ a)
res4 <- lmInfl(LM4)
lmPlot(res4)
pvalPlot(res4)
inflPlot(res4)
slsePlot(res4)
stability(res4)

## As Example #1, but with weights.
## Removal of #18 results in p = 0.04747.
set.seed(123)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(20, 0, 1)
LM5 <- lm(b ~ a, weights = 1:20)
res5 <- lmInfl(LM5)
lmPlot(res5)
stability(res5)
```

`lmMult` *Checks and analyzes leave-multiple-out (LMO) p-values in linear regression*

Description

This function calculates leave-multiple-out (LMO) *p*-values for an increasing number of data points and identifies those resulting in "significance reversal" of the model, i.e. in the slope's *p*-value traversing the user-defined α-level.

Usage

```
lmMult(model, max = 5, n = 10000, alpha = 0.05,
       method = c("pearson", "spearman"), verbose = TRUE)
```

Arguments

`model`	the linear model of class `lm`.
`max`	the maximum number of points to eliminate.

`n`	the number of samples to draw for each 1...`max`.
`alpha`	the α-level to use as the threshold border.
`method`	select either parametric (`"pearson"`) or rank-based (`"spearman"`) statistics.
`verbose`	logical. If `TRUE`, results for each 1...`max` will be printed to the console.

Details

The algorithm
1) calculates the *p*-value of the full model (all data points),
2) calculates a LMO-*p*-value for all `n` sampled groups of 1...`max` points removed,
3) checks for significance reversal in the resulting model and
4) returns all `n` samples and the corresponding *p*-values.

Value

A list with the following items:

`sample`	a matrix with all `max * n` iterations, where a 1 indicates the left-out sample(s), as well as the corresponding *p*-values and group.
`stat`	for each 1...`max` LMO's, the percentage of model significance reversers.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Example with single influencers and insignificant model (p = 0.115).
set.seed(123)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(20, 0, 1)
LM1 <- lm(b ~ a)
res1 <- lmMult(LM1)
multPlot(res1)
stability(res1)

## Large example with 100 data points and highly significant model (p = 6.72E-8).
## No significance reversal up to the elimination of 20 points.
set.seed(123)
a <- 1:100
b <- 5 + 0.08 * a + rnorm(100, 0, 5)
LM2 <- lm(b ~ a)
res2 <- lmMult(LM2, max = 20)
multPlot(res2)
stability(res2)
```

`lmThresh` *Finds and analyzes significance reversal regions for each response value*

Description

This function finds (by iterating through a grid of values for each response) the approximate response value range(s) in which the regression is significant (when inside) or not (when outside), as defined by `alpha`. Here, two scenarios can be tested: i) if `newobs = FALSE` (default), the model's significance is tested by shifting y_i along the search grid. If `newobs = TRUE`, y_i is kept fixed and a `new` observation y_{2i} is added and shifted along the search grid. Hence, this function tests the regression for the sensitivity of being reversed in its significance through minor shifting of the original or added response values, as opposed to the effect of point removal (`lmInfl`).

Usage

```
lmThresh(model, factor = 5, alpha = 0.05,
         method = c("pearson", "spearman"),
         steps = 10000, newobs = FALSE, ...)
```

Arguments

`model`	the linear model of class `lm`.
`factor`	a factor for the initial search grid. See 'Details'.
`alpha`	the α-level to use as the threshold border.
`method`	select either parametric (`"pearson"`) or rank-based (`"spearman"`) statistics.
`steps`	the number of steps within the search range. See 'Details'.
`newobs`	logical. Should the significance region for each y_i be calculated from shifting y_i or from keeping y_i fixed and adding a new observation $y2_i$?
`...`	other arguments to future methods.

Details

In a first step, a grid is created with a range from $y_i \pm \mathrm{factor} \cdot \mathrm{range}(y_{1 \ldots n})$ with `steps` cuts. For each cut, the *p*-value is calculated for the model when y_i is shifted to that value (`newobs = TRUE`) or a second observation y_{2i} is added to the fixed y_i (`newobs = TRUE`). When the original model $y = \beta_0 + \beta_1 x + \varepsilon$ is significant (*p* < `alpha`), there are two boundaries that result in insignificance: one decreases the slope β_1 and the other inflates the standard error $\mathrm{s.e.}(\beta_1)$ in a way that $P_t(\frac{\beta_1}{\mathrm{s.e.}(\beta_1)}, n - 2) > \alpha$. If the original model was insignificant, also two boundaries exists that either increase β_1 or reduce $\mathrm{s.e.}(\beta_1)$. Often, no boundaries are found and increasing the `factor` grid range may alleviate this problem.

This function is quite fast (~ 300ms/10 response values), as the slope's *p*-value is calculated from the `corr.test` function of the 'psych' package, which utilizes matrix multiplication and vectorized pt calculation. The vector of correlation coefficients r_i from the cor function is transformed to t-values by

$$t_i = \frac{r_i\sqrt{n-2}}{\sqrt{1-r_i^2}}$$

which is equivalent to that employed in the linear regression's slope test.

Value

A list with the following items:

`x`	the predictor values.
`y`	the response values.
`pmat`	the *p*-value matrix, with `length(x)` columns and `steps` rows.
`alpha`	the selected α-level.
`ySeq`	the grid sequence for which the algorithm calculates *p*-values when y_i is shifted within.
`model`	the original `lm` model.
`data`	the original `model.frame`.
`eosr`	the y-values of the ends of the significance region.
`diff`	the Δ value between y_i and the nearest border of significance reversal.
`closest`	the (approx.) value of the nearest border of significance reversal.
`newobs`	should a new observation be added?

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Significant model, no new observation.
set.seed(125)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(length(a), 0, 1)
LM1 <- lm(b ~ a)
res1 <- lmThresh(LM1)
threshPlot(res1)
stability(res1)

## Insignificant model, no new observation.
set.seed(125)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(length(a), 0, 2)
LM2 <- lm(b ~ a)
res2 <- lmThresh(LM2)
threshPlot(res2)
stability(res2)
```

```
## Significant model, new observation.
## Some significance reversal regions
## are within the prediction interval,
## e.g. 1 to 6 and 14 to 20.
set.seed(125)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(length(a), 0, 1)
LM3 <- lm(b ~ a)
res3 <- lmThresh(LM3, newobs = TRUE)
threshPlot(res3)
stability(res3)

## More detailed example to the above:
## a (putative) new observation within the
## prediction interval may reverse significance.
set.seed(125)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(length(a), 0, 1)
LM1 <- lm(b ~ a)
summary(LM1) # => p-value = 0.02688
res1 <- lmThresh(LM1, newobs = TRUE)
threshPlot(res1)
st <- stability(res1, pval = TRUE)
st$stats # => upper prediction boundary = 7.48
         # and eosr = 6.49
stabPlot(st, 1)
## reverse significance if we add a new response y_1 = 7
a <- c(1, a)
b <- c(7, b)
LM2 <- lm(b ~ a)
summary(LM2) # => p-value = 0.0767
```

`shinyInfl` *Initializes a Shiny App with all implemented methods of this package*

Description

A comprehensive Shiny App that facilitates the import of user-supplied data and subsequent detailed testing of all implemented methods in this package. Analysis results can be exported as plots and text.

Usage

```
shinyInfl()
```

Arguments

None.

Value

The analysis including plots and result tables.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## shinyInfl() # <= to initialize
```

`simInfl` *Simulates significance reversals and calculates their influence parameters*

Description

This function simulates linear regressions and stores the parameters and influence measures of all simulations that resulted in LOO significance reversal, developed for research purposes.

Usage

```
simInfl(x = 1:10, slope = 0.02, intercept = 1, error = 0.05, nrev = 1000, ...)
```

Arguments

`x`	the x values to be supplied to `lmExact`.
`slope`	the slope β_1 to be supplied to `lmExact`.
`intercept`	the intercept β_0 to be supplied to `lmExact`.
`error`	the ε value to be supplied to `lmExact`.
`nrev`	the number of desired significance reversals.
`...`	other parameters to `lmExact` and `lmInfl`.

Details

Loops over an undefined number of EXACT regressions (`lmExact`) with incrementing random seeds, stores all models and in case of significance reversal, parameters and influence measures (`lmInfl`). The simulation terminates when `nrev` reversals are counted.

Value

A list with the following two items:

`models`	the linear models of all reversals.
`mat`	the stored matrix with the resulting parameters and influence measures for all `nrev` reversals.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## Example with slight slope, intercept = 0.5 and 10 reversals.
res <- simInfl(x = 1:10, intercept = 0.5, slope = 0.02, error = 0.05, nrev = 10)

## Plot Cook's D versus delta-P values
## and insert common cut-off.
plot(res$mat[, "cook.d"], res$mat[, "dP"], pch = 16, cex = 0.5,
     xlab = "Cook's D", ylab = "delta-P")
thresh <- qf(0.5, 2, 8)  # threshold value for Qf(0.5, npar, df)
abline(v = thresh, col = "darkred", lwd = 2)

## Plot dfbeta slope versus delta-P values
## and insert common cut-off.
plot(res$mat[, "dfb.Slope"], res$mat[, "dP"], pch = 16, cex = 0.5,
     xlab = "dfbeta Slope", ylab = "delta-P")
thresh <- 2/sqrt(10)  # 2/sqrt(N)
abline(v = thresh, col = "darkred", lwd = 2)

## Plot dffits versus delta-P values
## and insert common cut-off.
plot(abs(res$mat[, "dffit"]), res$mat[, "dP"], pch = 16, cex = 0.5,
     xlab = "dffits", ylab = "delta-P")
thresh <- 2 * sqrt(2/10)  # 2 * sqrt(nPar/N)
abline(v = thresh, col = "darkred", lwd = 2)


## More illustrative with more reverser samples!
## Example with slight slope, intercept = 0.5 and 10 reversals.
res <- simInfl(x = 1:10, intercept = 0.5, slope = 0.02, error = 0.05, nrev = 200)
plot(res$mat[, "cook.d"], res$mat[, "dP"], pch = 16, cex = 0.5,
     xlab = "Cook's D", ylab = "delta-P")
thresh <- qf(0.5, 2, 8)  # threshold value for Qf(0.5, npar, df)
abline(v = thresh, col = "darkred", lwd = 2)
```

`stability` *Calculates stability values for results of 'lmInfl', 'lmMult' and 'lmThresh'*

Description

This function calculates stability values for LOO (`lmInfl`), LMO (`lmMult`) and response value shifting/addition (`lmThresh`).

Usage

```
stability(x, pval = FALSE, ...)
```

Arguments

`x`	a result of either `lmInfl`, `lmMult` or `lmThresh`.
`pval`	logical. If `TRUE`, for `lmThresh`, objects an exact p-value is calculated for a future response to reverse significance.
`...`	other parameters, not yet implemented.

Details

For results of `lmInfl`:
A [0, 1]-bounded stability measure $S = 1 - \frac{n}{N}$, with n = number of influencers (significance reversers) and N = total number of response values.

For results of `lmMult`:
For each 1...`max`, the percentage of all resamples that did *NOT* result in significance reversal.

For results of `lmThresh`:
A [0, 1]-bounded stability measure $S = 1 - \frac{n}{N}$, with n = number of response values where one of the ends of the significance region is within the prediction interval and N = total number of response values.
If `pval = TRUE`, the exact p-value is calculated in the following manner:

1) Mean square error (MSE) and prediction standard error (se) are calculated from the linear model:

$$\text{MSE} = \sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n-2} \qquad \text{se}_i = \sqrt{\text{MSE} \cdot \left(1 + \frac{1}{n} + \frac{(x_i - \bar{x}_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x}_i)^2}\right)}$$

2) Upper and lower prediction intervals boundaries are calculated for each $\hat{y}_i$:

$$\hat{y}_i \pm Q_t(\alpha/2, n-2) \cdot \text{se}_\text{i}$$

The prediction interval around $\hat{y}_i$ is a scaled/shifted t-distribution with density function

$$P_{tss}(y, n-2) = \frac{1}{\text{se}_\text{i}} \cdot P_t\left(\frac{y - \hat{y}_i}{\text{se}_\text{i}}, n-2\right)$$

, where P_t is the density function of the central, unit-variance t-distribution.
3) The probability of either shifting the response value (if `lmThresh(..., newobs = FALSE)`) or including a future response value y_{2i} (if `lmThresh(..., newobs = TRUE)`) to reverse the significance of the linear model is calculated as the integral between the end of the significance region (eosr) and the upper/lower $\alpha/2, 1-\alpha/2$ prediction interval:

$$P(\text{reverse}) = \int_{\text{eosr}}^{1-\alpha/2} P_{tss}(y, n-2)dy \quad \text{or} \quad \int_{\alpha/2}^{\text{eosr}} P_{tss}(y, n-2)dy$$

Value

The stability value.

Author(s)

Andrej-Nikolai Spiess

Examples

```
## See examples in 'lmInfl' and 'lmThresh'.

## The implemented strategy of calculating the
## probability of significance reversal, as explained above
## and compared to 'stabPlot'.
set.seed(125)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(length(a), 0, 1)
LM1 <- lm(b ~ a)
res1 <- lmThresh(LM1, newobs = TRUE)
st1 <- stability(res1, pval = TRUE)

## Let's check that the prediction interval encompasses 95%:
dt.scaled <- function(x, df, mu, s) 1/s * dt((x - mu)/s, df)
integrate(dt.scaled, lower = st1$stats[1, "lower"], st1$stats[1, "upper"],
          df = 18, mu = st1$stats[1, "fitted"], s = st1$stats[1, "se"])
## => 0.95 with absolute error < 8.4e-09

## This is the interval between "end of significance region" and upper
## prediction boundary:
integrate(dt.scaled, lower = st1$stats[1, "eosr.2"], st1$stats[1, "upper"],
          df = 18, mu = st1$stats[1, "fitted"], s = st1$stats[1, "se"])
## => 0.09264124 with absolute error < 1e-15

## We can recheck this value by P(B) - P(A):
pt.scaled <- function(x, df, mu, s) pt((x - mu)/s, df)
pA <- pt.scaled(x = st1$stats[1, "eosr.2"], df =  18, mu = st1$stats[1, "fitted"],
                s = st1$stats[1, "se"])
0.975 - pA
##  => 0.09264124 as above
```

Index