# Package: reverseR (via r-universe)

September 5, 2024

**Type** Package

**LazyLoad** yes

**LazyData** yes

**Title** Linear Regression Stability to Significance Reversal

**Version** 0.2

**Date** 2022-05-07

**Author** Andrej-Nikolai Spiess <aspiess@soilytix.com> Michal Burdukiewicz <michalburdukiewicz@gmail.com> Stefan Roediger [aut] <stefan.roediger@b-tu.de>

**Maintainer** Andrej-Nikolai Spiess <aspiess@soilytix.com>

**Description** Tests linear regressions for significance reversal through leave-one(multiple)-out.

**License** GPL (>= 2)

**Depends** R (>= 3.5.0), shiny, markdown, knitr, boot, boot.pval

**NeedsCompilation** no

**Repository** https://anspiess.r-universe.dev

**RemoteUrl** https://github.com/anspiess/reverser

**RemoteRef** HEAD

**RemoteSha** 3ec07fda74d8d9900b4926fe9e79ef3b17dae5d9

## Contents

---

bootLM *Nonparametric bootstrap linear model*

---

## Description

Nonparametric bootstrap (sampling cases with replacement) method for parameter estimation and confidence interval of a linear model.

## Usage

```
bootLM(formula, data = NULL, R = 10000, alpha = 0.05)
```

## Arguments

| | |
|---|---|
| `formula` | a formula of type `y ~ x` for the linear model. |
| `data` | an optional data frame, list or environment containing the variables in the model. |
| `R` | number of bootstrap samples. |
| `alpha` | the $\alpha$-level to use as the threshold border. |

## Details

For all $(x_i, y_i)$ datapoints, linear models are created by sampling `R` times - with replacement - from $n \in \{1 \ldots N\}$ and building models $Y_n = X_n\beta + \varepsilon$. This is also known as the .632-bootstrap, because the samples will, on average, contain $1 - e^{-1} = 0.632$ unique elements. Parameter estimates are obtained from each sampling, from which the average $\overline{P_n}$ and standard error $\frac{\sigma}{\sqrt{n}}$ is calculated as well as a quantile based confidence interval. *p*-values are calculated through inversion of the confidence interval (`boot.pval`).

## Value

A dataframe containing the estimated coefficients, their standard error, lower an upper confidence values and *p*-values.

## Author(s)

Andrej-Nikolai Spiess

## References

An Introduction to the Bootstrap.
Efron B, Tibshirani R.
Chapman & Hall (1993).

The Bootstrap and Edgeworth Expansion.
Hall P.
Springer, New York (1992).

Modern Statistics with R.
Thulin M.
Eos Chasma Press, Uppsala (2021).

## Examples

```
## Example #1 with single influencers and insignificant model (p = 0.115).
## Jackknife estimates are robust w.r.t. outlier #18.
set.seed(123)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(20, 0, 1)
LM1 <- lm(b ~ a)
bootLM(LM1, R = 1000)
```

---

`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`), studentized residuals (`rstudent`) and Hadi's measure against the $\Delta p$-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. `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, ...)
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. |
| `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**: $|\Delta\beta1_i| > 2/\sqrt{n}$ (page 28)
**dffits**: $|\mathrm{dffits}_i| > 2\sqrt{2/n}$ (page 28)
**covratio**: $|\mathrm{covr}_i - 1| > 3k/n$ (page 23)
**Cook's D**: $D_i > Q_F(0.5, k, n-k)$ (Cook & Weisberg, 1982)
**leverage**: $h_{ii} > 2k/n$ (page 17)
**studentized residual**: $t_i > Q_t(0.975, n-k-1)$ (page 20)
**Hadi's measure**: $H_i^2 > \mathrm{Med}(H_i^2) + 2 \cdot \mathrm{MAD}(H_i^2)$ (Hadi 1992)

### 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 (2004).

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.

Residuals and Influence in Regression.
Cook RD & Weisberg S.
Chapman & Hall, 1st ed, New York, USA (1982).

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

## Examples

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

---

`jackLM` *Jackknife linear model according to Quenouille 1956*

---

## Description

Jackknife (Leave-One-Out) method for parameter estimation and confidence interval of a linear model.

## Usage

```
jackLM(formula, data = NULL, alpha = 0.05)
```

## Arguments

| | |
|---|---|
| `formula` | a formula of type `y ~ x` for the linear model. |
| `data` | an optional data frame, list or environment containing the variables in the model. |
| `alpha` | the $\alpha$-level to use as the threshold border. |

## Details

For all $(x_i, y_i)$ datapoints, a linear model is created by leaving out each entry successively, $Y_{-i} = X_{-i}\beta + \varepsilon$. Pseudovalues from obtained and original coefficients are then created, $P_{-i} = (N \cdot \beta) - ((N-1) * \beta_{-i})$, from which the average $\overline{P_{-i}}$ and standard error $\frac{\sigma}{\sqrt{N}}$ is calculated to obtain the classical confidence interval $\overline{X}_n \pm t_{\alpha,\nu} \frac{S_n}{\sqrt{n}}$.

## Value

A dataframe containg the estimated coefficients, their standard error, lower an upper confidence values and *p*-values.

## Author(s)

Andrej-Nikolai Spiess

## References

Notes on bias in estimation.
Quenouille MH.
*Biometrika*, **43**, 1956, 353-36l.

## Examples

```
## Example #1 with single influencers and insignificant model (p = 0.115).
## Jackknife estimates are robust w.r.t. outlier #18.
set.seed(123)
a <- 1:20
b <- 5 + 0.08 * a + rnorm(20, 0, 1)
LM1 <- lm(b ~ a)
jackLM(LM1)
```

---

# 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 $\beta_1$ and intercept $\beta_0$,
**2)** exact *p*-value and intercept $\beta_0$, or
**3)** exact $R^2$ and intercept $\beta_0$.

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 $\beta_0$. |
| `slope` | the desired slope $\beta_1$. |
| `error` | if a single value, the standard deviation $\sigma$ 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 $\alpha$-level.

## Usage

```
lmInfl(model, alpha = 0.05, verbose = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `model` | the linear model of class `lm`. |
| `alpha` | the $\alpha$-level to use as the threshold border. |
| `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, $\Delta p$-values, slopes and standard errors attached.

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 $\beta_1$ is the difference of *t*-statistics

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

is inextricably linked to the changes in $p$-value $\Delta p$, calculated from

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

where $P_t$ is the Student's *t* cumulative distribution function with $\nu$ degrees of freedom, and where significance reversal is attained when $\alpha \in [p_{\beta 1}, p_{\beta 1(i)}]$. Interestingly, 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 $\Delta p$ in current literature on influence measures.

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

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

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

## 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 $\Delta$'s, LOO-standard errors and $R^2$s. Influence measures that exceed their specific threshold - see inflPlot - will be marked with asterisks. |
| raw | same as infl, but with pure numeric data. |
| sel | a vector with the influencers' indices. |
| alpha | the selected $\alpha$-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)
```

---

`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 $\alpha$-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 $y_i \pm \text{factor} \cdot \text{range}(y_{1...n})$ with steps cuts. For each cut, the *p*-value is calculated for the model when $y_i$ is shifted to that value (newobs = FALSE) 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 $\beta_1$ and the other inflates the standard error $\text{s.e.}(\beta_1)$ in a way that $P_t(\frac{\beta_1}{\text{s.e.}(\beta_1)}, n-2) > \alpha$. If the original model was insignificant, also two boundaries exists that either increase $\beta_1$ or reduce $\text{s.e.}(\beta_1)$. 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 $\alpha$-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 $\Delta$ 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
```

---

PNAS2015 *Small dataset from a 2015 PNAS paper*

---

## Description

The data was acquired by digitization of a graph from a 2015 PNAS paper. Contains three datapoints that exert significance reversal.

## Usage

```
data(PNAS2015)
```

## Author(s)

Andrej-Nikolai Spiess

## Examples

```
## See examples in 'lmInfl' and 'lmThresh'.
LM <- lm(y ~ x, data = PNAS2015)
lmInfl(LM)
```

`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 $\beta_1$ to be supplied to `lmExact`. |
| `intercept` | the intercept $\beta_0$ to be supplied to `lmExact`. |
| `error` | the $\varepsilon$ 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' and 'lmThresh'*

---

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| `x` | a result of either `lmInfl` 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 `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:

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

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

$$\hat{y}_i \pm Q_t(\alpha/2, n-2) \cdot \mathrm{se_i}$$

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

$$P_{tss}(y, n-2) = \frac{1}{\mathrm{se_i}} \cdot P_t\left(\frac{y - \hat{y}_i}{\mathrm{se_i}}, n-2\right)$$

, 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:

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

## 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