

# Package ‘api2lm’

July 22, 2025

**Type** Package

**Title** Functions and Data Sets for the Book “A Progressive Introduction to Linear Models”

**Version** 0.2

**Author** Joshua P. French

**Maintainer** Joshua P. French <joshua.french@ucdenver.edu>

**Description** Simplifies aspects of linear regression analysis, particularly simultaneous inference. Additionally, supports “A Progressive Introduction to Linear Models” by Joshua French (<<https://jfrench.github.io/LinearRegression/>>).

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.3

**Suggests** testthat (>= 3.0.0), ggplot2, knitr, rmarkdown, vdiff

**VignetteBuilder** knitr

**Depends** R (>= 3.5.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-06-08 22:32:54 UTC

## Contents

arg_check_leverage_test . . . . .	2
autoplot.confint_adjust . . . . .	3
coef_compare . . . . .	4
coef_matrix . . . . .	5
confint_adjust . . . . .	5
cooks_plot . . . . .	7
cooks_stats . . . . .	8

cooks_test . . . . .	9
crime2009 . . . . .	10
dfbetas_plot . . . . .	11
dfbeta_plot . . . . .	12
dffits_plot . . . . .	13
dffits_stats . . . . .	14
dffits_test . . . . .	15
dwaine . . . . .	16
get_residuals . . . . .	17
home_sales . . . . .	18
index_plot_lm . . . . .	19
index_plot_raw . . . . .	21
influence_plot . . . . .	22
influence_stats . . . . .	23
leverage_plot . . . . .	25
leverage_stats . . . . .	26
leverage_test . . . . .	27
outlier_plot . . . . .	28
outlier_stats . . . . .	29
outlier_test . . . . .	30
plot.confint_adjust . . . . .	31
predict_adjust . . . . .	32
print.coef_matrix . . . . .	34
print.confint_adjust . . . . .	35
print.predict_adjust . . . . .	36
residual_plot . . . . .	37
residual_plot_lm . . . . .	37
rloo . . . . .	39
rloo_lm . . . . .	40
rplot_raw . . . . .	41
sl_plot . . . . .	42
sl_plot_lm . . . . .	43
toluca . . . . .	44
<b>Index</b>	<b>46</b>

---

arg\_check\_leverage\_test

*Check arguments of leverage\_test*

---

### Description

Check arguments of leverage\_test

### Usage

arg\_check\_leverage\_test(model, n, ttype, threshold)

**Arguments**

model	A fitted model object from the <code>lm</code> function.
n	The number of leverage points to return. The default is all leverage points.
ttype	Threshold type. The default is "half". The other options are "2mean" and "custom". See Details.
threshold	A number between 0 and 1. Any observation with a leverage value above this number is declared a leverage point. This is automatically determined unless <code>ttype = "custom"</code> .

**Value**

A vector of statistics

---

`autoplot.confint_adjust`

*Plot confint\_adjust object*

---

**Description**

Plot a `confint_adjust` object produced by the `confint_adjust` function. The plotting function internally calls the `autoplot` function. Note: the `ggplot2` package must be loaded (i.e., `library(ggplot2)` or `ggplot2::autoplot` must be specifically called for this function to work. See Examples.

**Usage**

```
autoplot.confint_adjust(object, parm, ...)
```

**Arguments**

object	An <code>confint_adjust</code> object produced by the <code>confint_adjust</code> function.
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
...	Not used

**Value**

None.

**Author(s)**

Joshua French

**Examples**

```

fit <- lm(100/mpg ~ disp + hp + wt + am, data = mtcars)
# standard intervals
cia <- confint_adjust(fit)
# if ggplot2 package is available
if (require(ggplot2)) {
  autoplot(cia)
# select subset of plots
  autoplot(cia, parm = c("hp", "disp"))
}

```

coef\_compare

*Compare coefficients of 2 models***Description**

Compare the coefficients to two fitted models. The models must have the same coefficients.

**Usage**

```
coef_compare(model1, model2, digits = 3, verbose = TRUE)
```

**Arguments**

model1	A fitted model object from the <code>lm</code> function.
model2	A fitted model object from the <code>lm</code> function.
digits	A positive integer indicating how many significant digits are to be used for numeric and complex numbers. This argument is passed to <code>format</code> .
verbose	A logical value indicating whether the matrix should be printed. The default is TRUE.

**Value**

A matrix.

**Examples**

```

# fit model
lmod1 <- lm(murder ~ hs_grad + urban + poverty + single,
           data = crime2009)
#fit without DC
lmod2 <- lm(murder ~ hs_grad + urban + poverty + single,
           data = crime2009[-9, ])
#compare coefficients of models
coef_compare(lmod1, lmod2)

```

---

coef_matrix	<i>Return coefficient matrix</i>
-------------	----------------------------------

---

**Description**

coef\_matrix returns the coefficients element of the summary function, which is a matrix with columns for the estimated coefficients, their standard error, t-statistic and corresponding (two-sided) p-values.

**Usage**

```
coef_matrix(object)
```

**Arguments**

object            an object of class "lm", usually, a result of a call to `lm`.

**Value**

A  $p \times 4$  matrix with columns for the estimated coefficient, its standard error, t-statistic and corresponding (two-sided) p-value. Aliased coefficients are omitted. The additional class `coef_matrix` is added for custom printing.

**Author(s)**

Joshua P. French

**Examples**

```
## a fitted model
fit <- lm(100/mpg ~ disp + hp + wt + am, data = mtcars)
coef_matrix(fit)
print(coef_matrix(fit), digits = 3)
```

---

confint_adjust	<i>Adjust confidence intervals for multiple comparisons</i>
----------------	---

---

**Description**

A function to produce adjusted confidence intervals with a family-wise confidence level of at least level for `lm` objects (not applicable if no adjustment is used). Internally, the function is a slight revision of the code used in the `confint.lm` function.

**Usage**

```
confint_adjust(object, parm, level = 0.95, method = "none")
```

**Arguments**

object	a fitted model object.
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
level	the confidence level required.
method	A character string indicating the type of adjustment to make. The default choice is "none". The other option is "bonferroni".

**Details**

Let  $\alpha = 1 - \text{level}$ . Let  $p$  be the number of estimated coefficients in the fitted model. All intervals are computed using the formula  $\text{estimate} \pm m * \text{ese}$ , where  $m$  is a multiplier and  $\text{ese}$  is the estimated standard error of the estimate.

method = "none" (no correction) produces the standard t-based confidence intervals with multiplier  $qt(1 - \alpha/2, df = \text{object}\$df.\text{residual})$ .

method = "bonferroni" produces Bonferroni-adjusted intervals that use the multiplier  $m = qt(1 - \alpha/(2 * k), df = \text{object}\$df.\text{residual})$ , where  $k$  is the number of intervals being produced.

method = "wh" produces Working-Hotelling-adjusted intervals that are valid for all linear combinations of the regression coefficients, which uses the multiplier  $m = \text{sqrt}(p * \text{qf}(\text{level}, df1 = p, df2 = \text{object}\$df.\text{residual}))$ .

**Value**

A `confint_adjust` object, which is simply a data.frame with columns `term`, `lwr` (the lower confidence limit), and `upr` (the upper confidence limit).

**References**

Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilita. Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze, 8, 3-62.

Working, H., & Hotelling, H. (1929). Applications of the theory of error to the interpretation of trends. Journal of the American Statistical Association, 24(165A), 73-85. doi:10.1080/01621459.1929.10506274

Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2004). Applied Linear Statistical Models, 5th edition. New York: McGraw-Hill/Irwin. (p. 230)

**See Also**

[confint.lm](#)

**Examples**

```
## an extension of the documentation for confint.lm
fit <- lm(100/mpg ~ disp + hp + wt + am, data = mtcars)
# standard intervals
confint_adjust(fit)
# bonferroni-adjusted intervals
```

```
(cib <- confint_adjust(fit, method = "b"))
# plot results
plot(cib)
plot(cib, parm = c("hp", "disp"))
if (require(ggplot2)) {
  autoplot(cib)
  autoplot(cib, parm = c("hp", "disp"))
}
#' working-hotelling-adjusted intervals
(ciw_h <- confint_adjust(fit, method = "wh"))
```

---

cooks\_plot

*Index plot of Cook's distances for lm object*


---

### Description

cooks\_plot plots the Cook's distances from the `cooks.distance` function of a fitted `lm` object.

### Usage

```
cooks_plot(
  model,
  id_n = 3,
  add_reference = TRUE,
  ...,
  text_arglist = list(),
  abline_arglist = list(),
  extendrange_f = 0.08
)
```

### Arguments

<code>model</code>	A fitted model object from the <code>lm</code> function.
<code>id_n</code>	The number of points to identify with labels. The default is 3.
<code>add_reference</code>	A logical value indicating whether a reference line should be added. See details.
<code>...</code>	Additional arguments passed to the <code>plot</code> function.
<code>text_arglist</code>	Additional arguments passed to the <code>text</code> function, which is used to display the points that are identified.
<code>abline_arglist</code>	A named list specifying additional arguments passed to the <code>abline</code> function for the horizontal reference line added to the plot.
<code>extendrange_f</code>	Positive number(s) specifying the fraction by which the range of the residuals should be extended using the <code>extendrange</code> function. If longer than one, <code>f[1]</code> is used on the left and <code>f[2]</code> on the right.

**Details**

By default, a reference line is plotted at the  $0.5$  quantile of a  $F_{p,n-p}$  distribution where  $p = \text{length}(\text{stats}::\text{coef}(\text{model}))$  and  $n - p = \text{stats}::\text{df.residual}(\text{model})$ .

The vertical position of the reference line can be customized by setting the `h` argument of `abline_arglist`.

**Author(s)**

Joshua French

**See Also**

[plot](#), [text](#), [abline](#), [cooks.distance](#)

**Examples**

```
lmod <- lm(price ~ sqft_living, data = home_sales)
cooks_plot(lmod, id_n = 1)
```

---

cooks\_stats

*Cook's statistics*

---

**Description**

`cooks_stats` returns the ordered Cook's statistics (distances) decreasing in value of `model` to identify the most influential observations.

**Usage**

```
cooks_stats(model, n = 6L)
```

**Arguments**

<code>model</code>	A fitted model object from the <code>lm</code> function.
<code>n</code>	an integer vector of length up to <code>dim(x)</code> (or 1, for non-dimensioned objects). Values specify the indices to be selected in the corresponding dimension (or along the length) of the object. A positive value of <code>n[i]</code> includes the first/last <code>n[i]</code> indices in that dimension, while a negative value excludes the last/first <code>abs(n[i])</code> indices, including all remaining indices. NA or non-specified values (when <code>length(n) &lt; length(dim(x))</code> ) select all indices in that dimension. Must contain at least one non-missing value.

**Value**

A vector of statistics

**See Also**

[cooks.distance](#).



**Examples**

```
lmod <- lm(price ~ sqft_living, data = home_sales)
cooks_stats(lmod, n = 5)
```

---

cooks_test	<i>Identify influential observations</i>
------------	--

---

**Description**

cooks\_test returns the observations identified as influential based on the cooks statistics being larger than a threshold.

The threshold for this test is the 0.5 quantile of a  $F_{p, n-p}$  distribution where  $p = \text{length}(\text{stats}::\text{coef}(\text{model}))$  and  $n - p = \text{stats}::\text{df.residual}(\text{model})$ .

**Usage**

```
cooks_test(model, n = stats::nobs(model))
```

**Arguments**

model            A fitted model object from the `lm` function.

n                The number of outliers to return. The default is all influential observations.

**Value**

A vector of influential observations.

**See Also**

[cooks.distance](#)

**Examples**

```
lmod <- lm(price ~ sqft_living, data = home_sales)
cooks_test(lmod)
```

---

`crime2009`*2009 Crime Data*

---

**Description**

Data related to crime for the 50 U.S. states plus the District of Columbia. The data are taken from the `crime_data` data set available in the `statsmodels` package in Python. As stated in its documentation, "All data is for 2009 and was obtained from the American Statistical Abstracts except as indicated ...." The original documentation is available at <https://www.statsmodels.org/dev/datasets/generated/statecrime.html>.

The violent variable includes murder, forcible rape, robbery, and aggravated assault. Numbers for Illinois and Minnesota do not include forcible rapes. Footnote included with the American Statistical Abstract table reads: "The data collection methodology for the offense of forcible rape used by the Illinois and the Minnesota state Uniform Crime Reporting (UCR) Programs (with the exception of Rockford, Illinois, and Minneapolis and St. Paul, Minnesota) does not comply with national UCR guidelines. Consequently, their state figures for forcible rape and violent crime (of which forcible rape is a part) are not published in this table."

The single variable is calculated from 2009 1-year American Community Survey obtained obtained from Census. Variable is Male householder, no wife present, family household combined with Female householder, no husband present, family household, divided by the total number of Family households.

**Usage**

```
data(crime2009)
```

**Format**

A data frame with 51 observations and 7 variables:

`violent` Rate of violent crimes per 100,000 persons in the population.

`murder` Rate of murders per 100,000 persons in the population.

`hs_grad` Percentage of the population having graduated from high school or higher.

`poverty` Percentage of individuals with income below the poverty line.

`white` Percentage of the population that is only considered "white" for race based on the 2009 American Community Survey.

`single` Percentage of families made up of single individuals.

`urban` Percentage of the population living in Urbanized Areas as of 2010 Census, where Urbanized Areas are areas of 50,000 or more people.

**Source**

A public domain data set available in the `statsmodels` python package. All data is for 2009 and was obtained from the American Statistical Abstracts except as indicated. <https://www.statsmodels.org/dev/datasets/generated/statecrime.html>.

---

dfbetas_plot	<i>dfbetas index plots</i>
--------------	----------------------------

---

## Description

dfbetas\_plot creates index plot of the dfbetas statistics for each regressor.

## Usage

```
dfbetas_plot(  
  model,  
  id_n = 3,  
  regressors = ~.,  
  add_reference = TRUE,  
  ...,  
  text_arglist = list(),  
  abline_arglist = list(),  
  extendrange_f = 0.08  
)
```

## Arguments

model	A fitted model object from the <code>lm</code> function.
id_n	The number of points to identify with labels. The default is 3.
regressors	A formula describing the regressors for which to plot the dfbetas statistics. The default is all available regressors.
add_reference	A logical value indicating whether a reference line should be added. The default is TRUE.
...	Additional arguments passed to the <code>plot</code> function.
text_arglist	Additional arguments passed to the <code>text</code> function, which is used to display the points that are identified.
abline_arglist	A named list specifying additional arguments passed to the <code>abline</code> function for the horizontal reference line added to the plot.
extendrange_f	Positive number(s) specifying the fraction by which the range of the residuals should be extended using the <code>extendrange</code> function. If longer than one, <code>f[1]</code> is used on the left and <code>f[2]</code> on the right.

## Details

A horizontal reference line is added at -1 and +1 by default if `add_reference` is TRUE.

## Author(s)

Joshua French

**See Also**

[plot](#), [text](#), [abline](#) [dfbetas](#).

**Examples**

```
lmod <- lm(murder ~ hs_grad + urban + poverty + single,
          data = crime2009)
dfbetas_plot(lmod)
dfbetas_plot(lmod, regressors = ~ hs_grad, id_n = 4)
```

---

dfbeta\_plot

*dfbeta index plots*


---

**Description**

dfbeta\_plot creates an index plot of the dfbeta statistics for each regressor.

**Usage**

```
dfbeta_plot(
  model,
  id_n = 3,
  regressors = ~.,
  add_reference = TRUE,
  ...,
  text_arglist = list(),
  abline_arglist = list(),
  extendrange_f = 0.08
)
```

**Arguments**

model	A fitted model object from the <a href="#">lm</a> function.
id_n	The number of points to identify with labels. The default is 3.
regressors	A formula describing the regressors for which to plot the dfbetas statistics. The default is all available regressors.
add_reference	A logical value indicating whether a reference line should be added. The default is TRUE.
...	Additional arguments passed to the <a href="#">plot</a> function.
text_arglist	Additional arguments passed to the <a href="#">text</a> function, which is used to display the points that are identified.
abline_arglist	A named list specifying additional arguments passed to the <a href="#">abline</a> function for the horizontal reference line added to the plot.
extendrange_f	Positive number(s) specifying the fraction by which the range of the residuals should be extended using the <a href="#">extendrange</a> function. If longer than one, f[1] is used on the left and f[2] on the right.

## Details

A horizontal reference line is added at +/- the estimated standard error of each coefficient by default if `add_reference` is `TRUE`.

## Author(s)

Joshua French

## See Also

[plot](#), [text](#), [abline](#) [dfbeta](#).

## Examples

```
lmod <- lm(murder ~ hs_grad + urban + poverty + single,
           data = crime2009)
dfbeta_plot(lmod)
dfbeta_plot(lmod, regressors = ~ hs_grad + poverty,
            id_n = 1)
```

---

dffits\_plot

*Index plot of DFFITS values for lm object*

---

## Description

`dffits_plot` plots the DFFITS values from the `dffits` function of a fitted `lm` object.

## Usage

```
dffits_plot(
  model,
  id_n = 3,
  add_reference = TRUE,
  ...,
  text_arglist = list(),
  abline_arglist = list(),
  extendrange_f = 0.08
)
```

## Arguments

<code>model</code>	A fitted model object from the <code>lm</code> function.
<code>id_n</code>	The number of points to identify with labels. The default is 3.
<code>add_reference</code>	A logical value indicating whether a reference line should be added. The default is <code>TRUE</code> . See <a href="#">Details</a> .
<code>...</code>	Additional arguments passed to the <a href="#">plot</a> function.

- `text_arglist` Additional arguments passed to the `text` function, which is used to display the points that are identified.
- `abline_arglist` A named list specifying additional arguments passed to the `abline` function for the horizontal reference line added to the plot.
- `extendrange_f` Positive number(s) specifying the fraction by which the range of the residuals should be extended using the `extendrange` function. If longer than one, `f[1]` is used on the left and `f[2]` on the right.

### Details

By default, a reference line is plotted at  $\pm 2\sqrt{p/n}$ , where  $p = \text{length}(\text{stats}::\text{coef}(\text{model}))$  and  $n = \text{stats}::\text{nobs}(\text{model})$ . This can be customized by setting the `h` argument of `abline_arglist`.

### Author(s)

Joshua French

### See Also

[plot](#), [text](#), [abline](#), [dffits](#)

### Examples

```
lmod <- lm(price ~ sqft_living, data = home_sales)
dffits_plot(lmod, id_n = 6)
# customized plot
dffits_plot(lmod, id_n = 1,
            text_arglist = list(col = "blue", cex = 2),
            abline_arglist = list(col = "red", lwd = 2))
```

---

`dffits_stats`

*DFFITs statistics*

---

### Description

`dffits_stats` returns the ordered DFFITS values (decreasing in magnitude) of model to identify the observations with the highest DFFITS values.

### Usage

```
dffits_stats(model, n = 6L)
```

**Arguments**

model	A fitted model object from the <code>lm</code> function.
n	an integer vector of length up to <code>dim(x)</code> (or 1, for non-dimensional objects). Values specify the indices to be selected in the corresponding dimension (or along the length) of the object. A positive value of <code>n[i]</code> includes the first/last <code>n[i]</code> indices in that dimension, while a negative value excludes the last/first <code>abs(n[i])</code> , including all remaining indices. NA or non-specified values (when <code>length(n) &lt; length(dim(x))</code> ) select all indices in that dimension. Must contain at least one non-missing value.

**Value**

A vector of statistics

**See Also**

[dffits](#)

**Examples**

```
lmod <- lm(price ~ sqft_living, data = home_sales)
dffits_stats(lmod, n = 5)
```

---

dffits_test	<i>Identify influential observations</i>
-------------	--

---

**Description**

`dffits_test` returns the observations identified as influential based on the absolute value of the DFFITS statistics being larger than a threshold.

The threshold used is  $2\sqrt{p/n}$ , where  $p = \text{length}(\text{stats}::\text{coef}(\text{model}))$  and  $n = \text{stats}::\text{nobs}(\text{model})$ .

**Usage**

```
dffits_test(model, n = stats::nobs(model))
```

**Arguments**

model	A fitted model object from the <code>lm</code> function.
n	The number of outliers to return. The default is all influential observations.

**Value**

A vector of influential observations.

**See Also**[dffits](#)**Examples**

```
lmod <- lm(price ~ sqft_living, data = home_sales)
dffits_test(lmod)
```

---

dwaine

*Dwaine Studios data*

---

**Description**

Data from the Dwaine Studios data in Applied Linear Statistical Models, 5th edition, p. 237. From the book:

Dwaine Studios, Inc., operates portrait studios in 21 cities of medium size. These studios specialize in portraits of children. The company is considering an expansion into other cities of medium size and wishes to investigate where sales (`sales`) in a community can be predicted from the number of persons aged 16 or younger in the community (`targetpop`) and the per capita disposable personal income in the community (`dpi`). Data on these variables for the most recent year for the 21 cities in which Dwaine Studios is now operating are included in the data set.

**Usage**

```
data(dwaine)
```

**Format**

A data frame with 21 observations and 3 variables:

`targetpop` The number of persons aged 16 or younger in thousands of persons.

`dpi` Per capita disposable personal income in thousands of dollars.

`sales` Sales in thousands of dollars

**Author(s)**

Joshua P. French

**References**

Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2004). Applied Linear Statistical Models, 5th edition. New York: McGraw-Hill/Irwin.



---

get_residuals	<i>Extract residuals from a model</i>
---------------	---------------------------------------

---

### Description

Extracts different types of residuals from a fitted model. The types of residuals are discussed in Details.

### Usage

```
get_residuals(
  x,
  rtype = c("ordinary", "standardized", "studentized", "jackknife", "loo", "deleted",
            "internally studentized", "externally studentized")
)
```

### Arguments

<code>x</code>	An lm object
<code>rtype</code>	The desired residual type. The options are "ordinary", "standardized", "studentized", "jackknife", "loo", "deleted", "internally studentized", and "externally studentized".

### Details

For observations  $1, 2, \dots, n$ , let:

1.  $Y_i$  denote the response value for the  $i$ th observation.
2.  $\hat{Y}_i$  denote the fitted value for the  $i$ th observation.
3.  $h_i$  denote the leverage value for the  $i$ th observation.

We assume that  $\text{sd}(Y_i) = \sigma$  for  $i \in \{1, 2, \dots, n\}$  and that  $\hat{\sigma}$  is the estimate produced by `sigma(x)`, where `x` is the fitted model object.

The ordinary residual for the  $i$ th observation is computed as

$$\hat{\epsilon}_i = Y_i - \hat{Y}_i.$$

The variance of the  $i$ th ordinary residual under standard assumptions is  $\sigma^2(1 - h_i)$ .

The standardized residual for the  $i$ th observation is computed as

$$r_i = \frac{\hat{\epsilon}_i}{\hat{\sigma}\sqrt{1 - h_i}}.$$

The standardized residual is also known as the internally studentized residual.

Let  $\hat{Y}_{i(i)}$  denote the predicted value of  $Y_i$  for the model fit with all  $n$  observations except observation  $i$ . The leave-one-out (LOO) residual for observation  $i$  is computed as

$$l_i = Y_i - \hat{Y}_{i(i)} = \frac{\hat{\epsilon}_i}{1 - h_i}.$$

The LOO residual is also known as the deleted or jackknife residual.

The studentized residual for the  $i$ th observation is computed as

$$t_i = \frac{l_i}{\hat{\sigma}_{(i)}\sqrt{1 - h_i}},$$

where  $\hat{\sigma}_{(i)}$  is the leave-one-out estimate of  $\sigma$ .

The studentized residual is also known as the externally studentized residual.

### Value

A vector of residuals.

### Examples

```
lmod <- lm(Girth ~ Height, data = trees)
# ordinary residuals
rord <- get_residuals(lmod)
all.equal(rord, residuals(lmod))
# standardized residuals
rstand <- get_residuals(lmod, "standardized")
all.equal(rstand, rstandard(lmod))
# studentized residuals
rstud <- get_residuals(lmod, "studentized")
all.equal(rstud, rstudent(lmod))
# loo residuals
rl <- get_residuals(lmod, "loo")
all.equal(rl, rloo(lmod))
```

---

home\_sales

*Home sale prices in King County, WA*

---

### Description

The home\_sales data set is a data frame consisting of 216 rows and 8 columns. The data are a subset of home sales in King County, WA made between 2014-05-02 to 2015-05-27. The variables in the data set are:

- price: sale price (in log10 US dollars).
- bedrooms: number of bedrooms.
- bathrooms: number of bathrooms.
- sqft\_living: size of living space in square feet.
- sqft\_lot: lot size in square feet.

- floors: number of floors in home.
- waterfront: a factor variable with levels no and 'yes' that indicate whether the home has a waterfront view.
- condition: a factor variable indicating the condition of the house with levels ranging from poor to very good.

### Value

A data.frame.

### Source

The Center for Spatial Data Science, University of Chicago. <https://geodacenter.github.io/data-and-lab//KingCounty-HouseSales2015/>

These data were created by selectively choosing a subset of observations from the home\_prices data set in the `**KingCountyHomes**` package.

### Examples

```
data(home_sales)
summary(home_sales)
```

---

index_plot_lm	<i>Index plot of statistics from of an lm object</i>
---------------	--

---

### Description

index\_plot\_lm creates an index plot of statistics from an lm object.

### Usage

```
index_plot_lm(  
  model,  
  stat,  
  id_n = 3,  
  add_reference = FALSE,  
  ...,  
  text_arglist = list(),  
  abline_arglist = list(),  
  extendrange_f = 0.08  
)
```

**Arguments**

model	A fitted model object from the <code>lm</code> function.
stat	A function that can be applied to an <code>lm</code> object and returns a vector of observations for each observation used to fit the model.
id_n	The number of points to identify with labels. The default is 3.
add_reference	A logical value indicating whether a reference line should be added. The default is TRUE.
...	Additional arguments passed to the <code>plot</code> function.
text_arglist	Additional arguments passed to the <code>text</code> function, which is used to display the points that are identified.
abline_arglist	A named list specifying additional arguments passed to the <code>abline</code> function for the horizontal reference line added to the plot.
extendrange_f	Positive number(s) specifying the fraction by which the range of the residuals should be extended using the <code>extendrange</code> function. If longer than one, <code>f[1]</code> is used on the left and <code>f[2]</code> on the right.

**Author(s)**

Joshua French

**See Also**

[plot](#), [text](#), [lm](#), [rstudent](#), [hatvalues](#), [cooks.distance](#)

**Examples**

```
lmod <- lm(Petal.Length ~ Sepal.Length + Species,
          data = iris)
# outlier plot
# number of observations
n <- stats::nobs(lmod)
# loo residual degrees of freedom
rdf <- stats::df.residual(lmod) - 1

h <- c(-1, 1) * stats::qt(0.05/(2 * n), df = rdf)
index_plot_lm(lmod, stat = stats::rstudent,
              abline_arglist = list(h = h))

# leverage plot
index_plot_lm(lmod, stat = stats::hatvalues, id_n = 1)
# Cook's distance
index_plot_lm(lmod, stat = stats::cooks.distance,
              id_n = 3)
```

---

index_plot_raw	<i>Index plot helper function</i>
----------------	-----------------------------------

---

## Description

Index plot helper function

## Usage

```
index_plot_raw(  
  x,  
  y,  
  idd,  
  labels,  
  add_reference,  
  arglist,  
  text_arglist,  
  abline_arglist,  
  extendrange_f  
)
```

## Arguments

x	x-values to plot
y	y-values to plot
idd	Identified observations
labels	The labels to use for the identified points.
add_reference	Logical value
arglist	Named list for plot
text_arglist	Named list for text
abline_arglist	Named list for abline
extendrange_f	Positive number(s) specifying the fraction by which the range of the residuals should be extended using the <a href="#">extendrange</a> function. If longer than one, f[1] is used on the left and f[2] on the right.

---

influence_plot	<i>Influence plots</i>
----------------	------------------------

---

### Description

influence\_plot creates an influence plot for a fitted lm object. The y-axis is either the studentized (the default) or standardized residuals versus the leverage values for each observation. The size of the point associated with each observation is proportional to the value of the Cook's distance (the default) or the DFFITS statistic for the observation.

Details about the different types of residuals are discussed in the [get\\_residuals](#) function.

### Usage

```
influence_plot(
  model,
  rtype = c("studentized", "standardized"),
  criterion = c("cooks", "dffits"),
  id_n = 3,
  add_reference = TRUE,
  alpha = 0.05,
  size = c(1, 4.8),
  ...,
  text_arglist = list(),
  abline_arglist = list(),
  extendrange_f = 0.08
)
```

### Arguments

model	A fitted model object from the <a href="#">lm</a> function.
rtype	The residual type to plot on the y-axis. The default is "studentized". The other option is "standardized".
criterion	The criterion that decides the size of the points. The default is "cooks". The other option is "dffits".
id_n	The number of points to identify with labels with respect to largest absolute criterion. The default is 3 labels.
add_reference	A logical value indicating whether a reference line should be added. The default is TRUE.
alpha	The default quantile used for the horizontal reference lines. The default is 0.05. See Details.
size	A numeric vector of length 2 that provides guidelines for the size of the points.
...	Additional arguments passed to the <a href="#">plot</a> function.
text_arglist	Additional arguments passed to the <a href="#">text</a> function, which is used to display the points that are identified.

- `abline_arglist` A named list specifying additional arguments passed to the `abline` function for the horizontal reference line added to the plot.
- `extendrange_f` Positive number(s) specifying the fraction by which the range of the residuals should be extended using the `extendrange` function. If longer than one, `f[1]` is used on the left and `f[2]` on the right.

### Details

The range of the criterion statistic is mapped to  $\text{cex\_pt} = \text{size}[2]^2 - \text{size}[1]^2$  and then the size of the points is `sqrt(cex_pt)`.

If `add_reference` is TRUE, then horizontal reference lines are added at the  $\alpha/2$  and  $1-\alpha/2$  quantiles of a t distribution with degrees of freedom given by `stats::df.residual(model)`.

If `add_reference` is TRUE, then vertical reference lines are added at  $2p/n$  and 0.5 where  $p = \text{length}(\text{stats}::\text{coef}(\text{model}))$  and  $n = \text{stats}::\text{nobs}(\text{model})$ .

The vertical position of the reference lines can be customized by setting the `h` argument of `abline_arglist`. The horizontal position of the reference lines can be customized by setting the `v` argument of `abline_arglist`.

### Author(s)

Joshua French

### See Also

[plot](#), [text](#), [abline](#), [rstandard](#), [rstudent](#), [hatvalues](#) [cooks.distance](#), [dffits](#)

### Examples

```
lmod <- lm(murder ~ hs_grad + urban + poverty + single,
           data = crime2009)
# studentized residuals vs leverage
influence_plot(lmod, id_n = 3)
# standardized residuals vs leverage
influence_plot(lmod, rtype = "stan")
# similar plot from plot.lm
plot(lmod, which = 5)
```

---

influence\_stats

*Influence statistics*

---

### Description

`influence_stats` returns a data frame with influence-related statistics ordered from the largest to smallest magnitude of the criterion.

**Usage**

```
influence_stats(  
  model,  
  n = 6L,  
  rtype = c("studentized", "standardized"),  
  criterion = c("cooks", "dffits")  
)
```

**Arguments**

model	A fitted model object from the <code>lm</code> function.
n	an integer vector of length up to <code>dim(x)</code> (or 1, for non-dimensioned objects). Values specify the indices to be selected in the corresponding dimension (or along the length) of the object. A positive value of <code>n[i]</code> includes the first/last <code>n[i]</code> indices in that dimension, while a negative value excludes the last/first <code>abs(n[i])</code> , including all remaining indices. NA or non-specified values (when <code>length(n) &lt; length(dim(x))</code> ) select all indices in that dimension. Must contain at least one non-missing value.
rtype	The residual type to plot on the y-axis. The default is "studentized". The other option is "standardized".
criterion	The criterion that decides the size of the points. The default is "cooks". The other option is "dffits".

**Value**

A data frame of influence-related statistics.

**Author(s)**

Joshua French

**See Also**

[rstandard](#), [rstudent](#), [hatvalues](#) [cooks.distance](#), [dffits](#)

**Examples**

```
lmod <- lm(murder ~ hs_grad + urban + poverty + single,  
          data = crime2009)  
influence_stats(lmod, n = 3)  
influence_stats(lmod, rtype = "stan", crit = "df")
```



---

leverage\_plot

*Index plot of leverage values for lm object*


---

### Description

leverage\_plot plots the leverage (hat) values from the [hatvalues](#) function of a fitted lm object.

### Usage

```
leverage_plot(
  model,
  id_n = 3,
  add_reference = TRUE,
  ttype = "half",
  threshold = NULL,
  ...,
  text_arglist = list(),
  abline_arglist = list(),
  extendrange_f = 0.08
)
```

### Arguments

model	A fitted model object from the <a href="#">lm</a> function.
id_n	The number of points to identify with labels. The default is 3.
add_reference	A logical value indicating whether a reference line should be added. The default is TRUE.
ttype	Threshold type. The default is "half". The other options are "2mean" and "custom". See Details.
threshold	A number between 0 and 1. Any observation with a leverage value above this number is declared a leverage point. This is automatically determined unless ttype = "custom".
...	Additional arguments passed to the <a href="#">plot</a> function.
text_arglist	Additional arguments passed to the <a href="#">text</a> function, which is used to display the points that are identified.
abline_arglist	A named list specifying additional arguments passed to the <a href="#">abline</a> function for the horizontal reference line added to the plot.
extendrange_f	Positive number(s) specifying the fraction by which the range of the residuals should be extended using the <a href="#">extendrange</a> function. If longer than one, f[1] is used on the left and f[2] on the right.

**Details**

If `ttype = "half"`, the threshold is 0.5.

If `ttype = "2mean"`, the threshold is  $2p/n$ , where  $p = \text{length}(\text{stats}::\text{coef}(\text{model}))$  and  $n = \text{stats}::\text{nobs}(\text{model})$ , which is double the mean leverage value.

If `ttype = "custom"` then the user must manually specify threshold.

**Author(s)**

Joshua French

**See Also**

[plot](#), [text](#), [abline](#), [rstudent](#)

**Examples**

```
lmod <- lm(price ~ sqft_living, data = home_sales)
# reference line not visible on plot because all
# leverage values are less than 0.5
leverage_plot(lmod, id_n = 2)
# different reference line
leverage_plot(lmod, id_n = 6, ttype = "2mean")
# custom reference line
leverage_plot(lmod, id_n = 2, ttype = "custom",
              threshold = 0.15)
```

---

leverage_stats	<i>Leverage statistics</i>
----------------	----------------------------

---

**Description**

`leverage_stats` returns the ordered leverage values (decreasing) of `model` to identify the highest leverage observations.

**Usage**

```
leverage_stats(model, n = 6L)
```

**Arguments**

<code>model</code>	A fitted model object from the <code>lm</code> function.
<code>n</code>	an integer vector of length up to <code>dim(x)</code> (or 1, for non-dimensioned objects). Values specify the indices to be selected in the corresponding dimension (or along the length) of the object. A positive value of <code>n[i]</code> includes the first/last <code>n[i]</code> indices in that dimension, while a negative value excludes the last/first <code>abs(n[i])</code> , including all remaining indices. NA or non-specified values (when <code>length(n) &lt; length(dim(x))</code> ) select all indices in that dimension. Must contain at least one non-missing value.

**Value**

A vector of statistics

**Examples**

```
lmod <- lm(price ~ sqft_living, data = home_sales)
leverage_stats(lmod, n = 4)
```

---

leverage_test	<i>Identify leverage points</i>
---------------	---------------------------------

---

**Description**

leverage\_test returns the observations identified as a leverage point based on a threshold.

**Usage**

```
leverage_test(model, n = stats::nobs(model), ttype = "half", threshold = NULL)
```

**Arguments**

model	A fitted model object from the <a href="#">lm</a> function.
n	The number of leverage points to return. The default is all leverage points.
ttype	Threshold type. The default is "half". The other options are "2mean" and "custom". See Details.
threshold	A number between 0 and 1. Any observation with a leverage value above this number is declared a leverage point. This is automatically determined unless ttype = "custom".

**Details**

If ttype = "half", the threshold is 0.5.

If ttype = "2mean", the threshold is  $2p/n$ , where  $p = \text{length}(\text{stats::coef}(\text{model}))$  and  $n = \text{stats::nobs}(\text{model})$ , which is double the mean leverage value.

If ttype = "custom" then the user must manually specify threshold.

**Value**

A vector of statistics

**See Also**

[hatvalues](#)

**Examples**

```
lmod <- lm(price ~ sqft_living, data = home_sales)
# comparison of results using different threshold types
leverage_test(lmod)
leverage_test(lmod, ttype = "2mean", n = 7)
leverage_test(lmod, ttype = "custom", threshold = 0.1)
```

---

outlier\_plot

*Index plot of studentized residuals for lm object*


---

**Description**

outlier\_plot plots the studentized residuals (from the [rstudent](#) function) of a fitted `lm` object.

**Usage**

```
outlier_plot(
  model,
  id_n = 3,
  add_reference = TRUE,
  alpha = 0.05,
  ...,
  text_arglist = list(),
  abline_arglist = list(),
  extendrange_f = 0.08
)
```

**Arguments**

model	A fitted model object from the <a href="#">lm</a> function.
id_n	The number of points to identify with labels. The default is 3.
add_reference	A logical value indicating whether a reference line should be added. The default is TRUE. See Details.
alpha	The default lower quantile used for the reference line prior to the Bonferroni adjustment. The default is 0.05.
...	Additional arguments passed to the <a href="#">plot</a> function.
text_arglist	Additional arguments passed to the <a href="#">text</a> function, which is used to display the points that are identified.
abline_arglist	A named list specifying additional arguments passed to the <a href="#">abline</a> function for the horizontal reference line added to the plot.
extendrange_f	Positive number(s) specifying the fraction by which the range of the residuals should be extended using the <a href="#">extendrange</a> function. If longer than one, <code>f[1]</code> is used on the left and <code>f[2]</code> on the right.

## Details

If `add_reference = TRUE`, then reference lines are provided for the  $\alpha/(2n)$  and  $1 - \alpha/(2n)$  quantiles of a Student's  $t$  distribution with  $(df.residual(lmod) - 1)$  degrees of freedom, which are the standard quantiles used to identify outliers for a fitted model.

The vertical position of the reference line can be customized by setting the `h` argument of `abline_arglist`.

## Author(s)

Joshua French

## See Also

[plot](#), [text](#), [abline](#), [rstudent](#)

## Examples

```
lmod <- lm(price ~ sqft_living, data = home_sales)
outlier_plot(lmod, id_n = 1)
```

---

outlier\_stats

*Outlier statistics*

---

## Description

`outlier_stats` returns the ordered studentized residuals (decreasing based on magnitude) of `model` to identify the most unusual observations.

## Usage

```
outlier_stats(model, n = 6L)
```

## Arguments

<code>model</code>	A fitted model object from the <code>lm</code> function.
<code>n</code>	an integer vector of length up to <code>dim(x)</code> (or 1, for non-dimensioned objects). Values specify the indices to be selected in the corresponding dimension (or along the length) of the object. A positive value of <code>n[i]</code> includes the first/last <code>n[i]</code> indices in that dimension, while a negative value excludes the last/first <code>abs(n[i])</code> , including all remaining indices. NA or non-specified values (when <code>length(n) &lt; length(dim(x))</code> ) select all indices in that dimension. Must contain at least one non-missing value.

## Value

A vector of statistics.

## Examples

```
lmod <- lm(price ~ sqft_living, data = home_sales)
outlier_stats(lmod, n = 3)
```

---

outlier_test	<i>Identify outliers</i>
--------------	--------------------------

---

## Description

outlier\_test returns the observations identified as an outlier based on the Bonferroni correction for a studentized residuals.

## Usage

```
outlier_test(model, n = stats::nobs(model), alpha = 0.05)
```

## Arguments

model	A fitted model object from the <a href="#">lm</a> function.
n	The number of outliers to return. The default is all outliers.
alpha	The Bonferroni-adjusted threshold at which an outlier is identified. The default is 0.05.

## Value

A data frame with the outliers.

## See Also

[rstudent](#), [p.adjust](#)

## Examples

```
lmod <- lm(price ~ sqft_living, data = home_sales)
outlier_test(lmod)
outlier_test(lmod, alpha = 1, n = 7)
lmod2 <- lm(Petal.Length ~ Sepal.Length + Species, iris)
outlier_test(lmod2)
```

---

plot.confint\_adjust    *Plot confint\_adjust x*

---

### Description

Plot a confint\_adjust x produced by the [confint\\_adjust](#) function. See Examples.

### Usage

```
## S3 method for class 'confint_adjust'  
plot(x, parm, mar = c(5.1, 7.1, 4.1, 2.1), line = mar[2] - 1, ...)
```

### Arguments

x	An confint_adjust x produced by the <a href="#">confint_adjust</a> function.
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
mar	A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(5, 7, 4, 2) + 0.1.
line	The MARGin line, starting at 0 counting outwards, to draw the y-axis label. The default is 1 unit less than mar[2].
...	Additional arguments passed to plot.

### Details

The plot function doesn't automatically adjust the margins to account for the label names. If you need more space for your labels, then increase the second element of mar from 7.1 upward and line upward. Alternatively, if you need less space, then you can decrease both of these values. Or you could use the autoplot function that automatically controls the spacing.

### Value

None.

### Author(s)

Joshua P. French

### Examples

```
fit <- lm(100/mpg ~ disp + hp + wt + am, data = mtcars)  
# standard intervals  
cia <- confint_adjust(fit)  
plot(cia)  
# plot subset of intervals
```

```
plot(cia, parm = c("hp", "disp"))
# adjust margin and line for better formatting
plot(cia, parm = 2:3, mar = c(5.1, 4.1, 4.1, 2.1))
```

---

predict\_adjust

*Adjust prediction intervals for multiple comparisons*

---

## Description

A function to produce adjusted confidence/prediction intervals for predicted mean/new responses with a family-wise confidence level of at least level for lm objects (not applicable if no adjustment is used). Internally, the function is a slight revision of the code used in the [predict.lm](#) function.

## Usage

```
predict_adjust(
  object,
  newdata,
  se.fit = FALSE,
  scale = NULL,
  df = Inf,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  type = c("response", "terms"),
  method = "none",
  terms = NULL,
  na.action = stats::na.pass,
  pred.var = res.var/weights,
  weights = 1,
  ...
)
```

## Arguments

object	Object of class inheriting from "lm"
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
se.fit	A switch indicating if standard errors are required.
scale	Scale parameter for std.err. calculation.
df	Degrees of freedom for scale.
interval	Type of interval calculation. Can be abbreviated.
level	Tolerance/confidence level.
type	Type of prediction (response or model term). Can be abbreviated.
method	A character string indicating the type of adjustment to make. The default choice is "none". The other available options are "bonferroni", "wh" (Working-Hotelling), and "scheffe".



terms	If type = "terms", which terms (default is all terms), a <a href="#">character</a> vector.
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
pred.var	the variance(s) for future observations to be assumed for prediction intervals. See 'Details'.
weights	variance weights for prediction. This can be a numeric vector or a one-sided model formula. In the latter case, it is interpreted as an expression evaluated in newdata.
...	further arguments passed to or from other methods.

### Details

Let  $\alpha = 1 - \text{level}$ . All intervals are computed using the formula prediction  $\pm m * \text{epesd}$ , where  $m$  is a multiplier and  $\text{epesd}$  is the estimated standard deviation of the prediction error of the estimate.

method = "none" (no correction) produces the standard t-based confidence intervals with multiplier  $\text{stats::qt}(1 - \alpha/2, \text{df} = \text{object}\$df.\text{residual})$ .

method = "bonferroni" produces Bonferroni-adjusted intervals that use the multiplier  $m = \text{stats::qt}(1 - \alpha/(2 * k), \text{df} = \text{object}\$df.\text{residual})$ , where  $k$  is the number of intervals being produced.

The Working-Hotelling and Scheffe adjustments are distinct; the Working-Hotelling typically is related to a multiple comparisons adjustment for confidence intervals of the response mean while the Scheffe adjustment is typically related to a multiple comparisons adjustment for prediction intervals for a new response. However, references often uses these names interchangeably, so we use them equivalently in this function.

method = "wh" (Working-Hotelling) or method = "scheffe" and interval = "confidence" produces Working-Hotelling-adjusted intervals that use the multiplier  $m = \sqrt{p * \text{stats::qf}(\text{level}, \text{df1} = p, \text{df2} = \text{object}\$df.\text{residual})}$ , where  $p$  is the number of estimated coefficients in the model.

method = "wh" (Working-Hotelling) or method = "scheffe" and interval = "prediction" produces Scheffe-adjusted intervals that use the multiplier  $m = \sqrt{k * \text{stats::qf}(\text{level}, \text{df1} = k, \text{df2} = \text{object}\$df.\text{residual})}$ , where  $k$  is the number of intervals being produced.

### Value

predict\_adjust produces:

A vector of predictions if interval = "none".

A matrix of predictions and bounds with column names fit, lwr, and upr if interval is set. For type = "terms" this is a matrix with a column per term and may have an attribute "constant".

If se.fit is TRUE, a list with the following components is returned:

- fit: vector or matrix as above
- se.fit: standard error of predicted means
- residual.scale: residual standard deviations
- df: degrees of freedom for residual

## References

- Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilita. Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze, 8, 3-62.
- Working, H., & Hotelling, H. (1929). Applications of the theory of error to the interpretation of trends. Journal of the American Statistical Association, 24(165A), 73-85. doi:10.1080/01621459.1929.10506274
- Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2004). Applied Linear Statistical Models, 5th edition. New York: McGraw-Hill/Irwin.

## See Also

[predict.lm](#)

## Examples

```
fit <- lm(100/mpg ~ disp + hp + wt + am, data = mtcars)
newdata <- as.data.frame(rbind(
  apply(mtcars, 2, mean),
  apply(mtcars, 2, median)))
predict_adjust(fit, newdata = newdata,
  interval = "confidence",
  method = "none")
predict_adjust(fit, newdata = newdata,
  interval = "confidence",
  method = "bonferroni")
predict_adjust(fit, newdata = newdata,
  interval = "confidence",
  method = "wh")
predict_adjust(fit, newdata = newdata,
  interval = "prediction",
  method = "scheffe")
```

---

print.coef_matrix	<i>Print an object of class coef_matrix produced by the <a href="#">coef_matrix</a> function.</i>
-------------------	---

---

## Description

Print an object of class `coef_matrix` produced by the [coef\\_matrix](#) function.

## Usage

```
## S3 method for class 'coef_matrix'
print(x, digits = 2, ...)
```

## Arguments

x	An <code>coef_matrix</code> object produced by the <a href="#">coef_matrix</a> function.
digits	the minimum number of significant digits to be used: see <a href="#">print.default</a> .
...	Additional arguments to the <a href="#">print.data.frame</a> function, such as <code>digits</code> .

**Value**

A  $p \times 4$  matrix with columns for the estimated coefficient, its standard error, t-statistic and corresponding (two-sided) p-value.

**Author(s)**

Joshua French

**Examples**

```
fit <- lm(100/mpg ~ disp + hp + wt + am, data = mtcars)
(coefm <- coef_matrix(fit))
# print more digits
print(coefm, digits = 8)
```

---

print.confint\_adjust *Print confint\_adjust object*

---

**Description**

Print an object of class confint\_adjust produced by the [confint\\_adjust](#) function.

**Usage**

```
## S3 method for class 'confint_adjust'
print(x, ...)
```

**Arguments**

x                    An confint\_adjust object produced by the [confint\\_adjust](#) function.  
...                   Additional arguments to the [print.data.frame](#) function, such as digits.

**Value**

A data.frame with columns term, lwr, and upr, which are the coefficients for which inference is being made, and the lower and upper bounds of the confidence intervals for each coefficient, respectively.

**Author(s)**

Joshua French

**Examples**

```
fit <- lm(100/mpg ~ disp + hp + wt + am, data = mtcars)
(cia <- confint_adjust(fit))
print(cia, digits = 3)
```

---

print.predict\_adjust *Print predict\_adjust object*

---

### Description

Print an object of class predict\_adjust produced by the [predict\\_adjust](#) function.

### Usage

```
## S3 method for class 'predict_adjust'  
print(x, ...)
```

### Arguments

x                    An predict\_adjust object produced by the [predict\\_adjust](#) function.  
...                   Additional arguments to the [print.default](#) function, such as digits.

### Value

Depending on the interval argument of [predict\\_adjust](#):

A vector of predictions if interval = "none".

A matrix of predictions and bounds with column names fit, lwr, and upr if interval is set. For type = "terms" this is a matrix with a column per term and may have an attribute "constant".

If se.fit is TRUE, a list with the following components is returned:

- fit: vector or matrix as above
- se.fit: standard error of predicted means
- residual.scale: residual standard deviations
- df: degrees of freedom for residual

### Author(s)

Joshua French

### Examples

```
fit <- lm(100/mpg ~ disp + hp + wt + am, data = mtcars)  
(cia <- predict_adjust(fit))  
print(cia, digits = 3)
```

---

residual_plot	<i>Plot residuals of a fitted model</i>
---------------	---

---

**Description**

residual\_plot plots the residuals of a fitted model.

**Usage**

```
residual_plot(model, ...)
```

**Arguments**

model	A fitted model.
...	Currently unimplemented.

**Author(s)**

Joshua French

**Examples**

```
lmod <- lm(Girth ~ Height, data = trees)
residual_plot(lmod)
```

---

residual_plot.lm	<i>Plot residuals of a fitted lm object</i>
------------------	---

---

**Description**

residual\_plot.lm plots the residuals of a fitted lm object. In general, it is intended to provide similar functionality to `plot.lm` when `which = 1`, but can be used for different types of residuals and can also plot first-order predictor variables along the x-axis instead of only the fitted values.

Details about the different types of residuals are discussed in the `get_residuals` function.

**Usage**

```
## S3 method for class 'lm'
residual_plot(
  model,
  rtype = c("ordinary", "standardized", "studentized", "loo", "jackknife", "deleted",
            "internally studentized", "externally studentized"),
  xaxis = "fitted",
  id_n = 3,
  predictors = ~.,
```

```

smooth = stats::loess,
add_reference = TRUE,
add_smooth = TRUE,
...,
text_arglist = list(),
abline_arglist = list(),
smooth_arglist = list(),
lines_arglist = list(),
extendrange_f = 0.08
)

```

### Arguments

model	A fitted model object from the <code>lm</code> function.
rtype	The residual type to plot. The default is "ordinary". The other options are "standardized", "studentized", "loo", "jackknife", "deleted", "internally studentized", "externally studentized".
xaxis	The variable to use on the x-axis of the plot(s). The default is "fitted" to use fitted values. The other option is "predictors".
id_n	The number of points to identify with labels. The default is 3.
predictors	A formula describing the first-order predictors to plot the residuals against. The default is all available first-order predictors.
smooth	A function with a <code>formula</code> argument to smooth the desired plot. The default function is <code>loess</code> .
add_reference	A logical value indicating whether a reference line should be added. The default is TRUE.
add_smooth	A logical value indicating whether a smooth should be added to each plot produced. The default is TRUE.
...	Additional arguments passed to the <code>plot</code> function.
text_arglist	Additional arguments passed to the <code>text</code> function, which is used to display the points that are identified.
abline_arglist	A named list specifying additional arguments passed to the <code>abline</code> function for the horizontal reference line added to the plot.
smooth_arglist	A named list specifying additional arguments passed to the function provided in the <code>smooth</code> argument.
lines_arglist	A named list specifying additional arguments passed to the <code>lines</code> function for plotting the result of applying the smooth function.
extendrange_f	Positive number(s) specifying the fraction by which the range of the residuals should be extended using the <code>extendrange</code> function. If longer than one, <code>f[1]</code> is used on the left and <code>f[2]</code> on the right.

### Author(s)

Joshua French

**See Also**

[plot](#), [text](#), [abline](#), [lines](#) [loess](#).

**Examples**

```
lmod <- lm(Petal.Length ~ Sepal.Length + Species,
           data = iris)
# similarity with built-in plot.lm functionality
residual_plot(lmod)
plot(lmod, which = 1)
# residual plot for other residual types
residual_plot(lmod, rtype = "standardized", id_n = 0)
# another residual plot with several customizations
residual_plot(lmod,
              text_arglist = list(col = "blue", cex = 2),
              abline_arglist = list(lwd = 2, lty = 2,
                                    col = "brown"),
              lines_arglist = list(col = "purple"),
              )
# residual plot for predictors
residual_plot(lmod, xaxis = "pred", id_n = 2)
# residual plot for individual predictors
residual_plot(lmod, xaxis = "pred",
              predictors = ~ Sepal.Length, id_n = 2)
residual_plot(lmod, xaxis = "pred",
              predictors = ~ Species,)
```

---

rloo

---

*Compute leave-one-out residuals*


---

**Description**

rloo computes the leave-one-out residuals of model.

rjackknife and rdeleted are aliases for rloo.

**Usage**

```
rloo(model, ...)
```

```
rdeleted(model, ...)
```

```
rjackknife(model, ...)
```

**Arguments**

model	a fitted model object from the <a href="#">lm</a> function.
...	Currently unimplemented

**Author(s)**

Joshua French

**See Also**

[rloo.lm](#)

**Examples**

```
lmod <- lm(Girth ~ Height, data = trees)
rloo(lmod)
```

---

rloo.lm

*Compute leave-one-out residuals for 'lm' objects.*

---

**Description**

rloo.lm computes the leave-one-out residuals of the lm object stored in model.

rjackknife.lm and rdeleted.lm are aliases for rloo.lm.

**Usage**

```
## S3 method for class 'lm'
rloo(
  model,
  infl = stats::lm.influence(model, do.coef = FALSE),
  res = infl$wt.res,
  ...
)

## S3 method for class 'lm'
rdeleted(
  model,
  infl = stats::lm.influence(model, do.coef = FALSE),
  res = infl$wt.res,
  ...
)

## S3 method for class 'lm'
rjackknife(
  model,
  infl = stats::lm.influence(model, do.coef = FALSE),
  res = infl$wt.res,
  ...
)
```



**Arguments**

model	a fitted model object from the <code>lm</code> function.
infl	influence structure as returned by <code>lm.influence</code> .
res	(possibly weighted) residuals, with proper default.
...	Currently unimplemented

**Details**

Let  $\hat{\epsilon}_i$  denote the residual of the  $i$ th observation and  $h_i$  denote the leverage value of the  $i$ th observation. The leave-one-out residual for observation  $i$  is computed as

$$l_i = \frac{\hat{\epsilon}_i}{1 - h_i}.$$

**Author(s)**

Joshua French

**Examples**

```
lmod <- lm(Girth ~ Height, data = trees)
rloo(lmod)
```

---

rplot\_raw

*Helper function for residuals\_plot.lm*

---

**Description**

Helper function for residuals\_plot.lm

**Usage**

```
rplot_raw(
  x,
  y,
  idd,
  labels,
  xlab,
  smooth,
  add_reference,
  add_smooth,
  arglist,
  text_arglist,
  abline_arglist,
  smooth_arglist,
  lines_arglist,
  extendrange_f
)
```

**Arguments**

x	x-values to plot
y	y-values to plot
idd	Identified observations
labels	The labels to use for the identified points.
xlab	The x-axis label
smooth	The function for smoothing. Needs a 'formula' argument.
add_reference	Logical value
add_smooth	Logical value
arglist	Named list for plot
text_arglist	Named list for text
abline_arglist	Named list for abline
smooth_arglist	Named list for smooth
lines_arglist	Named list for lines of smooth
extendrange_f	Positive number(s) specifying the fraction by which the range of the residuals should be extended using the <a href="#">extendrange</a> function. If longer than one, f[1] is used on the left and f[2] on the right.

---

sl\_plot

*Spread-level plot of a fitted model*


---

**Description**

sl\_plot creates a spread-level plot for a fitted model.

**Usage**

```
sl_plot(model, ...)
```

**Arguments**

model	A fitted model.
...	Currently unimplemented.

**Author(s)**

Joshua French

**Examples**

```
lmod <- lm(Girth ~ Height, data = trees)
sl_plot(lmod)
```

---

sl_plot.lm	<i>Spread-level plot for lm object</i>
------------	--

---

### Description

sl\_plot.lm plots a spread-level plot of a fitted lm object. In general, it is intended to provide similar functionality to plot.lm when which = 3, but can be used for different types of residuals and can also plot first-order predictor variables along the x-axis instead of only the fitted values.

Details about the different types of residuals are discussed in the [get\\_residuals](#) function.

### Usage

```
## S3 method for class 'lm'
sl_plot(
  model,
  rtype = c("standardized", "studentized", "internally studentized",
            "externally studentized"),
  xaxis = "fitted",
  id_n = 3,
  predictors = ~.,
  smooth = stats::loess,
  add_smooth = TRUE,
  ...,
  text_arglist = list(),
  smooth_arglist = list(),
  lines_arglist = list()
)
```

### Arguments

model	A fitted model object from the <a href="#">lm</a> function.
rtype	The residual type to plot. The default is "ordinary". The other options are "standardized", "studentized", "loo", "jackknife", "deleted", "internally studentized", "externally studentized".
xaxis	The variable to use on the x-axis of the plot(s). The default is "fitted" to use fitted values. The other option is "predictors".
id_n	The number of points to identify with labels. The default is 3.
predictors	A formula describing the first-order predictors to plot the residuals against. The default is all available first-order predictors.
smooth	A function with a <a href="#">formula</a> argument to smooth the desired plot. The default function is <a href="#">loess</a> .
add_smooth	A logical value indicating whether a smooth should be added to each plot produced. The default is TRUE.
...	Additional arguments passed to the <a href="#">plot</a> function.

- `text_arglist` Additional arguments passed to the `text` function, which is used to display the points that are identified.
- `smooth_arglist` A named list specifying additional arguments passed to the function provided in the `smooth` argument.
- `lines_arglist` A named list specifying additional arguments passed to the `lines` function for plotting the result of applying the `smooth` function.

**Author(s)**

Joshua French

**See Also**

[plot](#), [text](#), [lines](#) [loess](#).

**Examples**

```
lmod <- lm(Petal.Length ~ Sepal.Length + Species,
           data = iris)
# similarity with built-in plot.lm functionality
sl_plot(lmod)
plot(lmod, which = 3)
# spread-level plot for other residual types
sl_plot(lmod, rtype = "studentized", id_n = 0)
# spread-level plot for predictors
sl_plot(lmod, xaxis = "pred", id_n = 2)
# spread-level plot for individual predictors
sl_plot(lmod, xaxis = "pred",
        predictors = ~ Sepal.Length,
        id_n = 2)
```

---

toluca

*Toluca Company data*

---

**Description**

Toluca Company data in *Applied Linear Statistical Models*, 5th edition, p. 19. From the book:

The Toluca Company manufactures refrigeration equipment as well as many replacement parts. In the past, one of the replacement parts has been produced periodically in lots of varying sizes. When a cost improvement program was undertaken, company officials wished to determine the optimum lot size for producing this part. The production of this part involves setting up the production process (which must be done no matter what is the lot size) and machining and assembly operations. One key input for the model to ascertain the optimum lot size was the relationship between lot size and labor hours required to produce the lot. To determine this relationship, data on lot size (`lot_size`) and work hours (`work_hours`) for 25 recent production runs were utilized.

**Usage**

```
data(toluca)
```

**Format**

A data frame with 25 observations and 2 variables:

lot\_size Number of replacement parts produced in the lot.

work\_hours Number of hours of work required to produce the lot.

**Author(s)**

Joshua P. French

**References**

Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2004). *Applied Linear Statistical Models*, 5th edition. New York: McGraw-Hill/Irwin.

# Index

- \* **datasets**
  - home\_sales, 18
- \* **data**
  - crime2009, 10
  - dwaine, 16
  - toluca, 44
- abline, 7, 8, 11–14, 20, 23, 25, 26, 28, 29, 38, 39
- arg\_check\_leverage\_test, 2
- autoplot, 3
- autoplot.confint\_adjust, 3
- character, 33
- coef\_compare, 4
- coef\_matrix, 5, 34
- confint.lm, 5, 6
- confint\_adjust, 3, 5, 31, 35
- cooks.distance, 7–9, 20, 23, 24
- cooks\_plot, 7
- cooks\_stats, 8
- cooks\_test, 9
- crime2009, 10
- dfbeta, 13
- dfbeta\_plot, 12
- dfbetas, 12
- dfbetas\_plot, 11
- dffits, 13–16, 23, 24
- dffits\_plot, 13
- dffits\_stats, 14
- dffits\_test, 15
- dwaine, 16
- extendrange, 7, 11, 12, 14, 20, 21, 23, 25, 28, 38, 42
- format, 4
- formula, 38, 43
- get\_residuals, 17, 22, 37, 43
- hatvalues, 20, 23–25, 27
- home\_sales, 18
- index\_plot\_lm, 19
- index\_plot\_raw, 21
- influence\_plot, 22
- influence\_stats, 23
- leverage\_plot, 25
- leverage\_stats, 26
- leverage\_test, 27
- lines, 38, 39, 44
- lm, 3–5, 7–9, 11–13, 15, 20, 22, 24–30, 38, 39, 41, 43
- lm.influence, 41
- loess, 38, 39, 43, 44
- outlier\_plot, 28
- outlier\_stats, 29
- outlier\_test, 30
- p.adjust, 30
- plot, 7, 8, 11–14, 20, 22, 23, 25, 26, 28, 29, 38, 39, 43, 44
- plot.confint\_adjust, 31
- plot.lm, 37, 43
- predict.lm, 32, 34
- predict\_adjust, 32, 36
- print.coef\_matrix, 34
- print.confint\_adjust, 35
- print.data.frame, 34, 35
- print.default, 34, 36
- print.predict\_adjust, 36
- rdeleted(rloo), 39
- rdeleted.lm(rloo.lm), 40
- residual\_plot, 37
- residual\_plot.lm, 37
- rjackknife(rloo), 39
- rjackknife.lm(rloo.lm), 40
- rloo, 39

rloo.lm, 40, 40  
rplot\_raw, 41  
rstandard, 23, 24  
rstudent, 20, 23, 24, 26, 28–30  
  
sl\_plot, 42  
sl\_plot.lm, 43  
  
text, 7, 8, 11–14, 20, 22, 23, 25, 26, 28, 29,  
38, 39, 44  
toluca, 44