

Package ‘speccurvieR’

July 23, 2025

Type Package

Title Easy, Fast, and Pretty Specification Curve Analysis

Version 0.4.2

Description Making specification curve analysis easy, fast, and pretty. It improves upon existing offerings with additional features and 'tidyverse' integration. Users can easily visualize and evaluate how their models behave under different specifications with a high degree of customization. For a description and applications of specification curve analysis see Simonsohn, Simmons, and Nelson (2020) <[doi:10.1038/s41562-020-0912-z](https://doi.org/10.1038/s41562-020-0912-z)>.

Encoding UTF-8

LazyData true

Imports ggplot2, magrittr, tidyr, dplyr, stringr, combinat, fixest, pbapply, parallel, lmttest, sandwich

RoxygenNote 7.3.1

License MIT + file LICENSE

URL <https://github.com/zayneember/speccurvieR>

BugReports <https://github.com/zayneember/speccurvieR/issues>

NeedsCompilation no

Author Zayne Sember [aut, cre, cph]

Maintainer Zayne Sember <zseember@ucsd.edu>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2024-10-09 21:30:01 UTC

Contents

bottles	2
controlExtractor	4
duplicate_remover	5
formula_builder	5

paste_factory	6
plotAIC	7
plotControlDistributions	8
plotCurve	9
plotDeviance	10
plotR2Adj	11
plotRMSE	12
plotVars	13
sca	14
scp	15
se_boot	16
se_compare	17
unAsIs	19
Index	20

bottles	<i>CalCOFI Bottle Data</i>
---------	----------------------------

Description

A subset of data from the California Cooperative Oceanic Fisheries Investigations. Each observation describes a sample of ocean water collected.

Usage

bottles

Format

'bottles' A data frame with 500 rows and 62 columns:

Cst_Cnt Cast count

Btl_Cnt Bottle Count

Sta_ID Line and Station

Depth_ID Depth ID

Depthm Bottle depth in meters

T_degC Water temperature in degrees Celsius

Salnty Salinity (Practical Salinity Scale 1978)

O2ml_L Milliliters of oxygen per liter of seawater

STheta Potential Density (Sigma Theta), Kg/M³

O2Sat Oxygen percent saturation

Oxy_μmol/Kg Oxygen micromoles per kilogram seawater

BtlNum Niskin bottle sample was collected from

RecInd Record Indicator

T_prec Temperature Precision
T_qual Quality Code
S_prec Salinity Precision
S_qual Quality Code
P_qual Quality Code
O_qual Quality Code
SThta Quality Code
O2Sat Quality Code
ChlorA Micrograms Chlorophyll-a per liter seawater
Chlqua Quality Code
Phaeop Micrograms Phaeopigment per liter seawater
Phaqua Quality Code
PO4uM Micromoles Phosphate per liter of seawater
PO4q Quality Code
SiO3uM Micromoles Silicate per liter of seawater
SiO3qu Quality Code
NO2uM Micromoles Nitrite per liter of seawater
NO2q Quality Code
NO3uM Micromoles Nitrate per liter of seawater
NO3q Quality Code
NH3uM Micromoles Ammonia per liter of seawater
NH3q Quality Code
C14As1 14C Assimilation of Replicate 1
C14A1p Precision of 14C Assimilation of Replicate 1
C14A1q Quality Code
C14As2 14C Assimilation of Replicate 2
C14A2p Precision of 14C Assimilation of Replicate 2
C14A2q Quality Code
DarkAs 14C Assimilation of Dark/Control Bottle
DarkAp Precision of 14C Assimilation of Dark/Control Bottle
Darkaq Quality Code
MeanAs Mean 14C Assimilation of Replicates 1 and 2
MeanAp Precision of Mean 14C Assimilation of Replicates 1 and 2
MeanAq Quality Code
IncTim Elapsed incubation time of primary productivity experiment
LightP Light intensities of the incubation tubes
R_Depth Reported Depth (from pressure) in meters

R_Temp Reported (Potential) Temperature in degrees Celsius
R_Sal Reported Salinity (from Specific Volume Anomaly, M³/Kg)
R_DYNHT Reported Dynamic Height in units of dynamic meters
R_Nuts Reported Ammonium concentration
R_Oxy_μmol/Kg Reported Oxygen micromoles/kilogram
DIC1 Dissolved Inorganic Carbon micromoles per kilogram solution
DIC2 Dissolved Inorganic Carbon on a replicate sample
TA1 Total Alkalinity micromoles per kilogram solution
TA2 Total Alkalinity on a replicate sample
pH1 pH (the degree of acidity/alkalinity of a solution)
pH2 pH on a replicate sample
DIC Quality Comment Quality Comment

Source

<<https://calcofi.org/data/oceanographic-data/bottle-database/>>

controlExtractor	<i>Extracts the control variable names and coefficients from an lm model summary.</i>
------------------	---

Description

Extracts the control variable names and coefficients from a model summary.

Usage

```
controlExtractor(model, x, feols_model = F)
```

Arguments

model	A model summary object.
x	A string containing the independent variable name.
feols_model	An indicator for whether ‘model’ is a ‘fixest::feols()’ model. Defaults to ‘FALSE’.

Value

A dataframe with two columns, ‘term’ contains the name of the control and ‘coef’ contains the coefficient estimate.

Examples

```

m <- summary(lm(Salnty ~ STheta + T_degC, bottles))
controlExtractor(model = m, x = "STheta");

m <- summary(lm(Salnty ~ STheta*T_degC + O2Sat, bottles))
controlExtractor(model = m, x = "STheta");

```

duplicate_remover	<i>Removes duplicate control variables</i>
-------------------	--

Description

Removes duplicate control variables from user input.

Usage

```
duplicate_remover(controls, x)
```

Arguments

controls	A vector of strings containing control variable names.
x	A string containing the independent variable name.

Value

A vector of strings containing control variable names

Examples

```
duplicate_remover(controls = c("control1", "control2*control3"),
  x = "independentVariable");
```

formula_builder	<i>Builds models formulae with every combination of control variables possible.</i>
-----------------	---

Description

Builds models formulae with every combination of control variables possible.

Usage

```
formula_builder(y, x, controls, fixedEffects = NA)
```

Arguments

y	A string containing the dependent variable name.
x	A string containing the independent variable name.
controls	A vector of strings containing control variable names.
fixedEffects	A string containing the name of a variable to use for fixed effects, defaults to 'NA' indicating no fixed effects desired.

Value

A vector of formula objects using every possible combination of controls.

Examples

```
formula_builder("dependentVariable", "independentVariable",
               c("control1", "control2"));
formula_builder("dependentVariable", "independentVariable",
               c("control1*control2"), fixedEffects="month");
```

paste_factory	<i>Paste together controls and independent variable</i>
---------------	---

Description

'paste_factory()' constructs the right hand side of the regression as a string i.e. "x + control1 + control2".

Usage

```
paste_factory(controls, x)
```

Arguments

controls	A vector of strings containing control variable names.
x	A string containing the independent variable name.

Value

A string concatenating independent and control variables separated by '+'.

Examples

```
paste_factory(controls = c("control1", "control2"),
              x = "independentVariable");
```

plotAIC

*Plots the AIC across model specifications.***Description**

plotAIC() plots the Akaike information criterion across model specifications. Only available for nonlinear regression models.

Usage

```
plotAIC(sca_data, title = "", showIndex = TRUE, plotVars = TRUE)
```

Arguments

sca_data	A data frame returned by 'sca()' containing model estimates from the specification curve analysis.
title	A string to use as the plot title. Defaults to an empty string, "".
showIndex	A boolean indicating whether to label the model index on the the x-axis. Defaults to 'TRUE'.
plotVars	A boolean indicating whether to include a panel on the plot showing which variables are present in each model. Defaults to 'TRUE'.

Value

If 'plotVars = TRUE' returns a grid grob (i.e. the output of a call to 'grid.draw'). If 'plotVars = FALSE' returns a ggplot object.

Examples

```
plotAIC(sca_data = sca(y = "Salnty", x = "T_degC",
                      controls = c("ChlorA", "O2Sat"),
                      data = bottles, progressBar = TRUE, parallel = FALSE),
        title = "AIC");
plotAIC(sca_data = sca(y = "Salnty", x = "T_degC",
                      controls = c("ChlorA*O2Sat"),
                      data = bottles, progressBar = FALSE,
                      parallel = FALSE),
        showIndex = FALSE, plotVars = FALSE);
plotAIC(sca_data = sca(y = "Salnty", x = "T_degC",
                      controls = c("ChlorA*N03uM", "O2Sat*N03uM"),
                      data = bottles,
                      progressBar = TRUE, parallel = TRUE, workers = 2));
```

plotControlDistributions

Plots control variable distributions.

Description

plotControlDistributions() plots the distribution of coefficients for each control variable included in the model specifications.

Usage

```
plotControlDistributions(sca_data, title = "", type = "density")
```

Arguments

sca_data	A data frame returned by ‘sca()’ containing model estimates from the specification curve analysis.
title	A string to use as the plot title. Defaults to an empty string, “”.
type	A string indicating what type of distribution plot to produce. When ‘type = "density"’ density plots are produced. When ‘type = "hist"’ or ‘type = "histogram"’ histograms are produced. Defaults to “density”.

Value

A ggplot object.

Examples

```
plotControlDistributions(sca_data = sca(y="Salnty", x="T_degC",
                                     controls = c("ChlorA", "O2Sat"),
                                     data = bottles,
                                     progressBar = TRUE, parallel = FALSE),
                       title = "Control Variable Distributions")
plotControlDistributions(sca_data = sca(y = "Salnty", x="T_degC",
                                     controls = c("ChlorA*O2Sat"),
                                     data = bottles,
                                     progressBar = FALSE, parallel = FALSE),
                       type = "hist")
plotControlDistributions(sca_data = sca(y = "Salnty", x = "T_degC",
                                     controls = c("ChlorA*N03uM",
                                                  "O2Sat*N03uM"),
                                     data = bottles, progressBar = TRUE,
                                     parallel = TRUE, workers = 2),
                       type = "density")
```

plotCurve	<i>Plots a specification curve.</i>
-----------	-------------------------------------

Description

plotCurve() takes the data frame output of sca() and produces a ggplot of the independent variable's coefficient (as indicated in the call to sca()) across model specifications. By default a panel is added showing which control variables are present in each model. Note that the ggplot output by this function can only be further customized when 'plotVars = FALSE', i.e. when the control variable panel is not included.

Usage

```
plotCurve(
  sca_data,
  title = "",
  showIndex = TRUE,
  plotVars = TRUE,
  ylab = "Coefficient",
  plotSE = "bar"
)
```

Arguments

sca_data	A data frame returned by 'sca()' containing model estimates from the specification curve analysis.
title	A string to use as the plot title. Defaults to an empty string, "".
showIndex	A boolean indicating whether to label the model index on the the x-axis. Defaults to 'TRUE'.
plotVars	A boolean indicating whether to include a panel on the plot showing which variables are present in each model. Defaults to 'TRUE'.
ylab	A string to be used as the y-axis label. Defaults to "Coefficient".
plotSE	A string indicating whether to display standard errors as bars or plots. For bars 'plotSE = "bar"', for ribbons 'plotSE = "ribbon"'. If any other value is supplied then no standard errors are included. Defaults to "bar".

Value

If 'plotVars = TRUE' returns a grid grob (i.e. the output of a call to 'grid.draw'). If 'plotVars = FALSE' returns a ggplot object.

Examples

```
plotCurve(sca_data = sca(y="Salnty", x="T_degC", c("ChlorA", "O2Sat"),
  data=bottles, progressBar=TRUE, parallel=FALSE),
  title = "Salinity and Temperature Models",
```

```

        showIndex = TRUE, plotVars = TRUE,
        ylab = "Coefficient value", plotSE = "ribbon");
plotCurve(sca_data = sca(y="Salnty", x="T_degC",
        c("ChlorA*O2Sat", "ChlorA", "O2Sat"),
        data=bottles, progressBar=FALSE, parallel=FALSE),
        showIndex = TRUE, plotVars = TRUE,
        plotSE = "ribbon");
plotCurve(sca_data = sca(y="Salnty", x="T_degC",
        c("ChlorA*NO3uM", "O2Sat", "ChlorA", "NO3uM"),
        data=bottles,
        progressBar = TRUE, parallel = TRUE, workers=2),
        plotSE="");

```

plotDeviance

Plots the deviance of residuals across model specifications.

Description

plotDeviance() plots the deviance of residuals across model specifications. Only available for linear regression models.

Usage

```
plotDeviance(sca_data, title = "", showIndex = TRUE, plotVars = TRUE)
```

Arguments

sca_data	A data frame returned by ‘sca()’ containing model estimates from the specification curve analysis.
title	A string to use as the plot title. Defaults to an empty string, “”.
showIndex	A boolean indicating whether to label the model index on the the x-axis. Defaults to ‘TRUE’.
plotVars	A boolean indicating whether to include a panel on the plot showing which variables are present in each model. Defaults to ‘TRUE’.

Value

If ‘plotVars = TRUE’ returns a grid grob (i.e. the output of a call to ‘grid.draw’). If ‘plotVars = FALSE’ returns a ggplot object.

Examples

```

plotDeviance(sca_data = sca(y = "Salnty", x = "T_degC",
        controls = c("ChlorA", "O2Sat"),
        data = bottles, progressBar = TRUE,
        parallel = FALSE),
        title = "Model Deviance");
plotDeviance(sca_data = sca(y = "Salnty", x = "T_degC",

```

```

        controls = c("ChlorA*O2Sat"),
        data = bottles, progressBar = FALSE,
        parallel = FALSE),
    showIndex = FALSE, plotVars = FALSE);
plotDeviance(sca_data = sca(y = "Salnty", x="T_degC",
    controls = c("ChlorA*N03uM", "O2Sat*N03uM"),
    data = bottles, progressBar = TRUE, parallel = TRUE,
    workers = 2));

```

plotR2Adj

Plots the adj. R-squared across model specifications.

Description

plotR2Adj() plots the adjusted R-squared across model specifications. Only available for linear regression models. Note when fixed effects are specified the within adjusted R-squared is used (i.e. 'fixest::r2()' with 'type="var2"').

Usage

```
plotR2Adj(sca_data, title = "", showIndex = TRUE, plotVars = TRUE)
```

Arguments

sca_data	A data frame returned by 'sca()' containing model estimates from the specification curve analysis.
title	A string to use as the plot title. Defaults to an empty string, "".
showIndex	A boolean indicating whether to label the model index on the x-axis. Defaults to 'TRUE'.
plotVars	A boolean indicating whether to include a panel on the plot showing which variables are present in each model. Defaults to 'TRUE'.

Value

If 'plotVars = TRUE' returns a grid grob (i.e. the output of a call to 'grid.draw'). If 'plotVars = FALSE' returns a ggplot object.

Examples

```

plotR2Adj(sca_data = sca(y = "Salnty", x = "T_degC",
    controls = c("ChlorA", "O2Sat"),
    data = bottles, progressBar = TRUE,
    parallel = FALSE),
    title = "Adjusted R^2");
plotR2Adj(sca_data = sca(y="Salnty", x="T_degC",
    controls = c("ChlorA*O2Sat"),
    data = bottles, progressBar = FALSE,
    parallel = FALSE),

```

```

      showIndex = FALSE, plotVars = FALSE);
plotR2Adj(sca_data = sca(y = "Salnty", x = "T_degC",
      controls = c("ChlorA*N03uM", "O2Sat*N03uM"),
      data = bottles,
      progressBar = TRUE, parallel = TRUE, workers = 2));

```

plotRMSE

Plots RMSE across model specifications.

Description

plotRMSE() plots the root mean square error across model specifications. Only available for linear regression models.

Usage

```
plotRMSE(sca_data, title = "", showIndex = TRUE, plotVars = TRUE)
```

Arguments

sca_data	A data frame returned by 'sca()' containing model estimates from the specification curve analysis.
title	A string to use as the plot title. Defaults to an empty string, "".
showIndex	A boolean indicating whether to label the model index on the the x-axis. Defaults to 'TRUE'.
plotVars	A boolean indicating whether to include a panel on the plot showing which variables are present in each model. Defaults to 'TRUE'.

Value

If 'plotVars = TRUE' returns a grid grob (i.e. the output of a call to 'grid.draw'). If 'plotVars = FALSE' returns a ggplot object.

Examples

```

plotRMSE(sca_data = sca(y="Salnty", x="T_degC", c("ChlorA", "O2Sat"),
      data=bottles, progressBar=TRUE, parallel=FALSE),
      title = "RMSE");
plotRMSE(sca_data = sca(y="Salnty", x="T_degC", c("ChlorA*O2Sat"),
      data=bottles, progressBar=FALSE, parallel=FALSE),
      showIndex = FALSE, plotVars = FALSE);
plotRMSE(sca_data = sca(y="Salnty", x="T_degC",
      c("ChlorA*N03uM", "O2Sat*N03uM"), data=bottles,
      progressBar = TRUE, parallel=TRUE, workers=2));

```

plotVars	<i>Plots the variables in each model.</i>
----------	---

Description

plotVars() plots the variables included in each model specification in order of model index. Returns a ggplot object that can then be combined with the output of other functions like plotRMSE() if further customization of each plot is desired.

Usage

```
plotVars(sca_data, title = "", colorControls = FALSE)
```

Arguments

sca_data	A data frame returned by 'sca()' containing model estimates from the specification curve analysis.
title	A string to use as the plot title. Defaults to an empty string, "".
colorControls	A boolean indicating whether to give each variable a color to improve readability. Defaults to 'FALSE'.

Value

A ggplot object.

Examples

```
plotVars(sca_data = sca(y = "Salnty", x = "T_degC",
  controls = c("ChlorA", "O2Sat"),
  data = bottles, progressBar = TRUE,
  parallel = FALSE),
  title = "Model Variable Specifications");
plotVars(sca_data = sca(y = "Salnty", x = "T_degC",
  controls = c("ChlorA*O2Sat"),
  data = bottles, progressBar = FALSE,
  parallel = FALSE),
  colorControls = TRUE);
plotVars(sca_data = sca(y = "Salnty", x = "T_degC",
  controls = c("ChlorA*NO3uM", "O2Sat*NO3uM"),
  data = bottles,
  progressBar = TRUE, parallel = TRUE, workers = 2));
```

Description

`sca()` is the workhorse function of the package—this estimates models with every possible combination of the controls supplied and returns a data frame where each row contains the pertinent information and parameters for a given model by default. This data frame can then be input to `plotCurve()` or any other plotting function in the package. Alternatively, if `'returnFormulae = TRUE'`, it returns a list of formula objects with every possible combination of controls.

Usage

```
sca(
  y,
  x,
  controls,
  data,
  weights = NULL,
  family = "linear",
  link = NULL,
  fixedEffects = NULL,
  returnFormulae = FALSE,
  progressBar = TRUE,
  parallel = FALSE,
  workers = 2
)
```

Arguments

<code>y</code>	A string containing the column name of the dependent variable in data.
<code>x</code>	A string containing the column name of the independent variable in data.
<code>controls</code>	A vector of strings containing the column names of the control variables in data.
<code>data</code>	A dataframe containing <code>y</code> , <code>x</code> , <code>controls</code> , and (optionally) the variables to be used for fixed effects or clustering.
<code>weights</code>	Optional string with the column name in <code>'data'</code> that contains weights.
<code>family</code>	A string indicating the family of models to be used. Defaults to "linear" for OLS regression but supports all families supported by <code>'glm()'</code> .
<code>link</code>	A string specifying the link function to be used for the model. Defaults to <code>'NULL'</code> for OLS regression using <code>'lm()'</code> or <code>'fixest::feols()'</code> depending on whether fixed effects are supplied. Supports all link functions supported by the family parameter of <code>'glm()'</code> .
<code>fixedEffects</code>	A string containing the column name of the variable in data desired for fixed effects. Defaults to <code>NULL</code> in which case no fixed effects are included.

<code>returnFormulae</code>	A boolean. When 'TRUE' a list of model formula objects is returned but the models are not estimated. Defaults to 'FALSE' in which case a dataframe of model results is returned.
<code>progressBar</code>	A boolean indicating whether the user wants a progress bar for model estimation. Defaults to 'TRUE'.
<code>parallel</code>	A boolean indicating whether to parallelize model estimation. Parallelization only offers a speed advantage when a large (> 1000) number of models is being estimated. Defaults to 'FALSE'.
<code>workers</code>	An integer indicating the number of workers to use for parallelization. Defaults to 2.

Value

When 'returnFormulae' is 'FALSE', a dataframe where each row contains the independent variable coefficient estimate, standard error, test statistic, p-value, model specification, and measures of model fit.

Examples

```
sca(y = "Salnty", x = "T_degC", controls = c("ChlorA", "O2Sat"),
    data = bottles, progressBar = TRUE, parallel = FALSE);
sca(y = "Salnty", x = "T_degC", controls = c("ChlorA*N03uM", "O2Sat*N03uM"),
    data = bottles, progressBar = TRUE, parallel = TRUE, workers = 2);
sca(y = "Salnty", x = "T_degC", controls = c("ChlorA", "O2Sat*N03uM"),
    data = bottles, progressBar = TRUE, parallel = FALSE,
    returnFormulae = TRUE);
```

scp	<i>Prepares the output of 'sca()' for plotting.</i>
-----	---

Description

Takes in the data frame output by 'sca()' and returns a list with the data frame and labels to make a plot to visualize the controls included in each spec curve model.

Usage

```
scp(sca_data)
```

Arguments

`sca_data` A data frame output by 'sca'.

Value

A list containing a data frame, control coefficients, and control names.

Examples

```
scp(sca(y = "Salnty", x = "T_degC", controls = c("ChlorA", "O2Sat"),
      data = bottles, progressBar=TRUE, parallel=FALSE));
```

se_boot	<i>Estimates bootstrapped standard errors for regression models</i>
---------	---

Description

Takes in a data frame, regression formula, and bootstrapping parameters and estimates bootstrapped standard errors for models with and without fixed effects.

Usage

```
se_boot(data, formula, n_x, n_samples, sample_size, weights = NULL)
```

Arguments

data	A data frame containing the variables provided in ‘formula’.
formula	A string containing a regression formula, with or without fixed effects.
n_x	An integer representing the number of independent variables in the regression.
n_samples	An integer indicating how many times the model should be estimated with a random subset of the data.
sample_size	An integer indicating how many observations are in each random subset of the data.
weights	Optional string with the column name in ‘data’ that contains weights.

Value

A named list containing bootstrapped standard errors for each coefficient.

Examples

```
se_boot(data = bottles, formula = "Salnty ~ T_degC + ChlorA + O2Sat",
      n_x = 3, n_samples = 4, sample_size = 300)

se_boot(data = data.frame(x1 = rnorm(50000, mean=4, sd=10),
      x2 = rnorm(50000, sd=50),
      ID = rep(1:100, 500),
      area = rep(1:50, 1000),
      y = rnorm(50000)),
      formula = "y ~ x1 + x2 | ID",
      n_x = 2, n_samples = 10, sample_size = 1000)
```

se_compare

*Compare different kinds of standard errors***Description**

se_compare() takes in a regression formula (with or without fixed effects), data, and the types of standard errors desired, including clustered, heteroskedasticity-consistent, and bootstrapped. It then returns a data frame with coefficient and standard error estimates for easy comparison and plotting.

Usage

```
se_compare(
  formula,
  data,
  weights = NULL,
  types = "all",
  cluster = NULL,
  clusteredOnly = FALSE,
  fixedEffectsOnly = FALSE,
  bootSamples = NULL,
  bootSampleSize = NULL
)
```

Arguments

formula	A string containing a regression formula, with or without fixed effects.
data	A data frame containing the variables provided in ‘formula’ and any clustering variables passed to ‘cluster’.
weights	Optional string with the column name in ‘data’ that contains weights.
types	A string or vector of strings specifying what types of standard errors are desired. Defaults to "all". The following types are supported for non-fixed effects models: With clustering: "HC0", "HC1", "HC2", "HC3". Without clustering: "iid" (i.e. normal standard errors), "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", "HC5", "bootstrapped". The following types are supported for fixed effects models: With clustering: "CL_FE" (clustered by fixed effects, i.e. the default standard errors reported by ‘feols()’ if no clusters are supplied), if clusters are supplied then the conventional clustered standard errors from ‘feols()’ are estimated for each clustering variable. Two- way clustered standard errors are not supported at this time. Without clustering: "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", "HC5", "bootstrapped".
cluster	A string or vector of strings specifying variables present in ‘data’ to be used for clustering standard errors.

<code>clusteredOnly</code>	A boolean indicating whether only standard errors with clustering should be estimated, defaults to 'FALSE'.
<code>fixedEffectsOnly</code>	A boolean indicating whether only standard errors for fixed effects models should be estimated, defaults to 'FALSE'.
<code>bootSamples</code>	An integer or vector of integers indicating how many times the model should be estimated with a random subset of the data. If a vector then every combination of 'bootSamples' and 'bootSampleSize' are estimated.
<code>bootSampleSize</code>	An integer or vector of integers indicating how many observations are in each random subset of the data. If a vector then every combination of 'bootSamples' and 'bootSampleSize' are estimated.

Value

A data frame where row represents an independent variable in the model and each column a type of standard error. Coefficient estimates for each variable are also included (column "estimate" for non-fixed effects model and column "estimate_FE" for fixed effects models). Columns are automatically named to specify the standard error type.

Some examples:

"iid" = normal standard errors, i.e. assuming homoskedasticity

"CL_FE" = standard errors clustered by fixed effects

"bootstrap_k8n300_FE" = bootstrapped standard errors for a fixed effects model where 'bootSamples = 8' and 'bootSampleSize = 300'

"CL_Depth_ID_FE" = standard errors clustered by the variable "Depth_ID" for a model with fixed effects

"HC0_Sta_ID" = HC0 standard errors clustered by the variable "Sta_ID"

Note: for fixed effects models the "(Intercept)" row will be all 'NA' because the intercept is not reported by 'feols()' when fixed effects are present.

Examples

```
se_compare(formula = "Salnty ~ T_degC + ChlorA + O2Sat | Sta_ID",
            data = bottles, types = "all", cluster = c("Depth_ID", "Sta_ID"),
            fixedEffectsOnly = FALSE, bootSamples=c(4, 8, 10),
            bootSampleSize=c(300, 500))

se_compare(formula = "Salnty ~ T_degC + ChlorA + O2Sat", data = bottles,
            types = "bootstrapped", bootSamples = c(8, 10),
            bootSampleSize = c(300, 500))

se_compare(formula = "Salnty ~ T_degC + ChlorA", data = bottles,
            types = c("HC0", "HC1", "HC3"))
```

`unAsIs`*Removes the 'AsIs' class attribute from the input.*

Description

Removes the 'AsIs' class attribute from the input. Taken from: <<https://stackoverflow.com/a/12866609>>

Usage

```
unAsIs(x)
```

Arguments

`x` An object with the 'AsIs' class attribute.

Value

An object without the 'AsIs' class attribute.

Examples

```
unAsIs(x = I(c(1:4)));
```

Index

- * **datasets**
 - bottles, [2](#)
- bottles, [2](#)
- controlExtractor, [4](#)
- duplicate_remover, [5](#)
- formula_builder, [5](#)
- paste_factory, [6](#)
- plotAIC, [7](#)
- plotControlDistributions, [8](#)
- plotCurve, [9](#)
- plotDeviance, [10](#)
- plotR2Adj, [11](#)
- plotRMSE, [12](#)
- plotVars, [13](#)
- sca, [14](#)
- scp, [15](#)
- se_boot, [16](#)
- se_compare, [17](#)
- unAsIs, [19](#)