Package 'supernova'

July 25, 2021

Type Package

Title Judd, McClelland, & Ryan Formatting for ANOVA Output

Version 2.4.2

Date 2021-07-24

Encoding UTF-8

LazyData yes

Description Produces ANOVA tables in the format used by Judd, McClelland, and Ryan (2017, ISBN: 978-1138819832) in their introductory textbook, Data Analysis. This includes proportional reduction in error and formatting to improve ease the transition between the book and R.

License GPL-3

Depends R (>= 3.4.0)

Suggests car, covr, lintr, lme4, testthat (>= 2.1.0), vdiffr, tidyr

Imports ggplot2, tibble, cli, magrittr, purrr, rlang, stringr, vctrs, pillar (>= 1.5.0), methods, backports

RoxygenNote 7.1.1

URL https://github.com/UCLATALL/supernova

BugReports https://github.com/UCLATALL/supernova/issues

Config/testthat/edition 3

NeedsCompilation no

Author Adam Blake [aut] (<https://orcid.org/0000-0001-7881-8652>), Jeff Chrabaszcz [aut], Ji Son [aut] (<https://orcid.org/0000-0002-4258-4791>), Jim Stigler [cre, aut] (<https://orcid.org/0000-0001-6107-7827>)

Maintainer Jim Stigler <jstigler@gmail.com>

Repository CRAN

Date/Publication 2021-07-25 04:20:02 UTC

R topics documented:

`class_data` *Teacher effectiveness data*

Description

These are hypothetical data for a small study comparing the effectiveness of three different teachers. Each teacher was randomly assigned a group of 35 high school students and asked to teach a 5-day unit on a new science topic for which none of the students had any prior knowledge. All three teachers used the same curriculum materials and lesson plans, and student learning was assessed using a common 30-item test at the end of the unit. The research question was: Were some teachers more effective than others? Did teachers differ in the amount of learning they were able to produce in their students?

Usage

```
class_data
```

Format

A data frame with 105 observations on the following 2 variables:

teacher coded as A, B, or C

outcome each student's score on the outcome test

drop_term *Drop a term from the given model*

Description

This function is needed to re-fit the models for Type III SS. If you have a model with an interactive term (e.g. `y ~ a + b + a:b`), when you try to refit without one of the lower-order terms (e.g. `y ~ a + a:b`) `lm()` will add it back in. This function uses a fitting function that operates underneath `lm()` to circumvent this behavior. (It is very similar to `drop1()`.)

Usage

```
drop_term(fit, term)
```

Arguments

`fit`	The model to update.
`term`	The term to drop from the model.

Value

An object of the class `lm`.

equation *Print the output of* `lm()` *with the fitted equation.*

Description

Print the output of `lm()` with the fitted equation.

Usage

```
equation(x, digits = max(3L, getOption("digits") - 3L))
```

Arguments

`x`	The fitted linear model to print.
`digits`	The minimal number of significant digits.

Value

Invisibly return the fitted linear model.

`estimate_extraction` *Extract estimates/statistics from a model*

Description

This collection of functions is useful for extracting estimates and statistics from a fitted model. They are particularly useful when estimating many models, like when bootstrapping confidence intervals. Each function can be used with an already fitted model as an `lm` object, or a formula and associated data can be passed to it. **All of these assume the comparison is the empty model.**

Usage

```
b0(object, data = NULL, ...)

b1(object, data = NULL, ...)

f(object, data = NULL, ...)

pre(object, data = NULL, ...)

sse(object, data = NULL, ...)

ssm(object, data = NULL, ...)

ssr(object, data = NULL, ...)

fVal(object, data = NULL, ...)

PRE(object, data = NULL, ...)

SSE(object, data = NULL, ...)

SSM(object, data = NULL, ...)

SSR(object, data = NULL, ...)
```

Arguments

`object`	A `lm` object, or `formula`.
`data`	If `object` is a formula, the data to fit the formula to as a `data.frame`.
`...`	Additional arguments passed through to `lm`.

Details

- `b0`: The intercept from the full model.
- `b1`: The slope b1 from the full model.

- `fVal`: The F value from the full model.
- `PRE`: The Proportional Reduction in Error for the full model.
- `SSE`: The SS Error (SS Residual) from the model.
- `SSM`: The SS Model (SS Regression) for the full model.
- `SSR`: Alias for SSM.

Value

The value of the estimate as a single number.

`Fingers` *Data from introductory statistics students at a university.*

Description

Students at a university taking an introductory statistics course were asked to complete this survey as part of their homework.

Usage

```
Fingers
```

Format

A data frame with 157 observations on the following 16 variables:

Sex Sex of participant.

RaceEthnic Racial or ethnic background.

FamilyMembers Members of immediate family (excluding self).

SSLast Last digit of social security number (NA if no SSN).

Year Year in school: 1=First, 2=Second, 3=Third, 4=Fourth, 5=Other

Job Current employment status: 0=not working, 1=part-time job, 2=full-time job

MathAnxious Agreement with this statement "In general I tend to feel very anxious about mathematics": 2=Strongly Agree, 1=Agree, 0=Neither Agree nor Disagree, -1=Disagree, -2=Strongly Disagree

Interest Interest in statistics and the course: 2=very interested in course and statistics, 1=somewhat interested, 0=no interest, -1=dread the course.

GradePredict Prediction for final grade in the course from the university's grade points per unit: 4.0=A, 3.7=A-, 3.3=B+, 3.0=B, 2.7=B-, 2.3=C+, 2.0=C, 1.7=C-, 1.3=Below C-

Thumb Length in mm from tip of thumb to the crease between the thumb and palm.

Index Length in mm from tip of index finger to the crease between the index finger and palm.

Middle Length in mm from tip of middle finger to the crease between the middle finger and palm.

Ring Length in mm from tip of ring finger to the crease between the middle finger and palm.

Pinkie Length in mm from tip of pinkie finger to the crease between the pinkie finger and palm

Height Height in inches.

Weight Weight in pounds.

`Fingers.messy` *Data from introductory statistics students at a university.*

Description

Students at a university taking an introductory statistics course were asked to complete this survey as part of their homework.

Usage

```
Fingers.messy
```

Format

A dataset with 210 observations on the following 16 variables.

Sex Sex of participant: 1=female, 2=male, 3=prefer not to answer

RaceEthnic Racial or ethnic background: 1=White, 2=African American, 3=Asian, 4=Latino, 5=Other

FamilyMembers Members of immediate family (excluding self).

SSLast Last digit of social security number (NA if no SSN).

Year Year in school.

Job Current employment status.

MathAnxious Agreement with this statement "In general I tend to feel very anxious about mathematics."

Interest Interest in statistics and the course.

GradePredict Prediction for final grade in the course from the university's grade points per unit.

Thumb Length in mm from tip of thumb to the crease between the thumb and palm.

Index Length in mm from tip of index finger to the crease between the index finger and palm.

Middle Length in mm from tip of middle finger to the crease between the middle finger and palm.

Ring Length in mm from tip of ring finger to the crease between the middle finger and palm.

Pinkie Length in mm from tip of pinkie finger to the crease between the pinkie finger and palm

Height Height in inches.

Weight Weight in pounds.

`generate_models` *Generate a List of Models for Computing Different Types of Sums of Squares*

Description

This function will return a list of lists where the top-level keys (names) of the items indicate the component of the full model (i.e. the term) that the generated models can be used to test. At each of these keys is a list with both the `complex` and simple models that can be compared to test the component. The `complex` models always include the target term, and the `simple` models are identical to the `complex` except the target term is removed. Thus, when the models are compared (e.g. using `anova`, except for Type III; see details below), the resulting values will show the effect of adding the target term to the model. There are three generally used approaches to determining what the appropriate comparison models should be, called Type I, II, and III. See the sections below for more information on these types.

Usage

```
generate_models(model, type = 3)

## S3 method for class 'formula'
generate_models(model, type = 3)

## S3 method for class 'lm'
generate_models(model, type = 3)
```

Arguments

`model`	The model to generate the models from, of the type `lm()`, `aov()`, or `formula()`.
`type`	The type of sums of squares to calculate: - Use `1`, `I`, and `sequential` for Type I. - Use `2`, `II`, and `hierarchical` for Type II. - Use `3`, `III`, and `orthogonal` for Type III.

Value

A list of the augmented models for each term, where the associated term is the key for each model in the list.

Type I

For Type I SS, or sequential SS, each term is considered in order after the preceding terms are considered. Consider the example model

`Y ~ A + B + A:B`

, where ":" indicates an interaction. To determine the Type I effect of `A`, we would compare the model `Y ~ A` to the same model without the term: `Y ~ NULL`. For `B`, we compare `Y ~ A + B` to `Y ~ A`; and for `A:B`, we compare `Y ~ A + B + A:B` to `Y ~ A + B`. Incidentally, the `anova()` function that ships with the base installation of R computes Type I statistics.

Type II

For Type II SS, or hierarchical SS, each term is considered in the presence of all of the terms that do not include it. For example, consider an example three-way factorial model

`Y ~ A + B + C + A:B + A:C + B:C + A:B:C`

, where ":" indicates an interaction. The effect of `A` is found by comparing `Y ~ B + C + B:C + A` to `Y ~ B + C + B:C` (the only terms included are those that do not include `A`). For `B`, the comparison models would be `Y ~ A + C + A:C + B` and `Y ~ A + C + A:C`; for `A:B`, the models would be `Y ~ A + B + C + A:C + B:C + A:B` and `Y ~ A + B + C + A:C + B:C`; and so on.

Type III

For Type III SS, or orthogonal SS, each term is considered in the presence of all of the other terms. For example, consider an example two-way factorial model

`Y ~ A + B + A:B`

, where `:` indicates an interaction between the terms. The effect of `A`, is found by comparing `Y ~ B + A:B + A` to `Y ~ B + A:B`; for `B`, the comparison models would be `Y ~ A + A:B + B` and `Y ~ A + A:B`; and for `A:B`, the models would be `Y ~ A + B + A:B` and `Y ~ A + B`.

Unfortunately, `anova()` cannot be used to compare Type III models. `anova()` does not allow for violation of the principle of marginality, which is the rule that interactions should only be tested in the context of their lower order terms. When an interaction term is present in a model, `anova()` will automatically add in the lower-order terms, making a model like `Y ~ A + A:B` unable to be compared: it will add the lower-order term `B`,and thus use the model `Y ~ A + B + A:B` instead. To get the appropriate statistics for Type III comparisons, use `drop1()` with the full scope, i.e. `drop1(model_fit,scope = . ~ .)`.

Examples

```
# create all type 2 comparison models
mod <- lm(Thumb ~ Height * Sex, data = Fingers)
mods_2 <- generate_models(mod, type = 2)

# compute the SS for the Height term
mod_Height <- anova(mods_2[["Height"]]$simple, mods_2[["Height"]]$complex)
mod_Height[["Sum of Sq"]][[2]]
```

`listwise_delete` *Remove cases with missing values.*

Description

Remove cases with missing values.

Usage

```
listwise_delete(obj, vars)

## S3 method for class 'data.frame'
listwise_delete(obj, vars = names(obj))

## S3 method for class 'lm'
listwise_delete(obj, vars = all.vars(formula(obj)))
```

Arguments

obj	The data.frame or lm object to process.
vars	The variables to consider.

Value

For data.frames, the vars are checked for missing values. If one is found on any of the variables, the entire row is removed (list-wise deletion). For linear models, the model is refit after the underlying data have been processed.

---

number number *vector*

Description

This creates a formatted double vector. You can specify the number of digits you want the value to display after the decimal, and the underlying value will not change. Additionally you can explicitly set whether scientific notation should be used and if numbers less than 0 should contain a leading 0.

Usage

```
number(x = numeric(), digits = 3L, scientific = FALSE, leading_zero = TRUE)

is_number(x)

as_number(x)
```

Arguments

x	• For number(): A numeric vector – For is_number(): An object to test – For as_number(): An object to coerce to a number
digits	The number of digits to display after the decimal point.
scientific	Whether the number should be represented with scientific notation (e.g. 1e2)
leading_zero	Whether a leading zero should be used on numbers less than 0 (e.g. .001)

Value

An S3 vector of class `supernova_number`. It should behave like a double, but be formatted consistently.

Examples

```
number(1:5, digits = 3)
```

`pairwise` *Compute all pairwise comparisons between category levels*

Description

This function is useful for generating and testing all pairwise comparisons of categorical terms in a linear model. This can be done in base R using functions like `pairwise.t.test` and `TukeyHSD`, but these functions are inconsistent both in their output format and their general approach to pairwise comparisons. `pairwise()` will return a consistent table format, and will make consistent decisions about how to calculate error terms and confidence intervals. See the **Details** section low for more on how the models are tested (and why your output might not match other functions).

Usage

```
pairwise(
  fit,
  correction = "Tukey",
  term = NULL,
  alpha = 0.05,
  var_equal = TRUE,
  plot = FALSE
)

pairwise_t(fit, term = NULL, alpha = 0.05, correction = "none")

pairwise_bonferroni(fit, term = NULL, alpha = 0.05)

pairwise_tukey(fit, term = NULL, alpha = 0.05)
```

Arguments

`fit`	A model fit by `lm()` or `aov()` (or similar).
`correction`	The type of correction (if any) to perform to maintain the family-wise error-rate specified by `alpha`: - **Tukey**: computes Tukey's Honestly Significant Differences (see `TukeyHSD()`) - **Bonferroni**: computes pairwise *t*-tests and then apply a Bonferroni correction - **none**: computes pairwise *t*-tests and reports the uncorrected statistics
`term`	If `NULL`, use each categorical term in the model. Otherwise, only use the given term.

alpha	The family-wise error-rate to restrict the tests to. If "none" is given for correction, this value is the value for each test (and is used to calculate the family-wise error-rate for the group of tests).
var_equal	If TRUE (default), treat the variances between each group as being equal, otherwise the Welch or Satterthwaite method is used to appropriately weight the variances. **Note:**, currently only TRUE is supported. Alternative methods forthcoming.
plot	Setting plot to TRUE will automatically call plot on the returned object.

Details

For simple one-way models where a single categorical variable predicts and outcome. You will get output similar to other methods of computing pairwise comparisons. Essentially, the differences on the outcome between each of the groups defined by the categorical variable are compared with the requested test, and their confidence intervals and p-values are adjusted by the requested correction.

However, when more than two variables are entered into the model, the outcome will diverge somewhat from other methods of computing pairwise comparisons. For traditional pairwise tests you need to estimate an error term, usually by pooling the standard deviation of the groups being compared. This means that when you have other predictors in the model, their presence is ignored when running these tests. For the functions in this package, we instead compute the pooled standard error by using the mean squared error (MSE) from the full model fit.

Let's take a concrete example to explain that. If we are predicting Thumb length from Sex, we can create that linear model and get the pairwise comparisons like this:

```
pairwise(lm(Thumb ~ Sex,data = supernova::Fingers))
```

The output of this code will have one table showing the comparison of males and females on thumb length. The pooled standard error is the same as the square root of the MSE from the full model.

In these data the Sex variable did not have any other values than *male* and *female*, but we can imagine situations where the data had other values like *other* or more refined responses. In these cases, the pooled SD would be calculated by taking the MSE of the full model (not of each group) and then weighting it based on the size of the groups in question (divide by n).

To improve our model, we might add Height as quantitative predictor:

```
pairwise(lm(Thumb ~ Sex + Height,data = supernova::Fingers))
```

Note that the output still only has a table for Sex. This is because we can't do a pairwise comparison using Height because there are no groups to compare. Most functions will drop or not let you use this variable during pairwise comparisons. Instead, pairwise() uses the same approach as in the 3+ groups situation: we use the MSE for the full model and then weight it by the size of the groups being compared. Because we are using the MSE for the full model, the effect of Height is accounted for in the error term even though we are not explicitly comparing different heights. **Importantly**, the interpretation of the outcome is different than in other traditional t-tests. Instead of saying, "there is a difference in thumb length based on the value of sex," we must add that this difference is found "after accounting for height."

Value

A list of tables organized by the terms in the model. For each term (categorical terms only, as splitting on a continuous variable is generally uninformative), the table describes all of the pairwise-comparisons possible.

`Servers` *Servers data*

Description

Data about tips collected from an experiment with 44 servers at a restaurant.

Usage

```
Servers
```

Format

A data frame with 44 observations on the following 2 variables.

ServerID A number assigned to each server.

Tip How much the tip was.

Details

Note: these data will be removed in future versions in favor of `Tables`.

`supernova` *supernova*

Description

An alternative set of summary statistics for ANOVA. Sums of squares, degrees of freedom, mean squares, and F value are all computed with Type III sums of squares, but for fully-between subjects designs you can set the type to I or II. This function adds to the output table the proportional reduction in error, an explicit summary of the whole model, separate formatting of p values, and is intended to match the output used in Judd, McClelland, and Ryan (2017).

Usage

```
supernova(fit, type = 3, verbose = TRUE)

## S3 method for class 'lm'
supernova(fit, type = 3, verbose = TRUE)

## S3 method for class 'lmerMod'
supernova(fit, type = 3, verbose = FALSE)

superanova(fit, type = 3, verbose = TRUE)
```

Arguments

fit	A model fit by lm() or lme4::lmer()
type	The type of sums of squares to calculate (see generate_models()). Defaults to the widely used Type III SS.
verbose	If FALSE, the description column is suppressed.

Details

superanova() is an alias of supernova()

Value

An object of the class supernova, which has a clean print method for displaying the ANOVA table in the console as well as a named list:

tbl	The ANOVA table as a data.frame
fit	The original lm or lmer object being tested
models	Models created by generate_models

References

Judd, C. M., McClelland, G. H., & Ryan, C. S. (2017). *Data Analysis: A Model Comparison Approach to Regression, ANOVA, and Beyond* (3rd ed.). New York: Routledge. ISBN:879-1138819832

Examples

```
supernova(lm(Thumb ~ Weight, data = Fingers))
format_p <- supernova(lm(Thumb ~ Weight, data = Fingers))
print(format_p, pcut = 8)
```

Survey — *Students at a university were asked to enter a random number between 1-20 into a survey.*

Description

Students at a university taking an introductory statistics course were asked to complete this survey as part of their homework.

Usage

```
Survey
```

Format

A data frame with 211 observations on the following 1 variable:

Any1_20 The random number between 1 and 20 that a student thought of.

Tables *Tables data*

Description

Data about tips collected from an experiment with 44 tables at a restaurant.

Usage

```
Tables
```

Format

A data frame with 44 observations on the following 2 variables.

TableID A number assigned to each table.

Tip How much the tip was.

TipExperiment *Data from an experiment about smiley faces and tips*

Description

Tables were randomly assigned to receive checks that either included or did not include a drawing of a smiley face. Data was collected from 44 tables in an effort to examine whether the added smiley face would cause more generous tipping.

Usage

```
TipExperiment
```

Format

A data frame with 44 observations on the following 3 variables.

TableID A number assigned to each table.

Tip How much the tip was.

Condition Which experimental condition the table was randomly assigned to.

update_in_env *Update a model in the environment the model was created in*

Description

`stats::update()` will perform the update in `parent.frame()` by default, but this can cause problems when the update is called by another function (so the parent frame is no longer the environment the user is in).

Usage

```
update_in_env(object, formula., ...)
```

Arguments

`object`	An existing fit from a model function such as `lm`, `glm` and many others.
`formula.`	Changes to the formula – see `update.formula` for details.
`...`	Additional arguments to the call, or arguments with changed values. Use `name = NULL` to remove the argument `name`.

Value

The updated model is returned.

variables *Extract the variables from a model formula*

Description

Extract the variables from a model formula

Usage

```
variables(object)

## S3 method for class 'supernova'
variables(object)

## S3 method for class 'formula'
variables(object)

## S3 method for class 'lm'
variables(object)

## S3 method for class 'lmerMod'
variables(object)
```

Arguments

object	A formula, lm or supernova object

Value

A list containing the outcome and predictor variables in the model.

Index