

# Package ‘groupedstats’

December 2, 2020

**Type** Package

**Title** Grouped Statistical Analyses in a Tidy Way

**Version** 2.0.0

**Maintainer** Indrajeet Patil <patilindrajeet.science@gmail.com>

**Description** Collection of functions to run statistical tests  
across all combinations of multiple grouping variables.

**License** GPL-3 | file LICENSE

**URL** <https://indrajeetpatil.github.io/groupedstats/>,  
<https://github.com/IndrajeetPatil/groupedstats/>

**BugReports** <https://github.com/IndrajeetPatil/groupedstats/issues/>

**Depends** R (>= 3.6.0)

**Imports** broomExtra, dplyr, effectsize, glue, lme4, magrittr,  
parameters, purrr, rlang, skimr, stats, tibble, tidyr

**Suggests** gapminder, ggplot2, knitr, rmarkdown

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** Indrajeet Patil [aut, cre, cph]  
(<<https://orcid.org/0000-0003-1995-6531>>, @patilindrajeets)

**Repository** CRAN

**Date/Publication** 2020-12-01 23:00:03 UTC

## R topics documented:

grouped_aov	2
grouped_glm	3
grouped_glmer	4

grouped_lm . . . . .	6
grouped_lmer . . . . .	7
grouped_proptest . . . . .	9
grouped_slr . . . . .	10
grouped_summary . . . . .	11
grouped_ttest . . . . .	12
grouped_wilcox . . . . .	14
lm_effsize_ci . . . . .	15
lm_effsize_standardizer . . . . .	16

<b>Index</b>	<b>18</b>
--------------	-----------

---

grouped_aov	<i>Running analysis of variance (aov) across multiple grouping variables.</i>
-------------	---

---

## Description

Running analysis of variance (aov) across multiple grouping variables.

## Usage

```
grouped_aov(
  data,
  grouping.vars,
  formula,
  effsize = "eta",
  output = "tidy",
  ...
)
```

## Arguments

data	A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way.
grouping.vars	Grouping variables.
formula	A formula specifying the model.
effsize	Character describing the effect size to be displayed: "eta" (default) or "omega".
output	A character describing what output is expected. Two possible options: "tidy" (default), which will return the results, or "tukey", which will return results from Tukey's Honest Significant Differences method for <i>post hoc</i> comparisons. The "glance" method to get model summary is currently not supported for this function.
...	Currently ignored.

**Examples**

```
# uses dataset included in the `groupedstats` package
set.seed(123)
library(groupedstats)

# effect size
groupedstats::grouped_aov(
  formula = wt ~ mpg,
  data = mtcars,
  grouping.vars = am,
  effsize = "eta"
)
```

---

grouped_glm	<i>Function to run generalized linear model (glm) across multiple grouping variables.</i>
-------------	---

---

**Description**

Function to run generalized linear model (glm) across multiple grouping variables.

**Usage**

```
grouped_glm(
  data,
  grouping.vars,
  ...,
  output = "tidy",
  tidy.args = list(conf.int = TRUE, conf.level = 0.95),
  augment.args = list()
)
```

**Arguments**

data	Dataframe (or tibble) from which variables are to be taken.
grouping.vars	Grouping variables.
...	Additional arguments to broom::tidy, broom::glance, or broom::augment S3 method.
output	A character describing what output is expected. Two possible options: "tidy" (default), which will return the results, or "glance", which will return model summaries.
tidy.args	A list of arguments to be used in the relevant S3 method.
augment.args	A list of arguments to be used in the relevant S3 method.

**Value**

A tibble dataframe with tidy results from linear model.

**See Also**

[grouped\\_lm](#), [grouped\\_lmer](#), [grouped\\_glmer](#)

**Examples**

```
groupedstats::grouped_glm(
  data = mtcars,
  formula = am ~ wt,
  grouping.vars = cyl,
  family = stats::binomial(link = "logit")
)
```

---

grouped_glmer	<i>Function to run generalized linear mixed-effects model (glmer) across multiple grouping variables.</i>
---------------	---

---

**Description**

Function to run generalized linear mixed-effects model (glmer) across multiple grouping variables.

**Usage**

```
grouped_glmer(
  data,
  grouping.vars,
  ...,
  output = "tidy",
  tidy.args = list(conf.int = TRUE, conf.level = 0.95, effects = "fixed", conf.method =
    "Wald"),
  augment.args = list()
)
```

**Arguments**

data	Dataframe (or tibble) from which variables are to be taken.
grouping.vars	Grouping variables.
...	Arguments passed on to <a href="#">lme4::glmer</a>
formula	a two-sided linear formula object describing both the fixed-effects and random-effects part of the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right. Random-effects terms are distinguished by vertical bars (" ") separating expressions for design matrices from grouping factors.

- family** a GLM family, see [glm](#) and [family](#).
- start** a named list of starting values for the parameters in the model, or a numeric vector. A numeric `start` argument will be used as the starting value of `theta`. If `start` is a list, the `theta` element (a numeric vector) is used as the starting value for the first optimization step (default=1 for diagonal elements and 0 for off-diagonal elements of the lower Cholesky factor); the fitted value of `theta` from the first step, plus `start[["fixef"]]`, are used as starting values for the second optimization step. If `start` has both `fixef` and `theta` elements, the first optimization step is skipped. For more details or finer control of optimization, see [modular](#).
- verbose** integer scalar. If  $> 0$  verbose output is generated during the optimization of the parameter estimates. If  $> 1$  verbose output is generated during the individual penalized iteratively reweighted least squares (PIRLS) steps.
- nAGQ** integer scalar - the number of points per axis for evaluating the adaptive Gauss-Hermite approximation to the log-likelihood. Defaults to 1, corresponding to the Laplace approximation. Values greater than 1 produce greater accuracy in the evaluation of the log-likelihood at the expense of speed. A value of zero uses a faster but less exact form of parameter estimation for GLMMs by optimizing the random effects and the fixed-effects coefficients in the penalized iteratively reweighted least squares step. (See Details.)
- subset** an optional expression indicating the subset of the rows of data that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
- weights** an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
- na.action** a function that indicates what should happen when the data contain NAs. The default action (`na.omit`, inherited from the 'factory fresh' value of `getOption("na.action")`) strips any observations with any missing values in any variables.
- offset** this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See [model.offset](#).
- contrasts** an optional list. See the `contrasts.arg` of [model.matrix.default](#).
- mustart** optional starting values on the scale of the conditional mean, as in [glm](#); see there for details.
- etastart** optional starting values on the scale of the unbounded predictor as in [glm](#); see there for details.
- devFunOnly** logical - return only the deviance evaluation function. Note that because the deviance function operates on variables stored in its environment, it may not return *exactly* the same values on subsequent calls (but the results should always be within machine tolerance).

output	A character describing what output is expected. Two possible options: "tidy" (default), which will return the results, or "glance", which will return model summaries.
tidy.args	A list of arguments to be used in the relevant S3 method.
augment.args	A list of arguments to be used in the relevant S3 method.

**Value**

A tibble dataframe with tidy results from linear model or model summaries.

---

grouped_lm	<i>Running linear model (lm) across multiple grouping variables.</i>
------------	--

---

**Description**

Running linear model (lm) across multiple grouping variables.

**Usage**

```
grouped_lm(
  data,
  grouping.vars,
  ...,
  output = "tidy",
  tidy.args = list(conf.int = TRUE, conf.level = 0.95),
  augment.args = list()
)
```

**Arguments**

data	Dataframe (or tibble) from which variables are to be taken.
grouping.vars	Grouping variables.
...	Additional arguments to broom::tidy, broom::glance, or broom::augment S3 method.
output	A character describing what output is expected. Two possible options: "tidy" (default), which will return the results, or "glance", which will return model summaries.
tidy.args	A list of arguments to be used in the relevant S3 method.
augment.args	A list of arguments to be used in the relevant S3 method.

**Value**

A tibble dataframe with tidy results from linear model.

**See Also**[grouped\\_slr](#), [grouped\\_tidy](#)**Examples**

```
# loading needed libraries
library(ggplot2)

# getting tidy output of results
grouped_lm(
  data = mtcars,
  grouping.vars = cyl,
  formula = mpg ~ am * wt,
  output = "tidy"
)

# getting model summaries
# diamonds dataset from ggplot2
grouped_lm(
  data = diamonds,
  grouping.vars = c(cut, color),
  formula = price ~ carat * clarity,
  output = "glance"
)
```

---

`grouped_lmer`*Linear mixed-effects model (lmer) across multiple grouping variables.*

---

**Description**

Linear mixed-effects model (lmer) across multiple grouping variables.

**Usage**

```
grouped_lmer(
  data,
  grouping.vars,
  ...,
  output = "tidy",
  tidy.args = list(conf.int = TRUE, conf.level = 0.95, effects = "fixed", conf.method =
    "Wald"),
  augment.args = list()
)
```

**Arguments**

<code>data</code>	Dataframe (or tibble) from which variables are to be taken.
<code>grouping.vars</code>	Grouping variables.
<code>...</code>	Arguments passed on to <code>lme4::lmer</code>
<code>formula</code>	a two-sided linear formula object describing both the fixed-effects and random-effects part of the model, with the response on the left of a <code>~</code> operator and the terms, separated by <code>+</code> operators, on the right. Random-effects terms are distinguished by vertical bars ( <code> </code> ) separating expressions for design matrices from grouping factors. Two vertical bars ( <code>  </code> ) can be used to specify multiple uncorrelated random effects for the same grouping variable. (Because of the way it is implemented, the <code>  </code> -syntax <i>works only for design matrices containing numeric (continuous) predictors</i> ; to fit models with independent categorical effects, see <code>dummy</code> or the <code>lmer_alt</code> function from the <b>afex</b> package.)
<code>REML</code>	logical scalar - Should the estimates be chosen to optimize the REML criterion (as opposed to the log-likelihood)?
<code>start</code>	a named <code>list</code> of starting values for the parameters in the model. For <code>lmer</code> this can be a numeric vector or a list with one component named <code>"theta"</code> .
<code>verbose</code>	integer scalar. If <code>&gt; 0</code> verbose output is generated during the optimization of the parameter estimates. If <code>&gt; 1</code> verbose output is generated during the individual penalized iteratively reweighted least squares (PIRLS) steps.
<code>subset</code>	an optional expression indicating the subset of the rows of data that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be included, or a character vector of the row names to be included. All observations are included by default.
<code>weights</code>	an optional vector of 'prior weights' to be used in the fitting process. Should be <code>NULL</code> or a numeric vector. Prior weights are <i>not</i> normalized or standardized in any way. In particular, the diagonal of the residual covariance matrix is the squared residual standard deviation parameter <code>sigma</code> times the vector of inverse weights. Therefore, if the weights have relatively large magnitudes, then in order to compensate, the <code>sigma</code> parameter will also need to have a relatively large magnitude.
<code>na.action</code>	a function that indicates what should happen when the data contain NAs. The default action ( <code>na.omit</code> , inherited from the 'factory fresh' value of <code>getOption("na.action")</code> ) strips any observations with any missing values in any variables.
<code>offset</code>	this can be used to specify an <i>a priori</i> known component to be included in the linear predictor during fitting. This should be <code>NULL</code> or a numeric vector of length equal to the number of cases. One or more <code>offset</code> terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See <code>model.offset</code> .
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>devFunOnly</code>	logical - return only the deviance evaluation function. Note that because the deviance function operates on variables stored in its environ-



ment, it may not return *exactly* the same values on subsequent calls (but the results should always be within machine tolerance).

output	A character describing what output is expected. Two possible options: "tidy" (default), which will return the results, or "glance", which will return model summaries.
tidy.args	A list of arguments to be used in the relevant S3 method.
augment.args	A list of arguments to be used in the relevant S3 method.

### Value

A tibble dataframe with tidy results from a linear mixed-effects model. Note that  $p$ -value is computed using `parameters::p_value`.

### Examples

```
# for reproducibility
set.seed(123)

# loading libraries containing data
library(gapminder)

# getting tidy output of results
# let's use only subset of the data
groupedstats::grouped_lmer(
  data = gapminder,
  formula = scale(lifeExp) ~ scale(gdpPercap) + (gdpPercap | continent),
  grouping.vars = year,
  REML = FALSE,
  tidy.args = list(effects = "fixed", conf.int = TRUE, conf.level = 0.95),
  output = "tidy"
)
```

---

grouped\_proptest      *Function to run proportion test on grouped data.*

---

### Description

Function to run proportion test on grouped data.

### Usage

```
grouped_proptest(data, grouping.vars, measure, ...)
```

**Arguments**

data	Dataframe (or tibble) from which variables are to be taken.
grouping.vars	Grouping variables.
measure	A variable for which proportion test needs to be carried out for each combination of levels of factors entered in grouping.vars.
...	Currently ignored.

**Value**

Dataframe with percentages and statistical details from a proportion test.

**Examples**

```
# for reproducibility
set.seed(123)

groupedstats::grouped_proptest(
  data = mtcars,
  grouping.vars = cyl,
  measure = am
)
```

---

grouped_slr	<i>Running simple linear regression (slr) on multiple variables across multiple grouping variables.</i>
-------------	---

---

**Description**

Running simple linear regression (slr) on multiple variables across multiple grouping variables.

**Usage**

```
grouped_slr(data, dep.vars, indep.vars, grouping.vars)
```

**Arguments**

data	Dataframe from which variables are to be taken.
dep.vars	List criterion or <b>dependent</b> variables for simple linear model ( $y$ in $y \sim x$ ).
indep.vars	List predictor or <b>independent</b> variables for simple linear model ( $x$ in $y \sim x$ ).
grouping.vars	List of grouping variables.

**Value**

A tibble dataframe with tidy results from simple linear regression analyses. The estimates are standardized, i.e. the lm model used is  $\text{scale}(y) \sim \text{scale}(x)$ , and not  $y \sim x$ .

**See Also**[grouped\\_lm](#), [grouped\\_tidy](#)**Examples**

```
# for reproducibility
set.seed(123)

# in case of just one grouping variable
groupedstats::grouped_slr(
  data = iris,
  dep.vars = c(Sepal.Length, Petal.Length),
  indep.vars = c(Sepal.Width, Petal.Width),
  grouping.vars = Species
)
```

---

grouped_summary	<i>Descriptive statistics for multiple variables for all grouping variable levels</i>
-----------------	---

---

**Description**

Descriptive statistics for multiple variables for all grouping variable levels

**Usage**

```
grouped_summary(
  data,
  grouping.vars,
  measures = NULL,
  measures.type = "numeric",
  topcount.long = FALSE,
  k = 2L,
  ...
)
```

**Arguments**

data	Dataframe from which variables need to be taken.
grouping.vars	A list of grouping variables. Please use unquoted arguments (i.e., use <code>x</code> and not <code>"x"</code> ).
measures	List variables for which summary needs to be computed. If not specified, all variables of type specified in the argument <code>measures.type</code> will be used to calculate summaries. <b>Don't</b> explicitly set <code>measures.type = NULL</code> in function call, which will produce an error because the function will try to find a column in a dataframe named "NULL".

<code>measures.type</code>	A character indicating whether summary for <i>numeric</i> ("numeric") or <i>factor/character</i> ("factor") variables is expected (Default: <code>measures.type = "numeric"</code> ). This function can't be used for both numeric <b>and</b> variables simultaneously.
<code>topcount.long</code>	If <code>measures.type = factor</code> , you can get the top counts in long format for plotting purposes. (Default: <code>topcount.long = FALSE</code> ).
<code>k</code>	Number of digits after decimal point (should be an integer) (Default: <code>k = 3L</code> ).
<code>...</code>	Currently ignored.

**Value**

Dataframe with descriptive statistics for numeric variables (n, mean, sd, median, min, max).

**Examples**

```
# for reproducibility
set.seed(123)

# another possibility
groupedstats::grouped_summary(
  data = iris,
  grouping.vars = Species,
  measures = Sepal.Length:Petal.Width,
  measures.type = "numeric"
)

# if no measures are chosen, all relevant columns will be summarized
groupedstats::grouped_summary(
  data = ggplot2::msleep,
  grouping.vars = vore,
  measures.type = "factor"
)

# for factors, you can also convert the dataframe to a long format with counts
groupedstats::grouped_summary(
  data = ggplot2::msleep,
  grouping.vars = c(vore),
  measures = c(genus:order),
  measures.type = "factor",
  topcount.long = TRUE
)
```

---

<code>grouped_ttest</code>	<i>Function to run t-test on multiple variables across multiple grouping variables.</i>
----------------------------	---

---

**Description**

Function to run *t*-test on multiple variables across multiple grouping variables.

**Usage**

```
grouped_ttest(  
  data,  
  dep.vars,  
  indep.vars,  
  grouping.vars,  
  paired = FALSE,  
  var.equal = FALSE  
)
```

**Arguments**

<code>data</code>	Dataframe from which variables are to be taken.
<code>dep.vars</code>	List dependent variables for a $t$ -test ( $y$ in $y \sim x$ ).
<code>indep.vars</code>	List independent variables for a $t$ -test ( $x$ in $y \sim x$ ).
<code>grouping.vars</code>	List of grouping variables.
<code>paired</code>	A logical indicating whether you want a paired $t$ -test (Default: <code>paired = FALSE</code> ; independent $t$ -test, i.e.).
<code>var.equal</code>	A logical variable indicating whether to treat the two variances as being equal. If <code>TRUE</code> , then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used (Default: <code>var.equal = FALSE</code> ; Welch's $t$ -test, i.e.).

**Value**

A tibble dataframe with tidy results from  $t$ -test analyses.

**Examples**

```
# for reproducibility  
set.seed(123)  
  
groupedstats::grouped_ttest(  
  data = dplyr::filter(.data = ggplot2::diamonds, color == "E" | color == "J"),  
  dep.vars = c(carat, price, depth),  
  indep.vars = color,  
  grouping.vars = clarity,  
  paired = FALSE,  
  var.equal = FALSE  
)
```

---

grouped_wilcox	<i>Function to run two-sample Wilcoxon tests on multiple variables across multiple grouping variables.</i>
----------------	--

---

### Description

Function to run two-sample Wilcoxon tests on multiple variables across multiple grouping variables.  
Running Wilcox test across multiple grouping variables.

### Usage

```
grouped_wilcox(  
  data,  
  dep.vars,  
  indep.vars,  
  grouping.vars,  
  paired = FALSE,  
  correct = TRUE  
)
```

### Arguments

data	Dataframe from which variables are to be taken.
dep.vars	List dependent variables for a two-sample Wilcoxon tests ( $y$ in $y \sim x$ ).
indep.vars	List independent variables for a two-sample Wilcoxon tests ( $x$ in $y \sim x$ ).
grouping.vars	List of grouping variables (if NULL, the entire dataframe will be used).
paired	A logical indicating whether you want a paired two-sample Wilcoxon tests (Default: paired = FALSE).
correct	A logical indicating whether to apply continuity correction in the normal approximation for the p-value (Default: correct = TRUE).

### Value

A tibble dataframe with tidy results from two-sample Wilcoxon tests analyses.

### See Also

[grouped\\_tidy](#)

### Examples

```
# for reproducibility  
set.seed(123)  
  
# only with one grouping variable  
groupedstats::grouped_wilcox(  
  data = iris,  
  dep.vars = "sepal.length",  
  indep.vars = "sepal.width",  
  grouping.vars = "species",  
  paired = FALSE,  
  correct = TRUE  
)
```

```

data = dplyr::filter(.data = ggplot2::diamonds, color == "E" | color == "J"),
dep.vars = depth:table,
indep.vars = color,
grouping.vars = clarity,
paired = FALSE
)

```

---

lm_effsize_ci	<i>Confidence intervals for (partial) eta-squared and omega-squared for linear models.</i>
---------------	--

---

### Description

This function will convert a linear model object to a dataframe containing statistical details for all effects along with effect size measure and its confidence interval. For more details, see `effectsize::eta_squared` and `effectsize::omega_squared`.

### Usage

```
lm_effsize_ci(object, effsize = "eta", partial = TRUE, conf.level = 0.95, ...)
```

### Arguments

object	The linear model object (can be of class <code>lm</code> , <code>aov</code> , <code>anova</code> , or <code>aovlist</code> ).
effsize	Character describing the effect size to be displayed: "eta" (default) or "omega".
partial	Logical that decides if partial eta-squared or omega-squared are returned (Default: TRUE). If FALSE, eta-squared or omega-squared will be returned. Valid only for objects of class <code>lm</code> , <code>aov</code> , <code>anova</code> , or <code>aovlist</code> .
conf.level	Numeric specifying Level of confidence for the confidence interval (Default: 0.95).
...	Ignored.

### Value

A dataframe with results from `stats::lm()` with partial eta-squared, omega-squared, and bootstrapped confidence interval for the same.

### Author(s)

Indrajeet Patil

**Examples**

```
# for reproducibility
set.seed(123)

# model
mod <-
  stats::aov(
    formula = mpg ~ wt + qsec + Error(displ / am),
    data = mtcars
  )

# dataframe with effect size and confidence intervals
groupedstats::lm_effsize_ci(mod)
```

---

lm\_effsize\_standardizer

*Standardize a dataframe with effect sizes for aov, lm, aovlist, etc. objects.*

---

**Description**

The difference between `lm_effsize_ci` and `lm_effsize_standardizer` is that the former has more opinionated column naming, while the latter doesn't. The latter can thus be more helpful in writing a wrapper around this function.

**Usage**

```
lm_effsize_standardizer(
  object,
  effsize = "eta",
  partial = TRUE,
  conf.level = 0.95,
  ...
)
```

**Arguments**

<code>object</code>	The linear model object (can be of class <code>lm</code> , <code>aov</code> , <code>anova</code> , or <code>aovlist</code> ).
<code>effsize</code>	Character describing the effect size to be displayed: "eta" (default) or "omega".
<code>partial</code>	Logical that decides if partial eta-squared or omega-squared are returned (Default: TRUE). If FALSE, eta-squared or omega-squared will be returned. Valid only for objects of class <code>lm</code> , <code>aov</code> , <code>anova</code> , or <code>aovlist</code> .
<code>conf.level</code>	Numeric specifying Level of confidence for the confidence interval (Default: 0.95).
<code>...</code>	Ignored.



**Examples**

```
set.seed(123)
groupedstats::lm_effsize_standardizer(
  object = stats::lm(formula = brainwt ~ vore, data = ggplot2::msleep),
  effsize = "eta",
  partial = FALSE,
  conf.level = 0.99
)
```

# Index

`dummy`, 8

`family`, 5

`glm`, 5

`grouped_aov`, 2

`grouped_glm`, 3

`grouped_glmer`, 4, 4

`grouped_lm`, 4, 6, 11

`grouped_lmer`, 4, 7

`grouped_proptest`, 9

`grouped_slr`, 7, 10

`grouped_summary`, 11

`grouped_tidy`, 7, 11, 14

`grouped_ttest`, 12

`grouped_wilcox`, 14

`list`, 8

`lm_effsize_ci`, 15

`lm_effsize_standardizer`, 16

`lme4::glmer`, 4

`lme4::lmer`, 8

`model.matrix.default`, 5

`model.offset`, 5, 8

`modular`, 5

`offset`, 5, 8

`sigma`, 8