

Package ‘finalfit’

September 3, 2025

Type Package

Title Quickly Create Elegant Regression Results Tables and Plots when Modelling

Version 1.1.0

Maintainer Ewen Harrison <ewen.harrison@ed.ac.uk>

Description Generate regression results tables and plots in final format for publication. Explore models and export directly to PDF and 'Word' using 'RMarkdown'.

License MIT + file LICENCE

Encoding UTF-8

LazyData true

BugReports <https://github.com/ewenharrison/finalfit/issues>

URL <https://github.com/ewenharrison/finalfit>

Imports bdsmatrix, boot, broom, cowplot, dplyr, forcats, GGally, ggplot2, lme4, magrittr, mice (>= 3.17.0), pillar, pROC, purrr, scales, stats, stringr, survival, tidyr (>= 1.0.0),

RoxxygenNote 7.3.2

Suggests cmprsk, coxme, Hmisc, knitr, lmtest, readr, rlang, rmarkdown, rstan, sandwich, survey, survminer, testthat, tibble

VignetteBuilder knitr

NeedsCompilation no

Author Ewen Harrison [aut, cre],
Tom Drake [aut],
Riinu Pius [aut]

Repository CRAN

Date/Publication 2025-09-03 19:00:02 UTC

Contents

boot_compare	3
boot_predict	4
check_recode	7
coefficient_plot	8
colon_s	10
coxphmulti	10
coxphuni	11
crrmulti	12
crruni	14
dependent_label	15
extract_variable_label	17
ff_column_totals	17
ff_expand	19
ff_formula	20
ff_glimpse	20
ff_interaction	21
ff_label	22
ff_merge	23
ff_metrics	25
ff_mode	27
ff_newdata	28
ff_parse_formula	29
ff_percent_only	30
ff_permute	30
ff_plot	32
ff_relabel	33
ff_relabel_df	34
ff_remove_p	35
ff_remove_ref	35
ff_row_totals	36
ff_stratify_helper	38
finalfit	39
fit2df	43
format_n_percent	51
glmmixed	52
glmmulti	53
glmmulti_boot	54
glmuni	55
hr_plot	56
labels_to_column	58
labels_to_level	59
lmmixed	59
lmmulti	61
lmuni	62
metrics_hoslem	63
missing_compare	64

missing_glimpse	65
missing_pairs	65
missing_pattern	66
missing_plot	67
missing_predictorMatrix	68
or_plot	70
p_tidy	71
rm_duplicates	72
rm_empty_block	73
round_tidy	73
summary_df	74
summary_factorlist	75
summary_factorlist_stratified	78
surv_plot	79
svyglmulti	80
svyglmuni	82
wcgs	84

Index	86
--------------	-----------

boot_compare	<i>Compare bootstrapped distributions</i>
--------------	---

Description

Not usually called directly. Included in `boot_predict`. Usually used in combination with A function that takes the output from `summary_factorlist(..., fit_id=TRUE)` and merges with any number of model dataframes, usually produced with a model wrapper followed by the `fit2df()` function (see examples).

Usage

```
boot_compare(
  bs.out,
  confint_level = 0.95,
  confint_sep = " to ",
  comparison = "difference",
  condense = TRUE,
  compare_name = NULL,
  digits = c(2, 3),
  ref_symbol = 1
)
```

Arguments

bs.out	Output from <code>boot::boot</code> ,
--------	---------------------------------------

confint_level	The confidence level to use for the confidence interval. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
confint_sep	String separating lower and upper confidence interval
comparison	Either "difference" or "ratio".
condense	Logical. FALSE gives numeric values, usually for plotting. TRUE gives table for final output.
compare_name	Name to be given to comparison metric.
digits	Rounding for estimate values and p-values, default c(2,3).
ref_symbol	Reference level symbol

Value

A dataframe of first differences or ratios for bootstrapped distributions of a metric of interest.
finalfit predict functions

See Also

[boot_predict](#) [finalfit_newdata](#)

Examples

```
# See boot_predict.
```

boot_predict	<i>Bootstrap simulation for model prediction</i>
--------------	--

Description

Generate model predictions against a specified set of explanatory levels with bootstrapped confidence intervals. Add a comparison by difference or ratio of the first row of newdata with all subsequent rows.

Usage

```
boot_predict(
  fit,
  newdata,
  type = "response",
  R = 100,
  estimate_name = NULL,
  confint_level = 0.95,
  conf.method = "perc",
  confint_sep = " to ",
  condense = TRUE,
```

```

boot_compare = TRUE,
compare_name = NULL,
comparison = "difference",
ref_symbol = "-",
digits = c(2, 3)
)

```

Arguments

fit	A model generated using <code>lm</code> , <code>glm</code> , <code>lmmulti</code> , and <code>glmmulti</code> .
newdata	Dataframe usually generated with <code>finalfit_newdata</code> .
type	the type of prediction required, see <code>predict.glm</code> . The default for <code>glm</code> models is on the scale of the response variable. Thus for a binomial model the default predictions are predicted probabilities.
R	Number of simulations. Note default <code>R=100</code> is very low.
estimate_name	Name to be given to prediction variable $\hat{y}$.
confint_level	The confidence level to use for the confidence interval. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
conf.method	Passed to the type argument of <code>boot::boot.ci()</code> . Defaults to "perc". The allowed types are "perc", "basic", "bca", and "norm". Does not support "stud" or "all"
confint_sep	String separating lower and upper confidence interval
condense	Logical. FALSE gives numeric values, usually for plotting. TRUE gives table for final output.
boot_compare	Include a comparison with the first row of newdata with all subsequent rows. See boot_compare .
compare_name	Name to be given to comparison metric.
comparison	Either "difference" or "ratio".
ref_symbol	Reference level symbol
digits	Rounding for estimate values and p-values, default <code>c(2,3)</code> .

Details

To use this, first generate newdata for specified levels of explanatory variables using `finalfit_newdata`. Pass model objects from `lm`, `glm`, `lmmulti`, and `glmmulti`. The comparison metrics are made on individual bootstrap samples distribution returned as a mean with confidence intervals. A p-value is generated on the proportion of values on the other side of the null from the mean, e.g. for a ratio greater than 1.0, p is the number of bootstrapped predictions under 1.0, multiplied by two so is two-sided.

Value

A dataframe of predicted values and confidence intervals, with the option of including a comparison of difference between first row and all subsequent rows of newdata.

See Also

[finalfit_newdata](#)

Examples

```
library(finalfit)
library(dplyr)

# Predict probability of death across combinations of factor levels
explanatory = c("age.factor", "extent.factor", "perfor.factor")
dependent = 'mort_5yr'

# Generate combination of factor levels
colon_s %>%
  finalfit_newdata(explanatory = explanatory, newdata = list(
    c("<40 years", "Submucosa", "No"),
    c("<40 years", "Submucosa", "Yes"),
    c("<40 years", "Adjacent structures", "No"),
    c("<40 years", "Adjacent structures", "Yes")
  )) -> newdata

# Run simulation
colon_s %>%
  glmmulti(dependent, explanatory) %>%
  boot_predict(newdata, estimate_name = "Predicted probability of death",
    compare_name = "Absolute risk difference", R=100, digits = c(2,3))

# Plotting
explanatory = c("nodes", "extent.factor", "perfor.factor")
colon_s %>%
  finalfit_newdata(explanatory = explanatory, rowwise = FALSE, newdata = list(
    rep(seq(0, 30), 4),
    c(rep("Muscle", 62), rep("Adjacent structures", 62)),
    c(rep("No", 31), rep("Yes", 31), rep("No", 31), rep("Yes", 31))
  )) -> newdata

colon_s %>%
  glmmulti(dependent, explanatory) %>%
  boot_predict(newdata, boot_compare = FALSE, R=100, condense=FALSE) -> plot

library(ggplot2)
theme_set(theme_bw())
plot %>%
  ggplot(aes(x = nodes, y = estimate, ymin = estimate_conf.low,
    ymax = estimate_conf.high, fill=extent.factor))+
  geom_line(aes(colour = extent.factor))+
  geom_ribbon(alpha=0.1)+
  facet_grid(.~perfor.factor)+
  xlab("Number of postive lymph nodes")+
  ylab("Probability of death")+
  labs(fill = "Extent of tumour", colour = "Extent of tumour")+
  ggtitle("Probability of death by lymph node count")
```

check_recode	<i>Check accurate recoding of variables</i>
--------------	---

Description

This was written a few days after the retraction of a paper in JAMA due to an error in recoding the treatment variable (<https://jamanetwork.com/journals/jama/fullarticle/2752474>). This takes a data frame or tibble, fuzzy matches variable names, and produces crosstables of all matched variables. A visual inspection should reveal any miscoding.

Usage

```
check_recode(  
  .data,  
  dependent = NULL,  
  explanatory = NULL,  
  include_numerics = TRUE,  
  ...  
)
```

Arguments

<code>.data</code>	Data frame or tibble.
<code>dependent</code>	Optional character vector: name(s) of dependent variable(s).
<code>explanatory</code>	Optional character vector: name(s) of explanatory variable(s).
<code>include_numerics</code>	Logical. Include numeric variables in function.
<code>...</code>	Pass other arguments to <code>agrep</code> .

Value

List of length two. The first is an index of variable combinations. The second is a nested list of crosstables as tibbles.

Examples

```
library(dplyr)  
data(colon_s)  
colon_s_small = colon_s %>%  
  select(-id, -rx, -rx.factor) %>%  
  mutate(  
    age.factor2 = forcats::fct_collapse(age.factor,  
      "<60 years" = c("<40 years", "40-59 years")),  
    sex.factor2 = forcats::fct_recode(sex.factor,  
      # Intentional miscode  
      "F" = "Male",  
      "M" = "Female")
```

```

)

# Check
colon_s_small %>%
  check_recode(include_numerics = FALSE)

out = colon_s_small %>%
  select(-extent, -extent.factor, -time, -time.years) %>%
  check_recode()
out

# Select a tibble and expand
out$counts[[9]]
# Note this variable (node4) appears miscoded in original dataset survival::colon.

# Choose to only include variables that you actually use.
# This uses standard Finalfit grammar.
dependent = "mort_5yr"
explanatory = c("age.factor2", "sex.factor2")
colon_s_small %>%
  check_recode(dependent, explanatory)

```

coefficient_plot

Produce a coefficient table and plot

Description

Produce a coefficient and plot from a `lm()` model.

Usage

```

coefficient_plot(
  .data,
  dependent,
  explanatory,
  random_effect = NULL,
  factorlist = NULL,
  lmfit = NULL,
  confint_type = "default",
  confint_level = 0.95,
  remove_ref = FALSE,
  breaks = NULL,
  column_space = c(-0.5, -0.1, 0.5),
  dependent_label = NULL,
  prefix = "",
  suffix = NULL,
  table_text_size = 4,
  title_text_size = 13,
  plot_opts = NULL,

```

```

    table_opts = NULL,
    ...
)

```

Arguments

<code>.data</code>	Dataframe.
<code>dependent</code>	Character vector of length 1: name of dependent variable (must be numeric/continuous).
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>random_effect</code>	Character vector of length 1, name of random effect variable.
<code>factorlist</code>	Option to provide output directly from <code>summary_factorlist()</code> .
<code>lmfit</code>	Option to provide output directly from <code>lmmulti()</code> and <code>lmmixed()</code> .
<code>confint_type</code>	For for lmer models, one of <code>c("default", "Wald", "profile", "boot")</code> Note <code>"default" == "Wald"</code> .
<code>confint_level</code>	The confidence level required.
<code>remove_ref</code>	Logical. Remove reference level for factors.
<code>breaks</code>	Manually specify x-axis breaks in format <code>c(0.1, 1, 10)</code> .
<code>column_space</code>	Adjust table column spacing.
<code>dependent_label</code>	Main label for plot.
<code>prefix</code>	Plots are titled by default with the dependent variable. This adds text before that label.
<code>suffix</code>	Plots are titled with the dependent variable. This adds text after that label.
<code>table_text_size</code>	Alter font size of table text.
<code>title_text_size</code>	Alter font size of title text.
<code>plot_opts</code>	A list of arguments to be appended to the ggplot call by "+".
<code>table_opts</code>	A list of arguments to be appended to the ggplot table call by "+".
<code>...</code>	Other parameters.

Value

Returns a table and plot produced in ggplot2.

See Also

Other finalfit plot functions: `ff_plot()`, `hr_plot()`, `or_plot()`, `surv_plot()`

Examples

```
library(finalfit)
library(ggplot2)

# Coefficient plot
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "nodes"
colon_s %>%
  coefficient_plot(dependent, explanatory)

colon_s %>%
  coefficient_plot(dependent, explanatory, table_text_size=4, title_text_size=14,
    plot_opts=list(xlab("Beta, 95% CI"), theme(axis.title = element_text(size=12))))
```

colon_s	<i>Chemotherapy for Stage B/C colon cancer</i>
---------	--

Description

This is a modified version of `survival::colon`. These are data from one of the first successful trials of adjuvant chemotherapy for colon cancer. Levamisole is a low-toxicity compound previously used to treat worm infestations in animals; 5-FU is a moderately toxic (as these things go) chemotherapy agent. There are two records per person, one for recurrence and one for death

Usage

```
data(colon_s)
```

Format

A data frame with 929 rows and 33 variables

Source

`colon`

coxphmulti	<i>Cox proportional hazards multivariable models: finalfit model wrapper</i>
------------	--

Description

Using `finalfit` conventions, produces multivariable Cox Proportional Hazard regression models for a set of explanatory variables against a survival object.

Usage

```
coxphmulti(.data, dependent, explanatory, ...)
```

Arguments

<code>.data</code>	Data frame.
<code>dependent</code>	Character vector of length 1: name of survival object in form <code>Surv(time, status)</code> .
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>...</code>	Other arguments to pass to coxph .

Details

Uses [coxph](#) with `finalfit` modelling conventions. Output can be passed to [fit2df](#).

Value

A multivariable [coxph](#) fitted model output. Output is of class `coxph`.

See Also

[fit2df](#), [finalfit_merge](#)

Other `finalfit` model wrappers: [coxphuni\(\)](#), [crrmulti\(\)](#), [crruni\(\)](#), [glmmixed\(\)](#), [glmmulti\(\)](#), [glmmulti_boot\(\)](#), [glmuni\(\)](#), [lmmixed\(\)](#), [lmmulti\(\)](#), [lmuni\(\)](#), [svyglmmulti\(\)](#), [svyglmuni\(\)](#)

Examples

```
# Cox Proportional Hazards multivariable analysis.
library(finalfit)
library(dplyr)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "Surv(time, status)"
colon_s %>%
  coxphmulti(dependent, explanatory) %>%
  fit2df()
```

coxphuni

Cox proportional hazards univariable models: finalfit model wrapper

Description

Using `finalfit` conventions, produces multiple univariable Cox Proportional Hazard regression models for a set of explanatory variables against a survival object.

Usage

```
coxphuni(.data, dependent, explanatory)
```

Arguments

.data	Data frame.
dependent	Character vector of length 1: name of survival object in form <code>Surv(time, status)</code> .
explanatory	Character vector of any length: name(s) of explanatory variables.

Details

Uses `coxph` with `finalfit` modelling conventions. Output can be passed to `fit2df`.

Value

A list of univariable `coxph` fitted model outputs. Output is of class `coxphlist`.

See Also

`fit2df`, `finalfit_merge`

Other `finalfit` model wrappers: `coxphmulti()`, `crrmulti()`, `crruni()`, `glmmixed()`, `glmmulti()`, `glmmulti_boot()`, `glmuni()`, `lmmixed()`, `lmmulti()`, `lmuni()`, `svyglmmulti()`, `svyglmuni()`

Examples

```
# Cox Proportional Hazards univariable analysis.
library(finalfit)
library(dplyr)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "Surv(time, status)"
colon_s %>%
  coxphuni(dependent, explanatory) %>%
  fit2df()
```

crrmulti

Competing risks multivariable regression: finalfit model wrapper

Description

Using `finalfit` conventions, produces multivariable Competing Risks Regression models for a set of explanatory variables.

Usage

```
crrmulti(.data, dependent, explanatory, ...)
```

Arguments

<code>.data</code>	Data frame or tibble.
<code>dependent</code>	Character vector of length 1: name of survival object in form <code>Surv(time, status)</code> . Status default values should be 0 censored (e.g. alive), 1 event of interest (e.g. died of disease of interest), 2 competing event (e.g. died of other cause).
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>...</code>	Other arguments to <code>crr</code>

Details

Uses `crr` with `finalfit` modelling conventions. Output can be passed to `fit2df`.

Value

A multivariable `crr` fitted model class `crr`.

See Also

`fit2df`, `finalfit_merge`

Other `finalfit` model wrappers: `coxphmulti()`, `coxphuni()`, `crruni()`, `glmmixed()`, `glmmulti()`, `glmmulti_boot()`, `glmuni()`, `lmmixed()`, `lmmulti()`, `lmuni()`, `svyglmmulti()`, `svyglmuni()`

Examples

```
library(dplyr)
melanoma = boot::melanoma
melanoma = melanoma %>%
  mutate(
    # Cox PH to determine cause-specific hazards
    status_coxph = ifelse(status == 2, 0, # "still alive"
      ifelse(status == 1, 1, # "died of melanoma"
        0)), # "died of other causes is censored"

    # Fine and Gray to determine subdistribution hazards
    status_crr = ifelse(status == 2, 0, # "still alive"
      ifelse(status == 1, 1, # "died of melanoma"
        2)), # "died of other causes"
    sex = factor(sex),
    ulcer = factor(ulcer)
  )

dependent_coxph = c("Surv(time, status_coxph)")
dependent_crr = c("Surv(time, status_crr)")
explanatory = c("sex", "age", "ulcer")

# Create single well-formatted table
melanoma %>%
  summary_factorlist(dependent_crr, explanatory, column = TRUE, fit_id = TRUE) %>%
  ff_merge(
```

```

melanoma %>%
  coxphmulti(dependent_coxph, explanatory) %>%
  fit2df(estimate_suffix = " (Cox PH multivariable)")
) %>%
ff_merge(
  melanoma %>%
    crrmulti(dependent_crr, explanatory) %>%
    fit2df(estimate_suffix = " (competing risks multivariable)")
) %>%
select(-fit_id, -index) %>%
dependent_label(melanoma, dependent_crr)

```

crruni

Competing risks univariable regression: finalfit model wrapper

Description

Using finalfit conventions, produces univariable Competing Risks Regression models for a set of explanatory variables.

Usage

```
crruni(.data, dependent, explanatory, ...)
```

Arguments

.data	Data frame or tibble.
dependent	Character vector of length 1: name of survival object in form Surv(time, status). Status default values should be 0 censored (e.g. alive), 1 event of interest (e.g. died of disease of interest), 2 competing event (e.g. died of other cause).
explanatory	Character vector of any length: name(s) of explanatory variables.
...	Other arguments to crr

Details

Uses [crr](#) with finalfit modelling conventions. Output can be passed to [fit2df](#).

Value

A list of univariable [crr](#) fitted models class crrlist.

See Also

[fit2df](#), [finalfit_merge](#)

Other finalfit model wrappers: [coxphmulti\(\)](#), [coxphuni\(\)](#), [crrmulti\(\)](#), [glmmixed\(\)](#), [glmmulti\(\)](#), [glmmulti_boot\(\)](#), [glmuni\(\)](#), [lmmixed\(\)](#), [lmmulti\(\)](#), [lmuni\(\)](#), [svyglmmulti\(\)](#), [svyglmuni\(\)](#)

Examples

```
library(dplyr)
melanoma = boot::melanoma
melanoma = melanoma %>%
  mutate(
    # Cox PH to determine cause-specific hazards
    status_coxph = ifelse(status == 2, 0, # "still alive"
      ifelse(status == 1, 1, # "died of melanoma"
        0)), # "died of other causes is censored"

    # Fine and Gray to determine subdistribution hazards
    status_crr = ifelse(status == 2, 0, # "still alive"
      ifelse(status == 1, 1, # "died of melanoma"
        2)), # "died of other causes"
    sex = factor(sex),
    ulcer = factor(ulcer)
  )

dependent_coxph = c("Surv(time, status_coxph)")
dependent_crr = c("Surv(time, status_crr)")
explanatory = c("sex", "age", "ulcer")

# Create single well-formatted table
melanoma %>%
  summary_factorlist(dependent_crr, explanatory, column = TRUE, fit_id = TRUE) %>%
  ff_merge(
    melanoma %>%
      coxphmulti(dependent_coxph, explanatory) %>%
      fit2df(estimate_suffix = " (Cox PH multivariable)")
  ) %>%
  ff_merge(
    melanoma %>%
      crrmulti(dependent_crr, explanatory) %>%
      fit2df(estimate_suffix = " (competing risks multivariable)")
  ) %>%
  select(-fit_id, -index) %>%
  dependent_label(melanoma, dependent_crr)
```

dependent_label	<i>Make a label for the dependent variable</i>
-----------------	--

Description

Can be add dependent label to final results dataframe.

Usage

```
dependent_label(df.out, .data, dependent, prefix = "Dependent: ", suffix = "")
```

Arguments

<code>df.out</code>	Dataframe (results table) to be altered.
<code>.data</code>	Original dataframe.
<code>dependent</code>	Character vector of length 1: quoted name of dependent variable. Can be continuous, a binary factor, or a survival object of form <code>Surv(time, status)</code>
<code>prefix</code>	Prefix for dependent label
<code>suffix</code>	Suffix for dependent label

Value

Returns the label for the dependent variable, if specified.

Examples

```
library(dplyr)
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
explanatory_multi = c("age.factor", "obstruct.factor")
random_effect = "hospital"
dependent = 'mort_5yr'

# Separate tables
colon_s %>%
summary_factorlist(dependent, explanatory, fit_id=TRUE) -> example.summary

colon_s %>%
glmuni(dependent, explanatory) %>%
fit2df(estimate_suffix=" (univariable)") -> example.univariable

colon_s %>%
glmmulti(dependent, explanatory) %>%
fit2df(estimate_suffix=" (multivariable)") -> example.multivariable

colon_s %>%
glmmixed(dependent, explanatory, random_effect) %>%
fit2df(estimate_suffix=" (multilevel)") -> example.multilevel

# Pipe together
example.summary %>%
  finalfit_merge(example.univariable) %>%
  finalfit_merge(example.multivariable) %>%
  finalfit_merge(example.multilevel) %>%
  select(-c(fit_id, index)) %>%
  dependent_label(colon_s, dependent) -> example.final
example.final
```

 extract_variable_label

Extract variable labels from dataframe

Description

Variable labels can be created using [ff_label](#). Some functions strip variable labels (variable attributes), e.g. `forcats::fct_recode`. Use this function to create a vector of variable labels from a data frame. Then use [ff_relabel](#) to relabel variables in data frame.

Usage

```
extract_variable_label(.data)
```

Arguments

`.data` Dataframe containing labelled variables.

Examples

```
colon_s %>%
  extract_variable_label
```

 ff_column_totals

Add column totals to summary_factorlist() output

Description

Add column totals to `summary_factorlist()` output

Usage

```
ff_column_totals(
  df.in,
  .data,
  dependent,
  na_include_dependent = FALSE,
  percent = TRUE,
  digits = c(1, 0),
  label = NULL,
  prefix = "",
  weights = NULL
)
```

```
finalfit_column_totals(
  df.in,
```

```

    .data,
    dependent,
    na_include_dependent = FALSE,
    percent = TRUE,
    digits = c(1, 0),
    label = NULL,
    prefix = "",
    weights = NULL
  )

```

Arguments

<code>df.in</code>	<code>summary_factorlist()</code> output.
<code>.data</code>	Data frame used to create <code>summary_factorlist()</code> .
<code>dependent</code>	Character. Name of dependent variable.
<code>na_include_dependent</code>	Logical. When TRUE, missing data in the dependent variable is included in totals.
<code>percent</code>	Logical. Include percentage.
<code>digits</code>	Integer length 2. Number of digits for (1) percentage, (2) weighted count.
<code>label</code>	Character. Label for total row.
<code>prefix</code>	Character. Prefix for column totals, e.g "N=".
<code>weights</code>	Character vector of length 1: name of column to use for weights.

Value

Data frame.

Examples

```

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = 'mort_5yr'
colon_s %>%
  summary_factorlist(dependent, explanatory) %>%
  ff_column_totals(colon_s, dependent)

# Ensure works with missing data in dependent
colon_s = colon_s %>%
  dplyr::mutate(
    mort_5yr = forcats::fct_na_value_to_level(mort_5yr, level = "(Missing)")
  )
colon_s %>%
  summary_factorlist(dependent, explanatory) %>%
  ff_column_totals(colon_s, dependent)

```

ff_expand	<i>Summarise with mode and mean/median and expand given factors</i>
-----------	---

Description

When producing conditional estimates from a regression model, it is often useful to set variables not of interest to their mode for factors and mean or median for numerics when creating the newdata object, and combine these with all levels for factors of interest.

Usage

```
ff_expand(.data, ..., cont = "mean")  
  
finalfit_expand(.data, ..., cont = "mean")
```

Arguments

.data	A data frame or tibble.
...	Factors to expand.
cont	One of "mean" or "median": the summary estimate for continuous variables.

Value

A data frame or tibble with the mode for factors and mean/median for continuous variables, with given factors expanded to include all levels.

See Also

[ff_mode summary_df](#)

Examples

```
library(dplyr)  
colon_s %>%  
  select(-hospital) %>%  
  ff_expand(age.factor, sex.factor)
```

ff_formula	<i>Generate formula as character string</i>
------------	---

Description

Useful when passing finalfit dependent and explanatory lists to base R functions

Usage

```
ff_formula(dependent, explanatory, random_effect = NULL)
```

```
finalfit_formula(dependent, explanatory, random_effect = NULL)
```

Arguments

dependent	Optional character vector: name(s) of dependent variable(s).
explanatory	Optional character vector: name(s) of explanatory variable(s).
random_effect	Optional character vector: name(s) of random effect variable(s).

Value

Character vector

Examples

```
explanatory = c("age", "nodes", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"
ff_formula(dependent, explanatory)

explanatory = c("age", "nodes", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"
random_effect = "(age.factor | hospital)"
ff_formula(dependent, explanatory)
```

ff_glimpse	<i>Descriptive statistics for dataframe</i>
------------	---

Description

Everyone has a function like this, str, glimpse, glance etc. This one is specifically designed for use with finalfit language. It is different in dividing variables by numeric vs factor.

Usage

```
ff_glimpse(
  .data,
  dependent = NULL,
  explanatory = NULL,
  digits = 1,
  levels_cut = 5
)

finalfit_glimpse(
  .data,
  dependent = NULL,
  explanatory = NULL,
  digits = 1,
  levels_cut = 5
)
```

Arguments

.data	Dataframe.
dependent	Optional character vector: name(s) of dependent variable(s).
explanatory	Optional character vector: name(s) of explanatory variable(s).
digits	Significant digits for continuous variable summaries
levels_cut	Max number of factor levels to include in factor levels summary (in order to avoid the long printing of variables with many factors).

Value

Dataframe on summary data.

Examples

```
library(finalfit)
dependent = 'mort_5yr'
explanatory = c("age", "nodes", "age.factor", "extent.factor", "perfor.factor")
colon_s %>%
  finalfit_glimpse(dependent, explanatory)
```

ff_interaction	<i>Make an interaction variable and add to dataframe</i>
----------------	--

Description

Combine two factor variables to make an interaction variable. Factor level order is determined by the order in the variables themselves. Note, names of the factor variables should not be quoted. The name of the variable is created from the names of the two factors. The variable is also labelled with a name derived from any pre-existing labels.

Usage

```
ff_interaction(.data, ..., levels_sep = "_", var_sep = "_", label_sep = ":")

finalfit_interaction(
  .data,
  ...,
  levels_sep = "_",
  var_sep = "_",
  label_sep = ":"
)
```

Arguments

- .data Data frame.
- ... The unquoted names of two factors.
- levels_sep Quoted character: how levels are separated in new variable.
- var_sep Quoted character: how variable name is separated.
- label_sep Quoted character: how variable label is separated

Value

Original data frame with new variable added via ‘dplyr::mutate’.

Examples

```
colon_s %>%
  ff_interaction(sex.factor, perfor.factor) %>%
  summary_factorlist("mort_5yr", "sex.factor_perfor.factor")
```

ff_label	<i>Label a variable</i>
----------	-------------------------

Description

Label a variable

Usage

```
ff_label(.var, variable_label)

finalfit_label(.var, variable_label)
```

Arguments

- .var Quoted variable name
- variable_label Quoted variable label

Value

Labelled variable

See Also

[extract_variable_label](#) [ff_relabel](#)

Examples

```
colon_s$sex.factor %>%
  ff_label("Sex") %>%
  str()
```

ff_merge

Merge a [summary_factorlist\(\)](#) table with any number of model results tables.

Description

A function that takes the output from [summary_factorlist\(..., fit_id=TRUE\)](#) and merges with any number of model dataframes, usually produced with a model wrapper followed by the [fit2df\(\)](#) function (see examples).

Usage

```
ff_merge(
  factorlist,
  fit2df_df,
  ref_symbol = "-",
  estimate_name = NULL,
  last_merge = FALSE
)

finalfit_merge(
  factorlist,
  fit2df_df,
  ref_symbol = "-",
  estimate_name = NULL,
  last_merge = FALSE
)
```

Arguments

factorlist	Output from summary_factorlist(..., fit_id=TRUE) .
fit2df_df	Output from model wrappers followed by fit2df() .
ref_symbol	Reference symbol for model reference levels, typically "-" or "1.0".

estimate_name	If you have chosen a new 'estimate name' (e.g. "Odds ratio") when running a model wrapper (e.g. 'glmuni'), then you need to pass this new name to 'final-fit_merge' to generate correct table. Defaults to OR/HR/Coefficient
last_merge	Logical. Set to try for the final merge in a series to remove index and fit_id columns.

Value

Returns a dataframe of combined tables.

See Also

[summary_factorlist fit2df](#)

Examples

```
library(finalfit)
library(dplyr)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
explanatory_multi = c("age.factor", "obstruct.factor")
random_effect = "hospital"
dependent = "mort_5yr"

# Create separate tables
colon_s %>%
  summary_factorlist(dependent, explanatory, fit_id=TRUE) -> example.summary

colon_s %>%
  glmuni(dependent, explanatory) %>%
  fit2df(estimate_suffix=" (univariable)") -> example.univariable

colon_s %>%
  glmmulti(dependent, explanatory) %>%
  fit2df(estimate_suffix=" (multivariable)") -> example.multivariable

colon_s %>%
  glmmixed(dependent, explanatory, random_effect) %>%
  fit2df(estimate_suffix=" (multilevel)") -> example.multilevel

# Pipe together
example.summary %>%
  ff_merge(example.univariable) %>%
  ff_merge(example.multivariable) %>%
  ff_merge(example.multilevel, last_merge = TRUE)

# Using finalfit()
colon_s %>%
  finalfit(dependent, explanatory, keep_fit_id = TRUE) %>%
  ff_merge(example.multilevel, last_merge = TRUE)
```

ff_metrics	<i>Generate common metrics for regression model results</i>
------------	---

Description

Generate common metrics for regression model results

Usage

```
ff_metrics(.data)

## S3 method for class 'lm'
ff_metrics(.data)

## S3 method for class 'lmlist'
ff_metrics(.data)

## S3 method for class 'glm'
ff_metrics(.data)

## S3 method for class 'glmlist'
ff_metrics(.data)

## S3 method for class 'lmerMod'
ff_metrics(.data)

## S3 method for class 'glmerMod'
ff_metrics(.data)

## S3 method for class 'coxph'
ff_metrics(.data)

## S3 method for class 'coxphlist'
ff_metrics(.data)
```

Arguments

.data Model output.

Value

Model metrics vector for output.

Examples

```
library(finalfit)

# glm
```

```

fit = glm(mort_5yr ~ age.factor + sex.factor + obstruct.factor + perfor.factor,
  data=colon_s, family="binomial")
fit %>%
  ff_metrics()

# glmlist
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"
colon_s %>%
  glmmulti(dependent, explanatory) %>%
  ff_metrics()

# glmerMod
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
random_effect = "hospital"
dependent = "mort_5yr"
colon_s %>%
  glmmixed(dependent, explanatory, random_effect) %>%
  ff_metrics()

# lm
fit = lm(nodes ~ age.factor + sex.factor + obstruct.factor + perfor.factor,
  data=colon_s)
fit %>%
  ff_metrics()

# lmerMod
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
random_effect = "hospital"
dependent = "nodes"

colon_s %>%
  lmmixed(dependent, explanatory, random_effect) %>%
  ff_metrics()

# coxphlist
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "Surv(time, status)"

colon_s %>%
  coxphmulti(dependent, explanatory) %>%
  ff_metrics()

# coxph
fit = survival::coxph(survival::Surv(time, status) ~ age.factor + sex.factor +
  obstruct.factor + perfor.factor,
  data = colon_s)

fit %>%
  ff_metrics()

```

ff_mode	<i>Return the most frequent level in a factor</i>
---------	---

Description

When producing conditional estimates from a regression model, it is often useful to set variables not of interest to their mode when creating the newdata object.

Usage

```
ff_mode(...)
```

```
finalfit_mode(...)
```

Arguments

... Unquoted factor names.

Value

The most frequent level in a factor.

See Also

[summary_df](#) [ff_expand](#)

Examples

```
library(dplyr)
colon_s %>%
  summarise(age.factor = ff_mode(age.factor))

colon_s %>%
  select(sex.factor, rx.factor, obstruct.factor, perfor.factor) %>%
  summarise(across(everything(), ff_mode))

colon_s %>%
  reframe(across(where(is.factor), ff_mode))
# Note, 4 rows is returned in this example because 4 factor levels within `hospital`
# have the same frequency.
```

ff_newdata

*Generate newdata for simulations***Description**

Generate newdata while respecting the variable types and factor levels in the primary data frame used to run model.

Usage

```
ff_newdata(
  .data,
  dependent = NULL,
  explanatory = NULL,
  rowwise = TRUE,
  newdata
)

finalfit_newdata(
  .data,
  dependent = NULL,
  explanatory = NULL,
  rowwise = TRUE,
  newdata
)
```

Arguments

.data	Dataframe.
dependent	Optional character vector of length 1: name of dependent variable. Not usually specified in bootstrapping model predictions.
explanatory	Character vector of any length: name(s) of explanatory variables.
rowwise	Logical. Format newdata is provided in.
newdata	A list of rows or columns corresponding exactly to the order of explanatory variables. Useful errors generated if requirements not fulfilled

Details

Generate model predictions against a specified set of explanatory levels with bootstrapped confidence intervals. Add a comparison by difference or ratio of the first row of newdata with all subsequent rows.

Value

A list of multivariable [glm](#) fitted model outputs. Output is of class `glm`list.

See Also

[boot_predict](#) [boot_compare](#)

Examples

```
# See boot_predict.
library(finalfit)
library(dplyr)

# Predict probability of death across combinations of factor levels
explanatory = c("age.factor", "extent.factor", "perfor.factor")
dependent = 'mort_5yr'

# Generate combination of explanatory variable levels rowwise
colon_s %>%
  finalfit_newdata(explanatory = explanatory, newdata = list(
    c("<40 years", "Submucosa", "No"),
    c("<40 years", "Submucosa", "Yes"),
    c("<40 years", "Adjacent structures", "No"),
    c("<40 years", "Adjacent structures", "Yes")
  )) -> newdata

# Generate combination of explanatory variable levels colwise.
explanatory = c("nodes", "extent.factor", "perfor.factor")
colon_s %>%
  finalfit_newdata(explanatory = explanatory, rowwise = FALSE, newdata = list(
    rep(seq(0, 30), 4),
    c(rep("Muscle", 62), rep("Adjacent structures", 62)),
    c(rep("No", 31), rep("Yes", 31), rep("No", 31), rep("Yes", 31))
  )) -> newdata
```

ff_parse_formula

*Parse a formula to finalfit grammar***Description**

Parse a formula to finalfit grammar

Usage

```
ff_parse_formula(.formula)
```

Arguments

`.formula` an object of class "formula" (or one that can be coerced to that class).

Value

A list containing dependent, explanatory and random effects variables

Examples

```
ff_parse_formula(mort ~ age + sex + (1 | hospital))
```

ff_percent_only	Include only percentages for factors in summary_factorlist output
-----------------	---

Description

Include only percentages for factors in [summary_factorlist](#) output

Usage

```
ff_percent_only(.data)

finalfit_percent_only(.data)
```

Arguments

.data Output from [finalfit](#) or similar.

Value

Data frame.

Examples

```
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = 'mort_5yr'
colon_s %>%
  summary_factorlist(dependent, explanatory) %>%
  ff_percent_only()
```

ff_permute	Permuate explanatory variables to produce multiple output tables for common regression models
------------	---

Description

Permuate explanatory variables to produce multiple output tables for common regression models

Usage

```
ff_permute(
  .data,
  dependent = NULL,
  explanatory_base = NULL,
  explanatory_permute = NULL,
  multiple_tables = FALSE,
  include_base_model = TRUE,
  include_full_model = TRUE,
  base_on_top = TRUE,
  ...
)

finalfit_permute(
  .data,
  dependent = NULL,
  explanatory_base = NULL,
  explanatory_permute = NULL,
  multiple_tables = FALSE,
  include_base_model = TRUE,
  include_full_model = TRUE,
  base_on_top = TRUE,
  ...
)
```

Arguments

<code>.data</code>	Data frame or tibble.
<code>dependent</code>	Character vector of length 1: quoted name of dependent variable. Can be continuous, a binary factor, or a survival object of form <code>Surv(time, status)</code> .
<code>explanatory_base</code>	Character vector of any length: quoted name(s) of base model explanatory variables.
<code>explanatory_permute</code>	Character vector of any length: quoted name(s) of explanatory variables to permute through models.
<code>multiple_tables</code>	Logical. Multiple model tables as a list, or a single table including multiple models.
<code>include_base_model</code>	Logical. Include model using <code>explanatory_base</code> variables only.
<code>include_full_model</code>	Logical. Include model using all <code>explanatory_base</code> and <code>explanatory_permute</code> variables.
<code>base_on_top</code>	Logical. Base variables at top of table, or bottom of table.
<code>...</code>	Other arguments to finalfit

Value

Returns a list of data frame with the final model table.

Examples

```

explanatory_base = c("age.factor", "sex.factor")
explanatory_permute = c("obstruct.factor", "perfor.factor", "node4.factor")

# Linear regression
colon_s %>%
  finalfit_permute("nodes", explanatory_base, explanatory_permute)

# Cox proportional hazards regression
colon_s %>%
  finalfit_permute("Surv(time, status)", explanatory_base, explanatory_permute)

# Logistic regression
# colon_s %>%
#   finalfit_permute("mort_5yr", explanatory_base, explanatory_permute)

# Logistic regression with random effect (glmer)
# colon_s %>%
#   finalfit_permute("mort_5yr", explanatory_base, explanatory_permute,
#     random_effect = "hospital")

```

ff_plot

*Produce a table and plot***Description**

Wraps [or_plot](#), [hr_plot](#), and [coefficient_plot](#) and sends to the appropriate method depending on the dependent variable type.

Usage

```

ff_plot(.data, dependent, explanatory, ...)

finalfit_plot(.data, dependent, explanatory, ...)

```

Arguments

.data	Data frame.
dependent	Character vector of length 1.
explanatory	Character vector of any length: name(s) of explanatory variables.
...	Pass arguments or_plot , hr_plot , or coefficient_plot

Value

A table and a plot using ggplot2.

See Also

Other finalfit plot functions: [coefficient_plot\(\)](#), [hr_plot\(\)](#), [or_plot\(\)](#), [surv_plot\(\)](#)

Examples

```
# Coefficient plot
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "nodes"
colon_s %>%
  ff_plot(dependent, explanatory)

# Odds ratio plot
dependent = "mort_5yr"
colon_s %>%
  ff_plot(dependent, explanatory)

# Hazard ratio plot
dependent = "Surv(time, status)"
colon_s %>%
  ff_plot(dependent, explanatory, dependent_label = "Survival")
```

ff_relabel	<i>Relabel variables in a data frame</i>
------------	--

Description

Variable labels can be created using [ff_label](#). Some functions strip variable labels (variable attributes), e.g. `forcats::fct_recode`. Use this function to create a vector of variable labels from a data frame. Then use [ff_relabel](#) to relabel variables in data frame.

Usage

```
ff_relabel(.data, .labels)

finalfit_relabel(.data, .labels)
```

Arguments

.data	Data frame to be relabelled
.labels	Vector of variable labels (usually created using extract_variable_label).

Examples

```
# Label variable
colon_s$sex.factor %>%
  ff_label("Sex") %>%
  str()

# Make factor level "Unknown" NA
colon_s %>%
  dplyr::mutate_if(is.factor, forcats::fct_recode,
    NULL = "Unknown") %>%
  str()

# Reset data
data(colon_s)

# Extract variable labels
vlabels = colon_s %>% extract_variable_label()

# Run function where labels are lost
colon_s %>%
  dplyr::mutate_if(is.factor, forcats::fct_recode,
    NULL = "Unknown") %>%
  str()

# Relabel
colon_s %<>% ff_relabel(vlabels)
colon_s %>% str()
```

ff_relabel_df

*Relabel variables from data frame after tidyverse functions***Description**

Relabel variables from data frame after tidyverse functions

Usage

```
ff_relabel_df(.data, .df)

finalfit_relabel_df(.data, .df)
```

Arguments

.data	Data frame or tibble after applicaton of label stripping functions.
.df	Original data frame which contains labels.

Value

Data frame or tibble

ff_remove_p	<i>Remove p-value from output</i>
-------------	-----------------------------------

Description

This will work with `finalfit` and any `fit2df` output.

Usage

```
ff_remove_p(.data)

finalfit_remove_p(.data)
```

Arguments

`.data` Output from `finalfit` or similar.

Value

Data frame.

Examples

```
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = 'mort_5yr'
colon_s %>%
  finalfit(dependent, explanatory) %>%
  ff_remove_p()
```

ff_remove_ref	<i>Remove regression reference level row from table</i>
---------------	---

Description

This looks for a column with a name including "Coefficient", "OR", or "HR" (`finalfit` defaults) and removes any rows with "-" (the default for the reference level). Can also be combined to produce an `or_plot`, see below.

Usage

```
ff_remove_ref(.data, only_binary = TRUE)

finalfit_remove_ref(.data, only_binary = TRUE)
```

Arguments

<code>.data</code>	Output from <code>finalfit</code> or similar.
<code>only_binary</code>	Logical. Remove reference level only for two-level factors. When set to false, reference level for all factors removed.

Value

Data frame.

Examples

```
# Table example
explanatory = c("age.factor", "age", "sex.factor", "nodes", "obstruct.factor", "perfor.factor")
dependent = 'mort_5yr'
colon_s %>%
  finalfit(dependent, explanatory, add_dependent_label = FALSE) %>%
  ff_remove_ref() %>%
  dependent_label(colon_s, dependent)

# Plot example
explanatory = c("age.factor", "age", "sex.factor", "nodes", "obstruct.factor", "perfor.factor")
dependent = 'mort_5yr'
colon_s %>%
  summary_factorlist(dependent, explanatory, total_col = TRUE, fit_id=TRUE) %>%
  ff_merge(
    glmuni(colon_s, dependent, explanatory) %>%
    fit2df()) %>%
  ff_remove_ref() %>%
  dplyr::select(-`OR`) -> factorlist_plot

colon_s %>%
  or_plot(dependent, explanatory, factorlist = factorlist_plot)
```

ff_row_totals

Add row totals to summary_factorlist() output

Description

This adds a total and missing count to variables. This is useful for continuous variables. Compare this to `summary_factorlist(total_col = TRUE)` which includes a count for each dummy variable as a factor and mean (sd) or median (iqr) for continuous variables.

Usage

```
ff_row_totals(
  df.in,
  .data,
  dependent,
```

```

    explanatory,
    missing_column = TRUE,
    percent = TRUE,
    digits = 1,
    na_include_dependent = FALSE,
    na_complete_cases = FALSE,
    total_name = "Total N",
    na_name = "Missing N"
  )

finalfit_row_totals(
  df.in,
  .data,
  dependent,
  explanatory,
  missing_column = TRUE,
  percent = TRUE,
  digits = 1,
  na_include_dependent = FALSE,
  na_complete_cases = FALSE,
  total_name = "Total N",
  na_name = "Missing N"
)

```

Arguments

<code>df.in</code>	<code>summary_factorlist()</code> output.
<code>.data</code>	Data frame used to create <code>summary_factorlist()</code> .
<code>dependent</code>	Character. Name of dependent variable.
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>missing_column</code>	Logical. Include a column of counts of missing data.
<code>percent</code>	Logical. Include percentage.
<code>digits</code>	Integer length 1. Number of digits for percentage.
<code>na_include_dependent</code>	Logical. When TRUE, missing data in the dependent variable is included in totals.
<code>na_complete_cases</code>	Logical. When TRUE, missing data counts for variables are for complete cases across all included variables.
<code>total_name</code>	Character. Name of total column.
<code>na_name</code>	Character. Name of missing column.

Value

Data frame.

Examples

```

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = 'mort_5yr'
colon_s %>%
  summary_factorlist(dependent, explanatory) %>%
  ff_row_totals(colon_s, dependent, explanatory)

```

ff_stratify_helper	<i>Help making stratified summary_factorlist tables</i>
--------------------	---

Description

Help making stratified summary_factorlist tables

Usage

```
ff_stratify_helper(df.out, .data)
```

Arguments

df.out	Output from summary_factorlist
.data	Original data frame used for summary_factorlist.

Examples

```

library(dplyr)
explanatory = c("age.factor", "sex.factor")
dependent = "perfor.factor"

# Pick option below
split = "rx.factor"
split = c("rx.factor", "node4.factor")

# Piped function to generate stratified crosstabs table
colon_s %>%
  group_by(!!! syms(split)) %>% #Looks awkward, but avoids unquoted var names
  group_modify(~ summary_factorlist(.x, dependent, explanatory)) %>%
  ff_stratify_helper(colon_s)

```

finalfit

*Final output tables for common regression models***Description**

An "all-in-one" function that takes a single dependent variable with a vector of explanatory variable names (continuous or categorical variables) to produce a final table for publication including summary statistics. The appropriate model is selected on the basis of dependent variable and whether a random effect is specified.

finalfit.lm method (not called directly)

finalfit.glm method (not called directly)

finalfit.coxph method (not called directly)

Usage

```
finalfit(
  .data,
  dependent = NULL,
  explanatory = NULL,
  explanatory_multi = NULL,
  random_effect = NULL,
  formula = NULL,
  model_args = list(),
  weights = NULL,
  cont_cut = 5,
  column = NULL,
  keep_models = FALSE,
  metrics = FALSE,
  add_dependent_label = TRUE,
  dependent_label_prefix = "Dependent: ",
  dependent_label_suffix = "",
  keep_fit_id = FALSE,
  ...
)
```

```
finalfit.lm(
  .data,
  dependent,
  explanatory,
  explanatory_multi = NULL,
  random_effect = NULL,
  model_args = NULL,
  weights = NULL,
  cont_cut = 5,
  column = FALSE,
  keep_models = FALSE,
```

```
    metrics = FALSE,
    add_dependent_label = TRUE,
    dependent_label_prefix = "Dependent: ",
    dependent_label_suffix = "",
    keep_fit_id = FALSE,
    ...
)

finalfit.glm(
  .data,
  dependent,
  explanatory,
  explanatory_multi = NULL,
  random_effect = NULL,
  model_args = NULL,
  weights = NULL,
  cont_cut = 5,
  column = FALSE,
  keep_models = FALSE,
  metrics = FALSE,
  add_dependent_label = TRUE,
  dependent_label_prefix = "Dependent: ",
  dependent_label_suffix = "",
  keep_fit_id = FALSE,
  ...
)

finalfit.coxph(
  .data,
  dependent,
  explanatory,
  explanatory_multi = NULL,
  random_effect = NULL,
  model_args = NULL,
  column = TRUE,
  cont_cut = 5,
  keep_models = FALSE,
  metrics = FALSE,
  add_dependent_label = TRUE,
  dependent_label_prefix = "Dependent: ",
  dependent_label_suffix = "",
  keep_fit_id = FALSE,
  ...
)
```

Arguments

`.data` Data frame or tibble.

<code>dependent</code>	Character vector of length 1: quoted name of dependent variable. Can be continuous, a binary factor, or a survival object of form <code>Surv(time, status)</code> .
<code>explanatory</code>	Character vector of any length: quoted name(s) of explanatory variables.
<code>explanatory_multi</code>	Character vector of any length: quoted name(s) of a subset of explanatory variables to generate reduced multivariable model (must only contain variables contained in <code>explanatory</code>).
<code>random_effect</code>	Character vector of length 1, either, (1) name of random intercept variable, e.g. "var1", (automatically converted to "(1 var1)"); or, (2) the full lme4 specification, e.g. "(var1 var2)". Note parenthesis MUST be included in (2) but NOT included in (1).
<code>formula</code>	an object of class "formula" (or one that can be coerced to that class). Optional instead of standard dependent/explanatory format. Do not include if using dependent/explanatory.
<code>model_args</code>	List. A list of arguments to pass to <code>lm</code> , <code>glm</code> , <code>coxph</code> .
<code>weights</code>	Character vector of length 1: quoted name of weights variable. Passed to <code>summary_factorlist</code> , <code>lm</code> , and <code>glm</code> to provide weighted summary table and regression (e.g. IPTW). If wish weighted regression and non-weighted summary table, pass <code>weights</code> argument within <code>model_args</code> . Not available with survival dependent variable.
<code>cont_cut</code>	Numeric: number of unique values in continuous variable at which to consider it a factor.
<code>column</code>	Logical: Compute margins by column rather than row.
<code>keep_models</code>	Logical: include full multivariable model in output when working with reduced multivariable model (<code>explanatory_multi</code>) and/or mixed effect models (<code>random_effect</code>).
<code>metrics</code>	Logical: include useful model metrics in output in publication format.
<code>add_dependent_label</code>	Add the name of the dependent label to the top left of table.
<code>dependent_label_prefix</code>	Add text before dependent label.
<code>dependent_label_suffix</code>	Add text after dependent label.
<code>keep_fit_id</code>	Keep original model output coefficient label (internal).
<code>...</code>	Other arguments to pass to <code>fit2df</code> : <code>estimate_name</code> , <code>digits</code> , <code>confint_type</code> , <code>confint_level</code> , <code>confint_sep</code> .

Value

Returns a data frame with the final model table.

Examples

```
library(finalfit)
library(dplyr)
```

```

# Summary, univariable and multivariable analyses of the form:
# glm(dependent ~ explanatory, family="binomial")
# lmer(), lmmulti(), lmmixed(), glmer(), glmmulti(), glmmixed(), glmmultiboot(),
# coxphuni(), coxphmulti()

data(colon_s) # Modified from survival::colon
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = 'mort_5yr'
colon_s %>%
  finalfit(dependent, explanatory)

# Multivariable analysis with subset of explanatory
# variable set used in univariable analysis
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
explanatory_multi = c("age.factor", "obstruct.factor")
dependent = "mort_5yr"
colon_s %>%
  finalfit(dependent, explanatory, explanatory_multi)

# Summary, univariable and multivariable analyses of the form:
# lme4::glmer(dependent ~ explanatory + (1 | random_effect), family="binomial")

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
explanatory_multi = c("age.factor", "obstruct.factor")
random_effect = "hospital"
dependent = "mort_5yr"
# colon_s %>%
#   finalfit(dependent, explanatory, explanatory_multi, random_effect)

# Include model metrics:
colon_s %>%
  finalfit(dependent, explanatory, explanatory_multi, metrics=TRUE)

# Summary, univariable and multivariable analyses of the form:
# survival::coxph(dependent ~ explanatory)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "Surv(time, status)"

colon_s %>%
  finalfit(dependent, explanatory)

# Rather than going all-in-one, any number of subset models can
# be manually added on to a summary_factorlist() table using finalfit.merge().
# This is particularly useful when models take a long-time to run or are complicated.

# Note requirement for fit_id=TRUE.
# `fit2df` is a subfunction extracting most common models to a dataframe.

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = 'mort_5yr'
colon_s %>%
  finalfit(dependent, explanatory, metrics=TRUE)

```

```

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
explanatory_multi = c("age.factor", "obstruct.factor")
random_effect = "hospital"
dependent = 'mort_5yr'

# Separate tables
colon_s %>%
  summary_factorlist(dependent, explanatory, fit_id=TRUE) -> example.summary

colon_s %>%
  glmuni(dependent, explanatory) %>%
  fit2df(estimate_suffix=" (univariable)") -> example.univariable

colon_s %>%
  glmmulti(dependent, explanatory) %>%
  fit2df(estimate_suffix=" (multivariable)") -> example.multivariable

# Edited as CRAN slow to run these
# colon_s %>%
#   glmmixed(dependent, explanatory, random_effect) %>%
#   fit2df(estimate_suffix=" (multilevel)") -> example.multilevel

# Pipe together
example.summary %>%
  finalfit_merge(example.univariable) %>%
  finalfit_merge(example.multivariable, last_merge = TRUE)
# finalfit_merge(example.multilevel)

```

fit2df	<i>Extract model fit results to dataframe (generic): finalfit model extractors</i>
--------	--

Description

Takes output from finalfit model wrappers and extracts to a dataframe, convenient for further processing in preparation for final results table.

fit2df.lm is the model extract method for [lm](#).

fit2df.lmlist is the model extract method for lmuni and lmmulti.

fit2df.glm is the model extract method for standard [glm](#) models, which have not used finalfit model wrappers.

fit2df.glmboot is the model extract method for [glmmulti_boot](#) models.

fit2df.glmlist is the model extract method for glmuni and glmmulti.

fit2df.svyglm is the model extract method for svyglmuni and svyglmmulti.

fit2df.lmerMod is the model extract method for standard lme4::lmer models and for the finalfit::lmmixed model wrapper.

fit2df.glmerMod is the model extract method for standard lme4::glmer models and for the finalfit::glmmixed model wrapper.

fit2df.coxph is the model extract method for survival::coxph.

fit2df.coxphlist is the model extract method for coxphuni and coxphmulti.

fit2df.crr is the model extract method for cmprsk::crr.

fit2df.coxme is the model extract method for eoxme::coxme.

fit2df.crr is the model extract method for crruni and crrmulti.

fit2df.stanfit is the model extract method for our standard Bayesian hierarchical binomial logistic regression models. These models will be fully documented separately. However this should work for a single or multilevel Bayesian logistic regression done in Stan, as long as the fixed effects are specified in the parameters block as a vector named beta, of length P, where P is the number of fixed effect parameters. e.g. parameters(vector[P] beta;)

fit2df.mipo is the model extract method for the mipo object created using mice::pool.

Usage

```
fit2df(...)
```

```
## S3 method for class 'lm'
```

```
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  remove_intercept = TRUE,
  explanatory_name = "explanatory",
  estimate_name = "Coefficient",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  confint_level = 0.95,
  confint_sep = " to ",
  ...
)
```

```
## S3 method for class 'lmlist'
```

```
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  remove_intercept = TRUE,
  explanatory_name = "explanatory",
  estimate_name = "Coefficient",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  confint_level = 0.95,
  confint_sep = " to ",
)
```

```

    ...
)

## S3 method for class 'glm'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  remove_intercept = TRUE,
  explanatory_name = "explanatory",
  estimate_name = "OR",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  exp = TRUE,
  confint_type = "profile",
  confint_level = 0.95,
  confint_sep = "-",
  ...
)

## S3 method for class 'glmboot'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  remove_intercept = TRUE,
  explanatory_name = "explanatory",
  estimate_name = "OR",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  exp = TRUE,
  confint_level = 0.95,
  confint_sep = "-",
  ...
)

## S3 method for class 'glmmlist'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  remove_intercept = TRUE,
  explanatory_name = "explanatory",
  estimate_name = "OR",
  estimate_suffix = "",
  p_name = "p",

```

```

    digits = c(2, 2, 3),
    exp = TRUE,
    confint_type = "profile",
    confint_level = 0.95,
    confint_sep = "-",
    ...
)

## S3 method for class 'svyglmList'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  remove_intercept = TRUE,
  explanatory_name = "explanatory",
  estimate_name = "Coefficient",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  exp = FALSE,
  confint_type = "profile",
  confint_level = 0.95,
  confint_sep = "-",
  ...
)

## S3 method for class 'lmerMod'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  remove_intercept = TRUE,
  explanatory_name = "explanatory",
  estimate_name = "Coefficient",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  confint_type = "Wald",
  confint_level = 0.95,
  confint_sep = " to ",
  ...
)

## S3 method for class 'glmerMod'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,

```

```

        remove_intercept = TRUE,
        explanatory_name = "explanatory",
        estimate_name = "OR",
        estimate_suffix = "",
        p_name = "p",
        digits = c(2, 2, 3),
        exp = TRUE,
        confint_type = "Wald",
        confint_level = 0.95,
        confint_sep = "-",
        ...
    )

## S3 method for class 'coxph'
fit2df(
    .data,
    condense = TRUE,
    metrics = FALSE,
    explanatory_name = "explanatory",
    estimate_name = "HR",
    estimate_suffix = "",
    p_name = "p",
    digits = c(2, 2, 3),
    confint_sep = "-",
    ...
)

## S3 method for class 'coxphlist'
fit2df(
    .data,
    condense = TRUE,
    metrics = FALSE,
    explanatory_name = "explanatory",
    estimate_name = "HR",
    estimate_suffix = "",
    p_name = "p",
    digits = c(2, 2, 3),
    confint_sep = "-",
    ...
)

## S3 method for class 'crr'
fit2df(
    .data,
    condense = TRUE,
    metrics = FALSE,
    explanatory_name = "explanatory",
    estimate_name = "HR",

```

```

    estimate_suffix = "",
    p_name = "p",
    digits = c(2, 2, 3),
    confint_sep = "-",
    ...
)

## S3 method for class 'coxme'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  explanatory_name = "explanatory",
  estimate_name = "HR",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  confint_sep = "-",
  ...
)

## S3 method for class 'crrlist'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  explanatory_name = "explanatory",
  estimate_name = "HR",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  confint_sep = "-",
  ...
)

## S3 method for class 'stanfit'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  remove_intercept = TRUE,
  explanatory_name = "explanatory",
  estimate_name = "OR",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  confint_sep = "-",
  ...
)

```

```

)

## S3 method for class 'mipo'
fit2df(
  .data,
  condense = TRUE,
  metrics = FALSE,
  remove_intercept = TRUE,
  explanatory_name = "explanatory",
  estimate_name = "Coefficient",
  estimate_suffix = "",
  p_name = "p",
  digits = c(2, 2, 3),
  exp = FALSE,
  confint_level = 0.95,
  confint_sep = "-",
  ...
)

```

Arguments

...	Other arguments: X: Design matrix from stanfit modelling. Details documented else where.
.data	Output from finalfit model wrappers.
condense	Logical: when true, effect estimates, confidence intervals and p-values are pasted conveniently together in single cell.
metrics	Logical: when true, useful model metrics are extracted.
remove_intercept	Logical: remove the results for the intercept term.
explanatory_name	Name for this column in output
estimate_name	Name for this column in output
estimate_suffix	Appended to estimate name
p_name	Name given to p-value estimate
digits	Number of digits to round to (1) estimate, (2) confidence interval limits, (3) p-value.
confint_level	The confidence level required.
confint_sep	String to separate confidence intervals, typically "-" or " to ".
exp	Currently GLM only. Exponentiate coefficients and confidence intervals. Defaults to TRUE.
confint_type	One of c("profile", "default") for GLM models (confint.glm) or c("profile", "Wald", "boot") for glmer/lmer models (confint.merMod.). Not implemented for lm, coxph or coxphlist.

Details

fit2df is a generic (S3) function for model extract.

Value

A dataframe of model parameters. When metrics=TRUE output is a list of two dataframes, one is model parameters, one is model metrics. length two

Examples

```
library(finalfit)
library(dplyr)
library(survival)
# glm
fit = glm(mort_5yr ~ age.factor + sex.factor + obstruct.factor + perfor.factor,
  data=colon_s, family="binomial")
fit %>%
  fit2df(estimate_suffix=" (multivariable)")

# glmlist
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"
colon_s %>%
  glmmulti(dependent, explanatory) %>%
  fit2df(estimate_suffix=" (univariable)")

# glmerMod
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
random_effect = "hospital"
dependent = "mort_5yr"
colon_s %>%
  glmmixed(dependent, explanatory, random_effect) %>%
  fit2df(estimate_suffix=" (multilevel)")

# glmboot
## Note number of draws set to 100 just for speed in this example
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"
colon_s %>%
  glmulti_boot(dependent, explanatory, R = 100) %>%
  fit2df(estimate_suffix=" (multivariable (BS CIs))")

# lm
fit = lm(nodes ~ age.factor + sex.factor + obstruct.factor + perfor.factor,
  data=colon_s)
fit %>%
  fit2df(estimate_suffix=" (multivariable)")

# lmerMod
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
random_effect = "hospital"
dependent = "nodes"
```

```

colon_s %>%
  lmmixed(dependent, explanatory, random_effect) %>%
  fit2df(estimate_suffix=" (multilevel)")

# coxphlist
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "Surv(time, status)"

colon_s %>%
  coxphuni(dependent, explanatory) %>%
  fit2df(estimate_suffix=" (univariable)")

colon_s %>%
  coxphmulti(dependent, explanatory) %>%
  fit2df(estimate_suffix=" (multivariable)")

# coxph
fit = coxph(Surv(time, status) ~ age.factor + sex.factor + obstruct.factor + perfor.factor,
  data = colon_s)

fit %>%
  fit2df(estimate_suffix=" (multivariable)")

# crr: competing risks
melanoma = boot::melanoma
melanoma = melanoma %>%
  mutate(
    status_crr = ifelse(status == 2, 0, # "still alive"
      ifelse(status == 1, 1, # "died of melanoma"
        2)), # "died of other causes"
    sex = factor(sex),
    ulcer = factor(ulcer)
  )

dependent = c("Surv(time, status_crr)")
explanatory = c("sex", "age", "ulcer")
melanoma %>%
  summary_factorlist(dependent, explanatory, column = TRUE, fit_id = TRUE) %>%
  ff_merge(
    melanoma %>%
      crrmulti(dependent, explanatory) %>%
      fit2df(estimate_suffix = " (competing risks)")
  ) %>%
  select(-fit_id, -index) %>%
  dependent_label(melanoma, dependent)

```

Description

Internal, function, not called directly

Usage

```
format_n_percent(n, percent, digits, digits_n = 0, na_include = TRUE)
```

Arguments

n	Value
percent	Value
digits	Value
digits_n	Value. Used when using weighted frequency counts
na_include	When proportion missing, include in parentheses?

glmmixed	<i>Mixed effects binomial logistic regression models: finalfit model wrapper</i>
----------	--

Description

Using finalfit conventions, produces mixed effects binomial logistic regression models for a set of explanatory variables against a binary dependent.

Usage

```
glmmixed(.data, dependent, explanatory, random_effect, ...)
```

Arguments

.data	Dataframe.
dependent	Character vector of length 1, name of dependent variable (must have 2 levels).
explanatory	Character vector of any length: name(s) of explanatory variables.
random_effect	Character vector of length 1, either, (1) name of random intercept variable, e.g. "var1", (automatically converted to "(1 var1)"); or, (2) the full lme4 specification, e.g. "(var1 var2)". Note parenthesis MUST be included in (2) but NOT included in (1).
...	Other arguments to pass to lme4: glmer .

Details

Uses lme4::[glmer](#) with finalfit modelling conventions. Output can be passed to [fit2df](#). This is only currently set-up to take a single random effect as a random intercept. Can be updated in future to allow multiple random intercepts, random gradients and interactions on random effects if there is a need

Value

A list of multivariable lme4: [glmer](#) fitted model outputs. Output is of class glmerMod.

See Also

[fit2df](#), [finalfit_merge](#)

Other finalfit model wrappers: [coxphmulti\(\)](#), [coxphuni\(\)](#), [crrmulti\(\)](#), [crruni\(\)](#), [glmmulti\(\)](#), [glmmulti_boot\(\)](#), [glmuni\(\)](#), [lmmixed\(\)](#), [lmmulti\(\)](#), [lmuni\(\)](#), [svyglmmulti\(\)](#), [svyglmuni\(\)](#)

Examples

```
library(finalfit)
library(dplyr)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
random_effect = "hospital"
dependent = "mort_5yr"

colon_s %>%
  glmmixed(dependent, explanatory, random_effect) %>%
  fit2df(estimate_suffix=" (multilevel)")
```

glmmulti	<i>Binomial logistic regression multivariable models: finalfit model wrapper</i>
----------	--

Description

Using finalfit conventions, produces a multivariable binomial logistic regression model for a set of explanatory variables against a binary dependent.

Usage

```
glmmulti(.data, dependent, explanatory, family = "binomial", weights = "", ...)
```

Arguments

.data	Data frame.
dependent	Character vector of length 1: name of dependent variable (must have 2 levels).
explanatory	Character vector of any length: name(s) of explanatory variables.
family	Character vector quoted or unquoted of the error distribution and link function to be used in the model, see glm .
weights	Character vector of length 1: name of variabe for weighting. 'Prior weights' to be used in the fitting process.
...	Other arguments to pass to glm .

Details

Uses `glm` with `finalfit` modelling conventions. Output can be passed to `fit2df`.

Value

A multivariable `glm` fitted model.

See Also

`fit2df`, `finalfit_merge`

Other `finalfit` model wrappers: `coxphmulti()`, `coxphuni()`, `crrmulti()`, `crruni()`, `glmmixed()`, `glmmulti_boot()`, `glmuni()`, `lmmixed()`, `lmmulti()`, `lmuni()`, `svyglmmulti()`, `svyglmuni()`

Examples

```
library(finalfit)
library(dplyr)
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"

colon_s %>%
  glmmulti(dependent, explanatory) %>%
  fit2df(estimate_suffix=" (multivariable)")
```

glmmulti_boot	<i>Binomial logistic regression multivariable models with bootstrapped confidence intervals: finalfit model wrapper</i>
---------------	---

Description

Using `finalfit` conventions, produces a multivariable binomial logistic regression models for a set of explanatory variables against a binary dependent.

Usage

```
glmmulti_boot(.data, dependent, explanatory, R = 1000)
```

Arguments

<code>.data</code>	Dataframe.
<code>dependent</code>	Character vector length 1: name of dependent variable (must have 2 levels).
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>R</code>	Number of draws.

Details

Uses `glm` with `finalfit` modelling conventions. `boot::boot` is used to draw bootstrapped confidence intervals on fixed effect model coefficients. Output can be passed to `fit2df`.

Value

A multivariable `glm` fitted model with bootstrapped confidence intervals. Output is of class `glmboot`.

See Also

`fit2df`, `finalfit_merge`

Other `finalfit` model wrappers: `coxphmulti()`, `coxphuni()`, `crrmulti()`, `crruni()`, `glmmixed()`, `glmmulti()`, `glmuni()`, `lmmixed()`, `lmmulti()`, `lmuni()`, `svyglmmulti()`, `svyglmuni()`

Examples

```
library(finalfit)
library(dplyr)
## Note number of draws set to 100 just for speed in this example
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"

colon_s %>%
  glmmulti_boot(dependent, explanatory, R=100) %>%
  fit2df(estimate_suffix="(multivariable (BS CIs))")
```

glmuni	<i>Binomial logistic regression univariable models: finalfit model wrapper</i>
--------	--

Description

Using `finalfit` conventions, produces multiple univariable binomial logistic regression models for a set of explanatory variables against a binary dependent.

Usage

```
glmuni(.data, dependent, explanatory, family = "binomial", weights = "", ...)
```

Arguments

<code>.data</code>	Data frame.
<code>dependent</code>	Character vector of length 1: name of dependent variable (must have 2 levels).
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>family</code>	Character vector quoted or unquoted of the error distribution and link function to be used in the model, see <code>glm</code> .

`weights` Character vector of length 1: name of variable for weighting. 'Prior weights' to be used in the fitting process.

`...` Other arguments to pass to `glm`.

Details

Uses `glm` with `finalfit` modelling conventions. Output can be passed to `fit2df`.

Value

A list of univariable `glm` fitted model outputs. Output is of class `glm`list.

See Also

`fit2df`, `finalfit_merge`

Other `finalfit` model wrappers: `coxphmulti()`, `coxphuni()`, `crrmulti()`, `crruni()`, `glmmixed()`, `glmulti()`, `glmmulti_boot()`, `lmmixed()`, `lmmulti()`, `lmuni()`, `svyglmmulti()`, `svyglmuni()`

Examples

```
library(finalfit)
library(dplyr)
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"

colon_s %>%
  glmuni(dependent, explanatory) %>%
  fit2df(estimate_suffix=" (univariable)")
```

hr_plot

Produce a hazard ratio table and plot

Description

Produce hazard ratio table and plot from a Cox Proportional Hazards analysis, `survival::coxph()`.

Usage

```
hr_plot(
  .data,
  dependent,
  explanatory,
  factorlist = NULL,
  coxfit = NULL,
  remove_ref = FALSE,
  breaks = NULL,
```

```

column_space = c(-0.5, 0, 0.5),
dependent_label = "Survival",
prefix = "",
suffix = ": HR (95% CI, p-value)",
table_text_size = 4,
title_text_size = 13,
plot_opts = NULL,
table_opts = NULL,
...
)

```

Arguments

<code>.data</code>	Dataframe.
<code>dependent</code>	Character vector of length 1: name of survival object in form <code>Surv(time, status)</code> .
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>factorlist</code>	Option to provide output directly from <code>summary_factorlist()</code> .
<code>coxfit</code>	Option to provide output directly from <code>coxphmulti()</code> .
<code>remove_ref</code>	Logical. Remove reference level for factors.
<code>breaks</code>	Manually specify x-axis breaks in format <code>c(0.1, 1, 10)</code> .
<code>column_space</code>	Adjust table column spacing.
<code>dependent_label</code>	Main label for plot.
<code>prefix</code>	Plots are titled by default with the dependent variable. This adds text before that label.
<code>suffix</code>	Plots are titled with the dependent variable. This adds text after that label.
<code>table_text_size</code>	Alter font size of table text.
<code>title_text_size</code>	Alter font size of title text.
<code>plot_opts</code>	A list of arguments to be appended to the <code>ggplot</code> call by "+".
<code>table_opts</code>	A list of arguments to be appended to the <code>ggplot</code> table call by "+".
<code>...</code>	Other parameters passed to <code>fit2df()</code> .

Value

Returns a table and plot produced in `ggplot2`.

See Also

Other finalfit plot functions: `coefficient_plot()`, `ff_plot()`, `or_plot()`, `surv_plot()`

Examples

```
# HR plot
library(finalfit)
library(dplyr)
library(ggplot2)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "Surv(time, status)"
colon_s %>%
  hr_plot(dependent, explanatory, dependent_label = "Survival")

colon_s %>%
  hr_plot(dependent, explanatory, dependent_label = "Survival",
    table_text_size=4, title_text_size=14,
    plot_opts=list(xlab("HR, 95% CI"), theme(axis.title = element_text(size=12))))
```

labels_to_column	<i>Labels to column names</i>
------------------	-------------------------------

Description

Labels to column names

Usage

```
labels_to_column(.data)
```

Arguments

.data	Data frame or tibble.
-------	-----------------------

Value

Data frame or tibble

Examples

```
library(dplyr)
colon_s %>%
  select(sex.factor) %>%
  labels_to_column()
```

labels_to_level	<i>Labels to level</i>
-----------------	------------------------

Description

For use with `forcats::fct_relabel`.

Usage

```
labels_to_level(.data, .labels)
```

Arguments

<code>.data</code>	Data frame or tibble.
<code>.labels</code>	Output from <code>extract_variable_label</code> .

Value

Data frame or tibble

Examples

```
library(dplyr)
vlabels = extract_variable_label(colon_s)
colon_s %>%
  select(sex.factor, obstruct.factor) %>%
  tidyr::gather() %>%
  mutate(
    key = forcats::fct_relabel(key, labels_to_level, vlabels)
  )
```

lmmixed	<i>Mixed effects linear regression models: finalfit model wrapper</i>
---------	---

Description

Using `finalfit` conventions, produces mixed effects linear regression models for a set of explanatory variables against a continuous dependent.

Usage

```
lmmixed(.data, dependent, explanatory, random_effect, ...)
```

Arguments

<code>.data</code>	Dataframe.
<code>dependent</code>	Character vector of length 1, name of dependent variable (must be continuous vector).
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>random_effect</code>	Character vector of length 1, either, (1) name of random intercept variable, e.g. "var1", (automatically converted to "(1 var1)"); or, (2) the full lme4 specification, e.g. "(var1 var2)". Note parenthesis MUST be included in (2) but NOT included in (1).
<code>...</code>	Other arguments to pass to lme4: lmer .

Details

Uses lme4: [lmer](#) with finalfit modelling conventions. Output can be passed to [fit2df](#). This is only currently set-up to take a single random effect as a random intercept. Can be updated in future to allow multiple random intercepts, random gradients and interactions on random effects if there is a need.

Value

A list of multivariable lme4: [lmer](#) fitted model outputs. Output is of class lmerMod.

See Also

[fit2df](#)

Other finalfit model wrappers: [coxphmulti\(\)](#), [coxphuni\(\)](#), [crrmulti\(\)](#), [crruni\(\)](#), [glmmixed\(\)](#), [glmmulti\(\)](#), [glmmulti_boot\(\)](#), [glmuni\(\)](#), [lmmulti\(\)](#), [lmuni\(\)](#), [svyglmmulti\(\)](#), [svyglmuni\(\)](#)

Examples

```
library(finalfit)
library(dplyr)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
random_effect = "hospital"
dependent = "nodes"

colon_s %>%
  lmmixed(dependent, explanatory, random_effect) %>%
  fit2df(estimate_suffix=" (multilevel")
```

lmmulti

*Linear regression multivariable models: finalfit model wrapper***Description**

Using finalfit conventions, produces a multivariable linear regression model for a set of explanatory variables against a continuous dependent.

Usage

```
lmmulti(.data, dependent, explanatory, weights = "", ...)
```

Arguments

.data	Dataframe.
dependent	Character vector of length 1: name of dependent variable (must a continuous vector).
explanatory	Character vector of any length: name(s) of explanatory variables.
weights	Character vector of length 1: name of variable for weighting. 'Prior weights' to be used in the fitting process.
...	Other arguments to pass to lm .

Details

Uses [lm](#) with finalfit modelling conventions. Output can be passed to [fit2df](#).

Value

A multivariable [lm](#) fitted model.

See Also

[fit2df](#)

Other finalfit model wrappers: [coxphmulti\(\)](#), [coxphuni\(\)](#), [crrmulti\(\)](#), [crruni\(\)](#), [glmmixed\(\)](#), [glmmulti\(\)](#), [glmmulti_boot\(\)](#), [glmuni\(\)](#), [lmmixed\(\)](#), [lmuni\(\)](#), [svyglmmulti\(\)](#), [svyglmuni\(\)](#)

Examples

```
library(finalfit)
library(dplyr)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "nodes"

colon_s %>%
  lmmulti(dependent, explanatory) %>%
  fit2df()
```

lmuni

*Linear regression univariable models: finalfit model wrapper***Description**

Using finalfit conventions, produces multiple univariable linear regression models for a set of explanatory variables against a continuous dependent.

Usage

```
lmuni(.data, dependent, explanatory, weights = "", ...)
```

Arguments

.data	Dataframe.
dependent	Character vector of length 1, name of dependent variable (must be continuous vector).
explanatory	Character vector of any length: name(s) of explanatory variables.
weights	Character vector of length 1: name of variable for weighting. 'Prior weights' to be used in the fitting process.
...	Other arguments to pass to lm .

Details

Uses [lm](#) with finalfit modelling conventions. Output can be passed to [fit2df](#).

Value

A list of multivariable [lm](#) fitted model outputs. Output is of class `lm` list.

See Also

[fit2df](#)

Other finalfit model wrappers: [coxphmulti\(\)](#), [coxphuni\(\)](#), [crrmulti\(\)](#), [crruni\(\)](#), [glmmixed\(\)](#), [glmulti\(\)](#), [glmulti_boot\(\)](#), [glmuni\(\)](#), [lmmixed\(\)](#), [lmmulti\(\)](#), [svyglmulti\(\)](#), [svyglmuni\(\)](#)

Examples

```
library(finalfit)
library(dplyr)

explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "nodes"

colon_s %>%
  lmuni(dependent, explanatory) %>%
  fit2df()
```

metrics_hoslem	<i>Hosmer-Lemeshow goodness of fit test</i>
----------------	---

Description

Internal, not usually called directly

Usage

```
metrics_hoslem(y, yhat, g = 10, digits = c(2, 3))
```

Arguments

y	Observed y, usually of the form <code>fit\$y</code> .
yhat	Predicted <code>y_hat</code> , usually for the form <code>fit\$fitted</code>
g	Number of bins to calculate quantiles.
digits	Number of decimal places of form <code>c(2,3)</code> , where <code>digits[1]</code> is for chi-sq estimate and <code>digits[2]</code> is for p-value.

Value

Character string of chi-sq result, df, and p-value. Significant p-value suggests poor fit.

Author(s)

Adapted from Peter Solymos.

Source

<https://github.com/psolymos/ResourceSelection/blob/master/R/hoslem.test.R>

Examples

```
fit = glm(mort_5yr~age.factor+extent.factor, data=colon_s, family="binomial")
metrics_hoslem(fit$y, fit$fitted)
```

missing_compare	<i>Compare missing data</i>
-----------------	-----------------------------

Description

Compare missing data

Usage

```
missing_compare(
  .data,
  dependent,
  explanatory,
  p = TRUE,
  na_include = FALSE,
  ...
)
```

Arguments

.data	Dataframe.
dependent	Variable to test missingness against other variables with.
explanatory	Variables to have missingness tested against.
p	Logical: Include null hypothesis statistical test.
na_include	Include missing data in explanatory variables as a factor level.
...	Other arguments to summary_factorlist() .

Value

A dataframe comparing missing data in the dependent variable across explanatory variables. Continuous data are compared with an Analysis of Variance F-test by default. Discrete data are compared with a chi-squared test.

Examples

```
library(finalfit)

explanatory = c("age", "age.factor", "extent.factor", "perfor.factor")
dependent = "mort_5yr"

colon_s %>%
  ff_glimpse(dependent, explanatory)

colon_s %>%
  missing_pattern(dependent, explanatory)

colon_s %>%
  missing_compare(dependent, explanatory)
```

missing_glimpse	<i>Summary of missing values</i>
-----------------	----------------------------------

Description

Summary of missing values

Usage

```
missing_glimpse(.data, dependent = NULL, explanatory = NULL, digits = 1)
```

Arguments

.data	Data frame.
dependent	Optional character vector: name(s) of dependent variable(s).
explanatory	Optional character vector: name(s) of explanatory variable(s).
digits	Number of decimal places to show for percentage missing.

Value

Data frame.

Examples

```
colon_s %>%
  missing_glimpse()
```

missing_pairs	<i>Missing values pairs plot</i>
---------------	----------------------------------

Description

Compare the occurrence of missing values in all variables by each other. Suggest limit the number of variables to a maximum of around six. Dependent and explanatory are for convenience of variable selection, are optional, and have no other specific function.

Usage

```
missing_pairs(
  .data,
  dependent = NULL,
  explanatory = NULL,
  use_labels = TRUE,
  title = NULL,
  position = "stack",
  showXAxisPlotLabels = TRUE,
  showYAxisPlotLabels = FALSE
)
```

Arguments

<code>.data</code>	Data frame.
<code>dependent</code>	Character vector. Optional name of dependent variable.
<code>explanatory</code>	Character vector. Optional name(s) of explanatory variables.
<code>use_labels</code>	Use variable label names in plot labelling.
<code>title</code>	Character vector. Optional title for plot.
<code>position</code>	For discrete variables, choose "stack" or "fill" to show counts or proportions.
<code>showXAxisPlotLabels</code>	Show x-axis plot labels.
<code>showYAxisPlotLabels</code>	Show y-axis plot labels.

Value

A plot matrix comparing missing values in all variables against each other.

Examples

```
## Not run:
explanatory = c("age", "nodes", "age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = 'mort_5yr'
colon_s %>%
  missing_pairs(dependent, explanatory)

## End(Not run)
```

missing_pattern	<i>Characterise missing data for finalfit models</i>
-----------------	--

Description

Using finalfit conventions, produces a missing data matrix using `md.pattern`.

Usage

```
missing_pattern(
  .data,
  dependent = NULL,
  explanatory = NULL,
  rotate.names = TRUE,
  ...
)
```

Arguments

<code>.data</code>	Data frame. Missing values must be coded NA.
<code>dependent</code>	Character vector usually of length 1, name of dependent variable.
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>rotate.names</code>	Logical. Should the orientation of variable names on plot should be vertical.
<code>...</code>	pass other arguments such as <code>plot = TRUE</code> to md.pattern .

Value

A matrix with `ncol(x)+1` columns, in which each row corresponds to a missing data pattern (1=observed, 0=missing). Rows and columns are sorted in increasing amounts of missing information. The last column and row contain row and column counts, respectively.

Examples

```
library(finalfit)
library(dplyr)
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"

colon_s %>%
  missing_pattern(dependent, explanatory)
```

<code>missing_plot</code>	<i>Missing values occurrence plot</i>
---------------------------	---------------------------------------

Description

Create a plot of missing values by observations on the x-axis and variable on the y-axis. Dependent and explanatory are for convenience and are optional.

Usage

```
missing_plot(
  .data,
  dependent = NULL,
  explanatory = NULL,
  use_labels = TRUE,
  title = NULL,
  plot_opts = NULL
)
```

Arguments

<code>.data</code>	Data frame.
<code>dependent</code>	Character vector. Optional name of dependent variable.
<code>explanatory</code>	Character vector. Optional name(s) of explanatory variables.
<code>use_labels</code>	Use variable label names in plot labelling.
<code>title</code>	Character vector. Optional title for plot.
<code>plot_opts</code>	A list of arguments to be appended to the ggplot call by "+".

Value

Heat map of missing values in dataset.

Examples

```
colon_s %>%
  missing_plot()
```

missing_predictorMatrix

Create predictorMatrix for use with mice

Description

Create predictorMatrix for use with mice

Usage

```
missing_predictorMatrix(
  .data,
  drop_from_imputed = NULL,
  drop_from_imputer = NULL
)
```

Arguments

<code>.data</code>	Data frame.
<code>drop_from_imputed</code>	Quoted names of variables not to impute.
<code>drop_from_imputer</code>	Quoted names of variables not to use in imputation algorithm.

Value

Matrix formatted for predictorMatrix argument in mice.

Examples

```

library(mice)
library(dplyr)

# Create some extra missing data
## Smoking missing completely at random
set.seed(1)
colon_s$smoking_mcar =
  sample(c("Smoker", "Non-smoker", NA),
    dim(colon_s)[1], replace=TRUE,
    prob = c(0.2, 0.7, 0.1)) %>%
  factor() %>%
  ff_label("Smoking (MCAR)")

## Make smoking missing conditional on patient sex
colon_s$smoking_mar[colon_s$sex.factor == "Female"] =
  sample(c("Smoker", "Non-smoker", NA),
    sum(colon_s$sex.factor == "Female"),
    replace = TRUE, prob = c(0.1, 0.5, 0.4))

colon_s$smoking_mar[colon_s$sex.factor == "Male"] =
  sample(c("Smoker", "Non-smoker", NA),
    sum(colon_s$sex.factor == "Male"),
    replace=TRUE, prob = c(0.15, 0.75, 0.1))
colon_s$smoking_mar = factor(colon_s$smoking_mar)%>%
  ff_label("Smoking (MAR)")

explanatory = c("age", "sex.factor",
  "nodes", "obstruct.factor", "smoking_mar")
dependent = "mort_5yr"

colon_s %>%
  select(dependent, explanatory) %>%
  missing_predictorMatrix(drop_from_imputed =
    c("obstruct.factor", "mort_5yr")) -> predM

colon_s %>%
  select(dependent, explanatory) %>%
  mice(m = 2, predictorMatrix = predM) %>% # e.g. m=10 when for real
  # Run logistic regression on each imputed set
  with(glm(formula(ff_formula(dependent, explanatory)),
    family="binomial")) %>%
  pool() %>%
  summary(conf.int = TRUE, exponentiate = TRUE) %>%
  # Jiggle into finalfit format
  mutate(explanatory_name = rownames(.)) %>%
  select(explanatory_name, estimate, conf.low, conf.high, p.value) %>%
  condense_fit(estimate_suffix = " (multiple imputation)") %>%
  remove_intercept() -> fit_imputed

```

or_plot

*Produce an odds ratio table and plot***Description**

Produce an odds ratio table and plot from a `glm()` or `lme4::glmer()` model.

Usage

```
or_plot(
  .data,
  dependent,
  explanatory,
  random_effect = NULL,
  factorlist = NULL,
  glmfit = NULL,
  confint_type = NULL,
  confint_level = 0.95,
  remove_ref = FALSE,
  breaks = NULL,
  column_space = c(-0.5, 0, 0.5),
  dependent_label = NULL,
  prefix = "",
  suffix = NULL,
  table_text_size = 4,
  title_text_size = 13,
  plot_opts = NULL,
  table_opts = NULL,
  ...
)
```

Arguments

<code>.data</code>	Data frame.
<code>dependent</code>	Character vector of length 1: name of dependent variable (must have 2 levels).
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>random_effect</code>	Character vector of length 1, name of random effect variable.
<code>factorlist</code>	Option to provide output directly from <code>summary_factorlist()</code> .
<code>glmfit</code>	Option to provide output directly from <code>glmmulti()</code> and <code>glmmixed()</code> .
<code>confint_type</code>	One of <code>c("profile", "default")</code> for GLM models or <code>c("default", "Wald", "profile", "boot")</code> for glmer models.
<code>confint_level</code>	The confidence level required.
<code>remove_ref</code>	Logical. Remove reference level for factors.
<code>breaks</code>	Manually specify x-axis breaks in format <code>c(0.1, 1, 10)</code> .

column_space	Adjust table column spacing.
dependent_label	Main label for plot.
prefix	Plots are titled by default with the dependent variable. This adds text before that label.
suffix	Plots are titled with the dependent variable. This adds text after that label.
table_text_size	Alter font size of table text.
title_text_size	Alter font size of title text.
plot_opts	A list of arguments to be appended to the ggplot call by "+".
table_opts	A list of arguments to be appended to the ggplot table call by "+".
...	Other parameters.

Value

Returns a table and plot produced in ggplot2.

See Also

Other finalfit plot functions: [coefficient_plot\(\)](#), [ff_plot\(\)](#), [hr_plot\(\)](#), [surv_plot\(\)](#)

Examples

```
library(finalfit)
library(dplyr)
library(ggplot2)

# OR plot
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"
colon_s %>%
  or_plot(dependent, explanatory)

colon_s %>%
  or_plot(dependent, explanatory, table_text_size=4, title_text_size=14,
    plot_opts=list(xlab("OR, 95% CI"), theme(axis.title = element_text(size=12))))
```

p_tidy	<i>Round p-values but keep trailing zeros</i>
--------	---

Description

Internal function, not called directly

Usage

```
p_tidy(x, digits, prefix = "=")
```

Arguments

x	Numeric vector of values to round
digits	Integer of length one: value to round to.
prefix	Appended in front of values for use with condense_fit.

Details

e.g. for 3 decimal places I want 0.100, not 0.1. Note this function with convert 0.000 to <0.001. All other values are prefixed with "=" by default

Value

Vector of strings.

rm_duplicates	<i>Remove duplicates and replace</i>
---------------	--------------------------------------

Description

Remove duplicates and replace

Usage

```
rm_duplicates(.var, fromLast = FALSE, replacement = "")
```

Arguments

.var	Vector.
fromLast	Logical. Consider duplication from last to first.
replacement	Character for what to replace duplicate with.

Value

Character vector.

rm_empty_block	<i>Remove rows where all specified variables are missing</i>
----------------	--

Description

It is common to want to remove cases/rows where all variables in a particular set are missing, e.g. all symptom variables are missing in a health care dataset.

Usage

```
rm_empty_block(.data, ...)
```

Arguments

.data	Dataframe.
...	Unquoted variable/column names.

Value

Data frame.

Examples

```
# Pretend that we want to remove rows that are missing in group1, group2, and group3
# but keep rest of dataset.
colon_s %>%
  dplyr::mutate(
    group1 = rep(c(NA, 1), length.out = 929),
    group2 = rep(c(NA, 1), length.out = 929),
    group3 = rep(c(NA, 1), length.out = 929)
  ) %>%
  rm_empty_block(group1, group2, group3) %>%
  head()
```

round_tidy	<i>Round values but keep trailing zeros</i>
------------	---

Description

e.g. for 3 decimal places I want 1.200, not 1.2.

Usage

```
round_tidy(x, digits)
```

Arguments

x Numeric vector of values to round
 digits Integer of length one: value to round to.

Value

Vector of strings.

Examples

```
round_tidy(0.01023, 3)
```

summary_df	<i>Summarise with mode for factors and mean/median for numeric variables</i>
------------	--

Description

When producing conditional estimates from a regression model, it is often useful to set variables not of interest to their mode for factors and mean or median for numerics when creating the newdata object.

Usage

```
summary_df(.data, cont = "mean")
```

Arguments

.data A data frame or tibble.
 cont One of "mean" or "median": the summary estimate for continuous variables.

Value

A data frame or tibble with the mode for factors and mean/median for continuous variables.

See Also

[ff_mode](#) [ff_expand](#)

Examples

```
library(dplyr)
colon_s %>%
  select(age, sex.factor, obstruct.factor, perfor.factor) %>%
  summary_df()

colon_s %>%
  select(age, sex.factor, obstruct.factor, perfor.factor) %>%
  summary_df(cont = "median")
```

summary_factorlist	<i>Summarise a set of factors (or continuous variables) by a dependent variable</i>
--------------------	---

Description

A function that takes a single dependent variable with a vector of explanatory variable names (continuous or categorical variables) to produce a summary table.

Usage

```
summary_factorlist(
  .data,
  dependent = NULL,
  explanatory = NULL,
  formula = NULL,
  cont = "mean",
  cont_nonpara = NULL,
  cont_cut = 5,
  cont_range = TRUE,
  p = FALSE,
  p_cont_para = "aov",
  p_cat = "chisq",
  column = TRUE,
  total_col = FALSE,
  orderbytotal = FALSE,
  digits = c(1, 1, 3, 1, 0),
  na_include = FALSE,
  na_include_dependent = FALSE,
  na_complete_cases = FALSE,
  na_to_p = FALSE,
  na_to_prop = TRUE,
  fit_id = FALSE,
  add_dependent_label = FALSE,
  dependent_label_prefix = "Dependent: ",
  dependent_label_suffix = "",
  add_col_totals = FALSE,
  include_col_totals_percent = TRUE,
  col_totals_rowname = NULL,
  col_totals_prefix = "",
  add_row_totals = FALSE,
  include_row_totals_percent = TRUE,
  include_row_missing_col = TRUE,
  row_totals_colname = "Total N",
  row_missing_colname = "Missing N",
  catTest = NULL,
  weights = NULL
```

)

Arguments

<code>.data</code>	Dataframe.
<code>dependent</code>	Character vector of length 1: name of dependent variable (2 to 5 factor levels).
<code>explanatory</code>	Character vector of any length: name(s) of explanatory variables.
<code>formula</code>	an object of class "formula" (or one that can be coerced to that class). Optional instead of standard dependent/explanatory format. Do not include if using dependent/explanatory.
<code>cont</code>	Summary for continuous explanatory variables: "mean" (standard deviation) or "median" (interquartile range). If "median" then non-parametric hypothesis test performed (see below).
<code>cont_nonpara</code>	Numeric vector of form e.g. <code>c(1, 2)</code> . Specify which variables to perform non-parametric hypothesis tests on and summarise with "median".
<code>cont_cut</code>	Numeric: number of unique values in continuous variable at which to consider it a factor.
<code>cont_range</code>	Logical. Median is show with 1st and 3rd quartiles.
<code>p</code>	Logical: Include null hypothesis statistical test.
<code>p_cont_para</code>	Character. Continuous variable parametric test. One of either "aov" (analysis of variance) or "t.test" for Welch two sample t-test. Note continuous non-parametric test is always Kruskal Wallis (<code>kruskal.test</code>) which in two-group setting is equivalent to Mann-Whitney U /Wilcoxon rank sum test. For continous dependent and continuous explanatory, the parametric test p-value returned is for the Pearson correlation coefficient. The non-parametric equivalent is for the p-value for the Spearman correlation coefficient.
<code>p_cat</code>	Character. Categorical variable test. One of either "chisq" or "fisher".
<code>column</code>	Logical: Compute margins by column rather than row.
<code>total_col</code>	Logical: include a total column summing across factor levels.
<code>orderbytotal</code>	Logical: order final table by total column high to low.
<code>digits</code>	Number of digits to round to (1) mean/median, (2) standard deviation / interquartile range, (3) p-value, (4) count percentage, (5) weighted count.
<code>na_include</code>	Logical: make explanatory variables missing data explicit (NA).
<code>na_include_dependent</code>	Logical: make dependent variable missing data explicit.
<code>na_complete_cases</code>	Logical: include only rows with complete data.
<code>na_to_p</code>	Logical: include missing as group in statistical test.
<code>na_to_prop</code>	Logical: include missing in calculation of column proportions.
<code>fit_id</code>	Logical: allows merging via finalfit_merge .
<code>add_dependent_label</code>	Add the name of the dependent label to the top left of table.

dependent_label_prefix	Add text before dependent label.
dependent_label_suffix	Add text after dependent label.
add_col_totals	Logical. Include column total n.
include_col_totals_percent	Include column percentage of total.
col_totals_rowname	Logical. Row name for column totals.
col_totals_prefix	Character. Prefix to column totals, e.g. "N=".
add_row_totals	Logical. Include row totals. Note this differs from total_col above particularly for continuous explanatory variables.
include_row_totals_percent	Include row percentage of total.
include_row_missing_col	Logical. Include missing data total for each row. Only used when add_row_totals is TRUE.
row_totals_colname	Character. Column name for row totals.
row_missing_colname	Character. Column name for missing data totals for each row.
catTest	Deprecated. See p_cat above.
weights	Character vector of length 1: name of column to use for weights. Explanatory continuous variables are multiplied by weights. Explanatory categorical variables are counted with a frequency weight (sum(weights)).

Details

This function aims to produce publication-ready summary tables for categorical or continuous dependent variables. It usually takes a categorical dependent variable to produce a cross table of counts and proportions expressed as percentages or summarised continuous explanatory variables. However, it will take a continuous dependent variable to produce mean (standard deviation) or median (interquartile range) for use with linear regression models.

Value

Returns a factorlist dataframe.

See Also

`fit2df` `ff_column_totals` `ff_row_totals` `ff_label` `ff_glimpse` `ff_percent_only`. For lots of examples, see <https://finalfit.org/>

Examples

```
library(finalfit)
library(dplyr)
# Load example dataset, modified version of survival::colon
data(colon_s)

# Table 1 - Patient demographics ----
explanatory = c("age", "age.factor", "sex.factor", "obstruct.factor")
dependent = "perfor.factor"
colon_s %>%
  summary_factorlist(dependent, explanatory, p=TRUE)

# summary_factorlist() is also commonly used to summarise any number of
# variables by an outcome variable (say dead yes/no).

# Table 2 - 5 yr mortality ----
explanatory = c("age.factor", "sex.factor", "obstruct.factor", "perfor.factor")
dependent = "mort_5yr"
colon_s %>%
  summary_factorlist(dependent, explanatory)
```

summary_factorlist_stratified

Summarise a set of factors (or continuous variables) by a dependent variable

Description

A function that takes a single dependent variable with a vector of explanatory variable names (continuous or categorical variables) to produce a summary table.

Usage

```
summary_factorlist_stratified(
  .data,
  ...,
  split,
  colname_sep = "|",
  level_max_length = 10,
  n_common_cols = 2
)
```

Arguments

.data	Dataframe.
...	Arguments to summary_factorlist .
split	Quoted variable name to stratify columns by.

- colname_sep Separator for creation of new column name.
- level_max_length Maximum name for each factor level contributing to column name.
- n_common_cols Number of common columns in [summary_factorlist](#) table, usually 2.

Details

This function aims to produce publication-ready summary tables for categorical or continuous dependent variables. It usually takes a categorical dependent variable to produce a cross table of counts and proportions expressed as percentages or summarised continuous explanatory variables. However, it will take a continuous dependent variable to produce mean (standard deviation) or median (interquartile range) for use with linear regression models. Stratify a [summary_factorlist](#) table (beta testing)

Value

Dataframe.

Examples

```
# Table 1 - Perforation status stratified by sex ----
explanatory = c("age", "obstruct.factor")
dependent = "perfor.factor"

# Single split
colon_s %>%
  summary_factorlist_stratified(dependent, explanatory, split = c("sex.factor"))

# Double split
colon_s %>%
  summary_factorlist_stratified(dependent, explanatory, split = c("sex.factor", "age.factor"))
```

surv_plot	<i>Plot survival curves with number-at-risk table</i>
-----------	---

Description

Produce a survival curve plot and number-at-risk table using `survminer::ggsurvplot` and `finalfit` conventions.

Usage

```
surv_plot(.data, dependent, explanatory, ...)
```

Arguments

- .data Dataframe.
- dependent Character vector of length 1: Survival object of the form `Surv(time, status)`.
- explanatory Character vector of max length 2: quoted name(s) of explanatory variables.
- ... Arguments passed to [ggsurvplot](#).

Value

Returns a table and plot produced in ggplot2.

See Also

Other finalfit plot functions: [coefficient_plot\(\)](#), [ff_plot\(\)](#), [hr_plot\(\)](#), [or_plot\(\)](#)

Examples

```
library(finalfit)
library(dplyr)

# Survival plot
data(colon_s)
explanatory = c("perfor.factor")
dependent = "Surv(time, status)"
colon_s %>%
  surv_plot(dependent, explanatory, xlab="Time (days)", pval=TRUE, legend="none")
```

svyglmmulti

Multivariable survey-weighted generalised linear models

Description

Wrapper for [svyglm](#). Fit a generalised linear model to data from a complex survey design, with inverse-probability weighting and design-based standard errors.

Usage

```
svyglmmulti(design, dependent, explanatory, ...)
```

Arguments

design	Survey design.
dependent	Character vector of length 1: name of dependent variable (must have 2 levels).
explanatory	Character vector of any length: name(s) of explanatory variables.
...	Other arguments to be passed to svyglm .

Value

A list of univariable fitted model outputs. Output is of class `svyglm`list.

See Also

[fit2df](#), [finalfit_merge](#)

Other finalfit model wrappers: [coxphmulti\(\)](#), [coxphuni\(\)](#), [crrmulti\(\)](#), [crruni\(\)](#), [glmmixed\(\)](#), [glmulti\(\)](#), [glmulti_boot\(\)](#), [glmuni\(\)](#), [lmmixed\(\)](#), [lmmulti\(\)](#), [lmuni\(\)](#), [svyglmuni\(\)](#)

Examples

```
# Examples taken from survey::svyglm() help page.

library(survey)
library(dplyr)

data(api)
dependent = "api00"
explanatory = c("ell", "meals", "mobility")

library(survey)
library(dplyr)

data(api)

apistrat = apistrat %>%
  mutate(
    api00 = ff_label(api00, "API in 2000 (api00)"),
    ell = ff_label(ell, "English language learners (percent)(ell)"),
    meals = ff_label(meals, "Meals eligible (percent)(meals)"),
    mobility = ff_label(mobility, "First year at the school (percent)(mobility)"),
    sch.wide = ff_label(sch.wide, "School-wide target met (sch.wide)")
  )

# Linear example
dependent = "api00"
explanatory = c("ell", "meals", "mobility")

# Stratified design
dstrat = svydesign(id=~1, strata=~stype, weights=~pw, data=apistrat, fpc=~fpc)

# Univariable fit
fit_uni = dstrat %>%
  svyglmuni(dependent, explanatory) %>%
  fit2df(estimate_suffix = " (univariable)")

# Multivariable fit
fit_multi = dstrat %>%
  svyglmmulti(dependent, explanatory) %>%
  fit2df(estimate_suffix = " (multivariable)")

# Pipe together
apistrat %>%
  summary_factorlist(dependent, explanatory, fit_id = TRUE) %>%
  ff_merge(fit_uni) %>%
  ff_merge(fit_multi) %>%
  select(-fit_id, -index) %>%
  dependent_label(apistrat, dependent)

# Binomial example
## Note model family needs specified and exponentiation if desired
```

```

dependent = "sch.wide"
explanatory = c("ell", "meals", "mobility")

# Univariable fit
fit_uni = dstrat %>%
  svyglmuni(dependent, explanatory, family = "quasibinomial") %>%
  fit2df(exp = TRUE, estimate_name = "OR", estimate_suffix = " (univariable)")

# Multivariable fit
fit_multi = dstrat %>%
  svyglmuni(dependent, explanatory, family = "quasibinomial") %>%
  fit2df(exp = TRUE, estimate_name = "OR", estimate_suffix = " (multivariable)")

# Pipe together
apistrat %>%
  summary_factorlist(dependent, explanatory, fit_id = TRUE) %>%
  ff_merge(fit_uni) %>%
  ff_merge(fit_multi) %>%
  select(-fit_id, -index) %>%
  dependent_label(apistrat, dependent)

```

svyglmuni

*Univariable survey-weighted generalised linear models***Description**

Wrapper for [svyglm](#). Fit a generalised linear model to data from a complex survey design, with inverse-probability weighting and design-based standard errors.

Usage

```
svyglmuni(design, dependent, explanatory, ...)
```

Arguments

design	Survey design.
dependent	Character vector of length 1: name of dependent variable (must have 2 levels).
explanatory	Character vector of any length: name(s) of explanatory variables.
...	Other arguments to be passed to svyglm .

Value

A list of univariable fitted model outputs. Output is of class `svyglm`.

See Also

[fit2df](#), [finalfit_merge](#)

Other finalfit model wrappers: [coxphmulti](#)(), [coxphuni](#)(), [crrmulti](#)(), [crruni](#)(), [glmrmixed](#)(), [glmrmulti](#)(), [glmrmulti_boot](#)(), [glmuni](#)(), [lmmixed](#)(), [lmmulti](#)(), [lmuni](#)(), [svyglmrmulti](#)()

Examples

```
# Examples taken from survey::svyglm() help page.

library(survey)
library(dplyr)

data(api)
dependent = "api00"
explanatory = c("ell", "meals", "mobility")

library(survey)
library(dplyr)

data(api)

apistrat = apistrat %>%
  mutate(
    api00 = ff_label(api00, "API in 2000 (api00)"),
    ell = ff_label(ell, "English language learners (percent)(ell)"),
    meals = ff_label(meals, "Meals eligible (percent)(meals)"),
    mobility = ff_label(mobility, "First year at the school (percent)(mobility)"),
    sch.wide = ff_label(sch.wide, "School-wide target met (sch.wide)")
  )

# Linear example
dependent = "api00"
explanatory = c("ell", "meals", "mobility")

# Stratified design
dstrat = svydesign(id=~1, strata=~stype, weights=~pw, data=apistrat, fpc=~fpc)

# Univariable fit
fit_uni = dstrat %>%
  svyglmuni(dependent, explanatory) %>%
  fit2df(estimate_suffix = " (univariable)")

# Multivariable fit
fit_multi = dstrat %>%
  svyglmuni(dependent, explanatory) %>%
  fit2df(estimate_suffix = " (multivariable)")

# Pipe together
apistrat %>%
  summary_factorlist(dependent, explanatory, fit_id = TRUE) %>%
  ff_merge(fit_uni) %>%
  ff_merge(fit_multi) %>%
  select(-fit_id, -index) %>%
  dependent_label(apistrat, dependent)

# Binomial example
## Note model family needs specified and exponentiation if desired
```

```

dependent = "sch.wide"
explanatory = c("ell", "meals", "mobility")

# Univariable fit
fit_uni = dstrat %>%
  svyglmuni(dependent, explanatory, family = "quasibinomial") %>%
  fit2df(exp = TRUE, estimate_name = "OR", estimate_suffix = " (univariable)")

# Multivariable fit
fit_multi = dstrat %>%
  svyglmuni(dependent, explanatory, family = "quasibinomial") %>%
  fit2df(exp = TRUE, estimate_name = "OR", estimate_suffix = " (multivariable)")

# Pipe together
apistrat %>%
  summary_factorlist(dependent, explanatory, fit_id = TRUE) %>%
  ff_merge(fit_uni) %>%
  ff_merge(fit_multi) %>%
  select(-fit_id, -index) %>%
  dependent_label(apistrat, dependent)

```

wcgs

Western Collaborative Group Study

Description

3154 healthy young men aged 39-59 from the San Francisco area were assessed for their personality type. All were free from coronary heart disease at the start of the research. Eight and a half years later change in this situation was recorded.

Usage

```
data(wcgs)
```

Format

A data frame with 3154 observations on the following 13 variables.

id Subject ID

age Age: age in years

height Height: height in inches

weight Weight: weight in pounds

sbp Systolic blood pressure: mmHg

dbp Diastolic blood pressure: mmHg

chol Cholesterol: mg/100 ml

personality Personality type/Behavior pattern: a factor with levels A1, A2, B3, B4

personality_2L Dichotomous personality type / behavior pattern: A = aggressive; B = passive

ncigs Smoking: Cigarettes/day

smoking Smoking: No, Yes

arcus Corneal arcus: No, Yes

chd Coronary heart disease event: No Yes

typechd coronary heart disease is a factor with levels No, MI_SD (MI or sudden death), Silent_MI, Angina

timechd Observation (follow up) time: Days

Details

The WCGS began in 1960 with 3,524 male volunteers who were employed by 11 California companies. Subjects were 39 to 59 years old and free of heart disease as determined by electrocardiogram. After the initial screening, the study population dropped to 3,154 and the number of companies to 10 because of various exclusions. The cohort comprised both blue- and white-collar employees. At baseline the following information was collected: socio-demographic including age, education, marital status, income, occupation; physical and physiological including height, weight, blood pressure, electrocardiogram, and corneal arcus; biochemical including cholesterol and lipoprotein fractions; medical and family history and use of medications; behavioral data including Type A interview, smoking, exercise, and alcohol use. Later surveys added data on anthropometry, triglycerides, Jenkins Activity Survey, and caffeine use. Average follow-up continued for 8.5 years with repeat examinations

Source

Statistics for Epidemiology by N. Jewell (2004)

References

Coronary Heart Disease in the Western Collaborative Group Study Final Follow-up Experience of 8 1/2 Years Ray H. Rosenman, MD; Richard J. Brand, PhD; C. David Jenkins, PhD; Meyer Friedman, MD; Reuben Straus, MD; Moses Wurm, MD JAMA. 1975;233(8):872-877. doi:10.1001/jama.1975.03260080034016.

Index

- * **data**
 - colon_s, 10
 - wcgs, 84
- * **finalfit all-in-one functions**
 - finalfit, 39
- * **finalfit model extractors**
 - fit2df, 43
- * **finalfit model wrappers**
 - coxphmulti, 10
 - coxphuni, 11
 - crrmulti, 12
 - crruni, 14
 - glmrmixed, 52
 - glmrmulti, 53
 - glmrmulti_boot, 54
 - glmuni, 55
 - lrmixed, 59
 - lrmulti, 61
 - lrmuni, 62
 - svyglmrmulti, 80
 - svyglmuni, 82
- * **finalfit plot functions**
 - coefficient_plot, 8
 - ff_plot, 32
 - hr_plot, 56
 - or_plot, 70
 - surv_plot, 79
- * **finalfit wrappers**
 - summary_factorlist, 75
- agrep, 7
- boot, 55
- boot_compare, 3, 5, 29
- boot_predict, 3, 4, 4, 29
- check_recode, 7
- coefficient_plot, 8, 32, 33, 57, 71, 80
- colon, 10
- colon_s, 10
- confint.glm, 49
- confint.merMod, 49
- coxme, 44
- coxph, 11, 12, 44
- coxphmulti, 10, 12–14, 53–56, 60–62, 80, 82
- coxphuni, 11, 11, 13, 14, 53–56, 60–62, 80, 82
- crr, 13, 14, 44
- crrmulti, 11, 12, 12, 14, 53–56, 60–62, 80, 82
- crruni, 11–13, 14, 53–56, 60–62, 80, 82
- dependent_label, 15
- extract_variable_label, 17, 23, 33
- ff_column_totals, 17, 77
- ff_expand, 19, 27, 74
- ff_formula, 20
- ff_glimpse, 20, 77
- ff_interaction, 21
- ff_label, 17, 22, 33, 77
- ff_merge, 23
- ff_metrics, 25
- ff_mode, 19, 27, 74
- ff_newdata, 28
- ff_parse_formula, 29
- ff_percent_only, 30, 77
- ff_permute, 30
- ff_plot, 9, 32, 57, 71, 80
- ff_relabel, 17, 23, 33, 33
- ff_relabel_df, 34
- ff_remove_p, 35
- ff_remove_ref, 35
- ff_row_totals, 36, 77
- ff_stratify_helper, 38
- finalfit, 30, 31, 35, 36, 39
- finalfit_column_totals
 - (ff_column_totals), 17
- finalfit_expand(ff_expand), 19
- finalfit_formula(ff_formula), 20
- finalfit_glimpse(ff_glimpse), 20

- `finalfit_interaction` (`ff_interaction`),
21
- `finalfit_label` (`ff_label`), 22
- `finalfit_merge`, 11–14, 53–56, 76, 80, 82
- `finalfit_merge` (`ff_merge`), 23
- `finalfit_mode` (`ff_mode`), 27
- `finalfit_newdata`, 4–6
- `finalfit_newdata` (`ff_newdata`), 28
- `finalfit_percent_only`
(`ff_percent_only`), 30
- `finalfit_permute` (`ff_permute`), 30
- `finalfit_plot` (`ff_plot`), 32
- `finalfit_relabel` (`ff_relabel`), 33
- `finalfit_relabel_df` (`ff_relabel_df`), 34
- `finalfit_remove_p` (`ff_remove_p`), 35
- `finalfit_remove_ref` (`ff_remove_ref`), 35
- `finalfit_row_totals` (`ff_row_totals`), 36
- `fit2df`, 3, 11–14, 23, 24, 35, 41, 43, 52–56,
60–62, 77, 80, 82
- `format_npercent`, 51
-
- `ggsurvplot`, 79
- `glm`, 28, 41, 43, 53–56
- `glmer`, 44, 52, 53
- `glmmixed`, 11–14, 44, 52, 54–56, 60–62, 70,
80, 82
- `glmulti`, 5, 11–14, 53, 55, 56, 60–62, 70,
80, 82
- `glmulti_boot`, 11–14, 43, 53, 54, 56,
60–62, 80, 82
- `glmuni`, 11–14, 53–55, 55, 60–62, 80, 82
-
- `hr_plot`, 9, 32, 33, 56, 71, 80
-
- `labels_to_column`, 58
- `labels_to_level`, 59
- `lm`, 41, 43, 61, 62
- `lmer`, 43, 60
- `lmmixed`, 9, 11–14, 43, 53–56, 59, 61, 62, 80,
82
- `lmmulti`, 5, 9, 11–14, 53–56, 60, 61, 62, 80, 82
- `lmuni`, 11–14, 53–56, 60, 61, 62, 80, 82
-
- `md.pattern`, 66, 67
- `metrics_hoslem`, 63
- `missing_compare`, 64
- `missing_glimpse`, 65
- `missing_pairs`, 65
- `missing_pattern`, 66
-
- `missing_plot`, 67
- `missing_predictorMatrix`, 68
-
- `or_plot`, 9, 32, 33, 35, 57, 70, 80
-
- `p_tidy`, 71
- `predict.glm`, 5
-
- `rm_duplicates`, 72
- `rm_empty_block`, 73
- `round_tidy`, 73
-
- `summary_df`, 19, 27, 74
- `summary_factorlist`, 3, 9, 23, 24, 30, 41, 57,
64, 70, 75, 78, 79
- `summary_factorlist_stratified`, 78
- `surv_plot`, 9, 33, 57, 71, 79
- `svyglm`, 80, 82
- `svyglmulti`, 11–14, 53–56, 60–62, 80, 82
- `svyglmuni`, 11–14, 53–56, 60–62, 80, 82
-
- `wcgs`, 84