Package 'foresty'

September 25, 2026

Title Forest Plots and Subgroup Effects from Fitted Regression Models

Version 0.2.0

Description Draws forest plots of exposure effects from fitted regression models. Name an exposure and 'foresty' plots its effect. Name an effect modifier as well and it refits the model with the interaction term, estimates the exposure effect within each level of the modifier as a linear combination of the coefficients, and reports the joint interaction test beside those estimates. It takes one exposure and one modifier at a time, so the interaction is always a two-way one. Rows of the plot and of the table beside it share one scale, in a layout that can follow a journal's house style. The same results go to a self-contained HTML page holding the subgroup estimates, the joint test and the coefficient table. The 'car' package computes the linear combinations and their tests. Models fitted by stats::glm(), stats::lm(), the 'survival' package, the 'lme4' package, the 'geepack' package and the 'survey' package are supported, as is any fit supplying coef() and vcov(). Ordinal outcomes are supported through the 'MASS' package and nominal ones through the 'nnet' package, where the figure carries one row per level of the outcome and the interaction is tested jointly across the equations. Fits from the 'rms' package are refused, naming the function that fits the same model in their place. The estimation of an exposure effect within a level of a modifier, and the test of the difference between such estimates, follow Altman and Bland (2003) <doi:10.1136/bmj.326.7382.219> and VanderWeele and Knol (2014) <doi:10.1515/em-2013-0005>; the reporting of subgroup effects beside the interaction test follows Wang et al. (2007) <doi:10.1056/NEJMsr077003>, and the figure itself the forest plot described by Lewis and Clarke (2001) <doi:10.1136/bmj.322.7300.1479>.

License GPL-3

URL https://github.com/AkiShiroshita/foresty, https://akishiroshita.github.io/foresty/

BugReports https://github.com/AkiShiroshita/foresty/issues

Encoding UTF-8

Language en-GB

Depends R (>= 4.1)

Imports car (>= 3.1.0), checkmate (>= 2.1.0), ggplot2 (>= 3.4.0), grDevices, grid, patchwork (>= 1.1.0), scales (>= 1.2.0), stats, utils

Suggests base64enc, broom (>= 1.0.0), bslib (>= 0.5.0), data.table, geepack, gt (>= 0.9.0), Hmisc, knitr, lme4, MASS, nnet, ragg, rmarkdown, rms, sandwich (>= 3.0.0), shiny (>= 1.7.0), survey (>= 4.1), survival (>= 3.2.0), svglite, tibble, testthat (>= 3.0.0), zip

VignetteBuilder knitr

LazyData true

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Akihiro Shiroshita [aut, cre, cph], Yuki Kataoka [aut]

Maintainer Akihiro Shiroshita <akihirokun8@gmail.com>

Repository CRAN

Date/Publication 2026-09-25 18:40:02 UTC

Contents

`foresty-model-methods` *Model methods for foresty figures*

Description

These pass through to the model the figure was drawn from, so that a foresty result can be used wherever the fit itself would have been.

Usage

```
## S3 method for class 'foresty'
predict(object, ..., model = NULL)

## S3 method for class 'foresty'
coef(object, ..., model = NULL)

## S3 method for class 'foresty'
vcov(object, ..., model = NULL)

## S3 method for class 'foresty'
formula(x, ..., model = NULL)

## S3 method for class 'foresty'
nobs(object, ..., model = NULL)

## S3 method for class 'foresty'
model.frame(formula, ..., model = NULL)
```

Arguments

`object`, `x`, `formula`	A `foresty` object. The argument is named `formula` in `model.frame()` only because the generic names it that.
`...`	Passed on to the method for the underlying fit.
`model`	Which model is meant, when the figure covers several.

Value

Whatever the corresponding method for the underlying model returns.

Examples

```
fit <- glm(asthma ~ no2 + sex, family = binomial, data = foresty_cohort)
x <- foresty_interaction(fit, exposure = "no2", interaction = "sex")
head(predict(x, type = "response"))
formula(x)
nobs(x)
```

`foresty-tidiers` *Turn a foresty figure into a data frame*

Description

`tidy()` returns the estimates the figure was drawn from, one row apiece, in the columns `broom` uses and in the order it puts them, so the result reads beside a tidied model and drops into the same pipelines. `glance()` returns one row describing the fit.

Usage

```
## S3 method for class 'foresty'
tidy(
  x,
  what = c("estimates", "coefficients"),
  conf.int = TRUE,
  model = NULL,
  ...
)

## S3 method for class 'foresty'
glance(x, model = NULL, ...)

## S3 method for class 'foresty'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	A `foresty` object.
`what`	`"estimates"`, the default, returns the rows drawn on the figure. `"coefficients"` returns the whole coefficient table of the model instead, on the scale it was fitted on.
`conf.int`	Whether to include the confidence interval. Defaults to `TRUE`, an interval being the point of a forest plot.
`model`	Which model to take the coefficients from, when the figure covers several. `glance()` takes the counts of that model too; without it a figure of several models is glanced at as a figure – its measure, its confidence level, its test and how many models it holds – and the columns describing what one model was fitted to are `NA`, no one of them being the figure's.
`...`	Ignored.
`row.names`, `optional`	Ignored, present for consistency with the generic.

Details

`as.data.frame()` returns everything the figure carries instead, the display labels included, which is what to use when the estimates are going back into a plot rather than into a table. It is the one of the three that needs nothing installed.

`tidy()` and `glance()` are `broom`'s generics, and `foresty` registers its methods on them rather than carrying `broom` itself: install it if you want them, and call them as `broom::tidy(x)` or after `library(broom)`.

Value

A data frame.

Examples

```
fit <- glm(asthma ~ no2 + sex + maternal_age, family = binomial,
           data = foresty_cohort)
x <- foresty_interaction(fit, exposure = "no2", interaction = "sex")
as.data.frame(x)

if (requireNamespace("broom", quietly = TRUE)) {
  broom::tidy(x)
  broom::glance(x)
}
```

---

foresty_app *Build a forest plot by clicking*

Description

Opens a Shiny app on a model you have already fitted, so that the exposures, the modifiers and the layout can be chosen from menus rather than typed, retyped and re-run. Every figure the app draws is drawn by `foresty_interaction()`, `foresty_main()` or `foresty_combine()`, and the code that drew it is shown in a tab of its own for copying back into your script.

Usage

```
foresty_app(
  fit,
  measure = NULL,
  launch = interactive(),
  launch.browser = TRUE,
  ...
)
```

Arguments

`fit`	A fitted model, as passed to `foresty_main()` or `foresty_interaction()`. Its variables become the menus of exposures and modifiers.
`measure`	The effect measure, as in `foresty_main()`. `NULL`, the default, takes the one the model implies.
`launch`	Whether to start the app. `TRUE` in an interactive session. `FALSE` returns the Shiny app object without running it, which is what tests and `shiny::runApp()` want.
`launch.browser`	Passed to `shiny::runApp()`. `TRUE`, the default, opens the app in the system browser as soon as it starts, rather than leaving a URL to be clicked or drawing it into the viewer pane of an IDE. `FALSE` starts it and says where it is, which is what a remote session or a scripted screenshot wants.
`...`	Passed to `shiny::runApp()`, so that `port` and `host` can be set.

Value

The Shiny app object, invisibly when the app was run.

What the app does not do

It does not fit your model. The estimates a forest plot reports come out of the coefficients and the covariance matrix of a fit, so the model has to exist before there is anything to draw, and which model to fit is the part of the analysis that should be written down in a script rather than clicked together and forgotten. The app is for the part that is fiddling – `foresty_layout()` alone has some thirty arguments, and finding the figure a journal wants by editing a call and re-running it is slow.

It does not change the type of your variables either. A modifier stored as a number but taking only two values is drawn as the two subgroups it is, needing no change; one taking three or more is refused, and the refusal is shown where the figure would have been, because turning it into a factor means fitting a different model and that is a decision to make in the script, not a checkbox. See the `interaction` argument of `foresty_interaction()`.

More than one pair

The exposure and the modifier menus both take as many variables as you select, and what is drawn is every pair of them: three exposures and two modifiers are six figures. The overall effect, which comes from the model with no interaction term in it, is a checkbox of its own, so it can be drawn beside the subgroups rather than instead of them. Where more than one figure is drawn the app offers to combine them, which is `foresty_combine()` and follows its rules: figures of one exposure become the blocks of one figure, and figures of different exposures stay apart, since rows reporting the effects of different exposures are not read against each other.

How big a difference in the exposure each estimate is for – one unit, an interquartile range, an increment you name, its two ends either way round, the first to the third quantile, two values you name – is asked once for each continuous exposure rather than once for all of them, since two exposures rarely want the same answer. The lowest and highest values it takes are written under the choice, and are where naming two values starts from. Whichever was chosen is written on the figure wherever the exposure is named, one unit included – `"no2 (per 1)"`, `"no2 (per 10)"`, `"no2 (per IQR, 8.44)"` – since a reader comparing two figures across a screen should not have to know that a row saying nothing means one.

Which way round the comparison runs is written with an arrow rather than as "from ... to ...", since that is the part a reader gets wrong: `Lowest value -> Highest value` is the effect of being at the top of the exposure rather than the bottom, and **Reverse the direction** turns it into the same estimate the other way up, which is how a protective effect is drawn as one. The two boxes that name two values are headed the same way, and so is the figure: two values compared are written beside the exposure as `"no2 (4.102 -> 38.72)"` rather than as "38.72 vs 4.102", which says which two values were compared without saying which of them the estimate is the effect of moving to.

From one quantile to another is not the interquartile range, though it begins at the quartiles: the range is a width, and an effect per one of it is a step along a slope, whereas the quantiles are two values of the exposure and are compared as `at = c(from, to)`. So it can be asked of a splined exposure, which has no single slope to report a width per, and the two probabilities can be set to anything – the 10th against the 90th, the median against the top decile – rather than only 0.25 and 0.75.

One reference group instead of one per subgroup

A subgroup figure reads each subgroup against its own reference level, so the rows of one subgroup are not comparisons with the rows of another. Where the exposure and the modifier both have levels there is a second question: what every combination of the two comes to against one of them, which is the table of groups against a baseline group a paper reports. **Model** asks it once per effect modifier chosen – *Compare every combination of the exposure and ... with one group* – and the menus under the box say which combination the rest are read against: a level of the modifier, and a level of each exposure that has levels.

Every row is then that whole cell – this level of the exposure, this level of the modifier – against the one chosen, all of it out of the same interaction model, and the cell chosen is drawn as the reference it is. The p-value beside the rows is the same joint test of the interaction either way: it asks whether the effect of the exposure depends on the modifier, and it is not a test of the rows. See the `reference` argument of `foresty_interaction()`.

A continuous exposure has no levels to combine, so a pair with one is drawn as it always was whatever the box says: its rows are the effect of a difference in the exposure rather than groups of people.

Figure style names

The rows, the axis and the columns of a figure are named after columns of a data set, and a column is called what the data set calls it. **Figure style** renames the outcome – which is otherwise taken from the left of your formula, `asthma_ever_dx` and all – replaces the label under the plot outright, and names each selected exposure. A model carrying person-time is also asked what unit to count it in – any number, 1,000 and 100,000 being how a paper usually reports a rate and six figures beside the plot being width the plot could have had.

A model too slow to redraw

Every change to a control refits the model once per pair of menus, and while that is happening the app looks exactly as it did before, so a line saying that it is working is shown above the tabs and the figure on the screen is the previous one until it goes.

On a model where that wait is not seconds but minutes, **Write the R code only** stops it: nothing is fitted and nothing is drawn, the menus and the style controls go on working, and the **R code** tab

goes on writing the call they come to, which is what to paste into the script and run once. The other tabs report on models the app fitted and say so instead.

Colors and the panel

A figure is drawn in one color, which is asked for per exposure and can be named as a place in a ColorBrewer palette – the third color of Dark2 – or typed as a hex code, which is how a figure is drawn in the color a journal or a slide deck already uses. **What the colors change with** draws it in more than one instead: a color per category of the rows, which is the levels of a categorical exposure or the subgroups of the modifier, or a color per row. Those come from a palette, or from hex codes typed beside it – `#1B9E77, #D95F02` – which is the same list `colors` takes. See `color_by` in `foresty_layout()`.

The tabs

Plot draws them, each figure carrying the exposure it is of under its title, which is a line that can be turned off on its own where the title says it already. A row of a forest plot is usually half of a comparison, and the columns of counts carry the other half beside it: a row of suburban children reads `822 vs 468`, those being the 822 the row is of and the 468 rural ones its odds ratio was estimated against. A line under the figure says which two groups they are, and for a multinomial fit that the estimate did not come out of the two of them alone. Nothing is said, and nothing is paired, on a figure whose rows compare no two groups of people – a continuous exposure and a binary outcome. *Count each row's own group only* goes back to one number a row. The line is on the screen rather than in the figure, so what the download buttons write is unchanged. See *What the counts beside the rows count* in `foresty_main()`. **Models** writes out, in R, how each figure was arrived at: the term added to your model, the linear combination each subgroup estimate is, and the test reported beside them. That code is meant to be run: it is the call `foresty` made, with the same design matrix, the same coefficients and covariance, the same degrees of freedom and the same test, so pasting it beside the model reproduces the numbers on the figure rather than approximating them. **summary(fit)** is the summary of the model you fitted and of every model the app fitted from it by adding an interaction term. Both tabs head each block with the outcome, the exposure and the effect modifier the figure under it is of, since a coefficient table says none of them. **R code** is the code twice over. *With foresty* is the code that drew what is beside it, deparsed rather than reconstructed, so it cannot drift from what you are looking at; several pairs are written as a loop over the pairs rather than as one call apiece. It names the model with the name you passed to `foresty_app()`, so pasting it into a script next to that model works as it stands. *How to calculate effect estimates for each subgroup* is where those numbers come from, in base R and the `car` package with nothing from this package in them: the interaction term, the linear combination of the coefficients each subgroup estimate is, and the joint test reported beside them. It is there for a reader deciding whether to take the dependency at all, and for one who would rather see what is being done on their behalf than take it on trust. It comes to the estimates on the Plot tab exactly – the same design matrix, the same coefficients and covariance, the same degrees of freedom and the same test – and draws nothing: drawing them is what the call above it is for. A multinomial fit is written out too: it holds one equation for each level of the outcome other than the one it was fitted against, so an estimate is a difference of blocks of coefficients rather than a difference of two rows of the design, and the script does that placing where it does the rest of the arithmetic.

What comes out

The figure downloads as PNG and as SVG, at the size and resolution set beside the buttons; where the pairs are not being combined, one file per figure comes down in a zip. The reports do the same, one HTML page per model. The objects themselves download as an `.rds` file holding a named list – one element per pair, plus the combined figure where there is one – so that a session that ended in the app can go on in a script: `figures <- readRDS("...")` and then `summary(figures[[1]])`, `as.data.frame(figures[[1]])` or `foresty_report(figures[[1]], "x.html")`.

See Also

`foresty_interaction()`, `foresty_main()`, `foresty_combine()`, `foresty_layout()`.

Examples

```
fit <- glm(asthma ~ no2 + sex + maternal_smoking + maternal_age,
           family = binomial, data = foresty_cohort)

# Opens the app in an interactive session.
if (interactive()) {
  foresty_app(fit)
}
```

---

foresty_cohort *A simulated birth cohort*

Description

Four thousand simulated children with an air pollution exposure in infancy, asthma by school age, and a time to first wheeze episode. The data are made up, and are here so that the examples run without a real cohort.

Usage

```
foresty_cohort
```

Format

A data frame with 4,000 rows and 13 columns:

asthma Asthma by school age, 1 or 0.

asthma_severity Asthma severity by school age, an ordered factor: `None` < `Mild` < `Moderate` < `Severe`. Drawn on its own rather than from `asthma`.

wheeze The event indicator for `followup_years`: 1 where a first wheeze episode was seen, 0 where the child was censored without one.

wheeze_phenotype Wheeze phenotype, an unordered factor: `None`, `Transient` or `Persistent`.

followup_years Years to the wheeze episode or to censoring.

no2 Nitrogen dioxide during infancy, in parts per billion.

black_carbon Black carbon during infancy, in micrograms per cubic metre.

sex Child's sex, `Female` or `Male`.

maternal_smoking Smoking during pregnancy, `No` or `Yes`.

maternal_asthma Maternal history of asthma, `No` or `Yes`.

maternal_age Maternal age at delivery, in years.

birth_year Year of birth, 2005 to 2014, as a factor.

urbanicity `Rural`, `Suburban` or `Urban`. Exposure rises with it, so it confounds the comparison.

Details

The exposure effect was simulated to be about twice as large in boys as in girls, and to be the same whether or not the mother smoked, although maternal smoking raises the risk of asthma on its own. The two modifiers therefore show what a real interaction and an absent one look like when the subgroup estimates are drawn beside the joint test.

The outcome comes in four shapes, so that the same interaction can be followed through the model classes that carry it: binary (`asthma`), time to event (`wheeze` with `followup_years`), ordinal (`asthma_severity`) and nominal (`wheeze_phenotype`). The ordinal outcome was drawn from a latent logistic variable cut at three fixed thresholds, so it is a proportional odds model by construction; the nominal one was drawn from two multinomial logits whose exposure effects differ, so its levels cannot be collapsed into an ordering.

The four are four separate draws, sharing the covariates and the same exposure effect rather than describing one event four ways: each is there to be the outcome of a model, and a child may be an asthma case in one of them and not in another. So `asthma` is not the `asthma_severity` of the same child collapsed to two levels, and cross-tabulating one outcome against another says nothing about anything. Fit them one at a time.

Source

Simulated by `data-raw/foresty_cohort.R`.

`foresty_colors` *The qualitative palettes a figure can be colored from*

Description

The ColorBrewer qualitative palettes, as `RColorBrewer::brewer.pal()` gives them, so that `color` and `colors` in `foresty_layout()` can be answered by naming a palette and a place in it rather than by pasting a hex code. The values are the palettes' own and are held here so that the package needs nothing installed to draw in them.

Usage

```
foresty_colors(palette = c("Dark2", "Set1", "Set2"), n = NULL, start = 1)
```

Arguments

palette	Name of the palette: "Dark2", the default, which is the one a figure printed in color is usually drawn from, "Set1" or "Set2".
n	How many colors to return. NULL, the default, returns the whole palette. More than the palette holds cycles round it, since a figure with more categories than the palette has colors has to draw them somehow.
start	Which color of the palette to begin at, counting from 1. The palette is rotated rather than trimmed, so that every color is still available after it.

Value

A character vector of colors.

See Also

foresty_layout(), whose color and colors arguments these are for.

Examples

```
foresty_colors("Dark2")

# The third color of Dark2, for one figure drawn in one color.
foresty_colors("Dark2")[3]

# A palette beginning there, for a figure whose rows are colored by their
# category.
foresty_colors("Dark2", start = 3)
```

---

foresty_combine *Combine several foresty figures into one forest plot*

Description

Draws the rows of figures already made – an overall effect from foresty_main(), the same effect within the levels of sex from foresty_interaction(), within the levels of ethnicity from another – as one figure, each of them a block of rows under a heading of its own. This is the subgroup panel a paper prints: the overall estimate at the top, the subgroup analyses under it, and the interaction p-value beside each block.

Usage

```
foresty_combine(
  ...,
  emphasize = "auto",
  outcome = NULL,
  table = TRUE,
```

```
  columns = NULL,
  person_time = NULL,
  layout = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL
)
```

Arguments

...	Figures returned by foresty_main() or foresty_interaction(), in the order they are to be drawn, optionally named.
emphasize	Which blocks are drawn as the figure's summary rather than as another subgroup: "auto", the default, takes the overall estimates that are a single row; TRUE takes every overall estimate, a block of several rows included; a character vector names blocks by the names they are drawn under; and NULL or FALSE draws every block alike. See *Singling out the overall estimate*.
outcome	What to call the outcome, as outcome = "incident asthma". The default takes it from the left of the model's formula, which is the name of a column – asthma_ever_dx, evt5 – and is rarely what a figure should say the effect is an effect on. It is written wherever the outcome is named: the axis under the plot, the title the package writes for itself, and the HTML report. NA names none, leaving "Adjusted odds ratio" on its own for a figure whose caption says what of.
table	Whether to draw the table of numbers beside the plot. Defaults to TRUE, a forest plot being read from the numbers as much as from the marks; table = FALSE leaves a plain figure, and summary() and tidy() report the same numbers at the console either way.
columns	Which columns the table carries, from "estimate", "p", "n", "events", "person_time", "interaction_p" and, where both tests of an interaction were asked for, "interaction_p_lrt". The default shows the estimate and the p-value together with whichever of the others the models can supply. On a figure reporting a test of an interaction the p-value of each row is left off, being easily read as the test beside it; name it in columns to have it back.
person_time	The unit person-time is reported in. NULL, the default, takes whatever the figures being combined were drawn with, since that is a decision already made; a number, or a named number, overrides them all. See foresty_main().
layout	How the figure is drawn: the name of a style, as layout = "jama", or a layout built by foresty_layout() when something about it has to be changed. The styles are "classic", "jama", "nejm", "lancet", "bmj" and "revman".
title	Plot title. The default names the measure, the outcome it is a measure of and the exposure the figure reports, as "Adjusted odds ratio for asthma associated with NO2, overall and within each subgroup". The exposure is named once however many blocks it was drawn under, and with the comparison behind it where that is not the plain one unit – contrast = 10, or two values named by at – since the estimates cannot be read without it. NA draws none, and the journal styles draw none. A title given by hand is drawn on every figure of a

call covering more than one exposure, which is a reason to leave it to the default there.

`subtitle` Plot subtitle. `NULL`, the default, draws none.

`xlab` The label under the plot, which by default names the measure and the outcome, as `"Adjusted odds ratio for asthma"`. A string is drawn as it was given – `xlab = "Odds ratio (95% CI), NO2 per 10 ug/m3"` – and `NA` draws none. Renaming only the outcome is what `outcome` is for; this replaces the whole line.

Details

Nothing is refitted and nothing is re-estimated. Each figure keeps the estimates it was drawn from, so a block means exactly what it meant on its own figure, and the estimates behind the combined one are still reached with `summary()`, `tidy()` and `as.data.frame()`.

The figures have to agree on what they are measuring: the same effect measure and the same confidence level, since the rows are read against one axis. They do not have to come from the same model, and usually will not, each subgroup analysis being its own interaction model.

Value

A `ggplot2` object, of class `foresty`, carrying every estimate drawn on it – or, where the figures combined report more than one exposure, a list of one such object per exposure, named for them and of class `foresty_figures`. See *More than one exposure*.

More than one exposure

Every row of a combined figure is read against the rows above it, and rows reporting the effect of different exposures are not comparable that way, however alike their axes happen to be. So the figures handed in are sorted by the exposure they report and one figure is drawn for each: NO2 overall and within every subgroup on the first, black carbon overall and within every subgroup on the second, each titled with its own exposure.

A call covering one exposure – which is the usual one – returns that figure. A call covering several returns the figures in a list, named for the exposures, which draws them one after another when it is printed and holds `foresty` figures that are used singly as any other is:

```
figures <- foresty_combine(Overall = overall, Sex = by_sex)
figures                       # draws each in turn
figures[["NO2"]]              # one of them
ggplot2::ggsave("no2.png", figures[["NO2"]])
```

Naming the blocks

A named argument names its block. An unnamed one is named for itself: a `foresty_interaction()` figure by its modifier, a `foresty_main()` figure `"Overall"`.

A block holding a single row – the overall estimate, most often – is drawn as one row carrying the block's name, rather than as a heading with a single row indented under it.

Singling out the overall estimate

The overall estimate is not one of the subgroups; it is what the subgroups are read against, and a figure that draws it as another row of the same kind invites a reader to compare it with them as though it were one. It is therefore drawn apart from them: a filled diamond on its interval, half again the size of the squares under it, its label in bold, and a rule between it and the subgroups, drawn whether or not the style rules between subgroups. The interval is a plain line, as every other interval on the figure is, so the whole figure is read the same way and the diamond says which row is the summary.

`emphasize` says which blocks are drawn that way. `"auto"`, the default, takes the blocks that came from `foresty_main()`, which are the ones with no modifier behind them; a figure of nothing but those has none singled out, since every row would be. Name blocks to choose them yourself, as `emphasize = c("Overall", "Pooled")`, and pass `NULL` to draw every block alike. How they are drawn is set in `foresty_layout()`, through `emphasis_shape`, `emphasis_height`, `emphasis_face` and `emphasis_gap`.

What the counts beside the rows count

The table beside the plot reports the number of people behind each row and, where the outcome is an event, how many of them had it. Both are counted over the rows the model was fitted to and not over the data frame it was fitted from, so anyone missing the outcome, the exposure or any covariate in the formula is left out of them – which is the same complete-case set the estimates come from, and is what makes the two agree. A column that is not in the model does not affect them however much of it is missing.

A row of a forest plot is usually half of a comparison, so by default the other half is beside it: a row of suburban children reads `822 vs 468`, those being the 822 the row is of and the 468 rural ones its odds ratio was estimated against. `counts = "row"` in `foresty_layout()` holds the row's own group alone instead. What the two groups are depends on what the row compares:

- A **continuous exposure** compares no two groups of people – its row is a step along a slope – so the count is the one group the row is of: every row of the model's data in a `foresty_main()` figure, everyone in the subgroup in a `foresty_interaction()` one.
- A **categorical exposure** compares one of its levels with the level the contrast is from, so a row carries the two: `822 vs 468` people and `125 vs 59` events. The reference row is the group the others are compared with rather than a comparison of its own, so it carries one number.
- A **multinomial fit** (`nnet::multinom()`) of an exposure with no levels compares two levels of the outcome, and the two numbers are how many people were at each: a "Transient vs None" row of women reads `637 vs 1,050`. That is the only pair of counts such a row has – the events column would repeat it – so it goes under the sizes and the events column is left off. Of a categorical exposure, the row compares the exposure's levels within one comparison of the outcome, so the sizes are the two exposure groups whatever their outcome and the events are the people at the row's outcome level within each.
- An **ordinal fit** (`MASS::polr()`) has one set of coefficients for the whole outcome, so its rows carry sizes and no count of events.

A multinomial estimate is worth one caution the counts cannot give: it did not come out of the two groups beside it alone. All the equations are fitted over the whole outcome at once, so the people at the levels a row is not about bear on it too, and a row reading `637 vs 1,050` is not the logistic regression of those 1,687.

The figures whose rows compare two groups say what the counts are under the plot in `foresty_app()` and under the table of estimates in `foresty_report()`; the ones that do not say nothing.

Adjusting the figure

The result is a ggplot2 object, so layers, scales and themes are added to it as usual, and + always reaches the forest:

```
foresty_main(list(fit), "no2") + ggplot2::coord_cartesian(xlim = c(0.8, 2))
foresty_main(list(fit), "no2") + ggplot2::theme_minimal(base_size = 14)
```

& reaches every panel of a figure that has more than one, which is worth knowing about but rarely what you want: a scale or a coordinate system applied to the table of numbers beside the plot will spoil it. Use +.

See Also

`foresty_main()`, `foresty_interaction()`, `foresty_layout()`.

Examples

```
fit <- glm(asthma ~ no2 + sex + maternal_smoking + maternal_age,
           family = binomial, data = foresty_cohort)

overall <- foresty_main(list(fit), exposure = "no2",
                        labels = c(no2 = "NO2"))
by_sex <- foresty_interaction(fit, exposure = "no2", interaction = "sex")
by_smoking <- foresty_interaction(fit, exposure = "no2",
                                  interaction = "maternal_smoking")

foresty_combine(Overall = overall, Sex = by_sex,
                `Maternal smoking` = by_smoking)

# The rest of the ways the blocks can be combined, drawn one after
# another. Each is quick; there are simply several of them.

# In the layout of a journal, the interaction p-value written once against
# each block, and without the numbers beside the plot.
foresty_combine(overall, by_sex, by_smoking, layout = "jama")
foresty_combine(overall, by_sex, by_smoking, table = FALSE)

# Every block drawn alike, the overall estimate included.
foresty_combine(Overall = overall, Sex = by_sex, emphasize = NULL)

# Two exposures are two figures, one apiece, named for them.
fit_bc <- glm(asthma ~ black_carbon + sex + maternal_smoking + maternal_age,
              family = binomial, data = foresty_cohort)
figures <- foresty_combine(
  Overall = foresty_main(list(fit, fit_bc),
                         exposure = c(NO2 = "no2",
                                      `Black carbon` = "black_carbon")),
  Sex = by_sex,
```

```
  `Black carbon by sex` = foresty_interaction(fit_bc, "black_carbon", "sex")
)
names(figures)
figures[["NO2"]]
```

---

foresty_data *Draw a forest plot and its table from a data frame of estimates*

Description

Takes the estimates you already have – one row per row of the figure, with the effect and the two ends of its interval – and draws the figure that `foresty_app()` draws: the forest plot and, beside it, the table of numbers, aligned row for row, in any of the journal styles of `foresty_layout()`.

Usage

```
foresty_data(
  data,
  estimate = NULL,
  conf.low = NULL,
  conf.high = NULL,
  label = NULL,
  group = NULL,
  n = NULL,
  events = NULL,
  person_time = NULL,
  p = NULL,
  interaction_p = NULL,
  reference = NULL,
  emphasis = NULL,
  color = NULL,
  measure = "OR",
  outcome = NULL,
  ratio = NULL,
  adjusted = FALSE,
  ci_level = 0.95,
  table = TRUE,
  columns = NULL,
  person_time_unit = NULL,
  layout = NULL,
  label_header = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL
)
```

Arguments

`data`	The estimates, one row per row of the figure. A data frame, a `tibble`, a `data.table`, or anything `as.data.frame()` accepts.
`estimate`, `conf.low`, `conf.high`	The columns holding the effect and the two ends of its interval. `NULL`, the default, looks for them by name.
`label`	The column holding what each row is called. `NULL` looks for `label`, `term`, `subgroup`, `name` or `variable`, and falls back on the row numbers.
`group`	The column that blocks the rows into subgroups, or `NULL` for a figure of one block.
`n`, `events`, `person_time`	The columns of counts to write in the table beside the plot, where the data carries them.
`p`	The column of p-values for the rows themselves. A figure carrying a test of the interaction leaves this column off unless `columns` asks for it by name: two columns of p-values side by side are read for each other.
`interaction_p`	The column of p-values for the interaction, written once per block.
`reference`	A logical column marking rows that are a definition rather than an estimate – the reference level of a categorical exposure – which are drawn without an interval and written "1.00 (reference)". `NULL` treats a row whose interval is missing and whose estimate is the null value as one.
`emphasis`	A logical column marking the rows drawn for emphasis.
`color`	The column saying what color each row is drawn in, holding either the colors themselves or the names of categories. See *Colors*.
`measure`	What the estimates are, as one of `"OR"`, `"RR"`, `"HR"`, `"IRR"`, `"MD"` and `"Coefficient"`, or as a name of your own – `"Standardised mean difference"`, `"Prevalence ratio"` – in which case say with `ratio` which side of the null it is drawn about.
`outcome`	What the measure is a measure of, named so that the axis reads "Odds ratio for incident asthma" rather than "Odds ratio".
`ratio`	Whether the estimates are ratios, drawn about 1 on a scale where the null is 1, rather than differences drawn about 0. `NULL`, the default, takes it from `measure`, which can only be done for a measure named by one of the codes above; a measure described in words of your own has to say, and it is an error not to.
`adjusted`	Whether the estimates are adjusted, which is a word the axis and the heading of the table say if so. `foresty` cannot tell from a column of numbers, so it asks.
`ci_level`	The confidence level the intervals were formed at, which is what the heading of the table reports. It is not used to compute anything: the interval is the one in the data.
`table`	Whether to draw the table of numbers beside the plot.
`columns`	Which columns of that table to draw, and in what order. See `foresty_main()`.
`person_time_unit`	The unit person-time is reported in, as `person_time_unit = 1000`. Optionally named, to head the column.

`layout` The style of the figure, as a name or a `foresty_layout()`.

`label_header` What to write over the column of row labels. `NULL` writes "Subgroup" over a blocked figure with nothing emphasised, and nothing over the rest.

`title`, `subtitle`, `xlab`
The title over the figure, the line under it, and the label under the plot. `NULL` takes the one the measure implies and `NA` draws none.

Details

This is the entry point for numbers that did not come out of a model this package can read. A meta-analysis, a table being redrawn from a paper, a model fitted by something `foresty` does not support, a set of results typed out by hand: pass the data frame, name the columns holding the estimate and its interval, and the figure is the same figure.

Value

A `foresty` figure: a `ggplot`/`patchwork` object carrying its estimates, which prints as the figure and can be added to as any `ggplot` can.

The data

A `data.frame`, a `tibble`, a `data.table` or a matrix, all of which are read as the plain data frame the columns are taken from. One row is one row of the figure, drawn in the order the rows are in, so sort the data the way the figure should read.

The estimate and the two ends of its interval are the only columns the figure cannot be drawn without. They are looked for by name when they are not named here – `estimate`, `conf.low` and `conf.high`, and the usual alternatives (`lower`/`upper`, `lcl`/`ucl`, `ci_low`/`ci_high`) – so a frame that came out of `broom::tidy(conf.int = TRUE)` needs nothing said about it at all.

Subgroups

`group` names a column that blocks the rows: its values become the headings the rows sit under, in the order they first appear. A block holding a single row takes no heading of its own – the row is labelled with the block instead – which is what makes an "Overall" row a row rather than a heading with one line under it.

`emphasis` names a logical column marking the rows drawn for emphasis: a bolder label, a diamond rather than a square, and a rule setting the block off from the ones around it. It is how the overall estimate is set apart from the subgroups read against it. See `foresty_layout()`.

`interaction_p` names a column of p-values for the interaction. One test covers the block it was taken across, so it is written once, against the first row of the block, rather than repeated down the column.

Colors

`color` names the column saying what color each row is drawn in, which is how a figure drawn from a table colors the rows it chooses rather than the ones a rule would have chosen. The mark, its interval and the diamond of an emphasised row are all drawn in it.

The column holds either the colors themselves – `"red"`, `"#B24745"`, any name in `grDevices::colors()` – or the names of categories, which are taken in the order they first appear and drawn in the `colors` of `foresty_layout()`, one color per name. Which of the two a column is, is read off its values rather than asked for again, so a column holding some of each is refused: half of it would be drawn in the colors given and half in colors chosen for it.

A row whose color is `NA` is drawn in the one color the rest of the figure would have been. The reference level of a categorical exposure stays hollow, being a definition rather than an estimate. No legend is drawn: every row is labelled already. Naming this column is what the figure is colored by, whatever `color_by` in `foresty_layout()` says.

What is not here

The figure carries no model, because it was not given one. `summary()`, `foresty_report()`, `coef()`, `vcov()` and the coefficient table are properties of a model and refuse on a figure drawn this way, naming `foresty_main()` and `foresty_interaction()` as the way to have them. `as.data.frame()`, `broom::tidy()` and `print()` work as they do on any other `foresty` figure.

See Also

`foresty_main()` and `foresty_interaction()`, which start from a fitted model; `foresty_layout()` for the styles; `foresty_app()`, whose figures this one reproduces.

Examples

```
subgroups <- data.frame(
  subgroup  = c("Overall", "Female", "Male", "Under 35", "35 and over"),
  block     = c("Overall", "Sex", "Sex", "Maternal age", "Maternal age"),
  overall   = c(TRUE, FALSE, FALSE, FALSE, FALSE),
  estimate  = c(1.24, 1.05, 1.48, 1.11, 1.39),
  conf.low  = c(1.08, 0.86, 1.21, 0.90, 1.14),
  conf.high = c(1.42, 1.28, 1.81, 1.37, 1.69),
  n         = c(4000, 2009, 1991, 1832, 2168),
  events    = c(802, 327, 475, 341, 461),
  p_int     = c(NA, 0.012, NA, 0.106, NA)
)

foresty_data(
  subgroups,
  label = "subgroup", group = "block", emphasis = "overall",
  interaction_p = "p_int",
  measure = "OR", outcome = "asthma", adjusted = TRUE
)

# Every combination of two categorical variables against one of them, which
# is what foresty_interaction(reference = ) draws from a model: the blocks
# are the levels of the modifier, the rows the levels of the exposure, and
# one row of the figure is the group the rest are read against.
cells <- data.frame(
  ecog      = c("0-1", ">=2", "0-1", ">=2"),
  mutation  = c("Negative", "Negative", "Positive", "Positive"),
  estimate  = c(1.00, 1.94, 0.62, 1.21),
```

```
  conf.low  = c(NA, 1.42, 0.44, 0.83),
  conf.high = c(NA, 2.65, 0.87, 1.76),
  n         = c(212, 168, 190, 145),
  events    = c(126, 131, 92, 96),
  reference = c(TRUE, FALSE, FALSE, FALSE),
  p_int     = c(0.031, NA, NA, NA)
)

foresty_data(
  cells,
  label = "ecog", group = "mutation", reference = "reference",
  interaction_p = "p_int", measure = "HR", outcome = "death",
  adjusted = TRUE, layout = "jama"
)

# The rows drawn in colors the data carries: the colors themselves here,
# one per block, with the overall estimate left in the layout's own.
subgroups$color_group <- c(NA, "#D95F02", "#D95F02", "#7570B3", "#7570B3")

foresty_data(
  subgroups,
  label = "subgroup", group = "block", emphasis = "overall",
  color = "color_group", interaction_p = "p_int",
  measure = "OR", outcome = "asthma", adjusted = TRUE
)

# The same figure from the names of the categories, drawn in the palette
# the layout was given rather than in colors written into the data. It is
# the figure above with one column changed, so it is drawn on the second
# pass rather than adding its time to the first.

subgroups$color_group <- c(NA, "Sex", "Sex", "Age", "Age")

foresty_data(
  subgroups,
  label = "subgroup", group = "block", emphasis = "overall",
  color = "color_group", interaction_p = "p_int",
  measure = "OR", outcome = "asthma", adjusted = TRUE,
  layout = foresty_layout("classic", colors = "Set1")
)
```

`foresty_interaction` *Exposure effect within each level of an effect modifier*

Description

Takes a model that does not yet interact the exposure with the modifier, updates it with the interaction term, and estimates the exposure effect separately within every level of the modifier. The

estimates are drawn one level above the other, a row to a level, and can be written to a self-contained HTML page carrying the joint test of the interaction.

Usage

```
foresty_interaction(
  fit,
  exposure,
  interaction,
  measure = NULL,
  exponentiate = TRUE,
  labels = NULL,
  level_labels = NULL,
  reference = NULL,
  outcome = NULL,
  outcome_reference = NULL,
  outcome_reference_row = FALSE,
  ci_level = 0.95,
  contrast = NULL,
  at = NULL,
  vcov = NULL,
  cluster = NULL,
  test = c("lrt", "wald", "both"),
  table = TRUE,
  columns = NULL,
  person_time = NULL,
  layout = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  html = FALSE
)
```

Arguments

`fit`	A fitted model. If it already contains the exposure by modifier interaction it is used as it stands; otherwise it is updated to add it.
`exposure`	Name of the exposure variable, as a character string. Naming it, as `exposure = c(NO2 = "no2")`, gives it that label on the figure, which saves repeating it in `labels`.
`interaction`	Name of the effect modifier, as a character string, and named as `interaction = c(Sex = "sex")` where the figure is to call it something other than what the column is called. It must have at least two levels. A numeric column taking exactly two values counts as having them, so a flag coded 0 and 1 is read as those two subgroups without being wrapped in `factor()` and fitted again: it entered the model through the one coefficient a two-level factor would have given, and the estimates and the test are the same either way. Its rows are labelled with the values themselves, `0` and `1`, which is rarely what a figure should say, so

name them through `level_labels`. A modifier taking three or more numeric values is rejected: categorize it with `cut()` and refit, so that the subgroups being compared are the ones you intend and the test has the degrees of freedom the figure implies.

`measure` Effect measure, one of `"OR"`, `"RR"`, `"HR"`, `"IRR"`, `"MD"` or `"Coefficient"`. The default reads it from the model: a logistic regression gives an odds ratio, as do the ordinal and multinomial forms of one, a Cox model a hazard ratio, a Poisson model with an offset an incidence rate ratio, and a linear model a mean difference. Ratios are exponentiated; differences are not. What an offset holds is not recorded anywhere, so a Poisson model offset by something other than person-time – the size of a population, say – is named here instead, as `measure = "RR"`.

`exponentiate` Whether a ratio measure is drawn as a ratio. `TRUE`, the default, draws an odds ratio as an odds ratio. `FALSE` leaves it on the scale the model was fitted on: the figure reports a log odds ratio, read against zero rather than against one, and the estimates and their intervals come back on that scale from `tidy()` as well. It says nothing about a mean difference or a coefficient, which are on that scale already and are never exponentiated.

`labels` Named character vector giving the label to draw for a variable, as `c(no2 = "Nitrogen dioxide")`. Names not matched are left as they are. This is where the exposure is renamed, and naming it where it is chosen – `exposure = c(NO2 = "no2")` – comes to the same thing.

`level_labels` Named character vector giving the label to draw for a level of the modifier, as `c(F = "Female", M = "Male")`, or `c("0" = "No", "1" = "Yes")` for a modifier coded as a flag.

`reference` The one combination of the exposure and the modifier every row is compared with, as a named character vector naming a level of each: `reference = c(ecog = "0-1", egfr = "Negative")`. `TRUE` takes the first level of each. `NULL`, the default, compares nothing across subgroups: each row is the effect of the exposure inside its own subgroup. Both variables must be categorical for a combination of them to be a group at all. See *One reference group for the whole figure*.

`outcome` What to call the outcome, as `outcome = "incident asthma"`. The default takes it from the left of the model's formula, which is the name of a column – `asthma_ever_dx`, `evt5` – and is rarely what a figure should say the effect is an effect on. It is written wherever the outcome is named: the axis under the plot, the title the package writes for itself, and the HTML report. `NA` names none, leaving "Adjusted odds ratio" on its own for a figure whose caption says what of.

`outcome_reference` For a multinomial logistic regression, the level of the outcome every estimate is read against, as `outcome_reference = "None"`. `NULL`, the default, is the level the model itself was referred to, read off the fit as the level it holds no equation for rather than assumed to be any particular one. Naming another does not refit anything: the odds ratio of one level against another is the difference between their two equations, and the covariance of the pair is already in the model. Every other level is then drawn against it, one row apiece, each row saying which two levels it compares. It says nothing about a model of one equation, whose

reference is fixed by how the outcome is coded, and is refused there rather than ignored.

`outcome_reference_row`
: Whether that level is drawn as a row of its own. `FALSE`, the default, draws only the levels compared with it, each row saying which two levels it compares. `TRUE` adds a row for the reference level itself, the way the reference level of a categorical exposure is drawn: it carries no estimate, being the definition the other rows are differences from, so it is `1` on the ratio scale and `0` on the scale the model was fitted on, with no interval, no test and no p-value. It is one row whatever the exposure is, since it is the same definition at every value of it, and the counts beside it are of everybody in that level of the outcome for the same reason: the row is not about a value of the exposure, so it is not counted at one. On a figure of subgroups it is drawn once inside each and counted within it. It says nothing about a model of one equation.

`ci_level`
: Confidence level of the intervals. Defaults to `0.95`.

`contrast`
: For a continuous exposure, the increment the effect is reported per. `NULL`, the default, is one unit and is not written on the figure. An increment you name is: `contrast = 10` draws the row as `"NO2 (per 10)"` and `contrast = 1` draws it as `"NO2 (per 1)"`, the same estimate as the default said out loud. No unit is invented, since the package cannot know what a column's numbers mean; name the unit in `labels`, as `c(no2 = "NO2, ug/m3")`, and the row reads `"NO2, ug/m3 (per 10)"`. `contrast = "iqr"` takes the increment from the data instead: the interquartile range of the exposure as the model saw it, which is what an exposure with no natural unit is usually reported per. The range it came to is written beside the variable, as `"NO2 (per IQR, 8.44)"`, because an effect per interquartile range cannot be compared with anything unless the figure says which range that was. `contrast` says nothing about a categorical exposure, whose comparisons are its levels.

`at`
: The two values of the exposure to contrast, as `c(from, to)`. Which two they were is written beside the exposure, as `"NO2 (10 -> 20)"`, wherever the exposure is named: every row of a figure is that same comparison taken within another subgroup, so it is said once rather than on each of them. An exposure entered as a spline, or in any other way that spreads it over more than one coefficient, has no single effect to report and is drawn only when `at` names the two values; any other exposure may be given them too. `at` and `contrast` both say which two values are compared, so only one of them is accepted at a time, but they do not say it the same way: `contrast` is an increment taken from the middle of the exposure's own distribution, and `at` is the two values themselves. Where the exposure enters the model as it stands the two come to the same number, an increment being the same difference wherever it is taken; where it enters transformed – `log(no2)`, a spline, a polynomial – they do not, and `at` is the one that says where on the curve the difference was taken. For a categorical exposure the two values are two of its levels, and the figure is then that one comparison rather than a row for every level.

`vcov`
: Robust standard errors. `NULL`, the default, uses the model's own. `"robust"` gives the heteroskedasticity-consistent sandwich estimator (`HC1`), and `"HC0"` to `"HC4"` name one exactly; both come from the `sandwich` package. A function is called

on the fit, and a matrix is used as it stands. For a Cox model refit with `robust = TRUE`; a fit that is already robust is used as it is. A `survey::svyglm()` fit refuses both `vcov` and `cluster`: its variance is the design-based one already, and the clustering is part of the design it was fitted to.

`cluster` Cluster-robust standard errors, passed to `sandwich::vcovCL()`. It says which observations belong together, so it takes a column name, a vector of one identifier per observation, or a one-sided formula: `cluster = "practice_id"`, `cluster = data$practice_id` or `cluster = ~practice_id`.

`test` How the interaction is tested: `"lrt"`, the default, is the likelihood ratio test against the same model without the interaction terms; `"wald"` is the joint Wald test of those terms; `"both"` reports the two of them in columns of their own. See *Testing the interaction*.

`table` Whether to draw the table of numbers beside the plot. Defaults to `TRUE`, a forest plot being read from the numbers as much as from the marks; `table = FALSE` leaves a plain figure, and `summary()` and `tidy()` report the same numbers at the console either way.

`columns` Which columns the table carries, from `"estimate"`, `"p"`, `"n"`, `"events"`, `"person_time"`, `"interaction_p"` and, where both tests of an interaction were asked for, `"interaction_p_lrt"`. The default shows the estimate and the p-value together with whichever of the others the models can supply. On a figure reporting a test of an interaction the p-value of each row is left off, being easily read as the test beside it; name it in `columns` to have it back.

`person_time` The unit person-time is reported in, for a model that carries any. `NULL`, the default, reports the total the model was fitted over. A number divides by it, so `person_time = 1000` draws a column of thousands of person-years and heads it `"Person-time (per 1,000)"`, since a count of person-time that does not say what it counts cannot be read against another study's. Naming the number heads the column outright, as `person_time = c("Person-years (per 1,000)" = 1000)`. The unit reaches the figure, `summary()` and the HTML report alike. It does not refit the model or change its estimates, confidence intervals, or p-values: `person_time` controls only how the person-time column is written.

`layout` How the figure is drawn: the name of a style, as `layout = "jama"`, or a layout built by `foresty_layout()` when something about it has to be changed. The styles are `"classic"`, `"jama"`, `"nejm"`, `"lancet"`, `"bmj"` and `"revman"`.

`title` Plot title. The default names the measure, the outcome it is a measure of, the exposure, the modifier and the fact that every subgroup estimate comes out of one model carrying their interaction term, as `"Adjusted odds ratio for asthma associated with NO2 within each level of Sex, from one model containing their interaction term"`. `NA` draws none, and the journal styles draw none, a caption being where a journal puts that.

`subtitle` Plot subtitle. `NULL`, the default, draws none.

`xlab` The label under the plot, which by default names the measure and the outcome, as `"Adjusted odds ratio for asthma"`. A string is drawn as it was given – `xlab = "Odds ratio (95% CI), NO2 per 10 ug/m3"` – and `NA` draws none. Renaming only the outcome is what `outcome` is for; this replaces the whole line.

`html` Whether to write the HTML report. `FALSE`, the default, writes nothing, so nothing leaves the session unless it is asked for. `TRUE` writes it to a file named for the variables it is about, the exposure and the modifier joined by an underscore, so that NO2 by sex is written to `no2_sex.html` in the working directory. A path writes it there instead, as `html = "reports/no2_sex.html"`. The report can also be written at any time afterwards with `foresty_report()`.

Details

The point is to make an interaction readable. The p-value of an interaction term reports only that the subgroups depart from a common effect; it does not say which subgroup carries the effect, in which direction, or how far apart they are, and a small p-value from a large study can accompany subgroup estimates that are practically identical. Seeing the subgroup-specific estimates and the test together is what settles whether an interaction is worth reporting.

Each subgroup effect is a linear combination of the coefficients of the single interaction model, not a separate model fitted in each subgroup: in the reference level of the modifier it is the exposure term alone, and in the other levels it is the exposure term plus the relevant interaction terms. Every subgroup therefore shares one estimate of the covariate effects, and the interaction can be tested. The combinations and their tests are computed by `car::linearHypothesis()`.

Value

A `ggplot2` object, of class `foresty`, carrying the estimates and the interaction test. Print it to draw it. `summary()` reports the coefficients and the test, and `tidy()` returns the estimates as a data frame.

One reference group for the whole figure

By default every row is the effect of the exposure inside one subgroup, so each subgroup is read against its own reference level and the rows of one subgroup are not comparisons with the rows of another. Where the exposure and the modifier are both categorical there is a second question a figure can answer: what every combination of the two comes to against one of them. `reference` asks it. `reference = c(ecog = "0-1", egfr = "Negative")` names the one combination the rest are compared with, and each row is then that whole cell – this level of the exposure and this level of the modifier – against that cell, all of it out of the same interaction model. The cell named carries no estimate of its own and is drawn as the reference it is.

The two figures answer different questions and the difference is worth being clear about. Without `reference`, a row says how much the exposure matters in that subgroup, and the interaction test beside the rows says whether those effects differ. With it, a row says how far that combination of the two variables is from one chosen combination, which is what a table of six groups against one baseline group reports, and the interaction test beside them is the same test of the same coefficients: it still asks whether the effect of the exposure depends on the modifier, and it is not a test of the rows.

`reference = TRUE` takes the first level of each, which is the cell a model fitted under R's default treatment contrasts takes as its own baseline. Which cell that is makes no difference to the estimates – each row is a difference between two rows of the design matrix, so it comes out the same however the factors were coded – but it is what the rest of the figure is read against, so name the cell with `reference = c(...)` where the first level is not the one a reader should start from.

Testing the interaction

The p-value beside the subgroups is the joint test that every exposure by modifier coefficient is zero. It is the likelihood ratio test by default, comparing the likelihood of the model carrying the interaction with that of the same model without it.

`test = "wald"` takes the joint Wald test over the coefficients and their covariance instead, which is what a model summary reports and what a robust or cluster-robust variance changes. The two tests answer the same question and usually agree; they part company where the Wald approximation is poor – a small study, a rare outcome, a sparse subgroup, an estimate far from the null – and the likelihood ratio test is the more trustworthy of the two there, which is why it is the default. `test = "both"` reports each in a column of its own, which is a way of showing that the reported p-value does not turn on which test was chosen.

The likelihood ratio test is computed from the likelihood, so it has no robust form: it takes no account of a variance passed through `vcov` or `cluster`. Where it would not match the intervals drawn beside it for that reason, the Wald test is reported in its place and `foresty` says so; ask for the likelihood ratio test by name there and it is given, with a warning that it does not match them.

A model with no likelihood at all is a different case, and no likelihood ratio test is reported for one however it is asked for. A GEE is the one that comes up: it estimates its coefficients from estimating equations rather than from a likelihood, and its standard errors are the sandwich ones from the start, so the joint Wald test is the test of it. A quasi-likelihood family is the same case. `test = "lrt"` on either is an error naming the reason rather than a number that would look like a likelihood ratio test and not be one; the default and `test = "both"` report the Wald test and say that they did.

For a linear model the likelihood ratio test is the chi-square form rather than the exact F test, which is what the Wald test gives there.

A `survey::svyglm()` fit maximizes a pseudo-likelihood weighted by the design, so the ordinary likelihood ratio test does not apply to it. In its place `test = "lrt"` takes the Rao-Scott working likelihood ratio test that `survey` reports for two nested fits, through `survey::anova.svyglm()`, and `test = "wald"` the F test on the degrees of freedom of the design, which is what `survey::regTermTest()` reports. Both are design-based from the start, so neither is affected by the caveat above.

What the counts beside the rows count

The table beside the plot reports the number of people behind each row and, where the outcome is an event, how many of them had it. Both are counted over the rows the model was fitted to and not over the data frame it was fitted from, so anyone missing the outcome, the exposure or any covariate in the formula is left out of them – which is the same complete-case set the estimates come from, and is what makes the two agree. A column that is not in the model does not affect them however much of it is missing.

A row of a forest plot is usually half of a comparison, so by default the other half is beside it: a row of suburban children reads `822 vs 468`, those being the 822 the row is of and the 468 rural ones its odds ratio was estimated against. `counts = "row"` in `foresty_layout()` holds the row's own group alone instead. What the two groups are depends on what the row compares:

- A **continuous exposure** compares no two groups of people – its row is a step along a slope – so the count is the one group the row is of: every row of the model's data in a `foresty_main()` figure, everyone in the subgroup in a `foresty_interaction()` one.

- A **categorical exposure** compares one of its levels with the level the contrast is from, so a row carries the two: `822 vs 468` people and `125 vs 59` events. The reference row is the group the others are compared with rather than a comparison of its own, so it carries one number.
- A **multinomial fit** (`nnet::multinom()`) of an exposure with no levels compares two levels of the outcome, and the two numbers are how many people were at each: a "Transient vs None" row of women reads `637 vs 1,050`. That is the only pair of counts such a row has – the events column would repeat it – so it goes under the sizes and the events column is left off. Of a categorical exposure, the row compares the exposure's levels within one comparison of the outcome, so the sizes are the two exposure groups whatever their outcome and the events are the people at the row's outcome level within each.
- An **ordinal fit** (`MASS::polr()`) has one set of coefficients for the whole outcome, so its rows carry sizes and no count of events.

A multinomial estimate is worth one caution the counts cannot give: it did not come out of the two groups beside it alone. All the equations are fitted over the whole outcome at once, so the people at the levels a row is not about bear on it too, and a row reading `637 vs 1,050` is not the logistic regression of those 1,687.

The figures whose rows compare two groups say what the counts are under the plot in `foresty_app()` and under the table of estimates in `foresty_report()`; the ones that do not say nothing.

Adjusting the figure

The result is a `ggplot2` object, so layers, scales and themes are added to it as usual, and `+` always reaches the forest:

```
foresty_main(list(fit), "no2") + ggplot2::coord_cartesian(xlim = c(0.8, 2))
foresty_main(list(fit), "no2") + ggplot2::theme_minimal(base_size = 14)
```

`&` reaches every panel of a figure that has more than one, which is worth knowing about but rarely what you want: a scale or a coordinate system applied to the table of numbers beside the plot will spoil it. Use `+`.

See Also

`foresty_main()`, `foresty_layout()`, `foresty_report()`.

Examples

```
fit <- glm(asthma ~ no2 + sex + maternal_smoking + maternal_age,
           family = binomial, data = foresty_cohort)

# Sex, where the interaction is real.
by_sex <- foresty_interaction(fit, exposure = "no2", interaction = "sex")
by_sex
summary(by_sex)

# The rest of what the function can be asked for, drawn one after
# another. Each is quick; there are simply several of them.
```

```
# Maternal smoking, where it is not.
foresty_interaction(fit, exposure = "no2", interaction = "maternal_smoking")

# Naming the modifier labels it, and both tests of the interaction can be
# reported side by side.
foresty_interaction(fit, exposure = c(NO2 = "no2"),
                    interaction = c(Sex = "sex"), test = "both")

# Every combination of two categorical variables against one of them, which
# is the other question a two-way table of groups asks.
fit2 <- glm(asthma ~ urbanicity + sex + maternal_age, family = binomial,
            data = foresty_cohort)
foresty_interaction(fit2, exposure = "urbanicity", interaction = "sex",
                    reference = c(urbanicity = "Rural", sex = "Female"))

# The subgroup estimates, the numbers behind them and the test of the
# interaction, in the layout of a journal.
foresty_interaction(fit, exposure = "no2", interaction = "sex",
                    layout = "jama")

# A multinomial outcome. Each subgroup carries one row per comparison
# between outcome levels, and the interaction is tested jointly across the
# equations, so it has one degree of freedom for each of them.
if (requireNamespace("nnet", quietly = TRUE)) {
  fit_phenotype <- nnet::multinom(
    wheeze_phenotype ~ no2 + sex + maternal_smoking,
    data = foresty_cohort, trace = FALSE
  )
  print(foresty_interaction(fit_phenotype, exposure = "no2",
                            interaction = "sex", contrast = 10))
}
```

`foresty_layout` *Layout and style of a foresty figure*

Description

Every drawing decision the figure makes is held in one object: the colors, the marks, the rules between the rows, how the numbers beside the plot are written, and where the columns sit. Pass the name of a style to `foresty_main()` or `foresty_interaction()` as `layout = "jama"`, or build one here and change any part of it.

Usage

```
foresty_layout(
  style = c("classic", "jama", "nejm", "lancet", "bmj", "revman"),
  base_size = NULL,
  family = NULL,
```

```
  color = NULL,
  color_by = NULL,
  colors = NULL,
  theme = NULL,
  palette = NULL,
  point_shape = NULL,
  point_size = NULL,
  emphasis_shape = NULL,
  emphasis_height = NULL,
  emphasis_size = NULL,
  emphasis_face = NULL,
  emphasis_gap = NULL,
  interval_width = NULL,
  null_line = NULL,
  grid = NULL,
  axis_line = NULL,
  band = NULL,
  rules = NULL,
  separators = NULL,
  group_position = NULL,
  table_side = NULL,
  header_face = NULL,
  group_face = NULL,
  digits = NULL,
  p_format = NULL,
  decimal_mark = NULL,
  ci_separator = NULL,
  ci_brackets = NULL,
  column_gap = NULL,
  counts = NULL,
  headings = NULL,
  xlim = NULL,
  arrows = NULL,
  arrows_position = NULL,
  plot_width = NULL,
  min_plot_width = NULL,
  auto_labels = NULL
)
```

Arguments

style	Name of the style to start from, one of "classic", "jama", "nejm", "lancet", "bmj" and "revman". A foresty_layout object is also accepted, and is then the thing being modified.
base_size	Base font size in points. Everything else is drawn relative to it.
family	Font family, as "serif" or "Times New Roman". NULL leaves the device's default. How wide each column of text is made is worked out from a table of character widths rather than by asking the device, so a family far from the de-

fault one may leave a column slightly wide or slightly narrow; the columns still hold what is in them.

`color` One color for the estimates and their intervals, which is the usual thing to change. `palette` changes them apart, and `color_by` draws the rows in colors of their own instead. `foresty_colors()` names a color of a ColorBrewer palette without pasting a hex code, as `color = foresty_colors("Dark2")[3]`.

`color_by` What the colors change with, for a figure drawn in more than one. `"none"`, the default, draws the whole figure in `color`. `"category"` gives every category of the rows a color of its own: the levels of a categorical exposure – "Yes" and "No" – where the rows are levels, and the subgroups of the modifier where they are subgroups, which is what an interaction figure draws. A category keeps its color wherever it appears, so a combined figure reads across its blocks. `"row"` gives every row a color, whatever it is of. The reference level of a categorical exposure is drawn hollow either way, being a definition rather than an estimate. No legend is drawn: every row is labelled already, and a legend repeating the labels is a second copy of them to keep in step. A figure drawn from a table of estimates can say which row is which color instead of leaving it to a rule, by carrying a column of them: see `color` in `foresty_data()`.

`colors` The colors `color_by` draws the categories in, in order, and cycled where there are more categories than colors. The name of a ColorBrewer palette – `"Dark2"`, the default, `"Set1"` or `"Set2"` – or the colors themselves, as `c("#1B9E77", "#D95F02")`. `foresty_colors()` builds one starting from a chosen place in a palette, as `colors = foresty_colors("Dark2", start = 3)`.

`theme` The `ggplot2` theme the plot itself is drawn on: `"void"`, the default, which is the plain panel a forest plot is usually drawn as, or the name of one of `ggplot2`'s own – `"minimal"`, `"bw"`, `"classic"`, `"light"`, `"linedraw"`, `"grey"` or `"dark"` – for its background, its border and its grid. A theme object is also accepted, as is a theme function, which is called with `base_size` and `base_family` and so has to take them, as `ggplot2`'s own do. It is the plot's theme, not the figure's: the columns of numbers beside it are a table rather than a plot and keep their own. Everything the layout sets – the sizes, the colors of the text, whether the axis line and the grid are drawn – is applied over it, so `grid = TRUE` and `theme = "bw"` are not two answers to the same question.

`palette` Named character vector overriding single colors, as `c(estimate = "#B24745", null = "grey60")`. The names are `estimate`, `border` (the outline of the marks), `reference` (the fill of the reference level's hollow mark), `interval`, `null`, `rule`, `band`, `text`, `header`, `group` and `axis`.

`point_shape, point_size`
Plotting symbol and its size. The default is a filled square, as a forest plot is usually drawn with.

`emphasis_shape, emphasis_height, emphasis_size, emphasis_face`
How a row singled out for emphasis is drawn – the overall estimate on a `foresty_combine()` figure, which a reader should be able to find without hunting for it.

`emphasis_shape` is a plotting symbol, and the row is drawn as the other rows are – a mark on an interval – in that symbol at `emphasis_size`, which may be left `NULL` for `point_size` times 1.5. The default, `23`, is the filled diamond,

which says the row is the summary while leaving its interval to be read as the others are.

`emphasis_shape = "diamond"` draws instead the wide summary diamond a paper draws: a filled diamond whose two side vertices sit on the confidence limits and whose apex sits on the estimate, drawn in place of the interval rather than on top of it. `emphasis_height` is how tall that diamond is, as a fraction of the space between one row and the next, and `emphasis_size` says nothing about it.

`emphasis_face` is the font face of the row's label, and applies either way.

`emphasis_gap` Whether a row singled out for emphasis is set apart from the rest by a rule of its own, drawn whether or not `separators` rules between subgroups, so that the overall estimate is read as its own thing rather than as another subgroup.

`interval_width` Line width of the confidence intervals.

`null_line` Line type of the line at the null, as `"dashed"` or `"solid"`. `"none"` draws none.

`grid` Whether to draw vertical grid lines behind the estimates.

`axis_line` Whether to draw the axis line under the plot.

`band` Whether to shade alternate rows, which helps a reader carry the eye across a wide table of numbers.

`rules` Where to rule the figure: `"header"` draws a line under the column headings, `"full"` adds one above them and one under the last row, and `"none"` draws neither.

`separators` Whether to draw a thin rule between one subgroup and the next.

`group_position` Where the name of a subgroup goes: `"row"` puts it on a row of its own above the rows it covers, and `"column"` puts it in a column of its own to the left of them.

`table_side` Which side of the plot the numbers are written on, `"right"` or `"left"`.

`header_face, group_face`
Font face of the column headings and of the subgroup names, as `"bold"` or `"plain"`.

`digits` Digits the estimates are written to.

`p_format` How p-values are written. `"default"` writes `0.032` and `<0.001`; `"jama"` drops the leading zero and rounds as JAMA asks, to three places up to `0.01` and two above it.

`decimal_mark` Decimal point, `"."` unless a journal asks otherwise.

`ci_separator` What goes between the confidence limits. The default picks `"-"` when every number on the figure is positive and `" to "` when one is not, a hyphen between negative numbers being unreadable.

`ci_brackets` The pair of brackets the interval is written in, as `c("[", "]")`.

`column_gap` Space between one column of numbers and the next, in ems, an em being about two digits wide. Lower it to draw the numbers tighter and leave the plot more of the figure.

`counts` What the columns of counts hold on a row that compares two groups of people. `"compared"`, the default, holds both of them, the row's own group and the group its estimate is compared with, written as `822 vs 468`: the odds ratio on that row came out of those 822 and those 468, and a row carrying one of the two numbers

leaves the reader to find the other. "row" holds the row's own group alone, which is what a figure drawn before this option existed held. The rows that compare no two groups of people are unaffected either way – a step along a continuous exposure, and the reference rows, which are the group the others are compared with rather than a comparison of their own. See *What the counts beside the rows count* in foresty_main().

headings — Named character vector renaming the column headings, as c(n = "No. of patients"). The names are estimate, label, p, n, events, person_time, interaction_p, and interaction_p_wald and interaction_p_lrt for a figure reporting both tests of the interaction. The estimate's heading is written from the model, naming the measure and the confidence level; the outcome is named on the axis under the plot rather than twice over. label heads the column of row labels, and defaults to "Exposure" in foresty_main(), the modifier's name in foresty_interaction() and "Subgroup" in foresty_combine(), which leaves it unheaded when the figure carries an overall estimate as well as subgroups. What n and events are counts of, which is not the same thing for every figure, is in *What the counts beside the rows count* in foresty_main(); rename them to say it where a caption does not.

xlim — Limits of the plot, as c(0.5, 4). Intervals running past them are drawn with an arrow at the end, which is what keeps one wide interval from flattening the rest of the figure.

arrows — Two labels for the directions of the effect, as c("Favours treatment", "Favours control"), drawn under the plot with an arrow apiece. NULL draws none.

arrows_position — Whether those labels go at the "bottom" of the plot or at the "top".

plot_width — How much of the figure the plot itself takes. NULL, the default, gives it whatever the columns of text leave, so that widening the figure widens the plot and nothing else. A fraction, as 0.6, gives it that share of the figure whatever the figure is drawn at, and the columns of text share the rest. A number of centimetres, or a grid::unit(), fixes it exactly.

min_plot_width — The least of the figure the plot is allowed to be left with, as a fraction. What the columns of text leave depends on the width the figure is drawn at, which is not known while it is being built, so a wide table – a survival model carrying N, events and person-time – can leave the plot a centimetre and squash the axis under it into a row of overprinted numbers. This is the floor: where the columns of text would leave the plot less than this share of the width it is being drawn at, the plot is given that share and the columns of text share the rest in proportion to what they hold. 0 turns the floor off and lets the plot have whatever is left, however little that is. It has no effect where plot_width says outright how wide the plot is.

auto_labels — Whether foresty_interaction() writes its own title and subtitle. The journal styles leave them off, since that text belongs in the caption; a title passed by hand is always drawn.

Details

The styles are the ones a paper is usually asked for, read off the forest plots those journals print and the layouts of the meta package, which foresty follows here. They are approximations of a house

style, not the style sheet itself: a journal will still ask for its own fonts and sizes, and every element below can be overridden.

`"classic"` The default. Black marks on white, a dashed line at the null, a rule under the column headings and thin rules between subgroups.

`"jama"` Numbers to the left of the plot, rules above and below the whole block, navy squares, a solid line at the null, p-values without their leading zero (`.03`, `<.001`), and "No." and "P value" as headings. The title and subtitle that `foresty_interaction()` writes for itself are left off, the caption being where a journal puts them.

`"nejm"` Numbers to the right, blue squares, en dashes between the confidence limits, "P value" as the heading.

`"lancet"` As `"nejm"`, in the Lancet's blue, with the middle dot for a decimal point, so that 1.17 is written with one, and en dashes.

`"bmj"` The BMJ's violet, grey rules, and "to" between the confidence limits.

`"revman"` Cochrane's RevMan: blue squares and intervals written `1.15 [0.93, 1.42]`.

Value

An object of class `foresty_layout`.

See Also

`foresty_main()`, `foresty_interaction()`.

Examples

```
fit <- glm(asthma ~ no2 + sex + maternal_smoking + maternal_age,
           family = binomial, data = foresty_cohort)

foresty_interaction(fit, "no2", "sex", table = TRUE, layout = "jama")

# A style, changed where it needs to be.
foresty_interaction(
  fit, "no2", "sex", table = TRUE,
  layout = foresty_layout("jama", color = "#B24745", base_size = 11)
)

# Trimming a wide interval rather than letting it flatten the figure.
fit_urban <- glm(asthma ~ urbanicity + sex + maternal_age,
                 family = binomial, data = foresty_cohort)
foresty_main(list(fit_urban), "urbanicity", table = TRUE,
             layout = foresty_layout("classic", xlim = c(0.8, 2.5),
                                     arrows = c("Lower risk", "Higher risk")))
```

foresty_main *Forest plot of exposure effects across models*

Description

Draws the effect of one exposure from each of several fitted models, one row apiece, so that exposures that were each fitted in their own model come onto a single figure. None of the models may interact its exposure with anything; use `foresty_interaction()` for those.

Usage

```
foresty_main(
  fits,
  exposure,
  measure = NULL,
  exponentiate = TRUE,
  labels = NULL,
  outcome = NULL,
  outcome_reference = NULL,
  outcome_reference_row = FALSE,
  ci_level = 0.95,
  contrast = NULL,
  at = NULL,
  vcov = NULL,
  cluster = NULL,
  table = TRUE,
  columns = NULL,
  person_time = NULL,
  layout = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  html = FALSE
)
```

Arguments

`fits`	A list of fitted models. Models fitted by `stats::glm()`, `stats::lm()`, the `survival` package and `survey::svyglm()` are supported, as is any fit supplying `coef()`, `vcov()` and a model frame. A survey-weighted fit keeps the design-based variance it was fitted with and is referred to a t and an F on the degrees of freedom of its design, as `survey` itself does; the counts beside its rows are the unweighted numbers of people in the sample, taken over the rows the fit gave weight to, so a design that was `subset()` counts the subset.
`exposure`	Name of the exposure variable in each model, as a character vector: either one name for all of them, or one name per model. Naming an element, as `exposure`

= c(NO2 = "no2"), gives that variable its label on the figure, which saves repeating it in labels.

measure
: Effect measure, one of "OR", "RR", "HR", "IRR", "MD" or "Coefficient". The default reads it from the model: a logistic regression gives an odds ratio, as do the ordinal and multinomial forms of one, a Cox model a hazard ratio, a Poisson model with an offset an incidence rate ratio, and a linear model a mean difference. Ratios are exponentiated; differences are not. What an offset holds is not recorded anywhere, so a Poisson model offset by something other than person-time – the size of a population, say – is named here instead, as measure = "RR".

exponentiate
: Whether a ratio measure is drawn as a ratio. TRUE, the default, draws an odds ratio as an odds ratio. FALSE leaves it on the scale the model was fitted on: the figure reports a log odds ratio, read against zero rather than against one, and the estimates and their intervals come back on that scale from tidy() as well. It says nothing about a mean difference or a coefficient, which are on that scale already and are never exponentiated.

labels
: Named character vector giving the label to draw for a variable, as c(no2 = "Nitrogen dioxide"). Names not matched are left as they are. This is where the exposure is renamed, and naming it where it is chosen – exposure = c(NO2 = "no2") – comes to the same thing.

outcome
: What to call the outcome, as outcome = "incident asthma". The default takes it from the left of the model's formula, which is the name of a column – asthma_ever_dx, evt5 – and is rarely what a figure should say the effect is an effect on. It is written wherever the outcome is named: the axis under the plot, the title the package writes for itself, and the HTML report. NA names none, leaving "Adjusted odds ratio" on its own for a figure whose caption says what of.

outcome_reference
: For a multinomial logistic regression, the level of the outcome every estimate is read against, as outcome_reference = "None". NULL, the default, is the level the model itself was referred to, read off the fit as the level it holds no equation for rather than assumed to be any particular one. Naming another does not refit anything: the odds ratio of one level against another is the difference between their two equations, and the covariance of the pair is already in the model. Every other level is then drawn against it, one row apiece, each row saying which two levels it compares. It says nothing about a model of one equation, whose reference is fixed by how the outcome is coded, and is refused there rather than ignored.

outcome_reference_row
: Whether that level is drawn as a row of its own. FALSE, the default, draws only the levels compared with it, each row saying which two levels it compares. TRUE adds a row for the reference level itself, the way the reference level of a categorical exposure is drawn: it carries no estimate, being the definition the other rows are differences from, so it is 1 on the ratio scale and 0 on the scale the model was fitted on, with no interval, no test and no p-value. It is one row whatever the exposure is, since it is the same definition at every value of it, and the counts beside it are of everybody in that level of the outcome for the same reason: the row is not about a value of the exposure, so it is not counted at one.

On a figure of subgroups it is drawn once inside each and counted within it. It says nothing about a model of one equation.

`ci_level` Confidence level of the intervals. Defaults to `0.95`.

`contrast` For a continuous exposure, the increment the effect is reported per. `NULL`, the default, is one unit and is not written on the figure. An increment you name is: `contrast = 10` draws the row as `"NO2 (per 10)"` and `contrast = 1` draws it as `"NO2 (per 1)"`, the same estimate as the default said out loud. No unit is invented, since the package cannot know what a column's numbers mean; name the unit in `labels`, as `c(no2 = "NO2, ug/m3")`, and the row reads `"NO2, ug/m3 (per 10)"`. `contrast = "iqr"` takes the increment from the data instead: the interquartile range of the exposure as the model saw it, which is what an exposure with no natural unit is usually reported per. The range it came to is written beside the variable, as `"NO2 (per IQR, 8.44)"`, because an effect per interquartile range cannot be compared with anything unless the figure says which range that was. `contrast` says nothing about a categorical exposure, whose comparisons are its levels.

`at` The two values of the exposure to contrast, as `c(from, to)`. Which two they were is written beside the exposure, as `"NO2 (10 -> 20)"`, wherever the exposure is named: every row of a figure is that same comparison taken within another subgroup, so it is said once rather than on each of them. An exposure entered as a spline, or in any other way that spreads it over more than one coefficient, has no single effect to report and is drawn only when `at` names the two values; any other exposure may be given them too. `at` and `contrast` both say which two values are compared, so only one of them is accepted at a time, but they do not say it the same way: `contrast` is an increment taken from the middle of the exposure's own distribution, and `at` is the two values themselves. Where the exposure enters the model as it stands the two come to the same number, an increment being the same difference wherever it is taken; where it enters transformed – `log(no2)`, a spline, a polynomial – they do not, and `at` is the one that says where on the curve the difference was taken. For a categorical exposure the two values are two of its levels, and the figure is then that one comparison rather than a row for every level.

`vcov` Robust standard errors. `NULL`, the default, uses the model's own. `"robust"` gives the heteroskedasticity-consistent sandwich estimator (`HC1`), and `"HC0"` to `"HC4"` name one exactly; both come from the `sandwich` package. A function is called on the fit, and a matrix is used as it stands. For a Cox model refit with `robust = TRUE`; a fit that is already robust is used as it is. A `survey::svyglm()` fit refuses both `vcov` and `cluster`: its variance is the design-based one already, and the clustering is part of the design it was fitted to.

`cluster` Cluster-robust standard errors, passed to `sandwich::vcovCL()`. It says which observations belong together, so it takes a column name, a vector of one identifier per observation, or a one-sided formula: `cluster = "practice_id"`, `cluster = data$practice_id` or `cluster = ~practice_id`.

`table` Whether to draw the table of numbers beside the plot. Defaults to `TRUE`, a forest plot being read from the numbers as much as from the marks; `table = FALSE` leaves a plain figure, and `summary()` and `tidy()` report the same numbers at the console either way.

`columns` Which columns the table carries, from `"estimate"`, `"p"`, `"n"`, `"events"`, `"person_time"`, `"interaction_p"` and, where both tests of an interaction were asked for, `"interaction_p_lrt"`. The default shows the estimate and the p-value together with whichever of the others the models can supply. On a figure reporting a test of an interaction the p-value of each row is left off, being easily read as the test beside it; name it in `columns` to have it back.

`person_time` The unit person-time is reported in, for a model that carries any. `NULL`, the default, reports the total the model was fitted over. A number divides by it, so `person_time = 1000` draws a column of thousands of person-years and heads it `"Person-time (per 1,000)"`, since a count of person-time that does not say what it counts cannot be read against another study's. Naming the number heads the column outright, as `person_time = c("Person-years (per 1,000)" = 1000)`. The unit reaches the figure, `summary()` and the HTML report alike. It does not refit the model or change its estimates, confidence intervals, or p-values: `person_time` controls only how the person-time column is written.

`layout` How the figure is drawn: the name of a style, as `layout = "jama"`, or a layout built by `foresty_layout()` when something about it has to be changed. The styles are `"classic"`, `"jama"`, `"nejm"`, `"lancet"`, `"bmj"` and `"revman"`.

`title` Plot title. The default names the measure, the outcome it is a measure of, the exposure it is reported for and the fact that the estimate comes from a model with no interaction term in it, as `"Adjusted odds ratio for asthma associated with NO2, from one model without an interaction term"`. `NA` draws none, and the journal styles draw none, a caption being where a journal puts that.

`subtitle` Plot subtitle. `NULL`, the default, draws none.

`xlab` The label under the plot, which by default names the measure and the outcome, as `"Adjusted odds ratio for asthma"`. A string is drawn as it was given – `xlab = "Odds ratio (95% CI), NO2 per 10 ug/m3"` – and `NA` draws none. Renaming only the outcome is what `outcome` is for; this replaces the whole line.

`html` Whether to write the HTML report – the model it was drawn from, the estimates, the figure and the whole coefficient table, on a page that is a single file and can be sent on. `FALSE`, the default, writes nothing, so nothing leaves the session unless it is asked for. `TRUE` writes it to a file named for the exposures it is about, joined by underscores where there is more than one, so that a figure of NO2 is written to `no2.html` in the working directory. A path writes it there instead, as `html = "reports/no2.html"`. The report can also be written at any time afterwards with `foresty_report()`.

Details

The effect is the difference between two rows of the model's own design matrix, one at the baseline value of the exposure and one at the value it is compared with, so it is read off correctly whether the exposure is continuous, binary or a factor with several levels, and whatever contrast coding the fitting function used. The estimate and its confidence interval are computed by `car::linearHypothesis()`.

A categorical exposure gets one row per level, the reference level included and marked as such, with the levels named on the rows and the variable named once at the left of the figure.

Value

A ggplot2 object, of class foresty, carrying the estimates it was drawn from. Print it to draw it. summary() reports the coefficients and tests behind it, tidy() returns the estimates as a data frame, and predict() passes through to the underlying model.

A splined exposure

An exposure entered as a spline has no single effect to report: the difference it makes depends on where along the curve it is taken. So the two values are named, and the row says which they were:

```
fit <- glm(asthma ~ splines::ns(no2, 3) + sex, family = binomial, data = d)
foresty_main(list(fit), exposure = "no2", at = c(10, 20))
```

The basis has to be built inside the formula, by splines::ns(), splines::bs(), stats::poly() or another function of the variable, so the two design-matrix rows can be evaluated at the two values.

A basis computed before the fit and entered as columns of its own – Hmisc::rcspline.eval() written into spline1, spline2, spline3 and then fitted as y ~ spline1 + spline2 + spline3 – cannot be handled that way, because nothing in the fit records that those three columns are one variable or how to recompute them at another value. Name the exposure and the model refuses, saying that it is not in the model. Put the basis in the formula instead, which fits exactly the same model:

```
knots <- quantile(d$age, probs = c(0.05, 0.35, 0.65, 0.95))
fit <- glm(y ~ splines::ns(age, 4) + sex, family = binomial, data = d)
foresty_main(list(fit), exposure = "age", at = c(30, 60))
```

What the counts beside the rows count

The table beside the plot reports the number of people behind each row and, where the outcome is an event, how many of them had it. Both are counted over the rows the model was fitted to and not over the data frame it was fitted from, so anyone missing the outcome, the exposure or any covariate in the formula is left out of them – which is the same complete-case set the estimates come from, and is what makes the two agree. A column that is not in the model does not affect them however much of it is missing.

A row of a forest plot is usually half of a comparison, so by default the other half is beside it: a row of suburban children reads 822 vs 468, those being the 822 the row is of and the 468 rural ones its odds ratio was estimated against. counts = "row" in foresty_layout() holds the row's own group alone instead. What the two groups are depends on what the row compares:

- A **continuous exposure** compares no two groups of people – its row is a step along a slope – so the count is the one group the row is of: every row of the model's data in a foresty_main() figure, everyone in the subgroup in a foresty_interaction() one.
- A **categorical exposure** compares one of its levels with the level the contrast is from, so a row carries the two: 822 vs 468 people and 125 vs 59 events. The reference row is the group the others are compared with rather than a comparison of its own, so it carries one number.

- A **multinomial fit** (`nnet::multinom()`) of an exposure with no levels compares two levels of the outcome, and the two numbers are how many people were at each: a "Transient vs None" row of women reads `637 vs 1,050`. That is the only pair of counts such a row has – the events column would repeat it – so it goes under the sizes and the events column is left off. Of a categorical exposure, the row compares the exposure's levels within one comparison of the outcome, so the sizes are the two exposure groups whatever their outcome and the events are the people at the row's outcome level within each.
- An **ordinal fit** (`MASS::polr()`) has one set of coefficients for the whole outcome, so its rows carry sizes and no count of events.

A multinomial estimate is worth one caution the counts cannot give: it did not come out of the two groups beside it alone. All the equations are fitted over the whole outcome at once, so the people at the levels a row is not about bear on it too, and a row reading `637 vs 1,050` is not the logistic regression of those 1,687.

The figures whose rows compare two groups say what the counts are under the plot in `foresty_app()` and under the table of estimates in `foresty_report()`; the ones that do not say nothing.

Adjusting the figure

The result is a `ggplot2` object, so layers, scales and themes are added to it as usual, and `+` always reaches the forest:

```
foresty_main(list(fit), "no2") + ggplot2::coord_cartesian(xlim = c(0.8, 2))
foresty_main(list(fit), "no2") + ggplot2::theme_minimal(base_size = 14)
```

`&` reaches every panel of a figure that has more than one, which is worth knowing about but rarely what you want: a scale or a coordinate system applied to the table of numbers beside the plot will spoil it. Use `+`.

See Also

`foresty_interaction()`, `foresty_layout()`, `foresty_report()`.

Examples

```
fit_no2 <- glm(asthma ~ no2 + sex + maternal_smoking + maternal_age,
               family = binomial, data = foresty_cohort)
fit_bc <- glm(asthma ~ black_carbon + sex + maternal_smoking + maternal_age,
              family = binomial, data = foresty_cohort)

foresty_main(
  list(fit_no2, fit_bc),
  exposure = c("no2", "black_carbon"),
  labels = c(no2 = "NO2", black_carbon = "Black carbon")
)

# A categorical exposure: the levels are named, the variable once at the left.
fit_urban <- glm(asthma ~ urbanicity + sex + maternal_age,
                 family = binomial, data = foresty_cohort)
foresty_main(list(fit_urban), exposure = "urbanicity")
```

```
# Every variation the figure has, drawn one after another. They are
# skipped by the timed run of the examples only because there are many
# of them, not because any one is slow.

# In the layout of a journal, and without the numbers beside the plot.
foresty_main(list(fit_urban), exposure = "urbanicity", layout = "jama")
foresty_main(list(fit_urban), exposure = "urbanicity", table = FALSE)

# Per 10 units of the exposure rather than per 1, which the row says.
foresty_main(list(fit_no2), exposure = "no2", contrast = 10)

# A rate model. The offset is the time each child was followed for, so the
# measure is an incidence rate ratio and the person-time behind each row is
# drawn beside the counts.
fit_rate <- glm(asthma ~ no2 + sex + maternal_age +
                  offset(log(followup_years)),
                family = poisson, data = foresty_cohort)
foresty_main(list(fit_rate), exposure = "no2",
             labels = c(no2 = "NO2"), contrast = 10)

# A splined exposure: the two values being compared are named.
fit_spline <- glm(asthma ~ splines::ns(no2, 3) + sex + maternal_age,
                  family = binomial, data = foresty_cohort)
foresty_main(list(fit_spline), exposure = "no2", at = c(10, 20))

# On the scale the model was fitted on, as a log odds ratio about zero.
foresty_main(list(fit_no2), exposure = "no2", exponentiate = FALSE)

# The exposure, the outcome and the axis all named by hand.
foresty_main(list(fit_no2), exposure = c(`NO2, ug/m3` = "no2"),
             outcome = "incident asthma by age 8",
             xlab = "Adjusted odds ratio (95% CI)")

# Person-time reported per 1,000 rather than as the total.
foresty_main(list(fit_rate), exposure = "no2", person_time = 1000)

# A multinomial logistic regression has one equation per non-reference level
# of the outcome, so the exposure has one effect per level and the figure
# has one row per level. `outcome_reference` says which level they are all
# read against.
if (requireNamespace("nnet", quietly = TRUE)) {
  fit_phenotype <- nnet::multinom(
    wheeze_phenotype ~ no2 + sex + maternal_smoking,
    data = foresty_cohort, trace = FALSE
  )
  # Against "None", which is the level the model itself was fitted against,
  # and drawn as a row of its own so that the figure says so.
  print(foresty_main(list(fit_phenotype), exposure = "no2", contrast = 10,
                     outcome_reference_row = TRUE))

  # The same estimates read against another level instead.
  against_transient <- foresty_main(list(fit_phenotype), exposure = "no2",
```

```
                                    contrast = 10,
                                    outcome_reference = "Transient")
  summary(against_transient)
}
```

`foresty_report` *Write a foresty figure and its numbers to a self-contained HTML page*

Description

Writes what an interaction p-value was standing in for: the exposure effect within each level of the modifier, the joint test of the interaction, the ratio of the effects between levels, the whole coefficient table of the updated model, and the figure. Everything is inlined, so the file can be opened or sent on its own.

Usage

```
foresty_report(
  x,
  file,
  title = NULL,
  model = NULL,
  width = 10,
  height = NULL,
  open = FALSE
)
```

Arguments

`x`	An object returned by `foresty_interaction()` or `foresty_main()`.
`file`	Path to write to. A `.html` extension is added if missing.
`title`	Browser page title. The default names the exposure and the modifier; it is not displayed in the report body.
`model`	Which model to report the coefficients of, when the figure covers several. The default reports every one of them.
`width, height`	Size of the embedded plot in inches. `height` defaults to a size that fits the number of rows.
`open`	Whether to open the file when it has been written. Defaults to `FALSE`.

Details

A figure drawn from several models – `foresty_main()` over a list of them – describes each of them on the page and gives each its own coefficient table, headed with the exposure it was fitted for. `model` cuts those sections down to one. The rest of the page is the figure itself – the estimates, the test and the plot – and is the whole of it whichever model is named, a figure being one figure however many models went into it.

Value

The path written to, invisibly.

Examples

```
# Writing the page renders the figure and lays the tables out with `gt`,
# which is the slow part, so it is left out of the timed run.

fit <- glm(asthma ~ no2 + sex + maternal_age, family = binomial,
           data = foresty_cohort)
x <- foresty_interaction(fit, exposure = "no2", interaction = "sex")
foresty_report(x, file = file.path(tempdir(), "no2_by_sex.html"))

# The same page can be asked for as the figure is made. `html = TRUE`
# writes it under a name taken from the variables it is about -- here
# no2_sex.html -- into the working directory; a path writes it there
# instead, which is what these examples do so that nothing is written
# outside the session's temporary directory.
foresty_interaction(fit, exposure = "no2", interaction = "sex",
                    html = file.path(tempdir(), "no2_sex.html"))
foresty_main(list(fit), exposure = "no2",
             html = file.path(tempdir(), "no2.html"))
```

`print.foresty` *Draw a forest plot*

Description

Draws the figure, having first made sure the plot in it has room for its axis. A column of text is as wide as what it holds, and the plot is given what those columns leave, which is a decision that cannot be made while the figure is being built: how much they leave depends on the width the figure is drawn at. So a table wide enough – a survival model reporting N, events and person-time beside the estimates – can leave the plot a centimetre and the numbers under its axis printed one on top of another. The floor is applied here, where the width being drawn at is known: see the `min_plot_width` argument of `foresty_layout()`. Where it has to act, the columns of text stop being as wide as what they hold and share what the plot leaves in proportion to it, which is where the room for the axis comes from; a figure with room for its columns already is drawn exactly as it was built.

Usage

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

## S3 method for class 'foresty'
plot(x, ...)
```

```
## S3 method for class 'foresty'
grid.draw(x, recording = TRUE)
```

Arguments

x	A figure returned by foresty_main(), foresty_interaction() or foresty_combine().
...	Passed on to patchwork's and ggplot2's own print methods.
recording	Whether to record the drawing on the display list, as in grid::grid.draw().

Value

x, invisibly.

Examples

```
fit <- glm(asthma ~ no2 + sex, family = binomial, data = foresty_cohort)
print(foresty_interaction(fit, "no2", "sex", table = TRUE))
```

---

summary.foresty *Summarize a foresty figure*

Description

Reports what the figure was drawn from: the model, its whole coefficient table with standard errors and p-values in the manner of summary.glm(), the exposure or subgroup estimates on the reported scale, and the joint test of the interaction where there is one.

Usage

```
## S3 method for class 'foresty'
summary(object, model = NULL, ...)

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

Arguments

object	A foresty object.
model	Which model to report, when the figure covers several. Defaults to the only one.
...	Ignored.
x	A summary.foresty object.

Details

The coefficient table is on the scale the model was fitted on, as a model summary is, so a logistic regression reports log odds. The estimates underneath are on the reported scale, exponentiated where the measure is a ratio.

Value

An object of class `summary.foresty`, a list with elements `call`, `coefficients`, `estimates` and, for an interaction, `interaction_test`.

Examples

```
fit <- glm(asthma ~ no2 + sex + maternal_age, family = binomial,
           data = foresty_cohort)
summary(foresty_interaction(fit, exposure = "no2", interaction = "sex"))
```

Index