Package: ggfixest (via r-universe)

October 9, 2024

Title Dedicated 'ggplot2' Methods for 'fixest' Objects

Version 0.1.0.9001

Date 2023-12-12

Description Provides 'ggplot2' equivalents of fixest::coefplot() and fixest::iplot(), for producing nice coefficient plots and interaction plots. Enables some additional functionality and convenience features, including grouped multi-'fixest' object faceting and programmatic updates to existing plots (e.g., themes and aesthetics).

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

URL https://grantmcdermott.com/ggfixest/

BugReports https://github.com/grantmcdermott/ggfixest/issues

Depends ggplot2 (>= 2.2.0), fixest (>= 0.11.2)

Imports dreamerr, ggh4x, scales, marginaleffects (>= 0.10.0), stats, utils

Suggests knitr, rmarkdown, tinytest (>= 1.4.1), tinysnapshot (>= 0.0.3), magick, rsvg, svglite, fontquiver, data.table

VignetteBuilder knitr

Repository https://grantmcdermott.r-universe.dev

RemoteUrl https://github.com/grantmcdermott/ggfixest

RemoteRef HEAD

RemoteSha b44b7f34f3c8ee84d1f2532953cade5b870dfbd7

Contents

`aggr_es` *Aggregates event-study treatment effects.*

Description

Aggregates post- (and/or pre-) treatment effects of an "event-study" estimation, also known as a dynamic difference-in-differences (DDiD) model. The event-study should have been estimated using the `fixest` package, which provides a specialised `i()` operator for this class of models. By default, the function will return the average post-treatment effect (i.e. across multiple periods). However, it can also return the cumulative post-treatment effect and can be used to aggregate pre-treatment effects too. At its heart, `aggr_es()` is a convenience wrapper around `marginaleffects::hypotheses()`, which is used to perform the underlying joint hypothesis test.

Usage

```
aggr_es(
  object,
  rhs = 0,
  period = "post",
  aggregation = c("mean", "cumulative"),
  abbr_term = TRUE,
  ...
)
```

Arguments

`object`	A model object of class `fixest`, where the `i()` operator has been used to facilitate an "event-study" DiD design. See Examples.
`rhs`	Numeric. The null hypothesis value. Defaults to 0.
`period`	Keyword string or numeric sequence. Which group of periods are we aggregating? Can either be one of three convenience strings—i.e., "post" (the default), "prep", or "both"—or a numeric sequence that matches a subset of periods in the data (e.g. 6:8).
`aggregation`	Character string. The aggregation type. Either "mean" (the default) or "cumulative".
`abbr_term`	Logical. Should the leading "term" column of the return data frame be abbreviated? The default is TRUE. If FALSE, then the term column will retain the full hypothesis test string as per usual with `marginaleffects()`. Note that this information is retained as an attribute of the return object, regardless.
`...`	Additional arguments passed to `marginaleffects::hypotheses()`.

Value

A "tidy" data frame of aggregated (pre and/or post) treatment effects, plus inferential information about standard errors, confidence intervals, etc. Potentially useful information about the underlying hypothesis test is also provided as an attribute. See Examples.

See Also

`marginaleffects::hypotheses()`

Examples

```
library(ggfixest) ## Will load fixest too

est = feols(y ~ x1 + i(period, treat, 5) | id + period, base_did)

# Default hypothesis test is a null mean post-treatment effect
(post_mean = aggr_es(est))
# The underlying hypothesis is saved as an attribute
attributes(post_mean)["hypothesis"]

# Other hypothesis and aggregation options
aggr_es(est, aggregation = "cumulative") # cumulative instead of mean effects
aggr_es(est, period = "pre")             # pre period instead of post
aggr_es(est, period = "both")            # pre & post periods separately
aggr_es(est, period = 6:8)               # specific subset of periods
aggr_es(est, rhs = -1, period = "pre")   # pre period with H0 value of 1
# Etc.
```

`ggcoefplot` *Draw coefficient plots and interaction plots from* `fixest` *regression objects.*

Description

Draws the `ggplot2` equivalents of `fixest::coefplot` and `fixest::iplot`. These "gg" versions do their best to recycle the same arguments and plotting logic as their original base counterparts. But they also support additional features via the `ggplot2` API and infrastructure. The overall goal remains the same as the original functions. To wit: `ggcoefplot` plots the results of estimations (coefficients and confidence intervals). The function `ggiplot` restricts the output to variables created with `i`, either interactions with factors or raw factors.

Usage

```
ggcoefplot(
  object,
  geom_style = c("pointrange", "errorbar"),
  multi_style = c("dodge", "facet"),
```

```
  facet_args = NULL,
  theme = NULL,
  ...
)

ggiplot(
  object,
  geom_style = c("pointrange", "errorbar", "ribbon"),
  multi_style = c("dodge", "facet"),
  aggr_eff = NULL,
  aggr_eff.par = list(col = "grey50", lwd = 1, lty = 1),
  facet_args = NULL,
  theme = NULL,
  ...
)
```

Arguments

`object`	A model object of class `fixest` or `fixest_multi`, or a list thereof.
`geom_style`	Character string. One of `c('pointrange', 'errorbar', 'ribbon')` describing the preferred geometric representation of the coefficients. Note that ribbon plots not supported for `ggcoefplot`, since we cannot guarantee a continuous relationship among the coefficients.
`multi_style`	Character string. One of `c('dodge', 'facet')`, defining how multi-model objects should be presented.
`facet_args`	A list of arguments passed down to `ggplot::fact_wrap()`. E.g. `facet_args = list(ncol = 2, scales = 'free_y')`. Only used if `multi_style = 'facet'`.
`theme`	ggplot2 theme. Defaults to `theme_linedraw()` with some minor adjustments, such as centered plot title. Can also be defined on an existing ggiplot object to redefine theme elements. See examples.
`...`	Arguments passed down to, or equivalent to, the corresponding `fixest::coefplot/fixest::iplot` arguments. Note that some of these require list objects. Currently used are: • `keep` and `drop` for subsetting variables using regular expressions. The `fixest::iplot` help page includes more detailed examples, but these should generally work as you expect. One useful regexp trick worth mentioning briefly for event studies with many pre-/post-periods is `drop = "[[:digit:]]{2}"`. This will cause the plot to zoom in around single digit pre-/post-periods. • `group` a list indicating variables to group over. Each element of the list reports the coefficients to be grouped while the name of the element is the group name. Each element of the list can be either: i) a character vector of length 1, ii) of length 2, or iii) a numeric vector. Special patterns such as "^^var_start" can be used to more appealing plotting, where group labels are separated from their subsidiary labels. This can be especially useful for plotting interaction terms. See the Details section of `fixest::coefplot` for more information.

- `i.select` Integer scalar, default is 1. In `ggiplot`, used to select which variable created with `i()` to select. Only used when there are several variables created with `i`. See the Details section of `fixest::iplot` for more information.
- `main`, `xlab`, and `ylab` for setting the plot title, x- and y-axis labels, respectively.
- `zero` and `zero.par` for defining or adjusting the zero line. For example, `zero.par = list(col = 'orange')`.
- `ref.line` and `ref.line.par` for defining or adjusting the vertical reference line. For example, `ref.line.par = list(col = 'red', lty = 4)`.
- `pt.pch` and `pt.join` for overriding the default point estimate shapes and joining them, respectively.
- `col` for manually defining line, point, and ribbon colours.
- `ci_level` for changing the desired confidence level (default = 0.95). Note that multiple levels are allowed, e.g. `ci_level = c(0.8, 0.95)`.
- `ci.width` for changing the width of the extremities of the confidence intervals. Only used if `geom_style = "errorbar"` (or if multiple CI levels are requested for the default pointrange style). The default value is 0.2.
- `ci.fill.par` for changing the confidence interval fill. Only used when `geom_style = "ribbon"` and currently only affects the alpha (transparency) channel. For example, we can make the CI band lighter with `ci.fill.par = list(alpha = 0.2)` (the default alpha is 0.3).
- `dict` a dictionary for overriding coefficient names.
- `vcov`, `cluster` or `se` as alternative options for adjusting the standard errors of the model object(s) on the fly. See `summary.fixest` for details. Written here in superseding order; `cluster` will only be considered if `vcov` is not null, etc.

`aggr_eff` A keyword string or numeric sequence, indicating whether mean treatment effects for some subset of the model should be displayed as part of the plot. For example, the "post" keyword means that the mean post-treatment effect will be plotted alongside the individual period effects. Passed to `aggr_es`; see that function's documentation for other valid options.

`aggr_eff.par` List. Parameters of the aggregated treatment effect line, if plotted. The default values are `col = 'gray50'`, `lwd = 1`, `lty = 1`.

Details

These functions generally try to mimic the functionality and (where appropriate) arguments of `fixest::coefplot` and `fixest::iplot` as closely as possible. However, by leveraging the ggplot2 API and infrastructure, they are able to support some more complex plot arrangements out-of-the-box that would be more difficult to achieve using the base `coefplot`/`iplot` alternatives.

Value

A ggplot2 object.

Functions

- `ggiplot()`: This function plots the results of estimations (coefficients and confidence intervals). The function `ggiplot` restricts the output to variables created with i, either interactions with factors or raw factors.

See Also

`fixest::coefplot()`, `fixest::iplot()`.

Examples

```
library(ggfixest)

##
# Author note: The examples that follow deliberately follow the original
#   examples from the coefplot/iplot help pages. A few "gg-" specific
#   features are sprinkled within, with the final set of examples in
#   particular highlighting unique features of this package.


#
# Example 1: Basic use and stacking two sets of results on the same graph
#

# Estimation on Iris data with one fixed-effect (Species)
est = feols(Petal.Length ~ Petal.Width + Sepal.Length + Sepal.Width | Species, iris)

ggcoefplot(est)

# Show multiple CIs
ggcoefplot(est, ci_level = c(0.8, 0.95))

# By default, fixest model standard errors are clustered by the first fixed
# effect (here: Species).
# But we can easily switch to "regular" standard-errors
est_std = summary(est, se = "iid")

# You can plot both results at once in the same plot frame...
ggcoefplot(list("Clustered" = est, "IID" = est_std))
# ... or as separate facets
ggcoefplot(list("Clustered" = est, "IID" = est_std), multi_style = "facet") +
 theme(legend.position = "none")


#
# Example 2: Interactions
#


# Now we estimate and plot the "yearly" treatment effects

data(base_did)
```

```
base_inter = base_did

# We interact the variable 'period' with the variable 'treat'
est_did = feols(y ~ x1 + i(period, treat, 5) | id + period, base_inter)

# In the estimation, the variable treat is interacted
#  with each value of period but 5, set as a reference

# ggcoefplot will show all the coefficients:
ggcoefplot(est_did)


# Note that the grouping of the coefficients is due to 'group = "auto"'

# If you want to keep only the coefficients
# created with i() (ie the interactions), use ggiplot
ggiplot(est_did)

# We can see that the graph is different from before:
#  - only interactions are shown,
#  - the reference is present,
# => this is fully flexible

ggiplot(est_did, ci_level = c(0.8, 0.95))
ggiplot(est_did, ref.line = FALSE, pt.join = TRUE, geom_style = "errorbar")
ggiplot(est_did, geom_style = "ribbon", col = "orange")
# etc

# We can also use a dictionary to replace label values. The dicionary should
# take the form of a named vector or list, e.g. c("old_lab1" = "new_lab1", ...)

# Let's create a "month" variable
all_months = c("aug", "sept", "oct", "nov", "dec", "jan",
        "feb", "mar", "apr", "may", "jun", "jul")
# Turn into a dictionary by providing the old names
# Note the implication that treatment occured here in December (5 month in our series)
dict = all_months; names(dict) = 1:12
# Pass our new dictionary to our ggiplot call
ggiplot(est_did, pt.join = TRUE, geom_style = "errorbar", dict = dict)

#
# What if the interacted variable is not numeric?

# let's re-use our all_months vector from the previous example, but add it
# directly to the dataset
base_inter$period_month = all_months[base_inter$period]

# The new estimation
est = feols(y ~ x1 + i(period_month, treat, "oct") | id+period, base_inter)
# Since 'period_month' of type character, iplot/coefplot both sort it
ggiplot(est)

# To respect a plotting order, use a factor
```

```
base_inter$month_factor = factor(base_inter$period_month, levels = all_months)
est     = feols(y ~ x1 + i(month_factor, treat, "oct") | id + period, base_inter)
ggiplot(est)

# dict -> c("old_name" = "new_name")
dict = all_months; names(dict) = 1:12; dict
ggiplot(est_did, dict = dict)

#
# Example 3: Setting defaults
#

# The customization logic of ggcoefplot/ggiplot works differently than the
# original base fixest counterparts, so we don't have "gg" equivalents of
# setFixest_coefplot and setFixest_iplot. However, you can still invoke some
# global fixest settings like setFixest_dict(). SImple example:

base_inter$letter = letters[base_inter$period]
est_letters = feols(y ~ x1 + i(letter, treat, 'e') | id+letter, base_inter)

# Set global dictionary for capitalising the letters
dict = LETTERS[1:10]; names(dict) = letters[1:10]
setFixest_dict(dict)

ggiplot(est_letters)

setFixest_dict() # reset

#
# Example 4: group + cleaning
#

# You can use the argument group to group variables
# You can further use the special character "^^" to clean
#  the beginning of the coef. name: particularly useful for factors

est = feols(Petal.Length ~ Petal.Width + Sepal.Length +
       Sepal.Width + Species, iris)

# No grouping:
ggcoefplot(est)

# now we group by Sepal and Species
ggcoefplot(est, group = list(Sepal = "Sepal", Species = "Species"))

# now we group + clean the beginning of the names using the special character ^^
ggcoefplot(est, group = list(Sepal = "^^Sepal.", Species = "^^Species"))


#
# Example 5: Some more ggcoefplot/ggiplot extras
#
```

```
# We'll demonstrate using the staggered treatment example from the
# introductory fixest vignette.

data(base_stagg)
est_twfe = feols(
  y ~ x1 + i(time_to_treatment, treated, ref = c(-1, -1000)) | id + year,
  base_stagg
)
est_sa20 = feols(
  y ~ x1 + sunab(year_treated, year) | id + year,
  data = base_stagg
)

# Plot both regressions in a faceted plot
ggiplot(
  list('TWFE' = est_twfe, 'Sun & Abraham (2020)' = est_sa20),
  main = 'Staggered treatment', ref.line = -1, pt.join = TRUE
)

# So far that's no different than base iplot (automatic legend  aside). But an
# area where ggiplot shines is in complex multiple estimation cases, such as
# lists of fixest_multi objects. To illustrate, let's add a split variable
# (group) to our staggered dataset.
base_stagg_grp = base_stagg
base_stagg_grp$grp = ifelse(base_stagg_grp$id %% 2 == 0, 'Evens', 'Odds')

# Now re-run our two regressions from earlier, but splitting the sample to
# generate fixest_multi objects.
est_twfe_grp = feols(
  y ~ x1 + i(time_to_treatment, treated, ref = c(-1, -1000)) | id + year,
  data = base_stagg_grp, split = ~ grp
)
est_sa20_grp = feols(
  y ~ x1 + sunab(year_treated, year) | id + year,
  data = base_stagg_grp, split = ~ grp
)

# ggiplot combines the list of multi-estimation objects without a problem...
ggiplot(list('TWFE' = est_twfe_grp, 'Sun & Abraham (2020)' = est_sa20_grp),
    ref.line = -1, main = 'Staggered treatment: Split multi-sample')

# ... but is even better when we use facets instead of dodged errorbars.
# Let's use this an opportunity to construct a fancy plot that invokes some
# additional arguments and ggplot theming.
ggiplot(
  list('TWFE' = est_twfe_grp, 'Sun & Abraham (2020)' = est_sa20_grp),
  ref.line = -1,
  main = 'Staggered treatment: Split multi-sample',
  xlab = 'Time to treatment',
  multi_style = 'facet',
  geom_style = 'ribbon',
  facet_args = list(labeller = labeller(id = \(x) gsub(".*: ", "", x))),
  theme = theme_minimal() +
```

```
    theme(
      text = element_text(family = 'HersheySans'),
      plot.title = element_text(hjust = 0.5),
      legend.position = 'none'
    )
)

#
# Aside on theming and scale adjustments
#

# Setting the theme inside the `ggiplot()` call is optional and not strictly
# necessary, since the ggplot2 API allows programmatic updating of existing
# plots. E.g.
last_plot() +
 labs(caption = 'Note: Super fancy plot brought to you by ggiplot')
last_plot() +
 theme_grey() +
 theme(legend.position = 'none') +
 scale_fill_brewer(palette = 'Set1', aesthetics = c("colour", "fill"))
# etc.
```

`iplot_data` *Internal function for grabbing and preparing iplot data.*

Description

Grabs the underlying data used to construct `fixest::iplot`, with some added functionality and tweaks for the `ggiplot` equivalents.

Usage

```
iplot_data(
  object,
  .ci_level = 0.95,
  .keep = NULL,
  .drop = NULL,
  .dict = fixest::getFixest_dict(),
  .internal.only.i = TRUE,
  .i.select = 1,
  .aggr_es = NULL,
  .group = "auto",
  .vcov = NULL,
  .cluster = NULL,
  .se = NULL
)

coefplot_data(
```

```
  object,
  .ci_level = 0.95,
  .keep = NULL,
  .drop = NULL,
  .group = "auto",
  .dict = fixest::getFixest_dict(),
  .internal.only.i = FALSE,
  .i.select = 1,
  .aggr_es = "none",
  .vcov = NULL,
  .cluster = NULL,
  .se = NULL
)
```

Arguments

object	A model object of class fixest or fixest_multi, where the i() operator has been used to construct an interaction, or set of interactions.
.ci_level	A number between 0 and 1 indicating the desired confidence level, Defaults to 0.95.
.keep	Character vector used to subset the coefficients of interest. Passed down to fixest::iplot(..., keep = .keep) and should take the form of an acceptable regular expression.
.drop	Character vector used to subset the coefficients of interest (complement of .keep). Passed down to fixest::iplot(..., drop = .drop) and should take the form of an acceptable regular expression.
.dict	A dictionary (i.e. named character vector or a logical scalar). Used for changing coefficient names. Defaults to the values in getFixest_dict(). See the ?fixest::coefplot documentation for more information. Note: This argument applies dictionary changes directly to the return object for coefplot_data. However, it is ignored for iplot_data, since we want to preserve the numeric ordering for potential event study plots. (And imposing an ordered factor would create its own downstream problems in the case of continuous plot features like ribbons.) Instead, any dictionary replacement for ggiplot is deferred to the actual plot call and applied directly to the labels.
.internal.only.i	Logical variable used for some internal function handling when passing on to coefplot/iplot.
.i.select	Integer scalar, default is 1. In (gg)iplot, used to select which variable created with i() to select. Only used when there are several variables created with i. This is an index, just try increasing numbers to hopefully obtain what you want. Passed down to fixest::iplot(..., i.select = .i.select)
.aggr_es	A keyword string or numeric sequence indicating whether the aggregated mean treatment effects for some subset of the model should be added as a column to the returned data frame. Passed to aggr_es(..., aggregation = "mean").

`.group` A list, default is missing. Each element of the list reports the coefficients to be grouped while the name of the element is the group name. Passed down to `fixest::coefplot(..., group = .group)`. Example of valid uses:

- group=list(group_name="pattern")
- group=list(group_name=c("var_start", "var_end"))
- group=list(group_name=1:2)
- See the Details section of `?fixest::coefplot` for more.

`.vcov`, `.cluster`, `.se`
Alternative options for adjusting the standard errors of the model object on the fly. See `summary.fixest` for details (although note that the "." period prefix should be ignored in the latter's argument documentation). Written here in superseding order; `.cluster` will only be considered if `.vcov` is not null, etc.

Details

This function is a wrapper around `fixest::iplot(..., only.params = TRUE)`, but with various checks and tweaks to better facilitate plotting with `ggplot2` and handling of complex object types (e.g. lists of fixest_multi models)

Value

A data frame consisting of estimate values, confidence intervals, relative x-axis positions, and other aesthetic information needed to draw a ggplot2 object.

Functions

- `coefplot_data()`: Internal function for grabbing and preparing coefplot data

See Also

`fixest::iplot()`, `aggr_es()`.

Examples

```
library(fixest)

est_did = feols(y ~ x1 + i(period, treat, 5) | id+period,
                data = base_did)
iplot(est_did, only.params = TRUE) # The "base" version
iplot_data(est_did)                # The wrapper provided by this package

# Illustrative fixest_multi case, where the sample has been split by odd and
# even ID numbers.
est_split = feols(y ~ x1 + i(period, treat, 5) | id+period,
                  data = base_did, split = ~id%%2)
iplot(est_split, only.params = TRUE) # The "base" version
iplot_data(est_split)                # The wrapper provided by this package
```

Index