Package 'zfit'

January 17, 2022

Type Package

Title Fit Models in a Pipe

Version 0.3.0

Author Magnus Thor Torfason

Maintainer Magnus Thor Torfason <m@zulutime.net>

Description The goal of 'zfit' is to improve the usage of basic
model fitting functions within a piped work flow, in particular
when passing and processing a data.frame using 'dplyr' or
similar packages.

License MIT + file LICENSE

URL https://torfason.github.io/zfit/, https://github.com/torfason/zfit

Language en-US

Encoding UTF-8

RoxygenNote 7.1.2

Depends R (>= 3.5.0)

Suggests testthat (>= 3.0.0), dplyr, MASS, estimatr,

Imports

NeedsCompilation no

Repository CRAN

Date/Publication 2022-01-17 17:10:02 UTC

R topics documented:

---

zfit   *zfit: Fit Models in a Pipe*

Description

The goal of 'zfit' is to improve the usage of basic model fitting functions within a piped work flow, in particular when passing and processing a data.frame using 'dplyr' or similar packages.

Details

The goal of zfit is to make it easier to use a piped workflow with functions that don't have the "correct" order of parameters (the first parameter of the function does not match the object passing through the pipe). This issue is especially prevalent with model fitting functions, such as when passing and processing a data.frame (or tibble) before passing them to lm() or similar functions. The pipe passes the data object (data.frame/tibble) into the first parameter of the function, but the conventional estimation functions expect a formula to be the first parameter.

When using magrittr style pipes (%>%), this can be addressed by using special syntax, specifying data=. to pass the piped data into a parameter other than the first one. With R native pipes (|>), however, this is not possible and workaround are needed (such as constructing an anonymous function for each estimation or relying on complex rules about how piped arguments are interpreted in the presence of named parameters).

To address this, this package includes functions such as zlm() and zglm(). These are very similar to the core estimation functions such as lm() and glm(), but expect the first argument to be a (data.frame/tibble) rather than a formula (the formula becomes the second argument).

More importantly, the package includes two functions that make it trivial to construct a pipe-friendly version of any function. The zfitter() function takes any estimation function with the standard format of a formula and data parameter, and returns a version suitable for us in pipes (with the data parameter coming first). The zfitter() function also does some special handling to make make the call information more useful.

The zfunction() works for any function but omits the special handling for call parameters. Just pass the name of a function, and the name of the parameter that should receive the piped argument, and it returns a version of the function with that parameter coming first.

The package also includes the zprint() function, which is intended to simplify the printing of derived results, such as summary(), within the pipe, without affecting the modeling result itself. It also includes convenience functions for calling estimation functions using particular parameters, including zlogit() and zprobit(), and zpoisson(), to perform logistic or poisson regression within a pipe.

Note that some of the examples provided in the help and documentation use magrittr-style (%>%) pipe syntax, while others use the native pipe syntax (|>). The package has been tested with both types of pipe functionality and the results are identical, apart from the fact that %>% renames the piped argument to ., whereas the name of the piped argument is the complete nested function syntax of the pipe.

See Also

- [zlm](#) is the wrapper `lm`, probably the most common fitting function. The help file for this function includes several usage examples.

- [zglm](#) is a wrapper for `glm`, to fit generalized linear models.

- [zprint](#) is helpful for printing a `summary` of a model, but assigning the evaluated model to a variable

---

zfitter	*Create a pipe-friendly version of a given fitting function*

Description

This creates a pipe-friendly verion of a fitting function of the standard format — that is a function with a `formula` parameter followed by a `data` parameter.

Compared to just using zfunction(), this function includes some special handling to make the call information, which is usually reported by the summary() function more intuitive. Among other things, it shortens very long data names (longer than 32 characters by default), which otherwise are a nuisance when the data comes from the pipe, because the pipeline gets converted to a very long function call.

This function also stores the base name of the original fitting function, allowing one to use itsfull name, which is useful to just pull a single fitting function from a package without loading it.

Usage

```
zfitter(fun)
```

Arguments

fun   The fitting function to adapt. The name should not be quoted, rather, the actual function should be passed (prefixed with package if needed)

Examples

```
zlm_robust <- zfitter(estimatr::lm_robust)
zlm_robust(cars, speed~dist)

# The resulting function works well the native pipe ...
if ( getRversion() >= "4.1.0" ) {

  # Pipe cars dataset into zlm_robust for fitting
  cars |> zlm_robust( speed ~ dist )
}

# ... or with dplyr
if ( require("dplyr", warn.conflicts=FALSE) ) {
```

```
    # Pipe cars dataset into zlm_robust for fitting
    cars %>% zlm_robust( speed ~ dist )

    # Process iris with filter() before piping. Print a summary()
    # of the fitted model using zprint() before assigning the
    # model itself (not the summary) to m
    m <- iris %>%
      dplyr::filter(Species=="setosa") %>%
      zlm_robust(Sepal.Length ~ Sepal.Width + Petal.Width) %>%
      zprint(summary)
}
```

---

zfunction   *Create a pipe-friendly version of any function*

Description

`zfunction()` rearranges the arguments of any function moving the specified argument to the front of the list, so that this argument becomes the recipient of piping.

It returns a copy of the input function, that is completely identical except for the order of the arguments.

Usage

```
zfunction(fun, x)
```

Arguments

fun   The function to adapt. The name should not be quoted, rather, the actual function
             should be passed (prefixed with package if needed).

x   The name of the argument that should be moved to the front of the argument list
             . The name should not be quoted.

Examples

```
char_vector <- rownames(mtcars)

zgrep <- zfunction(grep, x)
grep("ll", char_vector, value=TRUE)
zgrep(char_vector, "ll", value=TRUE)
```

---

zglm	*Run a glm model in a pipe*

Description

These functions are wrappers for the [glm](#) function. The zglm function can be used to estimate any generalized linear model in a pipe. The zlogit, zprobit, and zpoisson functions can be used to estimate specific models. All of these functions rely on the glm function for the actual estimation, they simply pass the corresponding values to the family parameter of the glm function.

Usage of these functions is very similar to the [zlm](#) function (a wrapper for lm), for detailed examples, check out the entry for that function.

The zlogit function calls zglm, specifying family=binomial(link="logit").

The zprobit function calls zglm, specifying family=binomial(link="probit").

The zpoisson function calls zglm, specifying family="poisson".

Usage

```
zglm(
  data,
  formula,
  family = gaussian,
  weights,
  subset,
  na.action,
  start = NULL,
  etastart,
  mustart,
  offset,
  control = list(...),
  model = TRUE,
  method = "glm.fit",
  x = FALSE,
  y = TRUE,
  singular.ok = TRUE,
  contrasts = NULL,
  ...
)

zlogit(data, formula, ...)

zprobit(data, formula, ...)

zpoisson(data, formula, ...)
```

Arguments

data	A data.frame containing the model data.
formula	The formula to be fitted.
family	See the glm function.
weights	See the glm function.
subset	See the glm function.
na.action	See the glm function.
start	See the glm function.
etastart	See the glm function.
mustart	See the glm function.
offset	See the glm function.
control	See the glm function.
model	See the glm function.
method	See the glm function.
x	See the glm function.
y	See the glm function.
singular.ok	See the glm function.
contrasts	See the glm function.
...	Other arguments to be passed to the glm function.

Value

A fitted model.

See Also

- zlm is the wrapper for lm, probably the most common fitting function. The help file for zlm function includes several usage examples.

---

zlm	*Run an lm model in a pipe.*

Description

This function wraps around the lm function in order to make it more friendly to pipe syntax (with the data first).

Usage

```
zlm(
  data,
  formula,
  subset,
  weights,
  na.action,
  method = "qr",
  model = TRUE,
  x = FALSE,
  y = FALSE,
  qr = TRUE,
  singular.ok = TRUE,
  contrasts = NULL,
  offset,
  ...
)
```

Arguments

data	A data.frame containing the model data.
formula	The formula to be fitted.
subset	See the lm function.
weights	See the lm function.
na.action	See the lm function.
method	See the lm function.
model	See the lm function.
x	See the lm function.
y	See the lm function.
qr	See the lm function.
singular.ok	See the lm function.
contrasts	See the lm function.
offset	See the lm function.
...	Other arguments to be passed to the lm function.

Value

A fitted model.

See Also

- zglm is a wrapper for glm, to fit generalized linear models.

Examples

```
# Usage is possible without pipes
zlm( cars, dist ~ speed )

# zfit works well with dplyr
if ( require("dplyr", warn.conflicts=FALSE) ) {

  # Pipe cars dataset into zlm for fitting
  cars %>% zlm( speed ~ dist )

  # Process iris with filter before piping to zlm
  iris %>%
    filter(Species=="setosa") %>%
    zlm(Sepal.Length ~ Sepal.Width + Petal.Width)
}

# zfit also works well with the native pipe
if ( getRversion() >= "4.1.0" ) {

  # Pipe cars dataset into zlm for fitting
  cars |> zlm( speed ~ dist )

  # Extremely naive filtering function for piped usage
  filter_naive <- function(data, column, value) {
    data[data[,column]==value,]
  }

  # Process iris with filter() before piping. Print a summary()
  # of the fitted model using zprint() before assigning the
  # model itself (not the summary) to m
  m <- iris |>
    filter_naive("Species","setosa") |>
    zlm(Sepal.Length ~ Sepal.Width + Petal.Width) |>
    zprint(summary)
}
```

---

zprint   *Print the result of a function in a pipe but return original object*

Description

This function passes x to f and prints the result, but then returns the original x. It is useful in a pipe, when one wants a to print the derivative of an object in the pipe but then return or assign the original object. An example is printing the summary() of an estimated model but

Usage

```
zprint(x, f = NULL, ...)
```

Arguments

x	An object, typically in a pipe
f	A function to be applied to x before printing
...	Other arguments to be passed to f

Value

The original object x

Examples

```
if ( require("dplyr", warn.conflicts=FALSE) ) {

  # Print summary before assigning model to variable
  m <- lm( speed ~ dist, cars) %>%
    zprint(summary) # prints summary(x)
  m                 # m is the original model object

  # Print grouped data before filtering original
  cw_subset <- chickwts %>%
    zprint(count, feed, sort=TRUE) %>% # prints counts by feed
    filter(feed=="soybean")
  cw_subset # cw_subset is ungrouped, but filtered by feed
}
```

Index