Package 'mfp2'

November 2, 2026

Type Package

Title Multivariable Fractional Polynomial Models with Extensions

Version 1.1.0

Date 2026-11-02

Depends R (>= 4.2.0)

Imports stats, survival, nnet, utils, Rcpp

Suggests knitr, testthat (>= 3.2.0), rmarkdown, patchwork, fastglm (>= 0.1.1), MASS, numDeriv, rms (>= 7.0-0), ggplot2 (>= 3.4.0), geepack (>= 1.3.0)

Description Fits multivariable fractional polynomial models with selection of predictors and functional forms. The mfp2() function supports generalized linear models, including Gamma and inverse-Gaussian regression; negative-binomial, multinomial and ordinal regression; generalized estimating equations for clustered data; Cox and parametric survival models; and Fine-Gray competing-risks models. Extensions include approximate cumulative distribution transformations for sigmoid-shaped associations and spike-at-zero modelling for semi-continuous covariates. The mfpi() function assesses interactions between a categorical grouping variable and continuous, binary, or categorical predictors. Fractional polynomial modelling is described by Royston and Altman (1994) <doi:10.2307/2986270> and Royston and Sauerbrei (2008, ISBN:978-0-470-02842-1). The approximate cumulative distribution extension follows Royston (2014) <doi:10.1177/1536867X1401400206> and Royston and Sauerbrei (2016) <doi:10.1177/1536867X1601600109>. Spike-at-zero modelling follows Becher et al. (2012) <doi:10.1002/bimj.201100263>. Interaction analyses follow Royston and Sauerbrei (2004) <doi:10.1002/sim.1815>, Royston and Sauerbrei (2013) <doi:10.1002/sim.5813>, and Royston and Sauerbrei (2014) <doi:10.1002/sim.6308>.

License GPL-3

URL https://github.com/EdwinKipruto/mfp2

BugReports https://github.com/EdwinKipruto/mfp2/issues

Encoding UTF-8

LazyData true

NeedsCompilation yes

VignetteBuilder knitr

LinkingTo Rcpp

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Author Edwin Kipruto [aut, cre],
Michael Kammer [aut],
Patrick Royston [aut],
Willi Sauerbrei [aut]

Maintainer Edwin Kipruto <kiprutoedwin2626@gmail.com>

Repository CRAN

Date/Publication 2026-11-02 14:10:16 UTC

Contents

mfp2-package . . . 3
advanced_prostate_cancer . . . 5
art . . . 6
coef.mfp2 . . . 6
coef.mfpi . . . 7
confint.mfp2 . . . 9
fp . . . 10
fracplot . . . 13
gbsg . . . 15
gee_family . . . 16
get_selected_variable_names . . . 18
mfp2 . . . 19
mfpi . . . 37
multinomial_family . . . 48
negbin_family . . . 49
ordinal_family . . . 51
pima . . . 52
plot.mfp2 . . . 53
plot.mfpi . . . 60
predict.mfp2 . . . 64
predict.mfpi . . . 82
print.mfp2 . . . 90
print.mfpi . . . 91
print.mfpi_coef . . . 92
print.mfpi_prediction . . . 93
print.mfpi_prediction_list . . . 94
print.mfpi_vcov_list . . . 94

print.summary.mfp2 95
print.summary.mfpi 95
prostate 96
residuals.mfp2 97
summary.mfp2 98
summary.mfpi 101
survreg_family 103
transform_vector_fp 105
vcov.mfpi 108

Index **110**

`mfp2-package` *mfp2: Multivariable Fractional Polynomial Models and Extensions*

Description

The `mfp2` package fits multivariable fractional polynomial models with simultaneous selection of variables and functional forms for continuous predictors.

Main functions

- `mfp2()` fits multivariable fractional polynomial models.
- `fp()` specifies fractional-polynomial and related modelling options within a formula.
- `mfpi()` investigates interactions between a categorical grouping variable and continuous, binary, or categorical variables.

Fitted models can be examined using `print()`, `summary()`, and `coef()`, used for prediction with `predict()`, and visualised with `plot()`.

Supported models

`mfp2()` and `mfpi()` support every likelihood-based family supplied by base `stats::glm()`: Gaussian, binomial, Poisson, Gamma, and inverse-Gaussian regression, including the links accepted by their family objects. Quasi families are not supported because MFP selection requires a likelihood. Optional negative-binomial fitting requires the `fastglm` package. Multinomial logistic models use `multinomial_family()` and the `nnet` package. Their selected FP transformations are common across logits and their regression coefficients are logit specific. Proportional-odds ordinal models use `ordinal_family()` and the optional `rms` package. They estimate a common predictor effect across response cut-points.

Survival models are specified as separate families: `family = "cox"` for Cox proportional hazards and `survreg_family()` for the parametric distributions implemented by `survival::survreg()` are supported by both `mfp2()` and `mfpi()`. `finegray_family()` provides Fine–Gray proportional subdistribution hazards for `mfp2()` only.

Modelling extensions

In addition to standard fractional-polynomial modelling, `mfp2` provides:

- approximate cumulative distribution (ACD) transformations for selected continuous predictors;
- spike-at-zero (SAZ) modelling for semi-continuous predictors that contain a distinct zero component and a positive continuous component;
- interaction analysis between continuous, binary, or categorical predictors and categorical groups through `mfpi()`.

Spike-at-zero modelling can be requested in the formula interface with `fp(x, spike = TRUE)` or in the matrix interface with `spike_vars = "x"`. The procedure assesses whether the final model should retain both the zero-component indicator and the positive continuous component, or only one of them. Covariates requested through `zero`, `catzero`, or `spike` must be nonnegative: `x = 0` defines the zero component and `x > 0` defines the positive component. Negative values are rejected and must be explicitly recoded if they should represent the zero group.

Getting started

See `vignette("mfp2", package = "mfp2")` for an introduction to model fitting, interpretation, prediction, and plotting.

Further guidance is available in:

- `vignette("mfp-methodology", package = "mfp2")` for MFP methodology;
- `vignette("acd-transformations", package = "mfp2")` for ACD transformations;
- `vignette("spike-at-zero", package = "mfp2")` for spike-at-zero;
- `vignette("interactions", package = "mfp2")` for interaction analysis.

Author(s)

Maintainer: Edwin Kipruto <kiprutoedwin2626@gmail.com>

Authors:

- Edwin Kipruto <kiprutoedwin2626@gmail.com>
- Michael Kammer <michael.kammer@meduniwien.ac.at>
- Patrick Royston <j.royston@ucl.ac.uk>
- Willi Sauerbrei <wilhelm.sauerbrei@uniklinik-freiburg.de>

See Also

`mfp2()`, `fp()`, `mfpi()`, `multinomial_family()`, `ordinal_family()`, `survreg_family()`, `finegray_family()`, `predict.mfp2()`, `plot.mfp2()`

advanced_prostate_cancer *Advanced prostate cancer dataset used in Royston and Sauerbrei (2008) book.*

Description

Advanced prostate cancer dataset used in Royston and Sauerbrei (2008) book.

Usage

```
data(advanced_prostate_cancer)
```

Format

A dataset with 475 observations (338 deaths) and 17 variables.

patnr Patient number.

age Age at diagnosis in years (continuous).

wt Standardized weight (continuous).

sbp Systolic blood pressure (continuous).

dbp Diastolic blood pressure (continuous).

sz Size of primary tumour (continuous).

ap Serum acid phosphatase (continuous).

hg Haemoglobin (g/100 ml) (continuous).

sg Gleason stage-grade category (continuous).

pf Performance status (binary).

hx History of cardiovascular disease (binary).

bm Presence of bone metastases (binary).

stage Stage 4 vs stage 3 (binary).

ekg Abnormal electrocardiogram (binary).

rx Treatment group (binary).

survtime Time to death (overall survival).

cens Censoring (0 = censored, 1 = event).

Examples

```
data(advanced_prostate_cancer)
str(advanced_prostate_cancer)
```

art *Artificial dataset with continuous response*

Description

The ART data set mimics the GBSG breast cancer study in terms of the distribution of predictors and correlation structure.

Usage

```
data(art)
```

Format

The dataset has 250 observations and 11 variables (1 response and 10 covariates)

y Continuous response variable.

x1, x3, x5-x7, x10 Continuous covariates.

x2 Binary variable.

x4 Ordinal variable with 3 levels.

x8 Binary variable.

x9 Nominal variable with 3 levels.

Examples

```
data(art)
str(art)
```

coef.mfp2 *Extract Coefficients from an* mfp2 *Model*

Description

Returns coefficients from a fitted `mfp2()` model. This is a method for the generic `stats::coef()` function.

Usage

```
## S3 method for class 'mfp2'
coef(object, ...)
```

Arguments

`object`	A fitted `mfp2()` object.
`...`	Not used.

Value

A named numeric vector for scalar-response models. For multinomial models, a matrix with one row per non-reference outcome and one column per fitted term; powers are common across rows but coefficients are outcome specific.

See Also

`mfp2()`, `summary.mfp2()`, `predict.mfp2()`

Examples

```
data("prostate")
fit <- mfp2(
  lpsa ~ fp(age) + fp(cavol) + fp(weight) + svi,
  data = prostate,
  verbose = FALSE
)
coef(fit)
```

`coef.mfpi` *Extract Coefficients from MFPI Interaction Models*

Description

Extracts regression coefficients from the term-specific interaction models stored in an `mfpi()` fit. Because MFPI evaluates each continuous variable in a separate interaction model, the method returns one coefficient vector when a single interaction is requested and a named collection when several interactions are requested.

Usage

```
## S3 method for class 'mfpi'
coef(object, term = NULL, model = c("best", "all"), ...)
```

Arguments

`object`	A fitted `mfpi()` object.
`term`	Optional character vector naming continuous variables whose term-specific interaction models should be extracted. When `NULL`, all available terms in the requested `model` scope are returned.
`model`	Character scalar indicating which stored interaction models are eligible. `"best"` (the default) uses interactions retained by the MFPI selection criterion; `"all"` uses every evaluated term-specific winner, whether retained or not.
`...`	Not used. Supplying additional arguments produces an error.

Details

`mfpi()` does not fit all selected interactions together in one joint final model. Each variable in `interaction_vars` has its own term-specific interaction model. Consequently, `coef.mfpi()` reports coefficients from those stored interaction models rather than from the first-stage MFP adjustment model.

By default, only interactions retained by the MFPI selection criterion are available (`model = "best"`). Set `model = "all"` to extract the fitted interaction model for every evaluated variable, including interactions that were not retained. Use `term` to select one or more continuous variables.

The printed representation is designed for interpretation. For each interaction it reports the interaction variable, grouping variable, FLEX setting, and requested interaction form. The interaction-model FP powers are then shown separately for every group level, irrespective of FLEX. This gives one consistent display for FLEX1–FLEX4 and makes group-specific power choices under FLEX4 explicit.

The coefficients belonging to the group-specific continuous-variable functions are displayed in a table with columns for the original group level, the FP transformation, and the coefficient estimate. Coefficients for the grouping variable are shown in a separate table together with the fitted reference level. The table reports the coefficient attached to each non-reference factor level; it is deliberately not labelled as a direct comparison with the reference group because, in the presence of an interaction, that coefficient is only one component of the model parameterization and is not an overall group effect. For a grouping variable with K levels, there are K group-specific FP blocks and K - 1 group-variable coefficients. Adjustment coefficients are displayed separately, using stored MFP metadata to recover readable variable or transformation labels where available. A model intercept, when present, is also shown separately.

Transformation labels identify the FP basis before centering. When the MFPI fit uses `center = TRUE`, the corresponding fitted design columns are centered using the stored MFPI centering constants; the coefficient estimates are therefore coefficients of those centered design columns.

The numeric values remain ordinary regression coefficients. For a single requested interaction, the returned object is a numeric vector with an MFPI print class and can be used in numerical calculations. If several interactions are requested, a named list of such vectors is returned. Multinomial results contain the slope coefficients for each non-reference logit; their class-specific intercepts are not included.

The transformation labels reflect the interaction model actually fitted. Thus, under `flex4`, different group levels can display different FP transformations. Repeated FP powers use the standard repeated-power form; for example, powers `c(-2, -2)` are displayed as `x^-2` and `x^-2 * log(x)` (with any fitted shift included in `x`).

Value

If one interaction model is requested, a named numeric vector of regression coefficients with class `"mfpi_coef"`. If several interaction models are requested, a named list with class `"mfpi_coef_list"`, with one coefficient vector per interaction. The printed labels are user-facing; the original fitted coefficient names are retained in the `raw_names` attribute of each coefficient vector.

See Also

`mfpi()`, `vcov.mfpi()`, `summary.mfpi()`, `predict.mfpi()`

Examples

```
data("prostate", package = "mfp2")

fit <- mfpi(
  lpsa ~ fp(age) + svi + fp(cavol),
  data = prostate,
  group_var = "svi",
  interaction_vars = "cavol",
  interaction_forms = c(cavol = "fp2"),
  flex = "flex4",
  p_interact = 1,
  verbose = FALSE
)

# User-facing coefficient display for the selected interaction model.
coef(fit)

# Extract one term explicitly.
b <- coef(fit, term = "cavol")
is.numeric(b)

# Inspect an evaluated interaction even when it was not retained.
coef(fit, term = "cavol", model = "all")
```

---

confint.mfp2 *Confidence Intervals for* mfp2 *Model Parameters*

Description

Wald confidence intervals for the coefficients of a fitted `mfp2()` model, a method for the generic `stats::confint()`.

Usage

```
## S3 method for class 'mfp2'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	A fitted `mfp2()` object.
`parm`	Optional character vector of parameter names, or numeric indices, selecting which coefficients to report. Defaults to all.
`level`	The confidence level. Default `0.95`.
`...`	Passed to the underlying interval method for multinomial/ordinal models.

Details

The intervals are $\hat{\beta} \pm z_{1-\alpha/2}\,\mathrm{SE}(\hat{\beta})$, with standard errors taken from `stats::vcov()` of the fitted object. These are the same Wald intervals summarised by `summary.mfp2()` (which displays its 95% interval with the conventional rounded multiplier 1.96, whereas `confint()` uses the exact normal quantile and honours `level`), and they apply uniformly to every scalar-coefficient family. They are used in preference to the default profile-likelihood intervals of `stats::confint()` for `glm` fits because those refit the model on `mfp2`'s internally transformed design and fail; for GEE (`gee_family()`) the covariance is the robust sandwich estimator, which is the correct basis for GEE inference (and `geepack` supplies no `confint` method). Multinomial and ordinal models, whose coefficients are not a single named vector, defer to the interval method of the underlying fit.

Value

A matrix with a row per selected parameter and two columns giving the lower and upper confidence limits.

See Also

`mfp2()`, `summary.mfp2()`, `vcov()`

Examples

```
data("prostate")
fit <- mfp2(
  lpsa ~ fp(age) + fp(cavol) + fp(weight) + svi,
  data = prostate,
  verbose = FALSE
)
confint(fit)
confint(fit, parm = "svi.1", level = 0.9)
```

---

fp *Specify a Fractional-Polynomial Term*

Description

Use `fp()` or `fp2()` inside an `mfp2()` or `mfpi()` formula to set fractional-polynomial (FP) options for one continuous predictor. In `mfp2()`, continuous terms without `fp()` are also considered for FP selection, using its top-level settings. `fp()` and `fp2()` specify the same model term and record options for fitting; neither transforms the values when the formula is created.

Usage

```
fp(
  x,
  df = 4,
  alpha = 0.05,
```

```
  select = 0.05,
  shift = NULL,
  scale = NULL,
  center = TRUE,
  acd = FALSE,
  powers = NULL,
  zero = FALSE,
  catzero = FALSE,
  spike = FALSE,
  force_max_fp = FALSE,
  acdx = NULL
)

fp2(...)
```

Arguments

x	Continuous numeric predictor to include in the formula. Do not wrap a factor in `fp()`.
df	Maximum FP complexity: `1` for linear only, `2` for up to FP1, or `4` (default) for up to FP2. Higher positive even values permit higher FP degrees.
alpha	Significance level for comparisons between functional forms under p-value selection. Default `0.05`. For eligible spike-at-zero terms, also controls tests that remove components.
select	Significance level for variable inclusion under p-value selection. Default `0.05`; use `1` to force inclusion. It does not control inclusion under AIC or BIC selection.
shift	Numeric value added before FP transformation. `NULL` (default) inherits the setting from `mfp2()`; `0` disables shifting for this term.
scale	Positive divisor applied after shifting. `NULL` (default) inherits the setting from `mfp2()`; `1` disables scaling for this term. With `zero`, `catzero`, or `spike`, the automatic scale is `1`.
center	Whether to center the fitted terms. Default `TRUE`.
acd	Whether to request approximate cumulative distribution (ACD) modelling. Default `FALSE`.
powers	Numeric vector of candidate FP powers for this predictor. `NULL` (default) uses the global candidate set. Duplicate candidates are removed.
zero	If `TRUE`, transform only values with `x > 0`; exact zeros contribute zero to the continuous component. Requires nonnegative `x`. Default `FALSE`.
catzero	If `TRUE`, add a binary indicator for `x = 0` alongside the positive component. Implies `zero = TRUE`. Default `FALSE`.
spike	If `TRUE`, request spike-at-zero (SAZ) model selection, subject to eligibility checks. While eligible, it considers a positive component and a binary indicator for `x = 0`. Requires nonnegative `x`. Default `FALSE`.

force_max_fp	If TRUE, retain the most complex form permitted by df and bypass variable removal and functional-form simplification. For an eligible SAZ term, retain both components. Default FALSE.
acdx	Deprecated alias for acd. Use acd; supplying both arguments is an error.
...	Arguments passed by fp2() to fp().

Details

fp2() was introduced to avoid a name conflict with mfp::fp(). That conflict is now resolved: formulas passed to mfp2() or mfpi() use this package's fp() even when mfp is attached.

Add categorical predictors to the formula without fp() or fp2(). Convert numeric category codes with factor().

The default df = 4 permits selection of a linear effect, a first-degree FP (FP1), or a second-degree FP (FP2). Selection may also exclude the predictor. Use df = 2 to permit at most FP1, or df = 1 for a linear effect only. The fitting procedure may reduce the permitted complexity when a predictor has few distinct values.

Candidate powers default to c(-2, -1, -0.5, 0, 0.5, 1, 2, 3). A power of zero represents a logarithm. powers changes the candidates searched; it does not specify the powers of the final fitted function.

Options set inside fp() apply to this term. df, alpha, select, and center have their own defaults in fp() and ignore the values supplied to the surrounding mfp2() call; set them inside fp() to change them. shift, scale, and powers default to NULL and use the top-level values unless set here. Under p-value selection, select = 1 forces inclusion and alpha = 1 prevents simplification of the permitted functional form.

Value

The input vector with modelling options attached as attributes. The options take effect when the term is used in an mfp2() or mfpi() formula.

See Also

fp2(), mfp2(), mfpi()

Examples

```
data("prostate")

# Search up to FP1 for age.
fit_fp <- mfp2(
  lpsa ~ fp(age, df = 2) + svi,
  data = prostate,
  verbose = FALSE
)

# fp2() accepts the same options and produces the same model specification.
fit_fp2 <- mfp2(
  lpsa ~ fp2(age, df = 2) + svi,
  data = prostate,
```

```
  verbose = FALSE
)
```

`fracplot` *Deprecated Fractional-Polynomial Plotting Interface*

Description

`fracplot()` is retained temporarily for compatibility with code written for earlier versions of `mfp2`. Use `plot.mfp2()` or `plot()` instead.

Usage

```
fracplot(
  model,
  terms = NULL,
  partial_only = FALSE,
  type = c("terms", "contrasts"),
  ref = NULL,
  terms_seq = c("data", "equidistant"),
  alpha = 0.05,
  show_titles = TRUE,
  show_grid = FALSE,
  color_points = "#AAAAAA",
  color_line = "red",
  color_line_spike = "red",
  color_fill = "#000000",
  shape = 16,
  size_points = 1,
  size_points_spike = 2,
  linetype = "solid",
  linewidth = 1,
  alpha_fill = 0.1
)
```

Arguments

`model`	A fitted object of class `"mfp2"`.
`terms`	Names of the variables to plot, for example, `c("age", "cavol")`. By default (`NULL`), requests all variables kept in the final model. See **Choosing terms**.
`partial_only`	Whether to show only the fitted effect and confidence interval. Defaults to `FALSE`. See **Component-plus-residual plots**.
`type`	Either `"terms"` (default) or `"contrasts"`. See **Choosing the plot type**.
`ref`	Named list of reference values for contrast plots, for example, `list(age = 50)`. See **Contrasts and reference values**.

`terms_seq` Either `"data"` or `"equidistant"`. See **Evaluation values** for their meaning and defaults.

`alpha` Significance level for confidence intervals. The default, `0.05`, gives 95 percent intervals. See **Confidence intervals**.

`show_titles` Whether to show titles describing the fitted transformation or variable representation. Defaults to `TRUE`.

`show_grid` Whether to show the major and minor panel grid lines of `ggplot2::theme_bw()`. Defaults to `FALSE`, which draws plots without grid lines.

`color_points` Colour of component-plus-residual points, for example, `"grey60"` or `"#AAAAAA"`. All colour arguments accept ggplot2 colour names or hexadecimal codes. See ggplot2::aes_colour_fill_alpha.

`color_line` Colour of fitted curves and fitted points for binary predictors and factors, for example, `"black"` or `"blue"`. Also used for the zero endpoint of a term fitted with `zero = TRUE` alone.

`color_line_spike` Colour of the separate zero endpoint for a `catzero` or spike-at-zero term with a continuous component, for example, `"red"`.

`color_fill` Fill colour of confidence bands, for example, `"lightblue"` or `"#56B4E9"`.

`shape` Shape of component-plus-residual points. Common ggplot2 shape codes include `1` for open circles and `16` for solid circles. The default is `16`. See `ggplot2::scale_shape()`.

`size_points` Numeric size of component-plus-residual points, in millimetres, for example, `1` or `2`. The default is `1`. See `ggplot2::geom_point()`.

`size_points_spike` Numeric size of fitted points for binary predictors, factors, and zero endpoints. For example, use `3` for larger points. The default is `2`. See `ggplot2::geom_point()`.

`linetype` Line style of fitted curves, for example, `"solid"`, `"dashed"`, or `"dotted"`. The default is `"solid"`. See `ggplot2::scale_linetype()`.

`linewidth` Numeric width of fitted curves, for example, `0.5` or `1`. Larger values produce thicker lines. The default is `1`. See `ggplot2::geom_line()`.

`alpha_fill` Confidence-band transparency, from `0` (transparent) to `1` (opaque).

Details

The function forwards its arguments to `plot.mfp2()` and returns the same plot objects, except that its own defaults for `color_line` and `color_line_spike` are `"red"`, whereas `plot.mfp2()` uses `"black"`. The sections referred to in the argument descriptions, such as **Choosing terms**, are on the help page of `plot.mfp2()`.

Value

A named list of ggplot2 objects, one for each plotted term, identical to the result returned by `plot.mfp2()` called with the same arguments (including the colours).

See Also

`plot.mfp2()`

Examples

```
data("prostate")

fit <- mfp2(
  lpsa ~ fp(age) + fp(cavol) + fp(weight) + svi,
  data = prostate,
  verbose = FALSE
)

# Plotting requires ggplot2.
if (requireNamespace("ggplot2", quietly = TRUE)) {
  # Recommended interface
  plots <- plot(fit, terms = "cavol")
  print(plots)
}


if (requireNamespace("ggplot2", quietly = TRUE)) {
  # Deprecated compatibility interface
  old_plots <- fracplot(fit, terms = "cavol")
}
```

---

gbsg *Breast cancer dataset used in Royston and Sauerbrei (2008) book.*

Description

Breast cancer dataset used in Royston and Sauerbrei (2008) book.

Usage

```
data(gbsg)
```

Format

A dataset with 686 observations and 12 variables.

id Patient identifier.

age Age in years.

meno Menopausal status (0 = premeno, 1 = postmeno).

size Tumor size (mm).

grade Tumor grade.

nodes Number of positive lymph nodes.

enodes exp(-0.12*nodes).

pgr Progesterone receptor status.

er Estrogen receptor status.

hormon Tamoxifen treatment.

rectime Time (days) to death or cancer recurrence.

censrec Censoring (0 = censored, 1 = event).

Examples

```
data(gbsg)
str(gbsg)
```

`gee_family` *GEE Family Specification*

Description

Use `gee_family()` with `mfp2()` to model outcomes measured more than once within independent groups, such as repeated measurements from the same person. Supply the group identifier through `mfp2()`'s `id` argument. If measurement order matters, supply `waves`.

Usage

```
gee_family(
  family = stats::gaussian(),
  corstr = c("exchangeable", "independence", "ar1"),
  std.err = c("san.se", "jack", "j1s", "fij"),
  reference_dispersion = NULL,
  qbic_penalty = c("clusters", "observations")
)
```

Arguments

`family`	Response family: a family object, function, or name for `gaussian`, `binomial`, `poisson`, or `Gamma`, with any link the family supports, for example `binomial(link = "probit")`. The default is `stats::gaussian()`.
`corstr`	Assumed pattern of association between measurements in the same group: `"exchangeable"` (default), `"independence"`, or `"ar1"`.
`std.err`	Method for calculating standard errors: `"san.se"` (the default), `"jack"`, `"j1s"`, or `"fij"`. `"san.se"` gives robust sandwich standard errors with the finite-cluster correction $K/(K-1)$, where K is the number of clusters, as in Stata's `xtgee`; they are therefore slightly larger than those reported by `geepack::geeglm()`. The jackknife estimators are used unchanged.
`reference_dispersion`	Optional positive value that scales the quasi-likelihood part of GEE AIC and BIC scores. The score divides that part by this value; the penalty for model complexity is not divided by it. If `NULL` (the default), `mfp2()` estimates the value. It does not set the scale used for the final model's standard errors. It cannot be supplied with p-value selection.

qbic_penalty — For BIC selection, whether the complexity penalty uses the number of independent "clusters" (default) or the number of "observations". This setting does not affect AIC or p-value selection.

Details

All three selection criteria in mfp2() are available for GEE models. The default is criterion = "aic", which uses QICu. BIC selection uses a BIC-like score based on quasi-likelihood, and p-value selection uses approximate tests for GEE. Under AIC or BIC selection, select = 1 does not force a predictor into the model; use the keep argument of mfp2() instead.

Value

A family specification to supply as the family argument of mfp2().

References

Pan, W. (2001). Akaike's information criterion in generalized estimating equations. *Biometrics*, 57, 120–125.
Halekoh, U., Højsgaard, S. and Yan, J. (2006). The R package geepack for generalized estimating equations. *Journal of Statistical Software*, 15(2), 1–11.

See Also

mfp2(), geepack::geeglm()

Examples

```
set.seed(1)
d <- data.frame(id = rep(1:40, each = 3), x = runif(120, 1, 5))
d$y <- log(d$x) + rnorm(40, sd = 0.5)[d$id] + rnorm(120, sd = 0.3)

# GEE models require geepack.
if (requireNamespace("geepack", quietly = TRUE)) {
  fit_gee <- mfp2(
    y ~ fp(x, df = 2),
    data = d,
    family = gee_family(gaussian(), corstr = "exchangeable"),
    id = d$id,
    verbose = FALSE
  )
  coef(fit_gee)
}
```

`get_selected_variable_names` *Extract Names of Selected Variables from an* `mfp2` *Model*

Description

Returns the names of predictors retained in the final model after MFP selection, as recorded in the `selected` column of `object$fp_terms`. Usually a selected variable has at least one fractional-polynomial power that is not `NA`. The exception is a spike-at-zero term for which only the binary indicator for `x = 0` is retained: it is selected although all of its powers are `NA`. Factors are reported by their variable name, not by their contrast columns.

Usage

```
get_selected_variable_names(object)
```

Arguments

`object`	A fitted `mfp2()` object.

Value

Character vector of selected predictor names, ordered according to the `xorder` used in the `mfp2()` call.

See Also

`mfp2()`, `print.mfp2()`

Examples

```
# Gaussian model
data("prostate")
x <- as.matrix(prostate[, 2:8])
y <- as.numeric(prostate$lpsa)
fit <- mfp2(x, y, verbose = FALSE)
get_selected_variable_names(fit)
```

`mfp2` *Multivariable Fractional Polynomial Models with Extensions*

Description

`mfp2()` selects predictors for a regression model and chooses a linear or fractional-polynomial form for each selected continuous predictor. It also supports approximate cumulative distribution (ACD) transformations for sigmoid-shaped relationships and spike-at-zero modelling for predictors with many exact zeros. Models can be specified with a formula and data frame or with a numeric predictor matrix and an outcome. Both interfaces use the same selection procedure. The formula interface is the preferred choice for most users because it handles categorical predictors automatically; the matrix interface requires them to be coded as numeric columns. Supported regression models include Gaussian, binomial, Poisson, Gamma, inverse Gaussian, negative binomial, multinomial, and ordinal models. For time-to-event outcomes, it supports Cox, parametric survival, and Fine–Gray competing-risks models. It also supports generalized estimating equations (GEE) for clustered data.

Usage

```
mfp2(x, ...)

## Default S3 method:
mfp2(
  x,
  y,
  weights = NULL,
  offset = NULL,
  cycles = 5,
  scale = NULL,
  shift = NULL,
  df = 4,
  center = TRUE,
  subset = NULL,
  family = gaussian(),
  criterion = NULL,
  select = 0.05,
  alpha = 0.05,
  keep = NULL,
  xorder = c("ascending", "descending", "original"),
  powers = NULL,
  ties = c("efron", "breslow"),
  strata = NULL,
  nocenter = c(-1, 0, 1),
  acd_vars = NULL,
  ftest = FALSE,
  control = NULL,
  zero_vars = NULL,
  catzero_vars = NULL,
```

```
  spike_vars = NULL,
  min_saz_prop = 0.1,
  force_max_fp_vars = NULL,
  term_groups = NULL,
  verbose = TRUE,
  fitter = c("base", "fastglm"),
  id = NULL,
  waves = NULL,
  acdx = NULL,
  ...
)

## S3 method for class 'formula'
mfp2(
  formula,
  data,
  weights = NULL,
  offset = NULL,
  cycles = 5,
  scale = NULL,
  shift = NULL,
  df = 4,
  center = TRUE,
  subset = NULL,
  family = gaussian(),
  criterion = NULL,
  select = 0.05,
  alpha = 0.05,
  keep = NULL,
  xorder = c("ascending", "descending", "original"),
  powers = NULL,
  ties = c("efron", "breslow"),
  strata = NULL,
  nocenter = c(-1, 0, 1),
  ftest = FALSE,
  control = NULL,
  min_saz_prop = 0.1,
  verbose = TRUE,
  fitter = c("base", "fastglm"),
  id = NULL,
  waves = NULL,
  ...
)
```

Arguments

x	For `mfp2.default()`, a numeric matrix with one row per observation and one column per predictor. Column names are required.

`...` Used to pass arguments from `mfp2()` to its fitting method. Additional arguments supplied to the fitting method are not used; a warning names them.

`y` Outcome for `mfp2.default()`. In the formula interface, put the outcome on the left side of the formula. Its required format depends on `family`; see **Model families and responses**.

`weights` Optional numeric observation weights. All weights must be strictly positive. In the formula interface, an expression is evaluated first in `data` and then in the formula environment.

`offset` A known term added to the model, such as the log of exposure time. With formula input you can write `offset()` in the formula instead. When predicting for new data, supply the matching offset.

`cycles` Maximum number of MFP backfitting cycles. Default `5`.

`scale` Divisor used when preparing predictors for FP selection. `NULL` chooses scales automatically; `1` turns off automatic scaling. In `mfp2.default()`, use a named value such as `c(age = 10)` to set one column's scale. In formulas, a single unnamed value applies to every term that does not set its own; use `fp()` or `fp2()` for one term. Binary and ACD predictors always use scale `1`. Predictors with exact-zero handling (`zero`, `catzero`, or `spike`) use an automatic scale of `1`; an explicitly supplied scale is still used. See **Shifting, scaling, and centering**.

`shift` Value added to predictors before FP transformation. `NULL` chooses shifts automatically; `0` turns off automatic shifting. In `mfp2.default()`, use a named value such as `c(age = 20)` to set one column's shift. In formulas, a single unnamed value applies to every term that does not set its own; use `fp()` or `fp2()` for one term. Binary and linear predictors, and those using exact-zero handling, always have shift `0`. See **Shifting, scaling, and centering**.

`df` Maximum FP complexity: `1` allows linear only, `2` allows up to FP1, and `4` allows up to FP2. The default is `4`. Selection may choose a simpler form or omit the predictor. In `mfp2.default()`, use a named value such as `c(age = 2)` for one column. In `mfp2.formula()`, the top-level value applies only to terms outside `fp()` or `fp2()`; those terms default to `df = 4` unless set explicitly, for example `fp(age, df = 2)`. Predictors with few distinct values may be limited to a simpler form. See **Fractional-polynomial model selection**.

`center` If `TRUE` (default), center predictors after any FP or ACD transformation; `FALSE` leaves them uncentered. Continuous terms use their mean and binary terms use their lower observed value in the data. Centering does not change fitted values. In `mfp2.default()`, use one logical value for all predictors or a named vector for selected columns; unspecified columns use `TRUE`. In `mfp2.formula()`, the top-level value applies only to terms outside `fp()` or `fp2()`; set `center` inside the term to change it for an `fp()` term.

`subset` Optional observations used for model selection and fitting. Formula calls accept an expression evaluated in `data` and the formula environment. Matrix calls accept a logical vector or unique numeric/integer row positions. Automatic shift and scale values are estimated before the subset is applied. See the Subsetting section.

`family` Model family; default `"gaussian"`. Supported GLM families are `"gaussian"`, `"binomial"`, `"poisson"`, `"Gamma"`, and `"inverse.gaussian"`, supplied as a

name, function, or family object. Quasi families are not supported. Use `negbin_family()` for negative-binomial models, choosing a `"log"` (default), `"sqrt"`, or `"identity"` link; `"negbin"` is shorthand for `negbin_family(link = "log")`. Use `"multinomial"` or `multinomial_family()` for multinomial models; `"ordinal"` or `ordinal_family()` for cumulative-link ordinal models; `"cox"` for Cox models; `survreg_family()` for parametric survival models; `finegray_family()` for Fine–Gray models; or `gee_family()` for generalized estimating equations. GEE supports Gaussian, binomial, Poisson, and Gamma response families, but not negative-binomial responses.

`criterion` Model-selection criterion: `"pvalue"`, `"aic"`, or `"bic"`. If `NULL` (default), `"aic"` is used for GEE models (`gee_family()`) and `"pvalue"` for all other families. Supply `criterion = "pvalue"` explicitly to use robust Wald p-value selection for GEE models. With `"pvalue"`, `select` controls predictor inclusion and `alpha` controls FP functional form and spike-at-zero component comparisons. With `"aic"` or `"bic"`, the model with the lowest score is selected. For GEE models, `"aic"` and `"bic"` use quasi likelihood based scores; see **Model selection criteria** for their definitions.

`select` Significance level for predictor inclusion when `criterion = "pvalue"`; default `0.05`. In `mfp2.default()`, a single value applies to all predictors. A named vector sets the selection level for the specified predictors; all others use the default of `0.05`. In `mfp2.formula()`, the top-level value applies only to terms outside `fp()` or `fp2()`; those terms each default to `0.05` unless set explicitly, for example `fp(x, select = 1)`. Under p-value selection, `select = 1` forces inclusion. For spike-at-zero terms, `select` controls initial inclusion; `alpha` controls subsequent component removal. `select` does not affect AIC or BIC selection; use `keep` to retain a term under those criteria.

`alpha` Significance level for choosing between FP forms when `criterion = "pvalue"`; default `0.05`. A simpler form is chosen only when its comparison p-value is greater than `alpha`, so `alpha = 1` prevents this simplification. For spike-at-zero terms, `alpha` also controls tests that remove either component. In `mfp2.default()`, a single value applies to all predictors. A named vector sets values for specified predictors; all others use `0.05`. In `mfp2.formula()`, the top-level value applies only to terms outside `fp()` or `fp2()`. Set `alpha` within `fp()` or `fp2()` to change it for those terms. `alpha` does not affect AIC or BIC selection.

`keep` Character vector naming predictors to retain in the final model under any selection criterion. For a formula interface, use a predictor name from the formula, such as `keep = "age"`. For a matrix interface, use a column name from `x` or a name in `term_groups`. Keeping a predictor does not prevent selection of a simpler functional form.

`xorder` Order in which predictors are examined during backfitting: `"ascending"` (default), `"descending"`, or `"original"`. To rank predictors, each is tested as a whole by removing it from the full starting model. `"ascending"` visits the smallest p-value first, `"descending"` the largest, and `"original"` follows the input term order.

`powers` Optional named list of candidate FP powers for individual predictors, for example `list(age = c(0, 1, 2))`. Predictors not named use `c(-2, -1, -0.5, 0, 0.5, 1, 2, 3)`, where `0` denotes a logarithm. Each list element must contain at

least one finite numeric power; duplicate powers are removed. Use the predictor's column name as the list name. To allow only a linear effect, set `df = 1`. Powers supplied inside `fp()` or `fp2()` take precedence over this list.

`ties` Method for handling tied event times in Cox and Fine–Gray models. Supported values are `"efron"` (default) and `"breslow"`. `"exact"` is not supported. The argument has no effect for other families. See `survival::coxph()` for details.

`strata` Optional stratification for Cox, parametric survival, and Fine–Gray models. In `mfp2.default()`, supply one stratum label per observation as a vector or factor. To stratify by several variables, supply a matrix or data frame with one variable per column. In `mfp2.formula()`, put `strata(...)` in the formula. The separate `strata` argument is deprecated for formula fits; if both are supplied, the formula term takes precedence.

`nocenter` Numeric values that exempt a Cox or Fine–Gray predictor column from centering when every value in that column is among them. The default, `c(-1, 0, 1)`, leaves indicator columns uncentered, as in `survival::coxph()`. Set `NULL` to allow all predictor columns to be centered. Ignored for other model families.

`acd_vars` Character vector naming continuous columns of `x` to consider for approximate cumulative distribution (ACD) modelling. ACD functions are compared with simpler alternatives and need not be selected. Applies only to `mfp2.default()`; in formulas, use `fp(x, acd = TRUE)` or `fp2(x, acd = TRUE)`. See **ACD modelling**.

`ftest` Logical. If `TRUE`, use F-tests for selection in ordinary Gaussian regression models when `criterion = "pvalue"`. These tests cover variable selection, functional-form selection, and spike-at-zero testing, and may improve small-sample behaviour. Default `FALSE`. Not applicable to GEE or other model families.

`control` Optional named list of model-fitting settings. If `NULL`, the package uses its default settings for the selected model family. The available settings depend on the model family: see `stats::glm.control()` for ordinary GLMs, `survival::coxph.control()` for Cox and Fine–Gray models, `survival::survreg.control()` for parametric survival models, and `geepack::geese.control()` for GEE models. Multinomial models accept `maxit`, `reltol`, `abstol`, `trace`, and `MaxNWts`; ordinal models accept `maxit`, `eps`, `tol`, and `trace`.

`zero_vars` Names of nonnegative continuous predictors with exact zeros. Only positive values are transformed; zeros remain zero before centering. If a predictor is also in `acd_vars`, ACD is fitted using only its positive values. The shift is fixed at zero. Applies to `mfp2.default()` only; in formulas, use `fp(x, zero = TRUE)`. Negative values are rejected. See zero and spike-at-zero modelling section.

`catzero_vars` Names of nonnegative continuous predictors with exact zeros. Positive values are transformed, and a separate indicator for zero values is added to the model. This also enables `zero_vars` behavior for each predictor. Applies to `mfp2.default()` only; in formulas, use `fp(x, catzero = TRUE)`. Negative values are rejected. See the zero and spike-at-zero modelling section.

`spike_vars` Names of nonnegative continuous predictors to assess with spike-at-zero selection. The model may retain the transformed positive values, an indicator for zeros, or both. Selection uses two-stage tests when `criterion = "pvalue"` and compares candidate models when using AIC or BIC. Each group must meet

min_saz_prop; binary predictors are ineligible. Ineligible requests are ignored. Applies to mfp2.default() only; in formulas, use fp(x, spike = TRUE).

min_saz_prop
: Minimum proportion of observations required in each group (x = 0 and x > 0) for spike-at-zero selection. Must be strictly between 0 and 0.5; default 0.10. Predictors that do not meet this threshold are not assessed. In large samples, a lower threshold may still leave enough observations in each group.

force_max_fp_vars
: Names of predictors whose most complex permitted form should be kept in the model, regardless of the selection criterion. The maximum is set by df; when df = 1, it is a linear form. For an eligible predictor in spike_vars, both the zero indicator and the most complex permitted function of positive values are kept. Default NULL. Applies to mfp2.default() only; in formulas, use fp(x, force_max_fp = TRUE).

term_groups
: For mfp2.default(), an optional named list of columns to select and test together as one term. This is useful for dummy columns representing a categorical predictor. For example, list(race = c("raceB", "raceC")) treats both columns as the predictor race. Grouped columns must be linear (df = 1) and cannot use ACD or zero, catzero, or spike-at-zero modelling. Columns not listed are handled individually.

verbose
: Logical. If TRUE, print progress messages during model fitting. Default TRUE.

fitter
: Backend for GLM candidate fits during the FP search. "base" (default) uses stats::glm.fit(); "fastglm" uses the optional fastglm package. If fastglm is unavailable, ordinary GLMs use the base backend with a warning. Negative-binomial models always use fastglm and require it to be installed. Survival, multinomial, ordinal, and GEE models use their own fitters and ignore this setting.

id
: A subject or cluster identifier, with one value per observation. Use the same ID for observations from the same subject or cluster. IDs cannot be missing. Fine–Gray models require id for start–stop multi-state data; it is optional when there is one row per subject. GEE models always require id and group observations by cluster before fitting, preserving their order within each cluster. In formula fits, id is looked up first in data, then in the formula environment. It is not used by other model families.

waves
: Optional index giving the order of repeated observations within each id cluster in GEE models. Supply one positive integer per observation; values must be unique within each cluster. waves does not reorder rows. See geepack::geeglm() for further details. In formula fits, waves is looked up first in data, then in the formula environment.

acdx
: Deprecated compatibility alias for acd_vars in mfp2.default(). Use acd_vars in matrix interface.

formula
: For mfp2.formula(), a model formula. Use fp() or fp2() to set variable-specific FP, ACD, zero, catzero, or SAZ options.

data
: For mfp2.formula(), a data frame containing the variables in formula.

Value

An object of class `mfp2`, built from the final fitted model and extended with MFP selection and transformation information. Depending on the model family, it also inherits from:

- `glm` for Gaussian, binomial, Poisson, Gamma, and inverse-Gaussian models;
- `fastglm_nb` and `fastglm` for negative-binomial models;
- `geeglm` for GEE models;
- `multinom` and `nnet` for multinomial logistic models;
- `orm` for ordinal regression models;
- `coxph` for Cox and Fine–Gray models;
- `survreg` for parametric survival models.

Standard fitted-model components, such as coefficients, residuals, fitted values, and the model call, remain available. `mfp2()` adds the following components to describe its selection results:

convergence_mfp Logical value indicating whether the MFP selection algorithm converged.

fp_terms A data frame with one row per model term. It records the MFP complexity setting (`df_setting`) separately from the initial and final model degrees of freedom (`df_initial` and `df_final`), together with selection settings, selected status, searched-FP identity (`searched_fp`), fractional-polynomial powers, the ACD, zero, catzero, and spike settings actually applied after validation and eligibility checks, and `prop_zero` for retained spike-at-zero terms.

fp_powers A named list containing the selected fractional-polynomial powers for each model term. Terms excluded from the final model contain missing powers.

transformations A data frame describing the shift, scale, and centering setting of each model term. The shift and centering apply to the final fitted basis; the scale is used only during FP selection, so the final basis and its coefficients use shifted, unscaled values (see **Shifting, scaling, and centering**).

acd A named logical vector indicating which terms were assessed with the ACD extension. The selected powers identify whether the final representation retained an ACD component.

zero A named logical vector indicating which continuous terms were transformed using only their positive values.

catzero A named logical vector indicating which terms include a separate indicator for exact-zero values.

spike_dec A named numeric vector containing the final spike-at-zero decision for each term. A value of `1` retains both the zero indicator and continuous component, `2` retains the continuous component only, and `3` retains the zero indicator only. Non-spike terms use the standard value `2`.

call_mfp The call used to fit the model.

Use `print()`, `summary()`, `coef()`, `predict()`, and `plot()` for the main model operations. Use `get_selected_variable_names()` to obtain the names of terms retained in the selected model.

Only documented components form part of the stable public interface. Other components may change between package versions.

Methods (by class)

- `mfp2(default)`: Matrix interface: accepts a numeric predictor matrix `x` and response vector `y`. Categorical predictors must be pre-coded as numeric columns; group related columns with `term_groups`.
- `mfp2(formula)`: Formula interface: accepts a model formula and data frame. Factors are expanded automatically. Use `fp()` or `fp2()` to set variable-specific options.

Fractional-polynomial model selection

For a continuous predictor `x`, a linear effect has the form $\beta_1 x$. A first-degree fractional polynomial (FP1) replaces `x` with a selected power:

$$\beta_1 x^{p_1}.$$

A second-degree fractional polynomial (FP2) uses two powers:

$$\beta_1 x^{p_1} + \beta_2 x^{p_2}.$$

By convention, a power of zero means $\log(x)$. If the two powers are equal, the second term is $x^p \log(x)$; for two zero powers, the terms are $\log(x)$ and $\{\log(x)\}^2$. Here `x` denotes the positive input after any shifting and scaling. The default candidate powers are `c(-2, -1, -0.5, 0, 0.5, 1, 2, 3)`.

The `df` setting limits the maximum FP degree: `df = 1` allows only a linear effect, `df = 2` allows up to FP1, and the default `df = 4` allows up to FP2. Higher degrees can be allowed. Predictors with only two or three distinct values are restricted to `df = 1`. With four or five distinct values, a default or global `df` is capped at 2, but an explicitly set predictor-specific value is retained. Selection may still omit a predictor or choose a simpler form. Predictors in `keep` cannot be omitted, but their functional forms may still be selected. A selected FP1 with power 1 has the same fitted basis as the fixed linear form, but retains the FP power-search degrees of freedom. The `fp_terms$searched_fp` column distinguishes these cases: `TRUE` means the final continuous form came from an FP candidate class; `FALSE` means it came from the fixed linear candidate or no continuous form was retained. The flag identifies the form that won selection, not whether FP candidates were considered. For a single-predictor term with power 1, this gives `df_final = 2` for FP1 and `df_final = 1` for the fixed linear form.

With `criterion = "pvalue"`, a closed-test-style procedure uses `select` for predictor inclusion and `alpha` for functional-form comparisons. With `"aic"` or `"bic"`, candidate models are compared using the chosen information criterion.

See `vignette("mfp-methodology", package = "mfp2")` for methodological details.

Formula and matrix interfaces

For most users, the formula interface is the simpler choice: `mfp2(formula, data, ...)`. It handles factors automatically. Use `fp()` or `fp2()` to set options for individual continuous predictors; other predictors, including those added with `.`, use the applicable top-level settings. An `fp()` term has its own default `select = 0.05`, so set `select` inside the term if a different value is needed. Formulas may also include `offset()` and, for survival models, `strata()`.

The matrix interface, `mfp2(x, y, ...)`, takes a numeric predictor matrix. Categorical predictors must first be coded as numeric columns. If several columns represent one predictor, use `term_groups` to have them selected and tested together.

Both interfaces use the same selection procedure. Predictor names must be unique, non-empty, and free of backticks. Names given in options such as `keep` and `powers` must match predictors in the model.

Model families and responses

Set `family` to choose the model. Standard choices are `"gaussian"` for numeric outcomes, `"binomial"` for binary outcomes, `"poisson"` for counts, and `"Gamma"` or `"inverse.gaussian"` for positive numeric outcomes. Use `negbin_family()` for negative-binomial count regression with a `"log"`, `"sqrt"`, or `"identity"` link; `"negbin"` is shorthand for `negbin_family(link = "log")`. The first five also accept a GLM family function or object, so you can choose a link supported by `stats::glm()`. Quasi families are not supported.

Binomial responses may be values from 0 to 1, a two-level factor, or a two-column matrix of success and failure counts. Poisson and negative- binomial responses must be nonnegative integer counts. Count matrices must contain at least one trial in every row.

For outcomes with at least three categories, use `multinomial_family()` for unordered categories or `ordinal_family()` for ordered categories. Multinomial models also accept a matrix of counts by category. See **Multinomial models** and **Ordinal models** for response details.

For survival outcomes, use `"cox"` with a right-censored `survival::Surv()` response, `survreg_family()` for parametric survival regression, or `finegray_family()` for competing risks with a multi-state `Surv` response. The character values `"survreg"` and `"finegray"` use the respective default settings.

For clustered observations, use `gee_family()` with a Gaussian, binomial, Poisson, or Gamma response family. The response follows the rules for that family. See **Generalized estimating equations**.

Generalized estimating equations

Specify a GEE model with `gee_family()`. The required `id` argument identifies clusters; the package groups their observations before fitting. The optional `waves` argument specifies visit order within clusters. `gee_family()` sets the response family, working correlation, standard-error method, and scale settings. The final model is returned as a `geeglm` object.

The default selection criterion for GEE models is `criterion = "aic"`; all other families default to `criterion = "pvalue"`.

With `criterion = "pvalue"`, selection uses approximate comparisons of overall robust Wald statistics, following Stata's `mfp: xtgee` calculation. During verbose fitting, the selection tables display `Wald chi-sq` for candidates and `Wald diff.` for comparisons. The robust covariance uses the finite-cluster correction $K/(K-1)$, where K is the number of clusters.

With `criterion = "aic"` or `"bic"`, selection uses quasi-likelihood scores:

$$-2Q/\phi + 2p \quad \text{or} \quad -2Q/\phi + \log(n)p,$$

respectively. Here Q is the quasi-likelihood, ϕ is a single reference dispersion shared by all comparisons, and n is the number of clusters by default (or observations if `gee_family(qbic_penalty`

= "observations")). The reference dispersion is estimated once from a full model containing all predictors at their maximum allowed FP degrees, unless supplied through gee_family(reference_dispersion = ...). Reference powers are fixed from the permitted sets before the one reference fit. Geepack continues to estimate scale for each GEE fit; that fitted scale is used for fitting and reporting, not for these information-criterion comparisons. p counts mean-model coefficients, including the intercept, plus an allowance for selecting FP powers. The first is an MFP-adjusted QICu score; the second is a package-specific BIC-like score.

For AIC or BIC selection, print.mfp2() and summary.mfp2() show the full-linear and selected-MFP scores under Selection Scores. Smaller scores are preferred. For GEE p-value selection, they omit the Model Fit block; Wald comparisons appear during verbose fitting.

Shifting, scaling, and centering

FP functions involving logarithms or non-integer or negative powers need positive inputs. By default, mfp2() adds a shift when needed and chooses a power-of-ten scale for ordinary FP predictors to keep values numerically manageable during selection. ACD predictors may be shifted but are not scaled. Use shift = 0 or scale = 1 to turn off the corresponding automatic step for ordinary FP predictors.

Predictors with exact-zero handling (zero, catzero, or spike) that contain exact zeros are not scaled automatically (their automatic scale is 1). Their zeros stay at zero while only the positive values are transformed, so dividing by a scale factor would add a constant to logarithmic terms on the positive values only. That would make the selected powers depend on the arbitrary scale factor. Without scaling, FP selection compares the same models that are finally fitted. A scale you supply explicitly is still used.

In mfp2.default(), a single unnamed value applies to all predictors. A named value applies only to the specified column: for example, shift = c(age = 20) fixes the shift for age, while shifts for other columns are chosen automatically. The same rule applies to scale. Do not supply NA as an explicit shift or scale value.

A supplied shift is not increased automatically. If it leaves values nonpositive for a nonlinear FP that requires positive inputs, fitting stops with an error. Linear predictors (df = 1) and predictors using zero, catzero, or spike receive a shift of zero.

With center = TRUE, transformed continuous columns are centered on their mean in the fitted sample. For a predictor with exact-zero handling, its transformed values are set to zero at the original zeros *before* this mean is subtracted. Thus, its centered values at zero need not be zero. Binary columns have their lower observed value subtracted. Formula-generated factor columns follow the appropriate binary or mean-centering rule. Centering changes the intercept but not fitted values.

The final fitted FP basis uses shifted, unscaled values; scaling is used during selection. Supply new observations to predict.mfp2() on their original predictor scale. Prediction applies the fitted transformation and centering settings automatically.

Categorical predictors and grouped terms

In the formula interface, factors are expanded into contrast columns automatically. Each factor is selected or removed as one term. To keep a factor, give keep its name, not the names of its contrast columns:

```
mfp2(y ~ age + race, data = dat, keep = "race")
```

Factors are not FP-transformed, and ACD and zero options do not apply to them. Ordered factors use their configured contrasts, which are polynomial by default. They are tested as a whole, rather than with a single trend test.

In the matrix interface, create the indicator or contrast columns yourself. Use `term_groups` to select columns representing one categorical predictor together. For example, if `x` has columns `age`, `raceB`, and `raceC`:

```
mfp2(x, y, term_groups = list(race = c("raceB", "raceC")),
     keep = "race")
```

Here `keep = "race"` retains both `raceB` and `raceC`. With centering enabled, formula-generated treatment indicators keep their zero reference; other contrast columns are centered on their fitted-sample means. See **Shifting, scaling, and centering**.

ACD modelling

The approximate cumulative distribution (ACD) transformation can model a smooth S-shaped relationship between a continuous predictor and the outcome (Royston, 2014; Royston and Sauerbrei, 2016). It uses smoothed ranks to transform predictor values to a scale between 0 and 1.

In the matrix interface, specify predictors in `acd_vars`. In a formula, use `fp(x, acd = TRUE)`. Model selection considers ACD-based forms, ordinary fractional polynomials, a linear effect, and exclusion of the predictor. If the predictor has fewer than five distinct values, standard MFP is used instead. ACD predictors may be shifted but are not scaled. Selection uses the ACD function-selection procedure with `criterion = "pvalue"`; with `"aic"` or `"bic"`, candidate models are compared using the chosen criterion.

When ACD is combined with `zero = TRUE`, the transformation is fitted using positive values only; at least two distinct positive values are required. Exact zeros receive a transformed value of zero before centering. During prediction and plotting, positive values use the fitted transformation, and exact zeros again receive zero before centering.

See `vignette("acd-transformations", package = "mfp2")` for the ACD function-selection procedure.

Zero, catzero, and spike-at-zero modelling

Three options are available for nonnegative predictors that contain exact zeros.

- `zero` applies the continuous transformation to positive values only. Its value at zero is set to zero before centering.
- `catzero` transforms positive values in the same way as `zero` and adds a binary variable that equals 1 when the predictor is zero and 0 otherwise.
- `spike` uses the spike-at-zero selection algorithm to decide whether the final model includes the positive-value function, the zero indicator, both, or neither.

In the matrix interface, use `zero_vars`, `catzero_vars`, or `spike_vars`. In a formula, use `fp(x, zero = TRUE)`, `fp(x, catzero = TRUE)`, or `fp(x, spike = TRUE)`. `catzero` includes `zero` handling. An eligible `spike` request includes both `catzero` and `zero` handling. If ACD is also requested, its transformation uses positive values only.

Negative values cause an error and are never treated as zeros. These options do not apply to binary predictors. `zero` and `catzero` are ignored if the predictor has no exact zeros.

Spike-at-zero selection requires at least `min_saz_prop` of observations with zero values and at least that proportion with positive values. The default is `0.10`. If either group is too small, spike-at-zero selection is skipped. Separately requested `zero` or `catzero` handling remains in effect. With `criterion = "pvalue"`, spike-at-zero selection uses a two-stage procedure controlled by `select` and `alpha`. With `criterion = "aic"` or `"bic"`, it uses a one-stage procedure that compares all candidate models directly. Further details on spike-at-zero modelling are given by Kipruto and Sauerbrei (2026).

Subsetting

When you use `subset`, `mfp2()` determines automatic shifts and scales from the full input data before selecting observations. Model selection and fitting then use only the selected observations. The result may differ from a fit where the data were subset before calling `mfp2()`.

In the formula interface, you can select rows using a column of `data`. For example, if `dat` has a column called `group`, this fits the model using only rows where `group` is `"A"`:

```
mfp2(y ~ age, data = dat, subset = group == "A")
```

In the matrix interface, supply a logical vector or unique row numbers. The package applies the subset to `x`, `y`, `weights`, `offset`, and `strata` together. Row numbers retain the order you specify.

Use `subset` when analyses of different groups should use shifts and scales determined from the same full dataset. For cross-validation or removal of missing values, prepare the analysis data before fitting. The subset must leave more observations than predictor columns.

Survival models

Cox models require a right-censored `survival::Surv()` response, usually `Surv(time, event)` with `0` for censoring and `1` for an event. For stratification, use `strata()` in a formula or the `strata` argument with matrix input. The separate `strata` argument in the formula interface is deprecated. Cox settings such as `ties`, `nocenter`, and `control` follow `survival::coxph()`.

Parametric survival models use `survreg_family()` and return a `survreg` object. Here, `strata` allows separate scale parameters, as in `survival::survreg()`.

Fine–Gray models use `finegray_family()` with a multi-state `Surv` response. By default, `strata()` applies to both the censoring distribution and the baseline subdistribution hazard; change this with `strata_action`. Supply `id` for start–stop responses; it is optional when each subject has one row.

See `predict.mfp2()` for the data needed to predict survival probabilities.

Multinomial models

Use `family = "multinomial"` or `multinomial_family()` for an outcome with at least three unordered classes. The response can contain class labels or be a matrix of counts, with one column per class. The first class is the reference; use `multinomial_family(reference = ...)` to choose another.

For each continuous predictor, the selected FP powers are shared across classes, but each non-reference class has its own coefficients. Selection degrees of freedom account for both the class-specific coefficients and the search for shared powers. The final fit inherits from `multinom`.

Ordinal models

Use `ordinal_family()` when the outcome has at least three categories with a meaningful order. An ordered factor specifies that order. Otherwise, numeric values are ordered from smallest to largest, and character values and unordered factor levels are ordered alphabetically. For example, to set the order of named categories:

```
severity <- factor(
  c("low", "high", "medium", "low"),
  levels = c("low", "medium", "high"),
  ordered = TRUE
)
```

The default `link = "logistic"` fits a proportional-odds model. It uses the same fitted function for each predictor when comparing higher outcome categories with lower ones. The final fit is an `orm` object and requires the optional `rms` package (version 7.0-0 or later). See `ordinal_family()` for case weights, other links, and how to interpret coefficient signs.

Compatibility with the `mfp` package

Both `mfp` and `mfp2` provide `fp()`. In formulas passed to `mfp2()` or `mfpi()`, `fp()` always uses the `mfp2` version, even when both packages are loaded. `fp()` and `fp2()` are equivalent in these formulas:

```
fit1 <- mfp2(y ~ fp(x1) + fp(x2), data = dat)
fit2 <- mfp2(y ~ fp2(x1) + fp2(x2), data = dat)
```

Convergence and inference

MFP selection usually converges within 2–4 cycles. With `verbose = TRUE`, progress messages report when it converges. Lack of convergence involves oscillation between two or more models and is extremely rare. If selection does not converge within the allowed number of `cycles`, `mfp2()` returns the model obtained after the final cycle. Consider increasing `cycles`, changing `select` or `alpha`, or using `criterion = "aic"` or `"bic"`.

Standard errors, confidence intervals, and p-values from the final model do not account for uncertainty from selecting predictors or their functional forms, including ACD and spike-at-zero selection.

References

Royston, P. and Sauerbrei, W., 2008. *Multivariable Model - Building: A Pragmatic Approach to Regression Analysis based on Fractional Polynomials for Modelling Continuous Variables. John Wiley & Sons.*

Liang, K.-Y. and Zeger, S. L., 1986. *Longitudinal data analysis using generalized linear models. Biometrika, 73(1): 13–22.*

Pan, W., 2001. *Akaike's information criterion in generalized estimating equations. Biometrics, 57(1): 120–125.*

Sauerbrei, W., Meier-Hirmer, C., Benner, A. and Royston, P., 2006. *Multivariable regression model building by using fractional polynomials: Description of SAS, STATA and R programs. Comput Stat Data Anal, 50(12): 3464-85.*

Royston, P. 2014. *A smooth covariate rank transformation for use in regression models with a sigmoid dose-response function. Stata Journal 14(2): 329-341.*

Royston, P. and Sauerbrei, W., 2016. *mfpa: Extension of mfp using the ACD covariate transformation for enhanced parametric multivariable modeling. The Stata Journal, 16(1), pp.72-87.*

Sauerbrei, W. and Royston, P., 1999. *Building multivariable prognostic and diagnostic models: transformation of the predictors by using fractional polynomials. J Roy Stat Soc a Sta, 162:71-94.*

Sauerbrei, W., Kipruto, E. and Balmford, J., 2023. *Effects of influential points and sample size on the selection and replicability of multivariable fractional polynomial models. Diagnostic and Prognostic Research, 7(1), p.7.*

Becher, H., Lorenz, E., Royston, P. and Sauerbrei, W., 2012. *Analysing covariates with spike at zero: a modified FP procedure and conceptual issues. Biometrical journal, 54(5), pp.686-700.*

Lorenz, E., Jenkner, C., Sauerbrei, W. and Becher, H., 2019. *Modeling exposures with a spike at zero: simulation study and practical application to survival data. Biostatistics & Epidemiology, 3(1), pp.23-37.*

See Also

`fp()`, `fp2()`, `summary.mfp2()`, `coef.mfp2()`, `predict.mfp2()`, `plot.mfp2()`, `get_selected_variable_names()`, `transform_vector_fp()`, `gee_family()`, `multinomial_family()`, `ordinal_family()`, `survreg_family()`, `finegray_family()`

Examples

```
# Gaussian model: formula interface (recommended for most users)
data("prostate")
fit_formula <- mfp2(
  lpsa ~ fp(age) + fp(svi, df = 1) + fp(pgg45) + fp(cavol) +
    fp(weight) + fp(bph) + fp(cp),
  data = prostate,
  verbose = FALSE
)

fit_formula
summary(fit_formula)
get_selected_variable_names(fit_formula)
```

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plots <- plot(fit_formula)
  print(plots[[1]])
}

predict(fit_formula, newdata = prostate[1:5, ])

# Matrix interface
x <- as.matrix(prostate[, 2:8])
y <- prostate$lpsa
fit_matrix <- mfp2(x, y, verbose = FALSE)
fit_matrix

# Named partial settings apply only to the named columns. Here age is fixed,
# while shifts and scales for every other matrix column remain automatic.
fit_partial <- mfp2(
  x,
  y,
  shift = c(age = 20),
  scale = c(age = 10),
  verbose = FALSE
)
fit_partial$transformations[c("age", "weight"), c("shift", "scale")]


# ACD modelling
fit_acd_formula <- mfp2(
  lpsa ~ fp(cavol, acd = TRUE) + fp(age) + svi,
  data = prostate,
  verbose = FALSE
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(fit_acd_formula, terms = "cavol")[[1]]
}

# Spike-at-zero modelling
set.seed(1)
n <- 200
exposure <- numeric(n)
zero_rows <- sample(seq_len(n), size = round(0.25 * n))
exposure[-zero_rows] <- rgamma(
  n - length(zero_rows),
  shape = 2,
  rate = 0.5
)
age <- runif(n, 20, 80)
outcome <- 1.5 * (exposure == 0) +
  2 * log(ifelse(exposure > 0, exposure, 1)) +
  0.02 * age + rnorm(n)

d_spike <- data.frame(outcome, exposure, age)

# Formula interface
```

```
fit_spike_formula <- mfp2(
  outcome ~ fp(exposure, spike = TRUE) + fp(age),
  data = d_spike,
  verbose = FALSE
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(fit_spike_formula, terms = "exposure")[[1]]
}

# Matrix interface
x_spike <- as.matrix(d_spike[, c("exposure", "age")])
fit_spike_matrix <- mfp2(
  x_spike,
  d_spike$outcome,
  spike_vars = "exposure",
  verbose = FALSE
)

# Unordered factor: all treatment-contrast columns are selected jointly
set.seed(42)
d_factor <- data.frame(
  y = rnorm(60),
  age = runif(60, 20, 80),
  group = factor(rep(c("A", "B", "C"), each = 20))
)
fit_factor <- mfp2(
  y ~ age + group,
  data = d_factor,
  keep = "group",
  verbose = FALSE
)
predict(fit_factor, newdata = d_factor[1:4, ])

# Ordered factor: the default polynomial contrasts form one joint term
d_factor$severity <- ordered(
  rep(c("mild", "moderate", "severe"), length.out = nrow(d_factor)),
  levels = c("mild", "moderate", "severe")
)
fit_ordered <- mfp2(
  y ~ age + severity,
  data = d_factor,
  keep = "severity",
  verbose = FALSE
)
predict(
  fit_ordered,
  newdata = data.frame(
    age = c(40, 40),
    severity = ordered(
      c("mild", "severe"),
      levels = levels(d_factor$severity)
    )
  )
```

```
)

# Matrix interface: manually created dummy columns can be grouped as one term
x_grouped <- cbind(
  age = d_factor$age,
  groupB = as.integer(d_factor$group == "B"),
  groupC = as.integer(d_factor$group == "C")
)
fit_grouped <- mfp2(
  x_grouped,
  d_factor$y,
  term_groups = list(group = c("groupB", "groupC")),
  keep = "group",
  verbose = FALSE
)

# Examples for every supported model family ------------------------------
set.seed(2026)
n_family <- 120
family_data <- data.frame(
  x1 = runif(n_family, 0.5, 3),
  x2 = rnorm(n_family)
)
eta <- with(family_data, 0.2 + 0.3 * x1 - 0.2 * x2)
mu <- exp(eta)
simulate_inverse_gaussian <- function(mu, shape) {
  squared_normal <- stats::rnorm(length(mu))^2
  candidate <- mu + (mu^2 * squared_normal) / (2 * shape) -
    (mu / (2 * shape)) * sqrt(
      4 * mu * shape * squared_normal + mu^2 * squared_normal^2
    )
  choose_candidate <- stats::runif(length(mu)) <= mu / (mu + candidate)
  ifelse(choose_candidate, candidate, mu^2 / candidate)
}
family_data$y_gaussian <- eta + rnorm(n_family)
family_data$y_binomial <- rbinom(n_family, 1, plogis(eta - 1))
family_data$y_poisson <- rpois(n_family, mu)
family_data$y_gamma <- rgamma(n_family, shape = 5, scale = mu / 5)
family_data$y_inverse_gaussian <- simulate_inverse_gaussian(mu, shape = 8)
family_data$y_negbin <- rnbinom(n_family, mu = mu, size = 2)
family_data$y_multinomial <- sample.int(
  3L, n_family, replace = TRUE, prob = c(0.45, 0.35, 0.20)
)
family_data$y_ordinal <- ordered(
  cut(
    eta + stats::rlogis(n_family),
    breaks = c(-Inf, -0.5, 0.5, 1.5, Inf),
    labels = c("low", "medium", "high", "very high")
  )
)

fit_gaussian <- mfp2(
  y_gaussian ~ x1 + x2, data = family_data, family = "gaussian",
```

```
  df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
)
fit_binomial <- mfp2(
  y_binomial ~ x1 + x2, data = family_data, family = "binomial",
  df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
)
fit_poisson <- mfp2(
  y_poisson ~ x1 + x2, data = family_data, family = "poisson",
  df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
)
fit_gamma <- mfp2(
  y_gamma ~ x1 + x2, data = family_data,
  family = stats::Gamma(link = "log"),
  df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
)
fit_inverse_gaussian <- mfp2(
  y_inverse_gaussian ~ x1 + x2, data = family_data,
  family = stats::inverse.gaussian(link = "log"),
  df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
)

# Negative binomial uses the optional fastglm backend. `"negbin"` is
# shorthand for negbin_family(link = "log"); see ?negbin_family for the
# square-root and identity links.
if (requireNamespace("fastglm", quietly = TRUE)) {
  fit_negbin <- mfp2(
    y_negbin ~ x1 + x2, data = family_data,
    family = "negbin",
    df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
  )
}

# Integer class labels are converted to a multinomial factor response.
fit_multinomial <- mfp2(
  y_multinomial ~ x1 + x2, data = family_data,
  family = multinomial_family(),
  df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
)

# Ordinal regression uses the optional rms package.
if (requireNamespace("rms", quietly = TRUE) &&
    utils::packageVersion("rms") >= "7.0-0") {
  fit_ordinal <- mfp2(
    y_ordinal ~ x1 + x2, data = family_data,
    family = ordinal_family(),
    df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
  )
}

# Cox proportional hazards and parametric survival models.
event_time <- rexp(n_family, rate = exp(eta - 2))
censor_time <- rexp(n_family, rate = 0.15)
survival_data <- transform(
```

```
  family_data,
  time = pmin(event_time, censor_time),
  status = as.integer(event_time <= censor_time)
)
fit_cox <- mfp2(
  survival::Surv(time, status) ~ x1 + x2,
  data = survival_data, family = "cox",
  df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
)
fit_survreg <- mfp2(
  survival::Surv(time, status) ~ x1 + x2,
  data = survival_data, family = survreg_family(dist = "weibull"),
  df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
)

# Fine--Gray subdistribution hazards for a competing-risks response.
finegray_data <- transform(
  family_data,
  ftime = rexp(n_family, rate = 0.12),
  event = factor(
    sample(
      c("censor", "cause1", "cause2"), n_family, replace = TRUE,
      prob = c(0.35, 0.40, 0.25)
    ),
    levels = c("censor", "cause1", "cause2")
  )
)
fit_finegray <- mfp2(
  survival::Surv(ftime, event) ~ x1 + x2,
  data = finegray_data, family = finegray_family(etype = "cause1"),
  df = 1, select = 1, alpha = 1, cycles = 1, verbose = FALSE
)
```

`mfpi` *Test Interactions with a Categorical Grouping Variable*

Description

Test whether the association between an outcome and a continuous, binary, or categorical predictor differs between groups. Fractional polynomials (FPs) allow nonlinear associations for continuous predictors.

Usage

```
mfpi(x, ...)

## Default S3 method:
mfpi(
```

```
  x,
  y,
  group_var = NULL,
  interaction_vars = NULL,
  interaction_forms = NULL,
  flex = c("flex3", "flex1", "flex2", "flex4"),
  p_interact = 0.05,
  min_improvement = NULL,
  include_group_var = FALSE,
  show_models = FALSE,
  weights = NULL,
  offset = NULL,
  cycles = 10,
  scale = NULL,
  shift = NULL,
  df = 4,
  center = TRUE,
  subset = NULL,
  family = gaussian(),
  criterion = c("pvalue", "aic", "bic"),
  select = 0.05,
  alpha = 0.05,
  keep = NULL,
  force_max_fp_vars = NULL,
  xorder = c("ascending", "descending", "original"),
  powers = NULL,
  ties = c("efron", "breslow"),
  strata = NULL,
  nocenter = c(-1, 0, 1),
  acd_vars = NULL,
  zero_vars = NULL,
  catzero_vars = NULL,
  spike_vars = NULL,
  min_saz_prop = 0.1,
  ftest = FALSE,
  control = NULL,
  winsorize = FALSE,
  winsorize_probs = c(0.01, 0.99),
  center_type = c("grand", "group"),
  p_adjust_method = "none",
  verbose = TRUE,
  digits = 3,
  term_groups = NULL,
  fitter = c("base", "fastglm"),
  ...
)

## S3 method for class 'formula'
```

```
mfpi(
  formula,
  data,
  group_var = NULL,
  interaction_vars = NULL,
  interaction_forms = NULL,
  flex = c("flex3", "flex1", "flex2", "flex4"),
  p_interact = 0.05,
  min_improvement = NULL,
  include_group_var = FALSE,
  show_models = FALSE,
  weights = NULL,
  cycles = 10,
  scale = NULL,
  shift = NULL,
  df = 4,
  center = TRUE,
  subset = NULL,
  family = gaussian(),
  criterion = c("pvalue", "aic", "bic"),
  select = 0.05,
  alpha = 0.05,
  keep = NULL,
  force_max_fp_vars = NULL,
  xorder = c("ascending", "descending", "original"),
  powers = NULL,
  ties = c("efron", "breslow"),
  nocenter = c(-1, 0, 1),
  zero_vars = NULL,
  catzero_vars = NULL,
  spike_vars = NULL,
  min_saz_prop = 0.1,
  ftest = FALSE,
  control = NULL,
  winsorize = FALSE,
  winsorize_probs = c(0.01, 0.99),
  center_type = c("grand", "group"),
  p_adjust_method = "none",
  verbose = TRUE,
  digits = 3,
  fitter = c("base", "fastglm"),
  ...
)
```

Arguments

x	A numeric predictor matrix or data frame for the default method, without an intercept column. Include the grouping variable, all variables to test for interaction, and any adjustment variables. Data frames also accept logical predictors

and a factor or character grouping variable. Represent other categorical predictors using numeric dummy variables. For categorical predictors represented by multiple columns, use `term_groups` to identify which columns belong together so that each predictor is assessed as a whole rather than as separate dummy variables.

`...` Not used.

`y` The response for the default method, with one observation per row of `x`. Supply a response appropriate for `family`: a numeric response for ordinary GLMs, a binary response or two-column matrix of integer success and failure counts for binomial models, class labels or a class-count matrix for multinomial models, ordered categories for ordinal models, or a `survival::Surv()` object for survival models.

`group_var` A single character string naming the categorical grouping variable whose interactions with the variables in `interaction_vars` are to be tested. It must be included in `x` or `formula`. This argument must be supplied. The variable must have at least two observed groups in the data used for fitting and cannot also appear in `interaction_vars` or belong to `term_groups`. The reference is the first observed factor level, the first observed character value, or the smallest observed logical or numeric value. Supply a factor with the desired level order to choose the reference explicitly.

`interaction_vars`
A non-empty character vector naming variables to test separately for interaction with `group_var`. This argument must be supplied. Variables must be included in `x` or `formula`. For categorical predictors, name the complete term rather than individual contrast columns. Each analysis includes group main effects and the selected adjustment terms, excluding the variable currently being tested. The resulting interactions are not combined into a single model.

`interaction_forms`
An optional named character vector specifying the interaction form for each variable tested for interaction, for example `c(age = "fp1", stage = "linear")`. Use `"linear"` for a linear, binary, or categorical effect, `"fp1"` for a one-term fractional polynomial, or `"fp2"` for a two-term fractional polynomial, allowing repeated powers. `NULL` and unspecified entries default to `"fp1"` for continuous variables and `"linear"` for binary and categorical variables. FP requests for binary or categorical variables are changed to `"linear"` with a warning.

`flex` A single character string controlling FP power selection. `"flex1"` selects powers in the model without interaction and reuses them in the interaction model. `"flex2"` selects shared powers in the interaction model and reuses them in the model without interaction. `"flex3"` (default) selects powers separately for the two models but uses common powers across groups in the interaction model. `"flex4"` works like `"flex3"` but allows different FP powers across groups. All options estimate group-specific coefficients in the interaction model. Ignored for `"linear"` interactions. P-values for `"flex3"` and `"flex4"` are approximate because the selected models can be non-nested.

`p_interact` The p-value threshold for selecting interactions, greater than 0 and at most 1. Default: `0.05`. Used only with `criterion = "pvalue"`. An interaction is selected when its p-value is below this threshold. If a multiple-testing adjustment

is requested through p_adjust_method, the adjusted p-value is compared with the threshold.

min_improvement

A single finite positive minimum improvement required to select an interaction with criterion = "aic" or "bic". NULL uses 2. An interaction is selected when the information criterion for the model without interaction minus that for the interaction model is strictly greater than this value. Unused with criterion = "pvalue"; it does not affect adjustment-model selection.

include_group_var

A logical value indicating whether to force the grouping variable into the initial adjustment model. Default: FALSE, which excludes it from that stage. Group main effects are included in every subsequent interaction comparison regardless of this setting.

show_models A logical value indicating whether to print summaries of selected interaction models. Default: FALSE. Printing requires verbose = TRUE; this setting does not change which models are fitted or returned.

weights An optional numeric vector of observation weights greater than zero, with one value per observation in the original data. NULL gives equal weights. Weights are used in both adjustment-model selection and interaction fitting. In the formula method, weights can refer to a column in data.

offset An optional numeric offset for the default method, added to the linear predictor with its coefficient fixed at one in both adjustment-model selection and interaction fitting. Supply one value per observation in the original data. For multinomial models, supply a matrix with one row per observation and either C or C - 1 columns, where C is the number of outcome classes. NULL specifies no offset. In the formula method, include offset(...) in the formula.

cycles A positive integer giving the maximum number of MFP fitting cycles used to select the adjustment model. Default: 10.

scale Predictor scaling factors greater than zero, or NULL for automatic scaling. Use 1 to disable scaling. In the default method, supply a single value for all predictors or a named numeric vector specifying values for individual columns of x. Unspecified columns use automatic scaling. In the formula method, supply a single value or NULL; use fp() or fp2() for variable-specific settings. Predictors with structural zeros use an automatic scale of 1 unless a scale is supplied explicitly. Binary predictors, the grouping variable, and categorical interaction variables are not scaled.

shift Constants added to predictors before scaling and FP transformation, or NULL for automatic shifting. Use 0 to disable shifting. In the default method, supply a single value for all predictors or a named numeric vector specifying values for individual columns of x. Unspecified columns use automatic shifting. In the formula method, supply a single value or NULL; use fp() or fp2() for variable-specific settings. Values used in nonlinear FP transformations must be positive after shifting. A supplied shift is not increased automatically. Predictors with structural zeros, binary predictors, the grouping variable, and predictors modelled only linearly are not shifted.

df — The maximum FP complexity allowed in the adjustment model: 1 for linear, 2 for FP1, 4 for FP2, or a positive even integer 2m for an FP of degree m. Default: 4. In the default method, supply a single value for all predictors or a named numeric vector specifying values for individual columns of x. Unspecified columns use the default; categorical terms are treated as linear. In the formula method, supply a single value; use fp() or fp2() for variable-specific settings. Use interaction_forms to specify the interaction form.

center — Whether to centre transformed predictors. Default: TRUE. In the default method, supply a single logical value for all predictors or a named logical vector specifying values for individual columns of x. Unspecified columns use TRUE. In the formula method, supply a single value; use fp() or fp2() for variable-specific settings. For interaction functions, center_type determines the reference mean. Binary and categorical interaction variables are not centred. For predictors with structural zeros, the mean includes zero observations and is subtracted from every observation.

subset — An optional logical vector indicating which observations to include, or a vector of unique positive row numbers. Logical values must not be missing, and row numbers must be within the data range. NULL uses all observations. The formula method also accepts an expression using data. Supply responses, weights, offsets, and strata for the original data; the same subset is applied to all of them. At least five observations and two groups must remain, with at least two observations per group, or three if any FP2 interaction is requested. These minimum counts do not guarantee that the models can be fitted.

family — The regression family or model. Default: "gaussian". Supported GLM families are "gaussian", "binomial", "poisson", "Gamma", and "inverse.gaussian", supplied as a name, function, or family object. Use negbin_family() for negative-binomial regression with a "log", "sqrt", or "identity" link (requires fastglm); "negbin" is shorthand for negbin_family(link = "log"). Use "multinomial" or multinomial_family() for outcomes with at least three classes, supplied as class labels or a matrix of class counts. Use "ordinal" or ordinal_family() for ordered outcomes with at least three categories (requires rms version 7.0-0 or later). An ordered factor sets the category order; otherwise numeric categories are ordered numerically and character or unordered-factor categories alphabetically. Use "cox" for right-censored survival data or survreg_family() for parametric survival regression. Both require a survival::Surv() response. The response must meet the selected family's requirements. Quasi-likelihood, GEE, and Fine–Gray models are not supported.

criterion — The selection criterion for the adjustment model and interactions: "pvalue" (default), "aic", or "bic". With "pvalue", select and alpha control adjustment-model selection, while p_interact and p_adjust_method control interaction selection. With "aic" or "bic", min_improvement sets the improvement required to select an interaction.

select — The p-value threshold for retaining variables in the adjustment model, between 0 and 1 inclusive. Default: 0.05. Used only with criterion = "pvalue"; setting it to 1 retains the variable. In the default method, supply a single value for all predictors or a named numeric vector specifying values for individual columns of x. Unspecified columns use 0.05. In the formula method, supply a single

value; use fp() or fp2() for variable-specific settings. Use p_interact to set the threshold for selecting interactions.

alpha — The p-value threshold for selecting the FP degree in the adjustment model, between 0 and 1 inclusive. Default: 0.05. Used only with criterion = "pvalue". Setting alpha = 1 forces the maximum FP degree allowed by df, but the variable can still be removed during variable selection. Use select = 1 or keep to ensure that it remains in the model. In the default method, supply a single value for all predictors or a named numeric vector specifying values for individual columns of x. Unspecified columns use 0.05. In the formula method, supply a single value; use fp() or fp2() for variable-specific settings. Use interaction_forms to specify the interaction form.

keep — An optional character vector naming predictors that must remain in the initial adjustment model. For a categorical predictor, naming the complete term or any of its dummy variables retains the whole term. FP degrees and powers are still selected. The default is NULL, which does not force any predictors to remain in the model.

force_max_fp_vars — An optional character vector naming adjustment variables that must remain in the model at the maximum FP degree allowed by df, regardless of the selection criterion. FP powers are still selected from the data. The default is NULL. For spike-at-zero variables, the full permitted model is retained. This option cannot be used for grouped categorical predictors. In the formula method, you can also request the maximum FP degree for individual variables through fp() or fp2(). This argument does not change the interaction form specified by interaction_forms.

xorder — The order in which adjustment terms are assessed: "ascending" (default) orders terms from smallest to largest p-value for their contribution to the initial full linear model; "descending" reverses that order; "original" preserves predictor or formula term order. Each ordering p-value compares the full linear model with a model omitting the complete term while retaining the others.

powers — An optional named list of candidate FP powers, for example list(age = c(0, 0.5, 1)). The default is NULL, which uses c(-2, -1, -0.5, 0, 0.5, 1, 2, 3) for all variables. Unspecified variables also use these powers. Power 0 represents the natural logarithm, and power 1 represents a linear function. These powers are used for both adjustment and interaction functions. FP2 models allow repeated powers. In the formula method, powers specified in fp() or fp2() override this argument for the same variable.

ties — The method for handling tied event times in Cox models: "efron" (default) or "breslow". Ignored for other families.

strata — Optional survival-model strata for the default method. Supply a vector of group labels or a matrix or data frame of grouping variables, with one row per observation in the original data and no missing values. The default is NULL, meaning no stratification. Each stratum has its own baseline hazard in Cox models or scale parameter in survreg models. In the formula method, use strata(...) in the formula.

nocenter — A numeric vector of values used to determine which predictor columns the Cox fitter leaves uncentred. A column is not centred if all its values are in this vec-

tor. The default is `c(-1, 0, 1)`, as in `survival::coxph()`. Use `NULL` to allow centring of all columns. Applies only to Cox models and is separate from FP centring controlled by `center` and `center_type`.

`acd_vars` An optional character vector naming adjustment variables to transform using an approximate cumulative distribution (ACD). The default is `NULL`, meaning no ACD transformations are requested. This argument is available only in the default method; in the formula method, use `fp(variable, acd = TRUE)`. For variables with structural zeros, the ACD is estimated using positive values only. ACD transformations cannot be used for grouped categorical predictors and are disabled with a warning for variables tested for interaction.

`zero_vars` An optional character vector naming variables whose continuous function applies only to positive values, with zero contribution at zeros before centring. Values must not be negative. No separate zero indicator is added. The default is `NULL`. This option supports adjustment variables and variables tested for interaction. For interaction testing, each group needs at least two positive observations for a linear or FP1 form, or three for FP2. In the formula method, you can also use `fp(variable, zero = TRUE)`. Do not list the same adjustment variable in `catzero_vars`. This option is ignored for binary variables and variables without zeros.

`catzero_vars` An optional character vector naming adjustment variables to model using a continuous function for positive values and a separate indicator for zeros. Values must not be negative. The default is `NULL`. Do not list the same variable in `zero_vars`. In the formula method, you can also use `fp(variable, catzero = TRUE)`. This option cannot be used for grouped categorical predictors and is disabled with a warning for variables tested for interaction; any separately requested `zero_vars` handling is retained. It is ignored for binary variables and variables without zeros.

`spike_vars` An optional character vector naming adjustment variables for spike-at-zero modelling. Values must not be negative, and the zero and positive components must each meet `min_saz_prop`. Default: `NULL`. In the formula method, use `fp(variable, spike = TRUE)`. Not supported for binary or grouped categorical predictors, or variables tested for interaction.

`min_saz_prop` The minimum proportion of zero and positive observations required for spike-at-zero modelling. Must be greater than 0 and less than 0.5. Default: `0.10`, requiring at least 10% in each component. Variables that do not meet this requirement skip spike-at-zero selection.

`ftest` Whether to use F-tests instead of likelihood-ratio chi-square tests for Gaussian adjustment-model selection and interaction testing. Default: `FALSE`. Other families use chi-square tests and issue a warning if `TRUE` is supplied.

`control` Fitting controls used in both adjustment-model selection and interaction fitting. Use `stats::glm.control()`, `survival::coxph.control()`, or `survival::survreg.control()` as appropriate. Default: `NULL`, which uses the fitter's defaults. For `fastglm`, use `epsilon` and `maxit`; `trace = TRUE` is not supported.

`winsorize` A logical value indicating whether to cap continuous variables tested for interaction at the quantiles specified by `winsorize_probs`. Default: `FALSE`. No observations are removed, and the same limits are used for prediction. Binary, categorical, and structural-zero variables are excluded.

winsorize_probs
: Two probabilities specifying the lower and upper limits when winsorize = TRUE, in increasing order between 0 and 1. Default: c(0.01, 0.99), corresponding to the first and ninety-ninth percentiles.

center_type
: A character string specifying how interaction functions are centred: "grand" (default) uses means across all groups, while "group" uses means within each group. With "flex4", each group's function has its own centring value. The same values are used for prediction. Applies only when center = TRUE; binary and categorical variables tested for interaction are not centred.

p_adjust_method
: A character string specifying the multiple-testing adjustment for interaction p-values, using a method from stats::p.adjust(), such as "holm", "bonferroni", or "BH". Default: "none". With criterion = "pvalue", adjusted p-values are compared with p_interact. Each variable in interaction_vars counts as one test, including those with unavailable p-values. Unused for AIC or BIC selection.

verbose
: A logical value indicating whether to print fitting progress and selection results. Default: TRUE. Set to FALSE to suppress this output, including the model summaries requested by show_models.

digits
: A positive integer giving the number of decimal places in printed interaction results, excluding FP powers and degrees of freedom. Default: 3. This affects printed output rather than fitting precision.

term_groups
: An optional named list grouping dummy variables for categorical predictors in the default method, for example list(stage = c("stageII", "stageIII")). This ensures each predictor is assessed as a whole. Use the list name, such as "stage", in interaction_vars to test the complete predictor. Each column can belong to only one group, and group_var cannot be included. Grouped predictors are linear and cannot use special transformations or force_max_fp_vars. Default: NULL. The formula method groups categorical predictors automatically.

fitter
: The GLM fitting method: "base" (default) or "fastglm". The latter requires the optional fastglm package. Ordinary GLMs warn and retry with the base fitter if fastglm is unavailable or fitting fails. Negative-binomial models require fastglm and select it automatically. Survival models are unaffected by this argument.

formula
: A model formula containing the response, grouping variable, interaction variables, and adjustment variables. Supply original variables in data; categorical predictors are encoded and assessed as complete terms. Use fp() or fp2() to specify variable-specific adjustment-model and preprocessing settings. These settings override the corresponding global defaults but do not determine the interaction form, which is set by interaction_forms. Include offsets using offset(...) and survival strata using strata(...). See *Formula interface*.

data
: A data frame containing the variables used in formula. Formula expressions, weights, and subsetting expressions are evaluated using these data.

Details

mfpi() first selects an adjustment model using mfp2(). It then tests each variable in interaction_vars

separately for interaction with group_var, comparing a common association across groups with group-specific associations.

Both models include group main effects and the selected adjustment terms. The adjustment terms are held fixed, except that the variable being tested is modelled separately as part of the interaction comparison. A separate interaction model is returned for each tested variable; selected interactions are not combined into a single model.

Value

An object of class "mfpi" containing the adjustment model, interaction results, and fitted interaction models. Each variable is tested separately; selected interactions are not combined into a single model.

The main components are:

- all_model_metrics: results for every tested interaction, including p-values, test degrees of freedom, AIC and BIC comparisons between models with and without interaction, and selected FP powers.
- best_model_metrics: results for interactions that meet the selection rule specified by criterion and its associated threshold.
- all_interaction_models: fitted interaction models for all tested variables, including those whose interactions were not selected.
- best_interaction_model: fitted interaction models for variables whose interactions were selected.
- adjustment_model: the initial adjustment model selected using mfp2(). Its selected terms and transformations are used in the interaction analyses. It accepts prediction data on the original covariate scale.
- interaction_forms: the interaction form used for each tested variable: "linear", "fp1", or "fp2".
- group_level_map: the mapping between the internal group codes used for fitting and the original group labels.
- winsorize_limits: present only when winsorize = TRUE; the lower and upper limits used for Winsorisation, calculated after shifting and scaling. These limits are reused for prediction.
- criterion: the criterion used for model selection: "pvalue", "aic", or "bic".
- p_adjust_method: the multiple-testing adjustment method used for interaction p-values.
- call: the matched function call, recording the supplied arguments.

Use summary() to inspect interaction results, coef() and vcov() to extract coefficients and covariance matrices, and predict() and plot() to examine fitted associations and differences between groups.

Methods (by class)

- mfpi(default): Default method accepting a numeric matrix or data frame x and response vector y.
- mfpi(formula): Formula method using a model formula and data frame.

References

Kipruto, E., Royston, P. and Sauerbrei, W. Multivariable Fractional Polynomial Interaction Models in R.

Royston, P. and Sauerbrei, W. (2004). A new approach to modelling interactions between treatment and continuous covariates in clinical trials by using fractional polynomials. *Statistics in Medicine*, 23, 2509–2525. doi:10.1002/sim.1815.

Royston, P. and Sauerbrei, W. (2009). Two techniques for investigating interactions between treatment and continuous covariates in clinical trials. *The Stata Journal*, 9, 230–251. doi:10.1177/1536867X0900900204.

Royston, P. and Sauerbrei, W. (2013). Interaction of treatment with a continuous variable: simulation study of significance level for several methods of analysis. *Statistics in Medicine*, 32, 3788–3803. doi:10.1002/sim.5813.

Royston, P. and Sauerbrei, W. (2014). Interaction of treatment with a continuous variable: simulation study of power for several methods of analysis. *Statistics in Medicine*, 33, 4695–4708. doi:10.1002/sim.6308.

See Also

`mfp2()`, `fp()`, `coef.mfpi()`, `vcov.mfpi()`, `predict.mfpi()`, `plot.mfpi()`, `summary.mfpi()`, `multinomial_family()`, `ordinal_family()`, `survreg_family()`

Examples

```
data("prostate")

# Test FP1 and linear interactions with a categorical grouping variable.
# The grouping variable, svi, must appear in the formula and be named
# in group_var.
fit <- mfpi(
  lpsa ~ svi + fp(age) + fp(cavol) + fp(weight) + fp(bph) + fp(cp),
  data = prostate,
  family = gaussian(),
  group_var = "svi",
  p_interact = 0.9,
  interaction_vars = c("cavol", "age"),
  interaction_forms = c(cavol = "fp1", age = "linear"),
  flex = "flex1",
  verbose = FALSE
)

print(fit)
summary(fit)

# Examine the cavol interaction even if it was not selected.
pred <- predict(
  fit,
  terms = "cavol",
  model = "all",
  type = "both",
```

```
  grid = TRUE
)
pred$functions
pred$differences


# Plot the fitted cavol functions and differences between groups.
if (requireNamespace("patchwork", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  plot(fit, terms = "cavol")
}
```

`multinomial_family` *Multinomial Logistic Family*

Description

Use `multinomial_family()` with `mfp2()` or `mfpi()` when the outcome has at least three categories with no natural order. Choose one category as the reference; the model describes each other category in relation to it.

Usage

```
multinomial_family(reference = NULL)
```

Arguments

`reference`	Optional reference category. For a factor response, give a factor level; for a count-matrix response, give a column name or its one-based column number. If `NULL`, the first factor level or count-matrix column is used. Character and numeric category labels are converted to factor levels before the reference is chosen.

Details

Each predictor has one selected linear or curved form shared across the category comparisons. The strength and direction of its association may differ between categories.

The response can be a factor, a vector of character or numeric category labels, or a numeric matrix of category counts. It must contain at least three categories. Counts must be nonnegative whole numbers, every row must have at least one count, and every category must be represented in the weighted data.

Value

A family specification to supply as the `family` argument of `mfp2()` or `mfpi()`.

See Also

mfp2(), mfpi()

Examples

```
set.seed(43)
n <- 150
multinomial_data <- data.frame(
  x1 = stats::rnorm(n),
  x2 = stats::runif(n, -1, 1)
)
eta_b <- with(multinomial_data, 0.4 * x1 - 0.2 * x2)
eta_c <- with(multinomial_data, -0.3 * x1 + 0.5 * x2)
denominator <- 1 + exp(eta_b) + exp(eta_c)
probabilities <- cbind(
  A = 1 / denominator,
  B = exp(eta_b) / denominator,
  C = exp(eta_c) / denominator
)
multinomial_data$y <- factor(vapply(
  seq_len(n),
  function(i) sample(c("A", "B", "C"), 1, prob = probabilities[i, ]),
  character(1)
))

fit_multinomial <- mfp2(
  y ~ x1 + x2,
  data = multinomial_data,
  family = multinomial_family(reference = "A"),
  df = 1,
  select = 1,
  cycles = 1,
  verbose = FALSE
)
predict(
  fit_multinomial,
  newdata = multinomial_data[1:3, ],
  type = "response",
  se.fit = FALSE
)
```

---

negbin_family *Negative-Binomial Family*

Description

Use negbin_family() with mfp2() or mfpi() for overdispersed count outcomes. The regression coefficients and the negative-binomial dispersion parameter θ are estimated jointly by maximum likelihood with fastglm::fastglm_nb(), using the variance function $\mu + \mu^2/\theta$.

Usage

```
negbin_family(link = c("log", "sqrt", "identity"))
```

Arguments

`link`	Link function relating the expected count μ to the linear predictor η. One of `"log"` (the default, $\log \mu = \eta$), `"sqrt"` ($\sqrt{\mu} = \eta$), or `"identity"` ($\mu = \eta$). These are the links supported by `fastglm::fastglm_nb()`.

Details

`family = "negbin"` is shorthand for `negbin_family(link = "log")` and gives the same model.

The response must contain finite, nonnegative integer counts.

Coefficients, fitted functions, contrasts, and `type = "link"` predictions are on the scale of the chosen link: the log mean, the square root of the mean, or the mean itself. `type = "response"` predictions are expected counts. Response-scale standard errors use the delta method with the derivative of the chosen inverse link. An offset is added to the linear predictor on the same link scale; for example, `log(exposure)` is the usual exposure offset only for the log link.

The log link gives a positive mean for every linear predictor. The square-root and identity links do not: the square-root link requires a positive linear predictor, and the identity link requires a positive mean, for every observation. A fit that does not meet these requirements, or does not converge, is not used. During the FP power search such a candidate is skipped and reported in a warning, as for other candidates that cannot be fitted. If the null, linear, or final model cannot be fitted, model fitting stops with an error. The link is never changed, and fitted values are never adjusted, to make a fit succeed. Positive covariates, such as those with a positive `shift`, help keep these links valid.

Negative-binomial models require the suggested **fastglm** package (version 0.1.1 or later). Generalized estimating equations via `gee_family()` do not support negative-binomial responses.

Value

A family specification to supply as the `family` argument of `mfp2()` or `mfpi()`.

See Also

`mfp2()`, `mfpi()`, `fastglm::fastglm_nb()`

Examples

```
negbin_family()
negbin_family(link = "sqrt")

if (requireNamespace("fastglm", quietly = TRUE) &&
    "fastglm_nb" %in% getNamespaceExports("fastglm")) {
  set.seed(45)
  n <- 200
  count_data <- data.frame(x = stats::runif(n, 1, 4))
  count_data$y <- stats::rnbinom(
    n,
    mu = with(count_data, (0.8 + 0.6 * x)^2),
```

```
  size = 3
)

fit_sqrt <- mfp2(
  y ~ fp(x, df = 1),
  data = count_data,
  family = negbin_family(link = "sqrt"),
  verbose = FALSE
)
predict(fit_sqrt, newdata = count_data[1:3, ], type = "response")
}
```

`ordinal_family` *Ordinal Family*

Description

Use `ordinal_family()` with `mfp2()` or `mfpi()` when the outcome has categories with a meaningful order, such as low, medium, and high. The default `"logistic"` link gives a proportional-odds model. The optional **rms** package (version 7.0-0 or later) is required.

Usage

```
ordinal_family(link = c("logistic", "probit", "loglog", "cloglog", "cauchit"))
```

Arguments

`link`	Relationship used to model the ordered outcome. Choose `"logistic"` (default), `"probit"`, `"loglog"`, `"cloglog"`, or `"cauchit"`.

Details

Supply the outcome as an ordered factor to specify the category order. Numeric outcomes are ordered from smallest to largest. Character outcomes and unordered factors are ordered alphabetically; `mfp2()` reports the order it has assumed. If alphabetical order is not the intended order, convert the outcome to an ordered factor before fitting.

A positive predictor coefficient indicates an association with higher outcome categories. Optional case weights must be finite and strictly positive. They are used without normalisation in selection and final fitting. BIC uses the number of fitted rows as its sample size.

Value

A family specification to supply as the `family` argument of `mfp2()` or `mfpi()`.

See Also

`mfp2()`, `mfpi()`

Examples

```
ordinal_family()
ordinal_family(link = "probit")


if (requireNamespace("rms", quietly = TRUE) &&
    utils::packageVersion("rms") >= "7.0-0") {
  set.seed(44)
  n <- 180
  ordinal_data <- data.frame(
    x1 = stats::rnorm(n),
    x2 = stats::rbinom(n, 1, 0.5)
  )
  latent <- with(ordinal_data, 0.6 * x1 + 0.5 * x2 + stats::rlogis(n))
  ordinal_data$y <- ordered(
    cut(
      latent,
      breaks = stats::quantile(latent, probs = seq(0, 1, length.out = 5)),
      include.lowest = TRUE,
      labels = c("low", "medium", "high", "very high")
    )
  )

  fit_ordinal <- mfp2(
    y ~ x1 + x2,
    data = ordinal_data,
    family = ordinal_family(link = "logistic"),
    df = 1,
    select = 1,
    cycles = 1,
    verbose = FALSE
  )
  predict(
    fit_ordinal,
    newdata = ordinal_data[1:3, ],
    type = "response"
  )
}
```

`pima` *Pima Indians dataset used in Royston and Sauerbrei (2008) book.*

Description

The dataset arises from an investigation of potential predictors of the onset of diabetes in a cohort of 768 female Pima Indians of whom 268 developed diabetes. Missing values were imputed using the ice procedure for Stata.

Usage

```
data(pima)
```

Format

A dataset with 768 observations and 10 variables.

id Patient identifier.

pregnant Number of times pregnant.

glucose Plasma glucose concentration at 2h in an oral glucose tolerance test.

diastolic Diastolic blood pressure in mmHg.

triceps Triceps skin fold thickness in mm.

insulin 2-h serum insulin.

bmi Body mass index.

diabetes Diabetes pedigree function.

age Age in years.

y Binary outcome variable (diabetes, yes/no).

Examples

```
data(pima)
str(pima)
```

---

plot.mfp2 *Plot fitted term contributions and contrasts from an mfp2 model*

Description

Shows how selected variables contribute to the fitted model, with confidence intervals. Term plots can also include component-plus-residual points to help assess the fitted relationship.

Usage

```
## S3 method for class 'mfp2'
plot(
  x,
  terms = NULL,
  partial_only = FALSE,
  type = c("terms", "contrasts"),
  ref = NULL,
  terms_seq = c("data", "equidistant"),
  alpha = 0.05,
  show_titles = TRUE,
  show_grid = FALSE,
  color_points = "#AAAAAA",
```

```
  color_line = "black",
  color_line_spike = "black",
  color_fill = "#000000",
  shape = 16,
  size_points = 1,
  size_points_spike = 2,
  linetype = "solid",
  linewidth = 1,
  alpha_fill = 0.1,
  ...
)
```

Arguments

x	A fitted mfp2 model.
terms	Names of the variables to plot, for example, c("age", "cavol"). By default (NULL), requests all variables kept in the final model. See **Choosing terms**.
partial_only	Whether to show only the fitted effect and confidence interval. Defaults to FALSE. See **Component-plus-residual plots**.
type	Either "terms" (default) or "contrasts". See **Choosing the plot type**.
ref	Named list of reference values for contrast plots, for example, list(age = 50). See **Contrasts and reference values**.
terms_seq	Either "data" or "equidistant". See **Evaluation values** for their meaning and defaults.
alpha	Significance level for confidence intervals. The default, 0.05, gives 95 percent intervals. See **Confidence intervals**.
show_titles	Whether to show titles describing the fitted transformation or variable representation. Defaults to TRUE.
show_grid	Whether to show the major and minor panel grid lines of ggplot2::theme_bw(). Defaults to FALSE, which draws plots without grid lines.
color_points	Colour of component-plus-residual points, for example, "grey60" or "#AAAAAA". All colour arguments accept ggplot2 colour names or hexadecimal codes. See ggplot2::aes_colour_fill_alpha.
color_line	Colour of fitted curves and fitted points for binary predictors and factors, for example, "black" or "blue". Also used for the zero endpoint of a term fitted with zero = TRUE alone.
color_line_spike	Colour of the separate zero endpoint for a catzero or spike-at-zero term with a continuous component, for example, "red".
color_fill	Fill colour of confidence bands, for example, "lightblue" or "#56B4E9".
shape	Shape of component-plus-residual points. Common ggplot2 shape codes include 1 for open circles and 16 for solid circles. The default is 16. See ggplot2::scale_shape().
size_points	Numeric size of component-plus-residual points, in millimetres, for example, 1 or 2. The default is 1. See ggplot2::geom_point().

`size_points_spike`
: Numeric size of fitted points for binary predictors, factors, and zero endpoints. For example, use `3` for larger points. The default is `2`. See `ggplot2::geom_point()`.

`linetype`
: Line style of fitted curves, for example, `"solid"`, `"dashed"`, or `"dotted"`. The default is `"solid"`. See `ggplot2::scale_linetype()`.

`linewidth`
: Numeric width of fitted curves, for example, `0.5` or `1`. Larger values produce thicker lines. The default is `1`. See `ggplot2::geom_line()`.

`alpha_fill`
: Confidence-band transparency, from `0` (transparent) to `1` (opaque).

`...`
: Additional arguments are not used. Supplying `newdata` produces an error; other arguments are ignored with a warning.

Value

A named list of `ggplot2` objects, one per plotted variable. Individual plots can be printed by name or position and modified using `ggplot2`.

Choosing the plot type

Use `type` to choose what the fitted curve or fitted points represent:

- `"terms"` (default) shows a variable's fitted contribution, using the relationship selected during fitting.
- `"contrasts"` shows the difference from a reference value. See **Contrasts and reference values**.

Fitted contributions use the following model scales:

- Gaussian with an identity link: contributions to the mean response, in the same units as the response used for fitting. For example, fitting log-PSA gives contributions to mean log-PSA.
- Binomial with a logit link: log-odds.
- Poisson and negative-binomial with a log link: log mean.
- Negative-binomial with a square-root link: square root of the mean; with an identity link: the mean.
- Other GLM and GEE models: the scale defined by the fitted link function.
- Cox: log relative hazard.
- Fine–Gray: log relative subdistribution hazard.
- `survreg`: the fitted location scale, such as log time for Weibull models.

Term plots include the intercept for GLM, negative-binomial, GEE, and `survreg` models. Cox and Fine–Gray terms have no intercept. Offsets are excluded from fitted term contributions. See **Multinomial and ordinal models** for their scales, panels, and intercepts.

Choosing terms

Use `terms` to select variables kept in the final model. Use their original names: `"age"` rather than `"fp(age)"`, and `"race"` rather than individual contrast-column names such as `"raceB"` and `"raceC"`.

For example, for a model containing age, cancer volume, and the factor race:

```
# Plot age only.
age_plots <- plot(fit, terms = "age")
age_plots[["age"]]

# Create separate plots and combine them using patchwork, if installed.
plots <- plot(fit, terms = c("age", "cavol"))
patchwork::wrap_plots(plots, ncol = 2)

# Show all fitted race levels together.
race_plots <- plot(fit, terms = "race")
race_plots[["race"]]
```

Evaluation values

Use `terms_seq` to choose where a numeric variable's fitted effect is calculated:

- `"data"` uses the observed predictor values in the data used to fit the model.
- `"equidistant"` uses equally spaced values between the smallest and largest observed predictor values.

The default is `"data"` for term plots and `"equidistant"` for contrast plots. Both settings use the same fitted model.

```
# Evaluate the curve at observed ages.
plots <- plot(fit, terms = "age", terms_seq = "data")

# Evaluate it at equally spaced ages over the same range.
curves <- plot(
  fit, terms = "age", terms_seq = "equidistant", partial_only = TRUE
)
curves[["age"]]
```

See **Binary and categorical predictors** for discrete variables. To obtain estimates at specific new values, use `predict.mfp2()` with `newdata` and `terms_seq = "data"`, then plot its output. An example is included below.

Contrasts and reference values

A contrast is:
`contribution at the plotted value - contribution at the reference.`

For example, compare each age with age 50:

```
plots <- plot(
  fit, terms = "age", type = "contrasts", ref = list(age = 50)
)
plots[["age"]]
```

The contrast is zero at age 50. Above zero, the fitted contribution is larger than at age 50; below zero, it is smaller. Intercepts cancel from the comparison.

The interpretation follows the model scale: a Gaussian identity-link contrast is a difference in mean response; a binomial logit contrast is a log odds ratio; and a log-link count-model contrast is a log ratio of mean counts. Cox and Fine–Gray contrasts are log hazard ratios and log subdistribution hazard ratios, respectively. These comparisons hold the other predictors and any offset fixed and, where applicable, use the same stratum.

Supply references on the original variable scale. The fitted shift, transformation, and centering are applied automatically. For example, if age was fitted using `log(age + 10)`, supply `ref = list(age = 50)`, not `list(age = 60)` or `list(age = log(60))`.

For a formula factor, use a category label available during fitting. For example, if race has categories A, B, and C, compare each with race A:

```
plots <- plot(
  fit, terms = "race", type = "contrasts", ref = list(race = "A")
)
plots[["race"]]
```

If a reference is omitted, the function uses:

- the smaller value for a numeric predictor with two distinct values;
- the observed mean in the fitting data for other numeric predictors;
- the first fitted level for a formula factor.

See **Contrasts and reference values** in `predict.mfp2()` for further details.

Component-plus-residual plots

By default, term plots add one component-plus-residual point for each observation used to fit the model. Each point has:

- a horizontal position given by the observed predictor value;
- a height equal to the fitted contribution at that value plus a residual from the complete model.

These points leave out the fitted contributions of the other predictors, while keeping the plotted predictor's contribution and the residual from the complete model. This helps assess the fitted relationship while accounting for the other predictors. Residuals are calculated from the complete fitted model, including any offsets.

The type of residual added depends on the model:

- GLM, negative-binomial, and `survreg`: deviance residuals.
- GEE: deviance residuals based on fitted marginal means and prior weights.

- Cox: martingale residuals.
- Fine–Gray: martingale residuals from the weighted Cox fit, summed over each subject's expanded rows.
- Multinomial: binary deviance residuals comparing each non-reference category with all other categories. This differs from the comparison with the reference category shown by the fitted curve.
- Ordinal: deviance residuals for the cutoff shown in each panel.

For non-Gaussian models, these points provide a diagnostic display rather than an exact adjustment of the observed response for other predictors. Points always use observed predictor values, even when the fitted curve uses `terms_seq = "equidistant"`.

Set `partial_only = TRUE` to hide the component-plus-residual points:

```
curves <- plot(fit, terms = "age", partial_only = TRUE)
curves[["age"]]
```

Contrast plots always omit component-plus-residual points.

Confidence intervals

Confidence bands and vertical intervals describe uncertainty in the fitted contribution or contrast, rather than the spread of component-plus-residual points. They use the estimated uncertainty in the final model's coefficients and apply to each plotted value individually. They are not prediction intervals for future outcomes.

The intervals treat the selected model as fixed. They do not account for uncertainty from selecting variables, FP powers, or spike-at-zero components. For GEE models, they use the covariance selected for the fit.

Multinomial and ordinal models

Multinomial models return one plot per variable, with a panel for each non-reference category. A panel labelled `"B vs A"` shows the fitted contribution to the log-odds of category B versus reference category A, including that comparison's intercept for term plots. The fitted transformation is shared across panels, but the coefficients differ. Contrasts are shown separately for each category comparison.

Ordinal term plots have one panel per response cutoff. For ordered categories low, medium, and high, the panels compare medium/high with low, and high with low/medium. Each curve and confidence interval includes its cutoff intercept. The default logistic link uses cumulative log-odds; other ordinal links use their corresponding link scale.

Ordinal contrast plots show the shared predictor effect without cutoff intercepts and therefore use a single panel.

Binary and categorical predictors

Binary predictors and formula factors are displayed using fitted points and vertical confidence intervals rather than a continuous curve. Numeric binary predictors use their observed values on the horizontal axis; formula factors use their fitted category labels.

For numeric predictors with only two values in multinomial or ordinal models, use `terms_seq = "data"` to avoid displaying intermediate values.

Predictors containing zeros

For a term fitted with `zero = TRUE`, the continuous function is defined for positive values. Its curve and confidence band are drawn only above zero, with the fitted value at exactly zero shown as a separate point. Centering can make that contribution nonzero. This point comes from the continuous term and does not represent a separately estimated zero effect.

A `catzero` term also includes a separately estimated zero indicator. A spike-at-zero term follows the representation selected in the final model:

- a positive-value function with a zero indicator;
- the positive-value function alone;
- the zero indicator alone.

When a continuous component is retained, the positive-value curve is not connected to the zero endpoint. For `catzero` and spike-at-zero terms, that endpoint uses `color_line_spike`; for `zero = TRUE` alone, it uses `color_line`.

A binary-only spike term uses labels such as `"exposure = 0"` and `"exposure > 0"`, with fitted points in `color_line`.

See Also

`predict.mfp2()`, `mfp2()`

Examples

```
## Gaussian models: terms and contrasts
data("prostate")
fit_gaussian <- mfp2(
  lpsa ~ fp(age) + fp(cavol) + fp(weight) + svi,
  data = prostate,
  keep = c("age", "cavol"),
  verbose = FALSE
)

# Plotting requires ggplot2.
if (requireNamespace("ggplot2", quietly = TRUE)) {
  # Term contributions with component-plus-residual points.
  plots <- plot(fit_gaussian, terms = c("age", "cavol"))
  print(plots[["age"]])

  # Show only the fitted contribution and its confidence band.
  curves <- plot(fit_gaussian, terms = "age", partial_only = TRUE)
  print(curves[["age"]])

  # Compare each plotted age with age 50, on the original age scale.
  contrasts <- plot(
    fit_gaussian,
    terms = "age",
    type = "contrasts",
    ref = list(age = 50)
  )
```

```
  print(contrasts[["age"]])

  # Each element is a ggplot2 object and can be modified.
  print(curves[["age"]] + ggplot2::labs(x = "Age (years)"))
}

## Categorical predictors: use the factor name
fit_factor <- mfp2(
  Sepal.Length ~ Species,
  data = iris,
  keep = "Species",
  verbose = FALSE
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(fit_factor, terms = "Species", partial_only = TRUE)[["Species"]]
}

## A predictor with a spike at zero
set.seed(1)
d_spike <- data.frame(exposure = c(rep(0, 35), runif(85, 0.1, 8)))
d_spike$y <- 2 * (d_spike$exposure > 0) + log1p(d_spike$exposure) +
  rnorm(120)
fit_spike <- mfp2(
  y ~ fp(exposure, spike = TRUE),
  data = d_spike,
  keep = "exposure",
  verbose = FALSE
)
# Zero is shown separately from the curve for positive values.
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(fit_spike, terms = "exposure", partial_only = TRUE)[["exposure"]]
}

## Cox models: contributions to the log relative hazard
data("gbsg")
fit_cox <- mfp2(
  survival::Surv(rectime, censrec) ~ fp(age) + fp(nodes),
  data = gbsg,
  family = "cox",
  verbose = FALSE
)
# Points add martingale residuals to the fitted contribution.
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(fit_cox, terms = "nodes")[["nodes"]]
}
```

---

plot.mfpi *Plot MFPI Fitted Functions and Group Differences*

Description

Plot group-specific fitted functions and differences between groups for variables tested for interaction by `mfpi()`. Continuous predictors are shown as curves; binary and categorical predictors are shown as points at each fitted level.

Usage

```
## S3 method for class 'mfpi'
plot(
  x,
  terms = NULL,
  plot_type = c("both", "fitted", "difference"),
  auto_print = TRUE,
  show_title = TRUE,
  show_grid = TRUE,
  show_rug = FALSE,
  show_ci_fitted = FALSE,
  show_ci_diff = TRUE,
  show_null_line = FALSE,
  show_maineffect_line = TRUE,
  show_ci_maineffect = FALSE,
  legend_position = c("inside", "right", "left", "bottom", "top", "none"),
  legend_inside = c(0.02, 0.98),
  legend_justification = c(0, 1),
  colour_ref = "#2166AC",
  colour_group = "#D6604D",
  line_types = NULL,
  colour_diff = "#1B7837",
  colour_null = "grey50",
  colour_maineffect = "grey20",
  linewidth = 1,
  ribbon_alpha = 0.2,
  rug_alpha = 0.4,
  ...
)
```

Arguments

`x`	An object of class `"mfpi"` returned by `mfpi()`.
`terms`	An optional character vector naming variables tested for interaction. The default is `NULL`, which plots selected interactions only. Specify a variable to plot its interaction even if it was not selected.
`plot_type`	A character string specifying the plots to display: `"fitted"` for group-specific functions, `"difference"` for the comparison-group function minus the reference-group function, or `"both"` for side-by-side plots. Default: `"both"`, which requires the optional **patchwork** package.
`auto_print`	A logical value indicating whether to display plots automatically. Default: `TRUE`. Use `FALSE` to return the plots without displaying them.

show_title A logical value indicating whether to display titles and subtitles. Default: TRUE.

show_grid A logical value indicating whether to display panel grid lines. Default: TRUE. Use FALSE to remove them.

show_rug A logical value indicating whether to show observed continuous predictor values as rug marks along the horizontal axis. Default: FALSE.

show_ci_fitted A logical value indicating whether to display confidence intervals for group-specific functions. Default: FALSE.

show_ci_diff A logical value indicating whether to display confidence intervals for group differences. Default: TRUE.

show_null_line A logical value indicating whether to add a horizontal line at zero to difference plots. Default: FALSE.

show_maineffect_line
A logical value indicating whether to add the constant group difference estimated by the model without interaction to difference plots. Default: TRUE. The line is omitted if that model is unavailable.

show_ci_maineffect
A logical value indicating whether to display a confidence band around the main-effects reference line. Requires show_maineffect_line = TRUE and an available standard error. Default: FALSE.

legend_position
A character string specifying the legend position: "inside" (default), "right", "left", "bottom", "top", or "none".

legend_inside Two numeric values giving the horizontal and vertical position of an inside legend, between 0 and 1. The bottom-left corner is c(0, 0) and the top-right corner is c(1, 1). Default: c(0.02, 0.98).

legend_justification
Two numeric values specifying which part of an inside legend is anchored at legend_inside. Default: c(0, 1), which anchors the legend at its top-left corner.

colour_ref Colour of the reference-group curve, points, and confidence intervals. Default: "#2166AC".

colour_group Colour of the comparison-group curve, points, and confidence intervals. Default: "#D6604D".

line_types An optional character vector specifying curve styles. For example, c("solid", "dashed") gives the reference group a solid line and the comparison group a dashed line. Alternatively, assign styles to group labels, for example c(Placebo = "solid", Treatment = "dashed"), using the labels in your data. Default: NULL, which makes all lines solid.

colour_diff Colour of the difference curve, points, and confidence intervals. Default: "#1B7837".

colour_null Colour of the zero reference line. Default: "grey50".

colour_maineffect
Colour of the main-effects reference line. Default: "grey20".

linewidth A positive number controlling curve width. Default: 1.

ribbon_alpha Confidence-band transparency, from 0 (transparent) to 1 (opaque). Default: 0.2.

rug_alpha Rug-mark transparency, from 0 (transparent) to 1 (opaque). Default: 0.4.

... Not used. Supplied arguments generate a warning.

Value

Invisibly returns a named list with one element per plotted interaction variable. Each element contains a list of plots comparing individual groups with the reference group.

For multinomial models, each variable's results contain an additional list level for the non-reference response categories.

Individual plots are **ggplot2** objects. With `plot_type = "both"`, each group comparison contains a combined **patchwork** plot. Returned plots can be modified or saved using `ggplot2::ggsave()`. Use `auto_print = FALSE` to obtain them without displaying them.

An empty list is returned if no interactions were selected and `terms = NULL`.

Interpreting the plots

Each variable is plotted using its separate fitted interaction model. The reference group is the one used during fitting. With more than two groups, a separate plot is produced for each comparison with the reference. Multinomial models also produce separate plots for each non-reference response category.

Fitted functions include the predictor contribution, group main effects, and model intercept where applicable. They exclude contributions from other adjustment variables and offsets. Group differences include both the group main-effect difference and differences between the predictor functions (see Kipruto, Royston and Sauerbrei (2026) for details).

Plots use the model's linear-predictor scale. For example, Gaussian models with an identity link use the outcome scale, and binomial models with a logit link use the log-odds scale. See `predict.mfpi()` for interpretation by model family.

Confidence intervals use a fixed confidence level of 95%. They are pointwise intervals, shown as shaded bands for continuous predictors and error bars for binary and categorical predictors. They are conditional on the fitted model and do not account for uncertainty from selecting variables, transformations, or interactions.

Reference lines

Difference plots can show two horizontal reference lines.

The zero line indicates no difference between the groups. Add it using `show_null_line = TRUE`.

The main-effects line shows the group difference estimated by the model without interaction. That model assumes the predictor has the same effect in every group, so the group difference is constant across predictor values. This line is shown by default; use `show_maineffect_line = FALSE` to hide it. Use `show_ci_maineffect = TRUE` to add its 95% confidence band.

References

Kipruto, E., Royston, P. and Sauerbrei, W. (2026). Multivariable Fractional Polynomial Interaction Models in R. *Journal of Statistical Software*.

See Also

`mfpi()`, `predict.mfpi()`, `summary.mfpi()`

Examples

```
data("prostate")

# Include the grouping variable in the formula and name it in group_var.
fit <- mfpi(
  lpsa ~ fp(age) + svi + fp(pgg45) + fp(cavol) + fp(weight) +
    fp(bph) + fp(cp),
  data = prostate,
  family = gaussian(),
  group_var = "svi",
  interaction_vars = c("cavol", "age"),
  flex = "flex1",
  p_interact = 0.9,
  include_group_var = TRUE,
  center = FALSE,
  verbose = FALSE
)

# Plotting requires ggplot2.
if (requireNamespace("ggplot2", quietly = TRUE)) {
  # Plot an interaction even if it was not selected.
  plot(fit, terms = "cavol", plot_type = "difference")

  # Plot group-specific functions with confidence bands.
  plot(
    fit, terms = "cavol", plot_type = "fitted",
    show_ci_fitted = TRUE
  )

  # Return plots without displaying them.
  plots <- plot(
    fit, terms = "cavol", plot_type = "difference",
    auto_print = FALSE
  )
  print(plots$cavol[[1]])
}


# Side-by-side plots require patchwork and ggplot2.
if (requireNamespace("patchwork", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  plot(fit, terms = "cavol", plot_type = "both")
}
```

`predict.mfp2` *Predict from an* `mfp2` *Model*

Description

Obtain predictions from a model fitted with `mfp2()`. The function can return predictions from the complete model, term-specific fitted values, or differences from chosen reference values.

Usage

```
## S3 method for class 'mfp2'
predict(
  object,
  newdata = NULL,
  type = NULL,
  se.fit = FALSE,
  terms = NULL,
  terms_seq = c("equidistant", "data"),
  alpha = 0.05,
  ref = NULL,
  strata = NULL,
  newoffset = NULL,
  times = NULL,
  nseq = 100,
  add_intercept = TRUE,
  cox_reference = NULL,
  ordinal_cutoff = NULL,
  ...
)
```

Arguments

`object`	A fitted object of class `"mfp2"`.
`newdata`	An optional data frame or matrix containing observations to predict. See **Supplying prediction data**.
`type`	A character value selecting the prediction type. See **Choosing the prediction type**.
`se.fit`	Logical; return standard errors for complete-model predictions. Not available for multinomial class labels. Ignored for term and contrast predictions, which always include standard errors and confidence limits. See **Choosing the prediction type**.
`terms`	An optional character vector naming terms to return with `type = "terms"` or `type = "contrasts"`. Use original variable or group names. If `NULL`, all terms included in the final model are used.
`terms_seq`	For term and contrast predictions, choose which values to predict at. `"equidistant"` (the default) uses `nseq` evenly spaced values between the variable's minimum and maximum. `"data"` uses the observed values. Both use `newdata` when supplied, otherwise the fitting data. Factors and grouped terms always use the supplied or fitted rows.

`alpha` A single finite number strictly between 0 and 1 used to calculate confidence intervals for term and contrast results. The confidence level is `1 - alpha`. The default `alpha = 0.05` gives 95 percent confidence intervals.

`ref` A named list of reference values used with `type = "contrasts"`. Numeric references must be supplied on the original variable scale. See **Contrasts and reference values**.

`strata` Strata for new rows of a Cox or `survreg` model fitted with separately supplied strata. Supply a vector for one stratification variable, or a matrix or data frame for several. For formula strata, include the variables in `newdata` instead. See **Proportional-hazards predictions** and **Parametric survival predictions**.

`newoffset` Offsets for new rows: one finite numeric value per row, or, for multinomial models, a matrix with one column per category or non-reference category. See **Offsets**.

`times` Time points for Fine–Gray cumulative-incidence predictions, in the same units as the fitted follow-up time. Required for `type = "response"` in Fine–Gray models; unavailable otherwise. See **Proportional-hazards predictions**.

`nseq` A single finite positive integer giving the number of equally spaced values used when `terms_seq = "equidistant"`. The default is 100.

`add_intercept` Logical; defaults to `TRUE`. Includes the fitted intercept in term predictions where applicable and the reference-cutoff intercept in full-model ordinal link predictions. See **Choosing the prediction type**.

`cox_reference` Reference for Cox or Fine–Gray `"lp"` and `"risk"` predictions: `"zero"` (default), `"sample"`, or `"strata"`. See **Proportional-hazards predictions**.

`ordinal_cutoff` Cutoff number for ordinal term predictions. Includes the chosen cutoff's intercept and accounts for its uncertainty. The default, `NULL`, omits cutoff intercepts. See **Choosing the prediction type**.

`...` Additional arguments for the underlying prediction method, such as `p = 0.5` for `survreg` quantiles. Internal prediction methods ignore these arguments; negative-binomial predictions reject unknown arguments. See **Choosing the prediction type**.

Details

Supply predictors using their original values. The fitted shifts, FP or ACD functions, centering, zero handling, and spike-at-zero decisions are applied automatically.

Do not create fractional-polynomial basis columns, centered columns, or structural-zero indicator columns yourself.

Standard errors and confidence intervals are conditional on the final fitted model. They do not include uncertainty from variable selection or from choosing the fitted function.

Term and contrast predictions require enough valid covariance information to calculate their standard errors and confidence intervals. Negligible negative variances caused by numerical rounding are treated as zero. The same checks apply to full-model GEE, fastglm, ordinal, and multinomial standard errors. Prediction stops if a variance is non-finite or meaningfully negative and a valid interval cannot be calculated.

Value

Complete-model predictions usually return a numeric vector, or a list containing predictions and standard errors when `se.fit = TRUE`. Where a native prediction method is used, its return format is retained.

GEE predictions with `se.fit = TRUE` return a list containing `fit`, `se.fit`, and `residual.scale`.

Multinomial link and response predictions return matrices; class predictions return a factor. Ordinal link and mean predictions return numeric vectors; response predictions return a matrix with one column per ordered category. For these numeric predictions, `se.fit = TRUE` returns a list containing `fit` and `se.fit` with matching dimensions and observation names. See **Choosing the prediction type**.

Term and contrast predictions return a named list with one data frame per requested term, containing:

- `variable`: the evaluated value or factor level;
- `variable_pre`: for a single numeric term, the shifted value before the final transformation; for an indicator-only spike term, the zero indicator;
- `value`: the fitted term value or contrast;
- `se`: its standard error;
- `lower`, `upper`: its confidence limits.

Grouped terms may also include their member columns. Multinomial term and contrast results include a `logit` column identifying the non-reference category.

Supplying prediction data

Supply `newdata` with one row for each observation to predict.

If you fitted the model using the formula interface, supply the variables named in the formula. For example, for a model fitted with `y ~ age + race`, prediction data could be:

```
newdata <- data.frame(
  age = c(50, 60, 70),
  race = c("A", "B", "C")
)
```

Supply `race` itself; you do not need to create separate columns for its categories. The race categories `"A"`, `"B"`, and `"C"` must have been available during fitting.

If you fitted the model using the matrix interface, supply the same column names and use the same coding as in the fitting matrix. For example, if race had three categories during fitting, with race category A as the reference category, the fitting matrix could contain `age`, `raceB`, and `raceC`. The same prediction data would then be:

```
newdata <- cbind(
  age   = c(50, 60, 70),
  raceB = c(0, 1, 0),
  raceC = c(0, 0, 1)
)
```

The first row represents an observation aged 50 in race category A; the second, an observation aged 60 in race category B; and the third, an observation aged 70 in race category C. Race category A has zero in both `raceB` and `raceC`.

If you grouped these columns during fitting with `term_groups = list(race = c("raceB", "raceC"))`, include both `raceB` and `raceC` when predicting the group. In `terms`, use the group name `"race"`, rather than the column names. For example:

```
predict(fit, newdata = newdata, type = "terms", terms = "race")
```

For predictions from the whole model, include all variables kept in the final model. For term or contrast predictions, include only the variables needed for the requested terms.

Extra columns may be needed for offsets or survival groups (`strata`) specified in the formula. For Cox `type = "expected"` or `"survival"`, also include the time and event-status variables in `newdata`, using the names from the fitting formula. The time values specify when predictions are evaluated.

Values needed for prediction must not be missing or contain `Inf` or `-Inf`. Extra columns that are not used are ignored.

Supply values on their original scale; fitted shifts and transformations are applied automatically. Prediction stops if a transformation cannot use a supplied value. For example, a logarithm requires a positive value after the fitted shift is added.

Choosing the prediction type

The default is `"lp"` for Cox and Fine–Gray models and `"link"` for all other models. `"link"` and `"lp"` are interchangeable.

For GLM (including negative-binomial) and GEE models:

- `type = "link"` returns the linear predictor.
- `type = "response"` returns the predicted mean on the response scale.

For GEE models, standard errors use the covariance estimator chosen during fitting. Response-scale standard errors are approximated from those on the link scale.

For parametric `survreg` models:

- `type = "link"` returns the fitted location (linear predictor). `"linear"` is also accepted.
- `type = "response"` returns the fitted location transformed back to the response scale. This is not generally the mean survival time.
- `type = "quantile"` returns predicted quantiles on the response scale.
- `type = "uquantile"` returns predicted quantiles on the linear-predictor scale.

For `"quantile"` and `"uquantile"`, supply probabilities using `p`, for example, `p = 0.5` for the median. See `survival::predict.survreg()`.

For Cox models:

- `type = "lp"` returns the relative log-hazard.
- `type = "risk"` returns the relative hazard, calculated as `exp(lp)`.

- type = "expected" returns the predicted cumulative hazard up to the supplied follow-up time.
- type = "survival" returns the survival probability at that time, calculated as exp(-expected).

For Fine–Gray models:

- type = "lp" returns the relative log subdistribution hazard.
- type = "risk" returns the relative subdistribution hazard, calculated as exp(lp).
- type = "response" returns cumulative-incidence probabilities at the times supplied in times.

For ordinal models:

- type = "link" returns the linear predictor including the cutoff intercept selected by rms::orm(). Set add_intercept = FALSE to return only the predictor effects and any offset.
- type = "response" returns category probabilities, with one row per observation and one column per category.
- type = "mean" returns the expected category value. Numeric category labels are used when possible; otherwise, categories are scored 1, ..., K, where K is the number of categories.

For multinomial models:

- type = "link" returns log-odds relative to the reference category, with one row per observation and one column per non-reference category.
- type = "response" returns category probabilities, with one row per observation and one column per category.
- type = "class" returns the category with the highest predicted probability.

Use se.fit = TRUE to also obtain standard errors for complete-model numeric predictions. Ordinal probability and mean standard errors and multinomial probability standard errors use a first-order approximation (the delta method). Multinomial class labels have no standard error; use type = "response" to obtain uncertainty for their probabilities.

For all supported models:

- type = "terms" returns each requested term's contribution on the linear-predictor scale.
- type = "contrasts" returns the difference between each term's contribution and its contribution at a reference value.

Term predictions include the intercept by default for GLM, GEE, survreg, and multinomial models. Set add_intercept = FALSE to omit it. Cox and Fine–Gray terms have no intercept. Ordinal terms omit cutoff intercepts unless ordinal_cutoff is supplied. Intercepts cancel from contrast predictions.

Term and contrast predictions include standard errors and confidence intervals. For multinomial models, each term's result contains separate predictions for each non-reference category, identified by the logit column.

Term predictions

A single term result can represent an FP2 function, a factor, a zero-handled variable, or several matrix columns grouped with `term_groups`. Contributions from all parts of the term are combined in that result.

Use the original variable or group name in `terms`. Use `get_selected_variable_names()` to see the terms included in the final model. A requested term that is not in the final model is omitted with a warning.

For a single numeric term:

- `terms_seq = "equidistant"` uses `nseq` equally spaced values between the smallest and largest available values.
- `terms_seq = "data"` uses the values in `newdata`, or the fitted values when `newdata = NULL`.

Factors and other grouped terms are evaluated row by row.

For GLM (including negative-binomial), GEE, and `survreg` models, `add_intercept = TRUE` includes the fitted intercept in each term result; for multinomial models it includes each logit's own intercept. Set `add_intercept = FALSE` to return only the contribution from the requested term. This argument does not affect Cox, Fine–Gray, or contrast predictions.

Ordinal term predictions without `ordinal_cutoff` return the common term contribution and calculate its standard error from the covariance of that term's coefficients. Selecting a cutoff adds that cutoff's fitted intercept and includes its variance and covariance with the term coefficients in the standard error. Ordinal contrasts cancel the intercept at every cutoff.

Contrasts and reference values

Use `type = "contrasts"` to compare a term's contribution at different values with its contribution at a chosen reference value. Each contrast is calculated as:
`term contribution at the prediction value - term contribution at the reference`.

The contrast is zero at the reference value. A positive contrast means a larger contribution than at the reference; a negative contrast means a smaller contribution. Differences are reported on the model's linear-predictor scale, such as log-odds for logistic regression.

Example using the formula interface

For a model that includes `age` and the factor `race`, compare ages 50, 60, and 70 with age 50, and race categories A, B, and C with race category A:

```
newdata <- data.frame(
  age = c(50, 60, 70),
  race = c("A", "B", "C")
)

predict(
  fit_formula,
  newdata = newdata,
  type = "contrasts",
  terms = c("age", "race"),
  terms_seq = "data",
```

```
  ref = list(age = 50, race = "A")
)
```

The result contains separate age and race comparisons. The age contrasts compare each age with 50; the race contrasts compare each category with A. The contrast for age 50 is zero, as is the contrast for race category A. For example, if the fitted age contributions at ages 60 and 50 are 0.8 and 0.3, the contrast at age 60 is `0.8 - 0.3 = 0.5`.

Supply references as a named list: each name identifies a term, and its value specifies the reference. For formula factors, use a category label that was available during fitting, such as `"A"`.

Numeric references

Supply numeric references on the original variable scale. Any fitted shift, FP or ACD transformation, and centering are applied automatically. For example, if the fitted function uses `log(age + 10)`, supply `ref = list(age = 50)`, not 60 or `log(60)`.

References for grouped matrix columns

If race was represented by `raceB` and `raceC`, with race category A as the reference category, and grouped during fitting using `term_groups = list(race = c("raceB", "raceC"))`, compare with race category A by supplying both column values as zero:

```
newdata <- cbind(
  raceB = c(0, 1, 0),
  raceC = c(0, 0, 1)
)

predict(
  fit_matrix,
  newdata = newdata,
  type = "contrasts",
  terms = "race",
  ref = list(race = c(raceB = 0, raceC = 0))
)
```

The rows compare race categories A, B, and C with race category A. For comparison with race category B instead, use `ref = list(race = c(raceB = 1, raceC = 0))`.

For other grouped terms, including formula interactions, supply one numeric reference value for every member column. Use the column names supplied in `term_groups`, or those shown in the term prediction result.

Default references

If a reference is not supplied for a term, the function uses:

- the smaller value for a numeric term with two distinct values;
- the mean for another single numeric term;
- the first fitted category for a formula factor;
- the first available row for another grouped term.

Numeric and grouped-row defaults use `newdata` when supplied, otherwise the fitting data. Formula factors always use their first fitted category. Supply `ref` explicitly to keep the same comparison when `newdata` changes.

The model intercept cancels from the comparison and is not included in contrasts.

Zero and spike-at-zero variables

Supply zero-handled and spike-at-zero variables on their original scale. Do not create a separate zero-indicator column.

The zero handling selected during fitting is reused during prediction. These options require nonnegative covariates: `x = 0` belongs to the zero component and `x > 0` belongs to the positive component. Negative values in `newdata` are rejected; recode them explicitly before prediction if they should scientifically represent the zero group.

If the final spike-at-zero model includes both a continuous component and a zero indicator, both are used automatically. If it includes only one component, only that component is used.

The same rule is applied to a supplied reference. For a term that includes only the zero indicator, an exact-zero reference is treated as indicator 1 and a positive reference as indicator 0. Negative references are rejected.

Offsets

Offsets are included in predictions from the whole model. They are not included in term or contrast predictions.

When `newdata = NULL`, the offsets used during fitting are reused. For new observations, provide offsets as described below.

Offset in the formula

If the fitting formula contains `offset(...)`, include the variables needed to calculate it in `newdata`. The offset is calculated automatically. For example, for a model fitted with `count ~ age + offset(log(exposure))`:

```
newdata <- data.frame(
  age = c(50, 60),
  exposure = c(1, 2)
)

predict(fit_formula, newdata = newdata, type = "response")
```

Here, `log(exposure)` is calculated separately for each new observation.

Offset supplied separately

If the offset was supplied through the `offset` argument during fitting, supply the new offset values through `newoffset`. For example, if the fitting offset was `log(exposure)`:

```
predict(
  fit_offset,
  newdata = newdata["age"],
  newoffset = log(newdata$exposure),
  type = "response"
)
```

Supply the calculated offset, such as `log(exposure)`, rather than `exposure` itself. An explicit `newoffset` overrides a formula offset; the variables used only by that formula offset can then be omitted.

For most models, supply one finite `newoffset` value per prediction row. For example, `newoffset = c(0.1, 0.2)` supplies offsets for two observations. `newoffset` requires `newdata` and a model fitted with an offset.

Multinomial offsets

For multinomial models, offsets must form a matrix with one row per observation. Columns can represent all response categories or only the non-reference categories. Values are offsets on the logit scale.

A formula offset must produce such a matrix. For example, for response categories A, B, and C, with A as the reference category:

```
fit_multinomial <- mfp2(
  outcome ~ age + offset(cbind(A = off_A, B = off_B, C = off_C)),
  data = training,
  family = multinomial_family(reference = "A")
)

newdata <- data.frame(
  age   = c(50, 60),
  off_A = c(0, 0),
  off_B = c(0.2, 0.4),
  off_C = c(-0.1, 0.3)
)

predict(fit_multinomial, newdata = newdata, type = "response")
```

The offset matrix is calculated automatically from the columns in `newdata`. The offset for reference category A is subtracted from those for categories B and C.

Alternatively, supply the matrix directly through `newoffset`. This overrides the formula offset, so only `age` is needed in `newdata`:

```
newoffset <- cbind(
  A = newdata$off_A,
  B = newdata$off_B,
  C = newdata$off_C
)

predict(
  fit_multinomial,
  newdata = newdata["age"],
  newoffset = newoffset,
  type = "response"
)
```

Because the offsets for reference category A are zero here, supplying only columns B and C gives the same predictions:

```
predict(
  fit_multinomial,
  newdata = newdata["age"],
  newoffset = newoffset[, c("B", "C"), drop = FALSE],
  type = "response"
)
```

With columns for only the non-reference categories, supply offsets relative to the reference category. Name the columns using the corresponding response-category labels. The same matrix formats apply when the offset was supplied separately during fitting.

Proportional-hazards predictions

Cox and Fine–Gray `type = "lp"` predictions are relative log-hazards; `type = "risk"` returns their exponentials. For Fine–Gray models, these refer to the subdistribution hazard. Neither type returns an absolute hazard or a survival probability.

Reference for relative predictions

Use `cox_reference` with `"lp"` or `"risk"`:

- `"zero"` uses zero on the fitted predictor scale and is the default.
- `"sample"` uses the average fitted predictor values in the fitting sample.
- `"strata"` uses separate averages within each fitted stratum.

For example:

```
predict(
  fit_cox,
  newdata = data.frame(age = c(50, 60)),
  type = "risk",
  cox_reference = "sample"
)
```

Changing the reference changes the reported log-hazards and risk scores. Ratios of risk scores remain unchanged when a common reference is used. With `"strata"`, make comparisons within the same stratum. `cox_reference` is used only with `"lp"` and `"risk"`.

Cox survival probabilities and cumulative hazards

Use `type = "survival"` for survival probabilities and `type = "expected"` for cumulative hazards. When supplying `newdata`, also include the variables used in the fitted `Surv()` response. The time values specify when predictions are evaluated.

For example, suppose the model was fitted with `Surv(time, event) ~ age`, with time measured in days and event coded 0/1. To predict survival at 365 days:

```
newdata <- data.frame(
  age = c(50, 60),
  time = c(365, 365),
  event = c(0, 0)
)

predict(fit_cox, newdata = newdata, type = "survival")
```

The event-status column is required to construct the `Surv()` response, but its values do not affect these predictions. Use valid status values. With `type = "expected"`, the same data give cumulative hazards at 365 days.

Alternatively, include exactly one `Surv` column:

```
newdata <- data.frame(
  age = c(50, 60),
  followup = I(survival::Surv(c(365, 365), c(0, 0)))
)
```

For matrix-interface fits, supply follow-up information this way. When `newdata = NULL`, the follow-up times used during fitting are reused.

Stratified models

Each observation must belong to a stratum available during fitting. If the formula contains `strata(...)`, include its variables in `newdata`. For example, for a model fitted with `Surv(time, event) ~ age + strata(sex)`:

```
newdata <- data.frame(
  age = c(50, 60),
  sex = c("F", "M")
)

predict(fit_stratified, newdata = newdata, type = "risk")
```

Here, `"F"` and `"M"` must match fitted stratum labels. Do not also supply the `strata` argument for formula-level strata.

If strata were supplied separately during matrix-interface fitting, supply them separately for the new observations:

```
predict(
  fit_matrix,
  newdata = as.matrix(newdata["age"]),
  strata = newdata$sex,
  type = "risk"
)
```

Strata are matched by their labels, not by numeric positions. For several stratification variables, supply the same variables in the same column order used during fitting. New strata are not allowed.

Fine–Gray cumulative incidence

Use `type = "response"` with `times` to predict cumulative incidence. Supply times in the same units used during fitting. For example:

```
predict(
  fit_finegray,
  newdata = data.frame(age = c(50, 60)),
  type = "response",
  times = c(365, 730)
)
```

See Also

mfp2(), get_selected_variable_names(), stats::predict.glm(), survival::predict.coxph(), geepack::geeglm()

Examples

```
## Gaussian models: complete predictions, terms, and contrasts
data("prostate")

fit_gaussian <- mfp2(
  lpsa ~ fp(age) + fp(cavol) + fp(weight) + svi,
  data = prostate,
  family = "gaussian",
  keep = c("age", "cavol"),
  verbose = FALSE
)

# Predictions for the fitting data.
predict(fit_gaussian)

# Predictions and standard errors for five rows.
predict(
  fit_gaussian,
  newdata = prostate[1:5, ],
  type = "response",
  se.fit = TRUE
)

# For this Gaussian model, the response is lpsa.
# These predictions are therefore on the log-PSA scale.

# Term contributions, without the model intercept.
predict(
  fit_gaussian,
  type = "terms",
  terms = c("age", "cavol"),
  add_intercept = FALSE
)

# Evaluate exactly these ages, rather than an equally spaced grid.
# Only age is needed for an age term prediction.
predict(
  fit_gaussian,
  newdata = data.frame(age = c(40, 50, 60)),
  type = "contrasts",
  terms = "age",
  terms_seq = "data",
  ref = list(age = 50)
)

# References use the original variable scale, even for fitted FP functions.
predict(
```

```
  fit_gaussian,
  type = "contrasts",
  terms = c("age", "cavol"),
  ref = list(age = 50, cavol = 1)
)

## Gaussian models: a predictor with a spike at zero
set.seed(1)
d_spike <- data.frame(
  exposure = c(rep(0, 35), runif(85, 0.1, 8))
)
d_spike$y <- with(
  d_spike,
  2 * (exposure > 0) + log1p(exposure) + rnorm(120)
)

fit_spike <- mfp2(
  y ~ fp(exposure, spike = TRUE),
  data = d_spike,
  family = "gaussian",
  keep = "exposure",
  verbose = FALSE
)

# Compare exactly 0, 1, and 4 with exposure = 0.
predict(
  fit_spike,
  newdata = data.frame(exposure = c(0, 1, 4)),
  type = "contrasts",
  terms = "exposure",
  terms_seq = "data",
  ref = list(exposure = 0)
)

## Poisson models: formula offsets and separately supplied offsets
set.seed(2)
d_count <- data.frame(
  age = runif(80, 30, 75),
  exposure = runif(80, 0.5, 3)
)
d_count$count <- rpois(
  80,
  lambda = with(d_count, exposure * exp(0.02 * (age - 50)))
)

fit_poisson <- mfp2(
  count ~ fp(age) + offset(log(exposure)),
  data = d_count,
  family = "poisson",
  keep = "age",
  verbose = FALSE
)
```

```
# The formula calculates log(exposure) from the new rows.
nd_count <- d_count[1:4, c("age", "exposure")]
predict(fit_poisson, newdata = nd_count, type = "response")

# Response predictions are expected counts for the supplied exposure.
# Set exposure to 1 to obtain expected counts per unit of exposure.
nd_unit <- nd_count
nd_unit$exposure <- 1
predict(fit_poisson, newdata = nd_unit, type = "response")

fit_external_offset <- mfp2(
  count ~ fp(age),
  data = d_count,
  family = "poisson",
  offset = log(d_count$exposure),
  keep = "age",
  verbose = FALSE
)

# A separately supplied offset must be supplied again for new rows.
predict(
  fit_external_offset,
  newdata = nd_count["age"],
  newoffset = log(nd_count$exposure),
  type = "response"
)

## GEE models: predictions and standard errors (require geepack)
if (requireNamespace("geepack", quietly = TRUE)) {
  # Illustrate groups of four observations.
  d_count$id <- rep(seq_len(20), each = 4)
  fit_gee <- mfp2(
    count ~ fp(age) + offset(log(exposure)),
    data = d_count,
    family = gee_family(poisson(), corstr = "independence"),
    id = id,
    keep = "age",
    verbose = FALSE
  )

  # New group identifiers are not needed for these mean predictions.
  print(predict(
    fit_gee,
    newdata = nd_count,
    type = "response",
    se.fit = TRUE
  ))
}

## Multinomial models: category probabilities and matrix offsets
d_multi <- iris
d_multi$off_A <- 0.1
d_multi$off_B <- 0.05 * d_multi$Sepal.Width
```

```
d_multi$off_C <- -0.05 * d_multi$Sepal.Width

fit_multinomial <- mfp2(
  Species ~ fp(Sepal.Length, df = 1, select = 1) +
    offset(cbind(
      setosa = off_A,
      versicolor = off_B,
      virginica = off_C
    )),
  data = d_multi,
  family = multinomial_family(reference = "setosa"),
  verbose = FALSE
)

nd_multi <- d_multi[
  c(1, 51, 101),
  c("Sepal.Length", "off_A", "off_B", "off_C")
]

# The formula rebuilds the offset matrix from these columns.
predict(
  fit_multinomial,
  newdata = nd_multi,
  type = "response",
  se.fit = TRUE
)

# Equivalent offsets for only the non-reference categories.
# Subtract the setosa offset; do not simply drop its column.
relative_offsets <- with(
  nd_multi,
  cbind(
    versicolor = off_B - off_A,
    virginica = off_C - off_A
  )
)

# Explicit newoffset replaces the formula offset, so its source columns
# are no longer needed in newdata.
predict(
  fit_multinomial,
  newdata = nd_multi["Sepal.Length"],
  newoffset = relative_offsets,
  type = "response"
)

# Class predictions return labels; do not request their standard errors.
predict(fit_multinomial, newdata = nd_multi, type = "class")

## Ordinal models: category probabilities and cutoff intercepts
if (requireNamespace("rms", quietly = TRUE) &&
    utils::packageVersion("rms") >= "7.0-0") {
  set.seed(3)
```

```
  d_ordinal <- data.frame(age = runif(150, 30, 75))
  d_ordinal$severity <- cut(
    0.04 * (d_ordinal$age - 50) + rlogis(150),
    breaks = c(-Inf, -0.5, 0.5, Inf),
    labels = c("low", "medium", "high"),
    ordered_result = TRUE
  )

  fit_ordinal <- mfp2(
    severity ~ fp(age),
    data = d_ordinal,
    family = ordinal_family(),
    df = 1,
    select = 1,
    verbose = FALSE
  )

  nd_ordinal <- data.frame(age = c(40, 50, 60))
  print(predict(
    fit_ordinal,
    newdata = nd_ordinal,
    type = "response",
    se.fit = TRUE
  ))

  # The common predictor without a cutoff intercept.
  print(predict(
    fit_ordinal,
    newdata = nd_ordinal,
    type = "link",
    add_intercept = FALSE
  ))

  # Include cutoff 1: medium/high versus low.
  print(predict(
    fit_ordinal,
    newdata = nd_ordinal,
    type = "terms",
    terms = "age",
    terms_seq = "data",
    ordinal_cutoff = 1
  ))
}

## Cox models: relative risk and survival at a chosen time
data("gbsg")
fit_cox <- mfp2(
  survival::Surv(rectime, censrec) ~
    fp(age) + fp(nodes) + strata(grade),
  data = gbsg,
  family = "cox",
  df = 1,
  select = 1,
```

```
  verbose = FALSE
)

# Relative predictions need predictors and the strata variable.
nd_cox <- gbsg[1:3, c("age", "nodes", "grade")]
predict(fit_cox, newdata = nd_cox, type = "lp")
predict(
  fit_cox,
  newdata = nd_cox,
  type = "risk",
  cox_reference = "strata"
)

# Predict survival at 365 days for each profile.
# Use the response-variable names from the fitted formula.
nd_survival <- nd_cox
nd_survival$rectime <- 365
nd_survival$censrec <- 0L

# censrec is required to reconstruct Surv(); its value is a placeholder
# here, not knowledge of whether a future event will occur.
predict(fit_cox, newdata = nd_survival, type = "survival")

# "expected" returns cumulative hazard, not an event probability.
predict(fit_cox, newdata = nd_survival, type = "expected")

## Parametric survival models: predicted survival-time quantiles
fit_survreg <- mfp2(
  survival::Surv(rectime, censrec) ~ fp(age) + fp(nodes),
  data = gbsg,
  family = survreg_family(dist = "weibull"),
  df = 1,
  select = 1,
  verbose = FALSE
)

# Median survival time, in days. New follow-up values are not needed.
predict(
  fit_survreg,
  newdata = gbsg[1:3, c("age", "nodes")],
  type = "quantile",
  p = 0.5
)

## Fine--Gray models: cumulative incidence at chosen times
set.seed(4)
d_fg <- data.frame(
  time = rexp(180, rate = 1 / 1000),
  age = runif(180, 30, 75),
  event = factor(
    sample(c("censored", "relapse", "death"), 180, replace = TRUE),
    levels = c("censored", "relapse", "death")
  )
```

```
)

# The first event level represents censoring.
fit_finegray <- mfp2(
  survival::Surv(time, event) ~ fp(age, select = 1),
  data = d_fg,
  family = finegray_family(etype = "relapse"),
  verbose = FALSE
)

# Probability of relapse by 365 and 730 days, allowing for competing death.
# Unlike Cox survival prediction, supply times through the times argument.
predict(
  fit_finegray,
  newdata = data.frame(age = c(40, 50, 60)),
  type = "response",
  times = c(365, 730)
)
```

`predict.mfpi` *Predict from an MFPI Model*

Description

Obtain group-specific fitted functions, differences between groups, or predictions for individual observations from a model fitted by `mfpi()`.

Usage

```
## S3 method for class 'mfpi'
predict(
  object,
  newdata = NULL,
  terms = NULL,
  model = c("best", "all"),
  type = NULL,
  se.fit = TRUE,
  level = 0.95,
  grid = FALSE,
  n_grid = 200L,
  strata = NULL,
  cox_reference = NULL,
  newoffset = NULL,
  ...
)
```

Arguments

object	An object of class "mfpi" returned by mfpi().
newdata	An optional data frame or matrix containing prediction values. For fitted functions and group differences, supply the requested interaction variables. For individual predictions, also supply the grouping variable and required adjustment variables. The default is NULL, which uses the fitting data.
terms	An optional character vector naming variables previously tested for interaction. The default is NULL, which uses all variables available under the chosen model setting.
model	A character string specifying which interaction models to use: "best" (default) for selected interactions or "all" for all available fitted interactions, including those not selected.
type	A character string specifying the prediction type. Use "function", "difference", or "both" for fitted functions and group comparisons. For individual predictions, available types depend on the model family; see *Prediction types*. The default is NULL, which means "both".
se.fit	A logical value indicating whether to return standard errors. Default: TRUE. Availability for individual predictions depends on the family and prediction type; see *Uncertainty*.
level	The confidence level for fitted-function and group-difference intervals, greater than 0 and less than 1. Default: 0.95.
grid	A logical value indicating whether to evaluate continuous fitted functions on an equally spaced grid. Default: FALSE. Ignored with a warning for individual predictions.
n_grid	The number of grid points when grid = TRUE. An integer of at least 2; default: 200. For variables with structural zeros, zero may be included as an additional point.
strata	Optional stratum values for individual survival predictions when strata were supplied through the strata argument in the default method. For formula-based fits using strata(...), include the original stratification variables in newdata instead.
cox_reference	A character string specifying the covariate reference for Cox "lp" and "risk" predictions: "zero", "sample", or "strata". The default is NULL, which uses "zero". See *Cox predictions*.
newoffset	Optional replacement offsets for individual predictions. Supply one value per prediction row, or an appropriate offset matrix for multinomial models. Available only for models fitted with an offset; see *Offsets*.
...	Not used. Supplied arguments generate a warning.

Details

Each variable tested for interaction has a separate fitted model. Predictions for different variables therefore come from different models.

Value

For one term, an object of class "mfpi_prediction". For several terms, an object of class "mfpi_prediction_list": a named list containing one prediction object per term.

Each prediction object identifies the requested variable in term and the prediction type in type. Its main results are:

- functions: for "function" or "both", a data frame containing term, x, group, and fit. With se.fit = TRUE, it also contains se.fit, lower, and upper.
- differences: for "difference" or "both", a data frame containing term, x, contrast, group, reference, and fit. Here fit is the comparison-group function minus the reference-group function. With se.fit = TRUE, it also contains se.fit, lower, and upper.
- predictions: for individual prediction types, a data frame with one row per observation, containing term, fit, and, when requested and available, se.fit. Predictions for several response categories use separate fit.<class> and se.fit.<class> columns.

Multinomial functions and differences also contain a class column identifying the non-reference response category.

Selecting interaction models

Use terms to name variables previously tested for interaction by mfpi(). Adjustment variables cannot be requested unless they were also tested for interaction.

With model = "best", predictions are restricted to interactions selected during fitting. Use model = "all" to obtain predictions for an interaction that was not selected. Specifying terms does not override this restriction. If no interactions were selected, use model = "all".

When terms = NULL, predictions are returned for all variables available under the chosen model setting. An error is raised if no suitable fitted models are available.

Each requested variable uses its own fitted interaction model and associated adjustment terms.

Prediction types

For fitted functions and group comparisons, use:

- "function": the fitted function for each group.
- "difference": each comparison-group function minus the reference-group function at the same predictor values.
- "both": fitted functions and group differences.

The default, type = NULL, means "both", including when newdata is supplied. Binary and categorical predictors produce fitted values at their levels rather than continuous curves.

For predictions for individual observations, choose a type supported by the fitted family:

- GLMs: "link" returns the linear predictor; "response" applies the inverse link to obtain predictions on the response scale.
- Parametric survival models: "link" returns the fitted location; "response" applies the distribution's response transformation.

- Cox models: `"lp"` returns the relative log-hazard; `"risk"` returns its exponential; `"expected"` returns the cumulative hazard up to the supplied follow-up time; `"survival"` returns the survival probability at that time.
- Ordinal models: `"link"` returns the common covariate linear predictor without cut-point intercepts; `"response"` returns category probabilities; `"mean"` returns the expected category value. Numeric category labels are used as values when possible; otherwise scores `1, ..., K` are used.
- Multinomial models: `"link"` returns logits relative to the reference response category; `"response"` returns probabilities for all classes.

`"lp"` is accepted as an alias for `"link"` in non-Cox models, and `"link"` is accepted as an alias for `"lp"` in Cox models.

Interpreting fitted functions and differences

A fitted function includes the contribution of the variable tested for interaction, the group main effect, and the model intercept where applicable. It is estimated from the adjusted model but excludes the contributions of other adjustment variables and offsets.

Individual predictions include the required adjustment-variable contributions and any fitted offset.

A group difference is the comparison-group function minus the reference-group function at the same predictor value. It includes the group main-effect difference and the difference between the predictor functions. The reference group is the one used during fitting. With more than two groups, each other group is compared with this reference.

Fitted functions and differences are returned on the linear-predictor scale:

- Gaussian models with an identity link: outcome scale.
- Binomial models with a logit link: log-odds scale.
- Poisson and negative-binomial models with a log link: log of the expected count.
- Negative-binomial models with a square-root link: square root of the expected count; with an identity link: the expected count.
- Gamma models with an inverse link: reciprocal of the expected outcome.
- Inverse-Gaussian models with an inverse-squared link: reciprocal of the squared expected outcome.
- Ordinal models: common cumulative-link predictor scale, excluding cut-point intercepts.
- Multinomial models: log-odds relative to the reference response category, with separate results for each non-reference category.
- Parametric survival models: fitted location scale.
- Cox models: partial log-hazard scale, excluding the baseline hazard.

For GLMs using other links, results are on the scale of the chosen link function.

Supplying prediction data

The variables required in `newdata` depend on the prediction type.

For `"function"`, `"difference"`, or `"both"`, supply only the requested interaction variables. Each function is evaluated for every fitted group, so the grouping variable is not required.

For individual predictions, supply the interaction variable, grouping variable, and adjustment variables required by the relevant interaction model. Also supply any required offset variables, stratification variables, or follow-up information. The required adjustment variables can differ between interaction models.

For formula-based fits, use a data frame containing the original variables. Factor coding and required formula expressions are reconstructed automatically. For default-method fits, use the same column names and coding as during fitting. If `term_groups` was used, supply all dummy columns belonging to each required categorical term.

Supply continuous predictors on their original scale and binary or categorical predictors using the values or levels used during fitting. The fitted transformations, centring values, zero handling, and Winsorisation limits are applied automatically.

Predictors fitted with `zero`, `catzero`, or `spike` handling must not contain negative values. Zero identifies the zero component, and positive values identify the positive component. This requirement also applies when only the binary component of a spike-at-zero term was retained.

Required values must not be missing, and numeric values must be finite. New group or factor levels are not supported.

When `newdata = NULL`, predictions use the observations used for fitting.

Evaluation values and grids

For continuous fitted functions, `grid = FALSE` uses the values supplied in `newdata`, or the fitting values when `newdata = NULL`. With `grid = TRUE`, `n_grid` equally spaced values are generated over the corresponding range.

For variables with structural zeros, the grid covers the positive values. Zero is added separately if it occurs in the supplied or fitting values, giving `n_grid + 1` points. At least one positive value is needed to construct this grid.

Binary and categorical predictors use all fitted levels when `newdata = NULL` or `grid = TRUE`. With supplied `newdata` and `grid = FALSE`, they use the supplied values or levels. `n_grid` does not affect binary or categorical predictors.

Fitted Winsorisation limits are applied before constructing continuous grids. The returned `x` values are on the original predictor scale, after any Winsorisation.

`grid` does not apply to individual predictions.

Offsets

Offsets apply only to individual predictions. Do not supply `newoffset` for fitted functions or group differences.

With `newdata = NULL`, fitted offsets are reused unless replaced by `newoffset`.

For new observations from a formula-based fit, include the variables needed to evaluate the fitted `offset(...)` expression. Alternatively, supply `newoffset`, which overrides the formula offset. For offsets supplied separately through the default method, use `newoffset` for new observations.

Replacement offsets must be finite numeric values, with one value per prediction row. For multinomial models, supply a matrix with one row per observation and either one column per response category or one per non-reference category, following the fitted category convention.

`newoffset` is accepted only when the interaction model was fitted with an offset.

Cox predictions

`"lp"` and `"risk"` give relative predictions. Use `cox_reference` to choose the covariate reference:

- `"zero"`: zero on the fitted predictor scale; the default.
- `"sample"`: predictor means from the fitting sample.
- `"strata"`: predictor means within each fitted stratum.

Zero on the fitted predictor scale does not necessarily correspond to zero values of the original covariates because predictors may have been transformed or centred.

For the same reference, `risk = exp(lp)`. Changing `cox_reference` does not change the reference group used for fitted-function differences.

`"expected"` and `"survival"` also require follow-up information for each new observation. For formula-based fits, include the variables used in the fitted `Surv()` response. Alternatively, include exactly one column containing a right-censored `survival::Surv()` object, for example: `data.frame(age, group, followup = I(survival::Surv(time, event)))`.

Do not supply `cox_reference` for `"expected"` or `"survival"`, or for fitted functions and group differences.

Stratified models

Stratum information is required for individual predictions from stratified survival models.

If `strata(...)` was included in the fitted formula, include its original variables in `newdata`. Do not also supply the `strata` argument.

If strata were supplied separately through the default method, use `strata` to provide values for new observations. For one stratification variable, supply a vector with one value per prediction row. For several variables, supply a matrix or data frame in the same variable order used for fitting.

Stratum values are matched to those observed during fitting. Previously unseen strata are rejected. With `newdata = NULL`, fitted strata are reused unless replaced where supported.

Fitted functions and group differences do not require stratum information. Do not supply the `strata` argument for these prediction types.

Uncertainty

For fitted functions and group differences, `se.fit = TRUE` returns pointwise standard errors and normal-approximation confidence intervals at the requested `level`. These intervals describe uncertainty in the fitted functions.

For individual predictions, standard-error availability follows the prediction method used for the fitted family. Standard errors are available for multinomial logits and probabilities and for ordinal `"link"` predictions. Ordinal `"response"` and `"mean"` predictions return no standard errors and issue a warning if `se.fit = TRUE`; use `se.fit = FALSE` for these types.

`level` controls only fitted-function and group-difference intervals. Individual predictions return standard errors where supported, without confidence intervals.

All uncertainty estimates are conditional on the fitted model. They do not account for uncertainty from selecting variables, transformations, or interactions.

References

Kipruto, E., Royston, P. and Sauerbrei, W. (2026). Multivariable Fractional Polynomial Interaction Models in R. *Journal of Statistical Software*.

See Also

`mfpi()`, `plot.mfpi()`, `summary.mfpi()`

Examples

```
data("prostate")

# Include the grouping variable in the formula and name it in group_var.
fit <- mfpi(
  lpsa ~ fp(age) + svi + fp(pgg45) + fp(cavol) + fp(weight) +
    fp(bph) + fp(cp),
  data = prostate,
  family = gaussian(),
  group_var = "svi",
  interaction_vars = c("cavol", "age"),
  flex = "flex1",
  include_group_var = TRUE,
  center = FALSE,
  verbose = FALSE
)

# Fitted functions and differences, including unselected interactions.
curves <- predict(
  fit, terms = "cavol", model = "all", grid = TRUE
)
curves$functions
curves$differences

# Evaluate fitted functions at specified predictor values.
predict(
  fit,
  newdata = data.frame(cavol = c(0.2, 1, 3)),
  terms = "cavol",
  model = "all",
  type = "function"
)

# Obtain one response prediction per observation.
rows <- predict(
  fit,
  newdata = prostate[1:5, ],
```

```
  terms = "cavol",
  model = "all",
  type = "response"
)
rows$predictions

# Predictions for several terms come from separate fitted models.
several <- predict(
  fit,
  terms = c("cavol", "age"),
  model = "all",
  type = "function",
  grid = TRUE
)
several$cavol$functions
several$age$functions


data("advanced_prostate_cancer")
d <- advanced_prostate_cancer
d$rx <- factor(d$rx, levels = c(0, 1),
               labels = c("Placebo", "Treatment"))

fit_cox <- mfpi(
  survival::Surv(survtime, cens) ~
    rx + fp(age) + fp(wt) + fp(hg) + stage + strata(pf),
  data = d,
  family = "cox",
  group_var = "rx",
  interaction_vars = "age",
  interaction_forms = c(age = "linear"),
  flex = "flex1",
  select = 1,
  verbose = FALSE
)

# Group differences require only the variable tested for interaction.
predict(
  fit_cox,
  newdata = data.frame(age = c(50, 60, 70)),
  terms = "age",
  model = "all",
  type = "difference"
)

# Relative hazards use the full predictor profile and stratum information.
predict(
  fit_cox,
  newdata = d[1:5, ],
  terms = "age",
  model = "all",
  type = "risk",
  cox_reference = "strata"
```

```
)

# Survival probabilities also use the follow-up variables in newdata.
predict(
  fit_cox,
  newdata = d[1:5, ],
  terms = "age",
  model = "all",
  type = "survival"
)
```

`print.mfp2` *Print a Fitted MFP Model*

Description

Displays the choices made by `mfp2()` and the resulting fitted model. Use `print(fit)` for the full report, or set `detailed_settings = FALSE` for a shorter view.

Usage

```
## S3 method for class 'mfp2'
print(x, detailed_settings = TRUE, notes = TRUE, intercepts = NULL, ...)
```

Arguments

`x`	A fitted `mfp2` object.
`detailed_settings`	Show the Detailed Settings table. Default `TRUE`; set to `FALSE` for a shorter report.
`notes`	Show explanatory notes beneath the settings and model-fit sections. Default `TRUE`.
`intercepts`	Show ordinal threshold intercepts. By default (`NULL`), they are shown when there are fewer than 10 thresholds; use `TRUE` or `FALSE` to override this. Has no effect for other model families.
`...`	Additional print arguments. Use `digits` to choose the number of decimal places for displayed values (default 3); FP powers and degrees of freedom retain their own formatting.

Details

The report shows the original call and model details, selected and excluded variables, preprocessing, selected functions, final coefficients with standard errors, and model-fit statistics. Preprocessing shifts and scales display without unnecessary trailing zeros, independently of the `digits` print setting.

Function choices are grouped by type: standard MFP, approximate cumulative distribution (ACD), and spike-at-zero (SAZ). Selected variables appear before excluded variables in each table. In the SAZ table, `prop_zero` is the proportion of fitting observations with a structural zero; `binary` denotes an indicator for `x = 0`. For SAZ variables, the continuous component is transformed only when `x > 0`.

The optional Detailed Settings table shows each variable's initial and final degrees of freedom, selection settings, and transformation flags. The coefficients table labels each fitted basis term and its standard error. If the fit was centered, it also displays the centering constant and explains how to read the estimate.

Model-fit measures depend on the family. GLMs report deviances, Cox models report minus twice the partial log likelihood, and other families display their applicable statistics. Multinomial fits show one coefficient block per non-reference outcome. Ordinal fits may include threshold intercepts.

Value

Invisibly returns `x`, allowing the fitted object to be reused.

See Also

`mfp2()`, `summary.mfp2()`

Examples

```
data("prostate")
fit <- mfp2(lpsa ~ fp(age) + svi, data = prostate,
            select = 1, alpha = 1, verbose = FALSE)
print(fit, detailed_settings = FALSE)
```

---

print.mfpi *Print an "mfpi" Object*

Description

Displays the adjustment model, criterion-specific interaction model metrics, detailed group-level interaction FP powers, and the final interaction decision summary. The printed candidate table shows the metrics relevant to the selected criterion and the main-effect FP powers. Interaction FP powers are omitted from that table because the complete group-level mapping is printed immediately below it. Full diagnostics remain available in the returned object.

Usage

```
## S3 method for class 'mfpi'
print(x, digits = NULL, ...)
```

Arguments

x	An object of class "mfpi", as returned by mfpi().
digits	Optional non-negative integer controlling the number of decimal places used when printing numeric output other than FP powers and degrees of freedom. If NULL, x$digits is used when available; otherwise 3 is used.
...	Currently unused.

Details

The header reports the original call and fitted model type. Step 1 displays df_initial and df_final. For a grouped adjustment term, df_initial counts all member design columns in the initial model. Step 2 reports the grouping variable, interaction-selection criterion and rule, and the p-value adjustment method when applicable. The candidate table retains the stored metric names except that pvalue is printed as p-value. The group-level power table uses the display labels Group and Group level.

The final interaction summary reports selection explicitly as Yes or No without repeating the rule already shown in Step 2. With no multiplicity adjustment it prints one p-value column; otherwise it prints p_raw and p_adjusted.

Numeric values use fixed decimal places for display only, except FP powers and degrees of freedom, which retain their natural notation. The underlying metrics stored in the "mfpi" object are not modified, and model-selection decisions are based on the unrounded values computed during fitting.

Value

x invisibly.

See Also

mfpi(), summary.mfpi()

---

print.mfpi_coef *Print MFPI Coefficients*

Description

Prints coefficients returned by coef.mfpi() in a user-facing form. For each interaction model, the selected interaction powers are shown for every group level, followed by separate tables for group-specific FP terms, group-variable coefficients, and adjustment coefficients. These methods are called automatically and are not normally invoked directly.

Usage

```
## S3 method for class 'mfpi_coef'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'mfpi_coef_list'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object returned by coef.mfpi().
digits	Number of significant digits used for coefficient estimates.
...	Not used.

Value

Invisibly returns x.

---

print.mfpi_prediction *Print an MFPI Prediction*

Description

Prints the user-facing prediction results from an "mfpi_prediction" object without exposing internal prediction metadata by default.

Usage

```
## S3 method for class 'mfpi_prediction'
print(x, digits = NULL, ...)
```

Arguments

x	An object of class "mfpi_prediction", as returned by predict.mfpi().
digits	Optional non-negative integer controlling printed numeric precision. If NULL, 3 digits are used.
...	Currently unused.

Details

For fitted-function predictions, the displayed curves are group-specific partial fitted functions on the linear-predictor scale. Partial fitted functions comprise the intercept (when present), group main effect, and group-specific FP function estimated from the adjusted interaction model.

The stored prediction object is not modified. In particular, the underlying result tables retain the programmatic column names fit, se.fit, and x; these are printed as estimate, SE, and the actual term name (for example, age) only for readability. Internal components such as metadata remain accessible explicitly from the returned object.

Value

x invisibly.

print.mfpi_prediction_list *Print a List of MFPI Predictions*

Description

Prints each term-specific `"mfpi_prediction"` in an `"mfpi_prediction_list"` using the compact prediction print method.

Usage

```
## S3 method for class 'mfpi_prediction_list'
print(x, digits = NULL, ...)
```

Arguments

`x`	An object of class `"mfpi_prediction_list"`.
`digits`	Optional non-negative integer controlling printed numeric precision. If `NULL`, 3 digits are used.
`...`	Currently unused.

Value

`x` invisibly.

print.mfpi_vcov_list *Print a Collection of MFPI Variance-Covariance Matrices*

Description

Prints a compact index when `vcov.mfpi()` returns covariance matrices for several term-specific interaction models. Individual matrices can be displayed by requesting a single `term`.

Usage

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

Arguments

`x`	A multi-model object returned by `vcov.mfpi()`.
`...`	Not used.

Value

Invisibly returns `x`.

print.summary.mfp2 *Print a Summary of an* mfp2 *Model Fit*

Description

Renders the structured summary produced by summary.mfp2().

Usage

```
## S3 method for class 'summary.mfp2'
print(x, notes = x$notes, ...)
```

Arguments

x	An object of class "summary.mfp2".
notes	Logical. If FALSE, suppress the explanatory model-df note. Defaults to the value stored by summary.mfp2().
...	Not used.

Value

Invisibly returns x.

See Also

summary.mfp2(), print.mfp2()

print.summary.mfpi *Print a* "summary.mfpi" *Object*

Description

Displays the four-step MFPI summary produced by summary.mfpi().

Usage

```
## S3 method for class 'summary.mfpi'
print(x, digits = NULL, ...)
```

Arguments

x	An object of class "summary.mfpi".
digits	Optional non-negative integer controlling displayed decimal places other than fractional-polynomial powers and degrees of freedom. If NULL, x$digits is used when available; otherwise the MFPI print-helper default is used.
...	Currently unused.

Details

Steps 1–3 are delegated to `print.mfpi()` so that `print(fit)` and `print(summary(fit))` use the same adjustment-model and interaction- selection output. Step 4 adds a compact regression display for each retained interaction model.

Step 4 uses readable group labels and FP transformations rather than internal design-column names. The grouping variable is shown in Step 2 and the FLEX setting in the main MFPI header, so neither is repeated for every retained interaction. Each retained interaction is introduced by an underlined heading. Interaction powers are always printed by group level, including when the selected FLEX strategy gives all groups the same powers. Group-variable coefficients and adjustment coefficients are displayed in separate tables. For models with an intercept, it is displayed separately. Centered terms include a `Center` column and use the `Basis - Center` convention. Multinomial models use a separate outcome-versus-reference block for each non-reference outcome.

When one or more retained interaction variables have a nonzero MFPI shift, Step 4 prints one clarification immediately below the first shifted interaction's group-specific FP table. The constant shown inside an FP transformation is the shifting factor applied to all observations before the group-specific FP terms are constructed; structural zeros remain untransformed. Later shifted interactions retain the explicit constant in their transformation labels, but the note is not repeated. No shift note is printed when all retained interactions have zero shift.

The original `mfpi()` call is printed with Steps 1–3. Step 4 does not repeat the calls of the underlying retained regression models or print exponentiated-coefficient tables, significance stars, concordance, or global likelihood-ratio/Wald/score tests. Those quantities remain available from the stored fitted regression model when required, but they are not part of the MFPI interaction-selection result.

Value

`x` invisibly.

See Also

`summary.mfpi()`, `print.mfpi()`, `coef.mfpi()`, `vcov.mfpi()`

`prostate` *Prostate cancer dataset used in Royston and Sauerbrei (2008) book.*

Description

Prostate cancer dataset used in Royston and Sauerbrei (2008) book.

Usage

```
data(prostate)
```

Format

A dataset with 97 observations and 9 variables.

obsno Observation number.

age Age in years.

svi Seminal vesicle invasion (yes/no).

pgg45 Percentage Gleason score 4 or 5.

cavol Cancer volume (ml).

weight Prostate weight (g).

bph Amount of benign prostatic hyperplasia (g).

cp Amount of capsular penetration (g).

lpsa Log PSA concentration (outcome variable).

Examples

```
data(prostate)
str(prostate)
```

---

residuals.mfp2 *Extract Residuals from an* mfp2 *Model*

Description

Returns residuals from a fitted `mfp2()` model. For GEE and multinomial models, `type` can be `"deviance"` (the default), `"pearson"`, `"working"`, or `"response"`.

Usage

```
## S3 method for class 'mfp2'
residuals(object, ...)
```

Arguments

`object`	A fitted `mfp2()` model.
`...`	For GEE and multinomial models, use `type` to select `"deviance"` (the default), `"pearson"`, `"working"`, or `"response"` residuals. For other models, `type` and any other arguments follow the residual method for that model's family; see, for example, `stats::residuals.glm()` for GLM models.

Details

For GEE models, residuals are based on the marginal fitted means. Deviance residuals are calculated from the response family's deviance function and the model's prior weights. Pearson, working, and response residuals are obtained from the underlying GEE fit.

For multinomial models, residuals are calculated separately for each non-reference category. Each category is treated as a binary outcome, using its fitted class probability. The result has one column per non-reference category.

For other models, residual calculation is passed to the underlying fit, including its choice of default residual type.

Value

A numeric vector for GEE and other scalar-response models, or a numeric matrix with one row per observation and one column per non-reference category for multinomial models.

See Also

`mfp2()`, `plot()`, `stats::residuals()`

Examples

```
data("prostate")
fit <- mfp2(
  lpsa ~ fp(age) + fp(cavol) + fp(weight) + svi,
  data = prostate,
  verbose = FALSE
)
head(residuals(fit))
```

---

summary.mfp2 *Summarize an* mfp2 *Model Fit*

Description

Produces a structured, MFP-aware summary of a fitted `mfp2()` model. Unlike the raw model-specific summary output, this method separates interpretable linear terms from fractional-polynomial (FP) and other nonlinear terms, whose individual coefficients are curve parameters rather than per-unit effect sizes.

Usage

```
## S3 method for class 'mfp2'
summary(
  object,
  formulas = FALSE,
  basis = FALSE,
```

```
  raw = FALSE,
  notes = TRUE,
  digits = 3L,
  ...
)
```

Arguments

`object`	A fitted `mfp2()` object.
`formulas`	Logical. If `TRUE`, append the fitted-function formula for each nonlinear variable. Default `FALSE`.
`basis`	Logical. If `TRUE`, append a table of the individual fitted basis coefficients for nonlinear terms. Default `FALSE`.
`raw`	Logical. If `TRUE`, bypass the structured summary and return the raw model-specific summary object, with the fitting call replaced by the original `mfp2()` call. Default `FALSE`.
`notes`	Logical. If `FALSE`, suppress the explanatory model-df note when the structured summary is printed. Default `TRUE`.
`digits`	Number of decimal places used when printing numeric output other than fractional-polynomial powers and degrees of freedom. Defaults to 3.
`...`	Further arguments. When `raw = TRUE`, passed to the underlying summary method; otherwise ignored.

Details

For multinomial models, the structured summary reports the common selected functional forms and an outcome-specific coefficient table containing the estimate, standard error, z statistic, and p-value for each non-reference logit. Model-fit values are reported as minus twice the log likelihood.

The summary is organised into the following sections:

- **Selection Overview**: a compact overview of the selected functional form for each variable.
- **Ordinal Intercepts**: for ordinal models, the estimated threshold intercepts with standard errors, Wald statistics, p-values, and confidence intervals.
- **Linear Terms**: variables entering the model as a single linear term (including binary-only spike variables and factor levels), with the coefficient, standard error, test statistic, p-value, and confidence interval. GLM coefficients are always reported on their fitted link scale; supported survival models additionally report the usual exponentiated effect when it has a standard interpretation.
- **Nonlinear Terms**: one row per variable modelled by an FP1, FP2, ACD, or spike/catzero compound form. Each row reports a joint likelihood-ratio test (LRT) of all of that variable's terms, using the package's selection-adjusted degrees of freedom. The LRT compares the final MFP model with an otherwise unchanged model that omits the variable.
- **Model Fit**: full-linear and final-MFP deviances for GLMs, or minus twice the fitted likelihood for survival models (partial likelihood for Cox and Fine–Gray), with model degrees of freedom. GEE models instead show selection scores for AIC/BIC, and omit this block for p-value selection.

The joint LRT is a diagnostic: the variable was selected by the MFP procedure, not by this test. The degrees of freedom follow the Royston–Sauerbrei convention used throughout the package (2 df per FP1, 4 df per FP2), so the reported p-values acknowledge the FP power search rather than treating the powers as fixed in advance.

Coefficients for nonlinear terms are omitted from the default output. Set `formulas = TRUE` to append the fitted-function formulas, or `basis = TRUE` to append the individual fitted basis coefficients. For an active ACD component, either option also prints the fitted definition of $A(x)$, including the scale used during fitting. Set `raw = TRUE` to obtain the underlying model-specific summary object instead.

The structured header names the fitted model and its link where applicable. It also reports interpretation-critical nuisance parameters: the distribution, fixed or estimated scale, scale strata, and censoring pattern for `survreg` models; estimated or fixed dispersion for Gaussian, Gamma, and inverse-Gaussian GLMs; and the estimated theta for negative-binomial models. Ordinal and multinomial headers additionally report unweighted frequencies of the responses used for fitting.

Value

When `raw = FALSE`, an object of class `"summary.mfp2"`: a list with components `call`, `family`, `criterion`, `converged`, `n`, `nevents`, model-specific components `distribution`, `scale`, `scale_fixed`, `dispersion`, `dispersion_fixed`, `scale_strata`, `distribution_parameters`, `censoring`, `link`, `theta`, and `response_frequencies`, plus `function_table`, `linear_terms`, `nonlinear_terms`, `basis` (or `NULL`), `formulas` (or `NULL`), `acd_definitions` (or `NULL`), `ordinal_intercepts` (a threshold-inference data frame for ordinal models, otherwise `NULL`), `fit`, `raw_summary`, and `notes`. For compatibility with earlier versions, it also contains `coefficients`, `cov.scaled` and `cov.unscaled` when the underlying summary provides them (for example for GLM-type models); these are not printed. For multinomial models, `linear_terms`, `nonlinear_terms`, `basis`, `formulas`, `acd_definitions`, `ordinal_intercepts`, and `fit` are replaced by `coefficients` (the outcome-specific coefficient table), `model_fit_values`, `multinomial`, `class_levels`, `reference_class`, `n_logits`, and `digits`. A dedicated `print` method renders these. When `raw = TRUE`, the underlying summary object.

Coefficient statistics

The linear-terms table reports the coefficient statistics from the applicable GLM, Cox, or `survreg` summary. For a robust Cox fit, it uses the robust standard error so that the estimate, test statistic, and p-value remain consistent. When these statistics cannot be read directly from a compatible model summary, they are calculated from the fitted coefficients and `stats::vcov()` when possible.

See Also

`mfp2()`, `print.mfp2()`, `stats::summary.glm()`, `survival::summary.coxph()`, `survival::summary.survreg()`

Examples

```
data("prostate")

fit <- mfp2(
  lpsa ~ fp(age) + svi,
  data = prostate,
  select = 1,
  alpha = 1,
```

```
  verbose = FALSE
)

fit_summary <- summary(fit)

# The standardized coefficient statistics for selected linear terms.
fit_summary$linear_terms

# The original model-specific coefficient table remains available.
fit_summary$raw_summary$coefficients
```

`summary.mfpi` *Summarize an* `"mfpi"` *Object*

Description

Produces a structured summary of an `"mfpi"` fit.

Usage

```
## S3 method for class 'mfpi'
summary(object, ...)
```

Arguments

`object`	An object of class `"mfpi"`.
`...`	Currently unused.

Details

The printed summary has four steps. Steps 1–3 are the same model-building display produced by `print.mfpi()`: the selected adjustment model, all interaction candidates, and the final interaction-selection summary. Step 4 reports regression results for retained interaction models using MFPI-specific labels rather than the raw output of the underlying regression model.

The main MFPI header reports the FLEX setting, while Step 2 reports the grouping variable and interaction-selection settings. Step 4 avoids repeating those fit-level properties within each retained model. Instead, each retained interaction is introduced by an underlined interaction heading and reports its FP form and interaction powers for every group level. Regression coefficients are then separated into:

1. group-specific FP terms, displayed using the original group labels and readable FP transformations;
2. group-variable coefficients, with the fitted reference level stated explicitly;
3. a model intercept when the underlying model contains one; and
4. adjustment coefficients.

Centered basis terms report their stored centering constants using the same `Basis - Center` convention as `print.mfp2()`. Multinomial models use one outcome-versus-reference block per non-reference outcome, retaining the same coefficient-table separation within every block. Each coefficient table includes the estimate, standard error, model-specific test statistic (for example, `z` or `t`), and p-value. Significance stars, exponentiated coefficients, confidence-interval tables, the raw model call, and model-level likelihood-ratio/Wald/score tests are intentionally not printed. In particular, the usual global model tests condition on the selected regression specification and their standard degrees of freedom do not represent the additional model-selection complexity introduced by FP power selection.

Group-variable coefficients are coefficients of the fitted regression parameterization. When a group-by-continuous interaction is present, they are not overall comparisons between group levels; group comparisons depend on the interacting variable and should be obtained from `predict.mfpi()`.

Coefficient-level p-values are also conditional on the selected functional forms and do not account for uncertainty introduced by adjustment-model, FP, or interaction selection. They should therefore not be used for the MFPI interaction decision. The criterion reported in Step 3 is the relevant MFPI selection result.

Value

An object of class `"summary.mfpi"`. In addition to the fields needed to reproduce the MFPI Steps 1–3 display, the object contains `regression_displays`, a named list of structured coefficient displays for retained interaction models. Each display contains readable coefficient metadata, the corresponding model-conditional coefficient statistics, and model-specific nuisance metadata. Retained negative-binomial models report their own theta, while retained `survreg` models report their own scale(s). The normalized `family_string` is retained. Multinomial summaries also retain `class_levels`, `reference_class`, and `n_logits`; ordinal summaries retain `ordinal_levels`, `ordinal_link`, and `n_intercepts`. Printing the object produces the four-step summary described above.

See Also

`mfpi()`, `print.mfpi()`, `coef.mfpi()`, `vcov.mfpi()`, `predict.mfpi()`, `plot.mfpi()`

Examples

```
data("prostate", package = "mfp2")

fit <- mfpi(
  lpsa ~ fp(age) + svi + fp(cavol),
  data = prostate,
  group_var = "svi",
  interaction_vars = "cavol",
  interaction_forms = c(cavol = "fp2"),
  flex = "flex4",
  p_interact = 1,
  verbose = FALSE
)

summary(fit)
```

survreg_family *Survival Model Families*

Description

Use `survreg_family()` to choose a parametric survival model and its distribution. Use `finegray_family()` to study a particular event when other event types may occur first.

Usage

```
survreg_family(dist = "weibull", scale = 0, parms = NULL)

finegray_family(
  etype = NULL,
  timefix = TRUE,
  strata_action = c("both", "censoring", "baseline")
)
```

Arguments

`dist`	Survival-time distribution accepted by `survival::survreg()`. The default is `"weibull"`.
`scale`	Scale of the chosen distribution. The default, `0`, estimates it from the data; a positive value fixes it.
`parms`	Optional parameters for the chosen distribution, as accepted by `survival::survreg()`.
`etype`	Event type of interest in a Fine–Gray model: one of the event levels of the multi-state `Surv()` response, such as `"cause1"`. If `NULL`, the first event type (the first non-censoring level) is used. The censoring level cannot be used.
`timefix`	Whether to correct small differences in event times caused by rounding. The default is `TRUE`.
`strata_action`	How `strata()` terms in the model formula affect a Fine–Gray model: `"both"` The default. Allows both the censoring pattern and the baseline event pattern to differ between strata. `"censoring"` Allows the censoring pattern to differ between strata, while using a common baseline event pattern. `"baseline"` Allows the baseline event pattern to differ between strata, while estimating a common censoring pattern.

Details

Supply the resulting specification as the `family` argument to `mfp2()`. `survreg_family()` can also be used with `mfpi()`; Fine–Gray models cannot.

Both functions require a survival response. For Fine–Gray models, the response must identify the possible event types. Supply an `id` to `mfp2()` when observations use start and stop times.

Value

A family specification for the `family` argument of `mfp2()`. The specification from `survreg_family()` can also be used with `mfpi()`.

References

Fine JP, Gray RJ (1999). A proportional hazards model for the subdistribution of a competing risk. *Journal of the American Statistical Association*, 94(446), 496–509. doi:10.1080/01621459.1999.10474144

Zhou B, Latouche A, Rocha V, Fine JP (2011). Competing risks regression for stratified data. *Biometrics*, 67(2), 661–670. doi:10.1111/j.15410420.2010.01493.x

See Also

`survival::survreg()`, `mfp2()`, `mfpi()`

Examples

```
survreg_spec <- survreg_family(dist = "weibull")
finegray_spec <- finegray_family(etype = "cause1")


# Parametric Weibull survival model.
data("gbsg")
fit_survreg <- mfp2(
  survival::Surv(rectime, censrec) ~ age + nodes,
  data = gbsg,
  family = survreg_spec,
  df = 1,
  select = 1,
  cycles = 1,
  verbose = FALSE
)

# Fine--Gray model for the first of two competing event types.
set.seed(42)
n <- 160
x <- stats::rnorm(n)
cause1_time <- stats::rexp(n, rate = exp(0.3 * x) / 8)
cause2_time <- stats::rexp(n, rate = exp(-0.2 * x) / 10)
censor_time <- stats::rexp(n, rate = 1 / 15)
event <- ifelse(
  censor_time < pmin(cause1_time, cause2_time), 0L,
  ifelse(cause1_time <= cause2_time, 1L, 2L)
)
competing_data <- data.frame(
  time = pmin(cause1_time, cause2_time, censor_time),
  event = factor(
    event,
    levels = 0:2,
    labels = c("censor", "cause1", "cause2")
  ),
```

```
  x = x
)
fit_finegray <- mfp2(
  survival::Surv(time, event) ~ x,
  data = competing_data,
  family = finegray_spec,
  df = 1,
  select = 1,
  cycles = 1,
  verbose = FALSE
)
```

`transform_vector_fp` *Transform a Variable Using Fractional-Polynomial Powers*

Description

Constructs fractional-polynomial design columns from specified powers, analogous to `fracgen` in Stata.

Usage

```
transform_vector_fp(
  x,
  power = 1,
  scale = 1,
  shift = 0,
  name = NULL,
  zero = FALSE,
  check_binary = TRUE
)
```

Arguments

`x`	A numeric vector of predictor values.
`power`	A numeric vector of FP powers. Default is 1 (linear). `NA` values are ignored; if all powers are `NA`, `NULL` is returned.
`scale`	Positive numeric scaling factor applied to `x` after shifting. Default `1` (no scaling). If `NULL`, the scale factor is estimated automatically. Scaling does not change the selected powers.
`shift`	shift required for shifting x to positive values. Default is 0, meaning no shift is applied. If `NULL` then the shift is estimated automatically using the Royston and Sauerbrei formula iff any `x` <= 0. Ignored when `zero = TRUE`.
`name`	character used to define names for the output matrix. Default is `NULL`, meaning the output will have unnamed columns.

`zero` Logical indicating whether only values with `x > 0` should be transformed, with exact-zero values retained as structural zeros. If `TRUE`, `x` must be nonnegative and must have been validated by the caller; negative values must be explicitly recoded if they should represent the zero group. No shift is applied when `zero = TRUE`. If `FALSE` (default), all values are shifted (if needed) to ensure positivity.

`check_binary` a logical indicating whether or not input `x` is checked if it is a binary variable (i.e. has only two distinct values). The default `TRUE` usually only needs to changed when this function is to be used to transform data for predictions. See Details.

Details

This is a low-level utility for constructing fractional-polynomial design columns from specified powers. It does not perform model or power selection. Most users should specify continuous variables with `fp()` inside an `mfp2()` formula instead.

The FP transformation constructs design columns as follows. For each power pi in `power` = (p1, p2, ..., pn) it computes x^pi and returns the collection as a matrix. The input may be shifted and scaled before transformation. Centering, if desired, should be applied separately after transformation (see the Data processing section).

A special case are repeated powers, i.e. when some pi = pj. In this case, the fp transformations are given by x^pi and x^pi * log(x). In case more than 2 powers are repeated they are repeatedly multiplied with log(x) terms, e.g. pi = pj = pk leads to x^pi, x^pi * log(x) and x^pi * log(x)^2.

Note that the powers pi are assumed to be sorted. That is, this function sorts them, then proceeds to compute the transformation. For example, the output will be the same for `power = c(1, 1, 2)` and `power = c(1, 2, 1)`. This is done to make sense of repeated powers and to uniquely define FPs. In case an ACD transformation is used, there is a specific order in which powers are processed, which is always the same (but not necessarily sorted). Thus, throughout the whole package powers will always be given and processed in either sorted, or ACD specific order and the columns of the matrix returned by this function will always align with the powers used throughout this package.

Binary variables are not transformed unless `check_binary = FALSE`. With `check_binary = FALSE`, the requested FP power is applied even when `x` has one or two distinct values; use the fitted shift and scale parameters when transforming prediction values. For model fit, `check_binary` should always be at its default value.

Value

Returns a matrix of transformed variable(s). The number of columns depends on the number of powers provided, the number of rows is equal to the length of `x`. The columns are sorted by increased power. If all powers are `NA`, then this function returns `NULL`. Binary `x` with `check_binary = TRUE` is returned untransformed as a one-column matrix.

Data processing

Variables are shifted and then scaled before any power transformation is applied. Shifting ensures positive values; scaling avoids numerically extreme magnitudes. Scaling does not change the selected powers.

Centering is not performed by this function. When centering is desired, it should be applied after transformation, because centering before transformation alters the correlation structure among pre-

dictors. In mfp2(), centering is applied only to the final selected model when center = TRUE. See Sauerbrei et al (2006) for discussion.

If zero = TRUE, x must be nonnegative. Exact-zero values are not shifted and their transformed components remain zero; transformation is applied only to values with x > 0. Structural-zero membership is determined before the power transformation, so a positive value that transforms to zero is still part of the positive component. For performance, this low-level helper assumes that callers have already enforced the nonnegative-input contract. The public fitting and prediction interfaces perform that validation once, before any transformation work.

Domain restrictions

Fractional-polynomial transformations are applied after shifting and scaling. Users should ensure that the shifted and scaled values are in the domain required by the specified powers. In particular, powers involving logarithms require strictly positive input. This includes power = 0, which represents log(x), and repeated powers such as c(2, 2), where the second transformed variable is proportional to x^2 * log(x).

Negative powers are undefined at zero, and non-integer powers are not real-valued for negative input. Therefore, when using powers such as 0, -1, -0.5, 0.5, or repeated powers, the values of (x + shift) / scale should be strictly positive unless zero = TRUE.

Positive integer powers such as 1, 2, and 3 can be evaluated at zero and negative values, provided they are not part of a repeated-power FP basis.

These low-level transformation functions assume that the supplied x, shift, scale, and power values are compatible. If incompatible values are supplied, the transformation may produce nonfinite values or fail with an error from the underlying numerical code. In normal use through mfp2(), shift and scale parameters are estimated and checked during model fitting. During prediction, predict.mfp2() checks the fitted powers and stops with an informative error if new data fall outside the required transformation domain.

If zero = TRUE, exact-zero values are treated structurally: they are not shifted, and their transformed continuous components are set to zero.

References

Sauerbrei, W., Meier-Hirmer, C., Benner, A. and Royston, P., 2006. *Multivariable regression model building by using fractional polynomials: Description of SAS, STATA and R programs. Comput Stat Data Anal, 50(12): 3464-85.*

Examples

```
z = 1:10
transform_vector_fp(z)
```

vcov.mfpi *Extract Variance-Covariance Matrices from MFPI Interaction Models*

Description

Extracts estimated variance-covariance matrices from the term-specific interaction models stored in an `mfpi()` fit. The method uses the same model scope, term selection, coefficient order, and user-facing labels as `coef.mfpi()`.

Usage

```
## S3 method for class 'mfpi'
vcov(object, term = NULL, model = c("best", "all"), ...)
```

Arguments

`object`	A fitted `mfpi()` object.
`term`	Optional character vector naming continuous variables whose term-specific interaction models should be extracted. When `NULL`, all available terms in the requested `model` scope are returned.
`model`	Character scalar indicating which stored interaction models are eligible. `"best"` (the default) uses interactions retained by the MFPI selection criterion; `"all"` uses every evaluated term-specific winner, whether retained or not.
`...`	Not used. Supplying additional arguments produces an error.

Details

Each continuous variable evaluated by `mfpi()` has a separate interaction model. There is therefore no single covariance matrix spanning coefficients from different MFPI interactions. When one term is requested, `vcov.mfpi()` returns that model's ordinary covariance matrix. When several terms are requested, it returns a named list containing one covariance matrix per term-specific model.

Row and column names correspond exactly to `names(coef(object, ...))` for the same interaction model. Group-specific FP coefficients are labelled using the original group levels and readable FP transformations. Group-variable coefficients use labels such as `treatment [Drug A]`; the reference level is shown by the `coef()` print method rather than encoded as a pairwise contrast in the coefficient name.

The covariance matrices are conditional on the functional forms and model selected by MFPI. They do not incorporate the additional uncertainty caused by FP power selection, adjustment-model selection, or interaction selection.

Value

If one interaction model is requested, a numeric square matrix with row and column names matching `coef.mfpi()`. If several interaction models are requested, a named list with class `"mfpi_vcov_list"`, containing one matrix per interaction. The original fitted coefficient names are retained in the `raw_names` attribute of each matrix.

See Also

coef.mfpi(), mfpi(), summary.mfpi(), predict.mfpi()

Examples

```
data("prostate", package = "mfp2")

fit <- mfpi(
  lpsa ~ fp(age) + svi + fp(cavol),
  data = prostate,
  group_var = "svi",
  interaction_vars = "cavol",
  interaction_forms = c(cavol = "fp2"),
  flex = "flex4",
  p_interact = 1,
  verbose = FALSE
)

V <- vcov(fit, term = "cavol")
dim(V)
identical(rownames(V), names(coef(fit, term = "cavol")))

# `model = "all"` is useful when the evaluated interaction was not retained.
vcov(fit, term = "cavol", model = "all")
```

Index

∗ **data**
advanced_prostate_cancer, 5
art, 6
gbsg, 15
pima, 52
prostate, 96
∗ **package**
mfp2-package, 3

advanced_prostate_cancer, 5
art, 6

coef.mfp2, 6
coef.mfp2(), *32*
coef.mfpi, 7
coef.mfpi(), *47*, *92, 93*, *96*, *102*, *108, 109*
confint.mfp2, 9

fastglm::fastglm_nb(), *49, 50*
finegray_family *(*survreg_family*)*, 103
finegray_family(), *3, 4*, *22*, *27*, *30*, *32*
fp, 10
fp(), *3, 4*, *24*, *26*, *32*, *47*, *106*
fp2 *(*fp*)*, 10
fp2(), *12*, *24*, *26*, *31, 32*
fracplot, 13

gbsg, 15
gee_family, 16
gee_family(), *22*, *27*, *32*, *50*
geepack::geeglm(), *16, 17*, *24*, *76*
geepack::geese.control(), *23*
get_selected_variable_names, 18
get_selected_variable_names(), *25*, *32*, *70*, *76*
ggplot2::aes_colour_fill_alpha, *14*, *54*
ggplot2::geom_line(), *14*, *55*
ggplot2::geom_point(), *14*, *54, 55*
ggplot2::ggsave(), *63*
ggplot2::scale_linetype(), *14*, *55*
ggplot2::scale_shape(), *14*, *54*
ggplot2::theme_bw(), *14*, *54*

mfp2, 19
mfp2(), *3, 4*, *6, 7*, *9, 10*, *12*, *16–18*, *45–51*, *59*, *65*, *76*, *90, 91*, *97–100*, *104*, *106, 107*
mfp2-package, 3
mfpi, 37
mfpi(), *3, 4*, *7, 8*, *10*, *12*, *48–51*, *61*, *63*, *82–84*, *88*, *92*, *102*, *104*, *108, 109*
multinomial_family, 48
multinomial_family(), *3, 4*, *22*, *27*, *30*, *32*, *42*, *47*

negbin_family, 49
negbin_family(), *22*, *27*, *42*

ordinal_family, 51
ordinal_family(), *3, 4*, *22*, *27*, *31, 32*, *42*, *47*

pima, 52
plot(), *98*
plot.mfp2, 53
plot.mfp2(), *4*, *13, 14*, *32*
plot.mfpi, 60
plot.mfpi(), *47*, *88*, *102*
predict.mfp2, 64
predict.mfp2(), *4*, *7*, *28*, *30*, *32*, *56, 57*, *59*
predict.mfpi, 82
predict.mfpi(), *8*, *47*, *63*, *102*, *109*
print.mfp2, 90
print.mfp2(), *18*, *28*, *95*, *100*
print.mfpi, 91
print.mfpi(), *96*, *101, 102*
print.mfpi_coef, 92
print.mfpi_coef_list *(*print.mfpi_coef*)*, 92
print.mfpi_prediction, 93
print.mfpi_prediction_list, 94
print.mfpi_vcov_list, 94

print.summary.mfp2, 95
print.summary.mfpi, 95
prostate, 96

residuals.mfp2, 97

stats::coef(), *6*
stats::confint(), *9, 10*
stats::glm(), *3*, *27*
stats::glm.control(), *23*, *44*
stats::glm.fit(), *24*
stats::p.adjust(), *45*
stats::predict.glm(), *76*
stats::residuals(), *98*
stats::residuals.glm(), *97*
stats::summary.glm(), *100*
stats::vcov(), *10*, *100*
summary.mfp2, 98
summary.mfp2(), *7*, *10*, *28*, *32*, *91*, *95*
summary.mfpi, 101
summary.mfpi(), *8*, *47*, *63*, *88*, *92*, *95, 96*, *109*
survival::coxph(), *23*, *30*, *44*
survival::coxph.control(), *23*, *44*
survival::predict.coxph(), *76*
survival::predict.survreg(), *68*
survival::summary.coxph(), *100*
survival::summary.survreg(), *100*
survival::Surv(), *27*, *30*, *40*, *42*, *87*
survival::survreg(), *3*, *30*, *103, 104*
survival::survreg.control(), *23*, *44*
survreg_family, 103
survreg_family(), *3, 4*, *22*, *27*, *30*, *32*, *42*, *47*

transform_vector_fp, 105
transform_vector_fp(), *32*

vcov(), *10*
vcov.mfpi, 108
vcov.mfpi(), *8*, *47*, *94*, *96*, *102*