

Package: mfp2 (via r-universe)

October 8, 2026

Type Package

Title Multivariable Fractional Polynomial Models with Extensions

Version 1.0.2

Date 2025-10-01

Depends R (>= 4.2.0)

Imports ggplot2 (>= 3.4.0), stats, survival, utils

Suggests knitr, testthat (>= 3.0.0), xfun, rmarkdown, formatR,
patchwork, spelling

Description Multivariable fractional polynomial algorithm simultaneously selects variables and functional forms in both generalized linear models and Cox proportional hazard models. Key references are Royston and Altman (1994) <doi:10.2307/2986270> and Royston and Sauerbrei (2008, ISBN:978-0-470-02842-1). In addition, the implementation can model semi-continuous covariates with a “spike at zero” using a two-stage selection procedure. This extension follows the framework proposed by Becher et al. (2012) <doi:10.1002/bimj.201100263>. The package also includes the approximate cumulative distribution (ACD) transformation to model a sigmoid relationship between variable x and an outcome variable y, as described in Royston (2014) <doi:10.1177/1536867X1401400206> and Royston and Sauerbrei (2016) <doi:10.1177/1536867X1601600>. This feature distinguishes it from a standard fractional polynomial function, which lacks the ability to achieve such modeling.

License GPL-3

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

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

Encoding UTF-8

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

Maintainer Edwin Kipruto <edwin.kipruto@uniklinik-freiburg.de>

RoxygenNote 7.3.3
Roxygen list(markdown = TRUE)
VignetteBuilder knitr
Config/testthat/edition 3
Language en-US
Repository <https://edwinkipruto.r-universe.dev>
Date/Publication 2026-05-11 13:31:01 UTC
RemoteUrl <https://github.com/edwinkipruto/mfp2>
RemoteRef HEAD
RemoteSha 1bf3e78b3cd5e434325e80c507b564f07ff15478

Contents

apply_acd	3
apply_shift_scale	4
art	5
assign_df	5
backscale_matrix	6
calculate_f_test	6
calculate_lr_test	8
calculate_model_metrics	8
calculate_number_fp_powers	9
calculate_standard_error	10
center_matrix	11
coef.mfp2	12
contr.cumulative	12
convert_powers_list_to_matrix	13
create_dummy_variables	14
create_fp_terms	15
deviance_gaussian	16
ensure_length	17
find_best_fp_cycle	17
find_best_fp_step	20
find_best_fp1_for_acd	24
find_best_fpm_step	24
find_scale_factor	27
find_shift_factor	28
fit_acd	29
fit_cox	30
fit_glm	31
fit_linear_step	32
fit_model	34
fit_null_step	35
fp	36
fracplot	37

gbsg	40
generate_combinations_with_replacement	41
generate_powers_fp	41
generate_transformations_fp	42
get_selected_variable_names	44
mfp2	45
name_transformed_variables	60
order_variables	60
pima	61
predict.mfp2	62
prepare_newdata_for_predict	66
print.mfp2	67
print_mfp_step	67
prostate	68
reset_acd	69
select_ic	70
select_linear	73
select_ra2	76
select_ra2_acd	79
summary.mfp2	81
transform_matrix	82
transform_vector_fp	85
transform_vector_power	87

Index**89**

apply_acd	<i>Function to apply Approximate Cumulative Distribution (ACD)</i>
-----------	--

Description

Applies the acd transformation as outlined in Royston (2014) and Royston and Sauerbrei (2016). Designed to work with the output of `fit_acd()`, Please refer to the corresponding documentation for more details.

Usage

```
apply_acd(x, beta0, beta1, power, shift, scale, zero, ...)
```

Arguments

<code>x</code>	a numeric vector.
<code>beta0, beta1</code>	each a numeric value, representing the coefficients of the FP1 model for the ACD transformation.
<code>power</code>	a numeric value, estimated power to be used in the FP1 model for the ACD transformation.
<code>shift</code>	a numeric value that is used to shift the values of <code>x</code> to positive values.

scale	a numeric value used to scale x.
zero	Logical indicating whether only positive values of the variable should be transformed, with nonpositive values (zero or negative) set to zero. If TRUE, transformation is applied only to positive values; nonpositive values are replaced with zero before transformation.
...	not used.

Value

The transformed input vector x.

apply_shift_scale	<i>Shift and scale vector x</i>
-------------------	---------------------------------

Description

A function that is used to shift x values to positive values if it contains negative or zero values. If all values of x are positive then the original values of x is returned without shifting but scaled if the scaling factor is not equal to 1. If x has already been shifted and scaled then the function does nothing.

Usage

```
apply_shift_scale(x, scale = NULL, shift = NULL)
```

Arguments

x	A vector of predictor variable
scale	scaling factors for x of interest. Must be positive integers. Default is NULL and scaling factors are automatically estimated using find_scale_factor() function else it uses user supplied scaling factors. If no scaling is needed just use scale = 1
shift	adjustment factors required for shifting x to positive values. Default is NULL and adjustment factors are estimated automatically using find_shift_factor() function

Value

A numeric value that has been shifted and scaled.

Examples

```
x = 1:1000
apply_shift_scale(x)
```

art	<i>Artificial dataset with continuous response</i>
-----	--

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 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.

assign_df	<i>Helper to assign degrees of freedom</i>
-----------	--

Description

Determine the number of unique values in a variable. To be used in `mfp2()`.

Usage

```
assign_df(x, df_default = 4)
```

Arguments

x input matrix.

df_default default df to be used. Default is 4.

Details

Variables with fewer than or equal to three unique values, for example, will be assigned $df = 1$. $df = 2$ will be assigned to variables with 4-5 unique values, and $df = 4$ will be assigned to variables with unique values greater than or equal to 6.

Value

Vector of length `ncol(x)` with degrees of freedom for each variable in `x`.

Examples

```
x <- matrix(1:100, nrow = 10)
assign_df(x)
```

<code>backscale_matrix</code>	<i>Backscale Columns of a Matrix (Internal)</i>
-------------------------------	---

Description

Multiplies each column of a numeric matrix by a corresponding scalar value from a named vector. Typically used to reverse prior scaling (i.e., backscaling). This is an internal helper function and not intended for direct use by package users.

Usage

```
backscale_matrix(x, scalex)
```

Arguments

- `x` A numeric matrix with column names, or `NULL`.
- `scalex` A named numeric vector. Each name must match a column name of `x`.

Value

A matrix with backscaled columns, or `NULL` if `x` is `NULL`.

<code>calculate_f_test</code>	<i>Function to compute F-statistic and p-value from deviances</i>
-------------------------------	---

Description

Alternative to likelihood ratio tests in normal / Gaussian error models.

Usage

```
calculate_f_test(deviances, dfs_resid, n_obs, d1 = NULL)
```

Arguments

deviances	a numeric vector of length 2 with deviances. Typically ordered in increasing order (i.e. null model first, then full model) and used to test the difference $\text{deviances}[1] - \text{deviances}[2]$.
dfs_resid	a numeric vector with residual degrees of freedom.
n_obs	a numeric value with the number of observations.
d1	a numeric value giving d1 in the formula below directly as the number of additional degrees of freedom in model 2 compared to model 1. In this case dfs_resid must be a single numeric value giving the residual df for model 2. This interface is sometimes more convenient than to specify both residual dfs.

Details

Uses formula on page 23 from here: <https://www.stata.com/manuals/rfp.pdf>:

$$F = \frac{d_2}{d_1} \left(\exp\left(\frac{D_2 - D_1}{n}\right) - 1 \right),$$

where D refers to deviances of two models 1 and 2. d_1 is the number of additional parameters used in in model 2 as compared to model 1, i.e. $\text{dfs_resid}[1] - \text{dfs_resid}[2]$. d_2 is the number of residual degrees of freedom minus the number of estimated powers for model 2, i.e. $\text{dfs_resid}[2]$. #* The p-value then results from the use of a F-distribution with (d_1 , d_2) degrees of freedom.

Note that this computation is completely equivalent to the computation of a F-test using sum of squared errors as in e.g. Kutner et al. (2004), p 263. The formula there is given as

$$F = \frac{SSE(R) - SSE(F)}{df_R - df_F} / \frac{SSE(F)}{df_F},$$

where the df terms refer to residual degrees of freedom, and R and F to the reduced (model 1) and full model (model 2), respectively.

Value

A list with three entries giving the test statistic and p-value for the F-test for the comparison of $\text{deviance}[1]$ to $\text{deviance}[2]$.

- statistic: test statistic.
- pvalue: p-value.
- dev_diff: difference in deviances tested.

References

Kutner, M.H., et al., 2004. *Applied linear statistical models*. McGraw-Hill Irwin.

calculate_lr_test	<i>Function to calculate p-values for likelihood-ratio test</i>
-------------------	---

Description

Function to calculate p-values for likelihood-ratio test

Usage

```
calculate_lr_test(logl, dfs)
```

Arguments

logl	a numeric vector of length 2 with log-likelihoods. Typically ordered in increasing order (i.e. null model first, then full model) and used to test the ratio $\text{logl}[1] / \text{logl}[2]$.
dfs	a numeric vector with degrees of freedom.

Details

Uses Wilk's theorem that $-2\log(\text{LR})$ (LR = likelihood ratio) asymptotically approaches a Chi-square distribution under the null hypothesis that both likelihoods are equal.

Model likelihoods can then be compared by computing $D = -2 \log(\text{likelihood reduced model} / \text{likelihood full model})$, and then use a Chi-square distribution with $\text{df_full} - \text{df_reduced}$ degrees of freedom to derive a p-value.

This is basically the same way as `stats::anova()` implements the likelihood ratio test.

Value

A list with two entries for the likelihood ratio test for the ratio $\text{logl}[1] / \text{logl}[2]$.

- statistic: test statistic.
- pvalue: p-value

calculate_model_metrics	<i>Function to compute model metrics to be used within mfp2</i>
-------------------------	---

Description

Mostly used within an mfp step to compare between the different fp models of a variable.

Usage

```
calculate_model_metrics(obj, n_obs, df_additional = 0)
```

Arguments

obj	a list returned by <code>fit_model()</code> representing a glm or Cox model fit.
n_obs	a numeric value indicating the number of observations for the data used to fit obj.
df_additional	a numeric value indicating the number of additional degrees of freedom to be accounted for in the computations of AIC and BIC. These may be necessary when a model uses FP terms, as these add another degree of freedom per estimated power.

Value

A numeric vector with the following entries:

- df: number of degrees of freedom of model (i.e. coefficients plus df_additional).
- deviance_rs: "deviance", i.e. minus twice the log likelihood. This is not the usual definition of deviance used by R, which is defined as twice the difference between the log likelihoods of the saturated model (one parameter per observation) and the null (or reduced) model. It is, however, the definition used in Royston and Sauerbrei (2008) and in mfp. For selection of fps this does not really play a role, as the common factor would be cancelled anyway when comparing models based on deviances.
- sse: sum of squared residuals as returned by `fit_model()`.
- deviance_gaussian: deviance computed by `deviance_gaussian()`, applicable to Gaussian models and used for F-test computations.
- aic: Akaike information criterion, defined as $-2\log L + 2(df + df_additional)$.
- bic: Bayesian information criterion, defined as $-2\log L + \log(n_obs)(df + df_additional)$.
- df_resid: residual degrees of freedom, defined as $n_obs - df$. For consistency with stata we subtract the scale parameter from df.

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.

calculate_number_fp_powers

Calculates the total number of fractional polynomial powers in adjustment variables.

Description

This function takes a list `x` containing fractional polynomial powers for all variables and calculates the total number of powers across the variables.

Usage

```
calculate_number_fp_powers(x)
```

Arguments

`x` A list of fractional polynomial powers for all variables.

Value

Numeric value denoting total number of fractional polynomial powers in the adjustment variables.

```
calculate_standard_error
```

Helper function to compute standard error of a partial predictor

Description

To be used in `predict.mfp2()`.

Usage

```
calculate_standard_error(model, X, xref = NULL)
```

Arguments

`model` fitted mfp2 object.

`X` transformed input matrix with variables of interest for partial predictor.

`xref` transformed reference value for variable of interest. Default is NULL, in which case this function computes standard errors without reference values.

Details

See pages 91-92 and following in the book by Royston and Sauerbrei 2008 for the formulas and mathematical details.

Value

Standard error.

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.

center_matrix	<i>Simple function to center data</i>
---------------	---------------------------------------

Description

Simple function to center data

Usage

```
center_matrix(mat, centers = NULL, zero = NULL)
```

Arguments

mat	a transformed data matrix.
centers	a vector of centering values. Length must be equal to the number of columns in mat. If NULL (default) then centering values are determined by the function (see Details).
zero	Optional named logical vector indicating which columns treat zero values specially. Names must match mat columns. Default NULL means no zero-specific handling.

Details

Centering is done by column means for continuous variables (more than 2 distinct values) and by the minimum for binary variables. For variables with zero = TRUE, the mean is computed only over the non-zero values, while zero values remain at zero.

It is assumed all categorical variables in the data are represented by binary dummy variables.

Value

Transformed data matrix. Has an attribute scaled:center that stores values used for centering.

Examples

```
mat <- matrix(1:100, nrow = 10)
colnames(mat) <- paste0("x", 1:ncol(mat))
zero <- setNames(rep(FALSE, ncol(mat)), colnames(mat))
center_matrix(mat, zero = zero)
```

coef.mfp2	<i>Extract coefficients from object of class mfp2</i>
-----------	---

Description

This function is a method for the generic `stats::coef()` function for objects of class `mfp2`.

Usage

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

Arguments

object	an object of class <code>mfp2</code> , usually, a result of a call to <code>mfp2()</code> .
...	not used.

Value

Named numeric vector of coefficients extracted from the model object.

contr.cumulative	<i>Cumulative (Threshold) Contrast Coding for Ordered Factors</i>
------------------	---

Description

Generates a contrast matrix for ordered factors in which each column represents a cumulative comparison: level k versus all lower levels. This is sometimes called cumulative or sequential dummy coding.

Usage

```
contr.cumulative(n)
```

Arguments

n	Integer or character vector. If an integer, specifies the number of levels. If a character vector, its elements are used as factor levels.
---	--

Details

For an ordered factor with levels $A < B < C < D$, the resulting matrix produces three columns: Column 1 compares B, C, D versus A; Column 2 compares C, D versus A and B; Column 3 compares D versus A, B, and C. Each element of the matrix is 0 or 1, with 1 indicating that the observation belongs to the "higher" category for that threshold.

Column Names: Column names are formatted as varname_1, varname_2, etc. These names are syntactically valid, unique, and correspond to thresholds in order: varname_1 represents the first threshold (level 2 vs level 1), varname_2 represents the second threshold (level 3 vs levels 1 and 2), etc.

This contrast matrix can be assigned to an ordered factor via `contrasts()` before fitting a model, e.g., in `mfp2.formula()`. This approach preserves ordinal information while allowing threshold-type interpretation of regression coefficients.

Value

A numeric matrix with `length(n)` rows and `length(n)-1` columns. Each column is a dummy variable encoding the cumulative comparison of higher levels against lower levels. Row names correspond to factor levels.

Examples

```
# Create a data frame
data <- data.frame(grade = c("A", "B", "C", "D", "A"))
# Convert the column to an ordered factor
data$grade <- factor(data$grade, levels = c("A", "B", "C", "D"), ordered = TRUE)
# Assign the cumulative contrasts to the ordered factor
contrasts(data$grade) <- contr.cumulative(levels(data$grade))
```

```
convert_powers_list_to_matrix
```

Helper to convert a nested list with same or different length into a matrix

Description

To be used in `fit_mfp()`.

Usage

```
convert_powers_list_to_matrix(power_list)
```

Arguments

`power_list` list of powers created in `fit_mfp()`.

Value

a matrix.

`create_dummy_variables`*Simple function to create dummy variables for ordinal and nominal variables*

Description

Simple function to create dummy variables for ordinal and nominal variables

Usage

```
create_dummy_variables(  
  data,  
  var_ordinal = NULL,  
  var_nominal = NULL,  
  drop_variables = FALSE  
)
```

Arguments

<code>data</code>	A dataframe containing the ordinal variable.
<code>var_ordinal</code>	Names of ordinal variables in the data for which dummy variables should be created.
<code>var_nominal</code>	Names of nominal variables in the data for which dummy variables should be created.
<code>drop_variables</code>	Specifies whether to drop the original variables after dummy variables have been created. The default value is FALSE, and the original variables are kept in the data.

Details

This function creates dummy variables based on ordinal and categorical coding described in the Royston and Sauerbrei (2008) book (Chapter 3, Table 3.5). It uses the levels of the categorical variable if they exist; otherwise, it will extract the unique values of the variable, sort them, and use them as levels. We recommend that the user sets the levels of categorical variables and specifies their reference group. You can use the `factor()` function in base R. If the levels are 1, 2, and 3, then 1 will be the reference group. On the other hand, if the levels are 3, 2, and 1, then 3 will be the reference group. In brief, the first level will be taken as the reference group.

Value

A dataframe with new dummy variables.

Examples

```
data("gbsg")
# create dummy variable for grade using ordinal coding
gbsg <- create_dummy_variables(gbsg, var_ordinal = "grade", drop_variables = TRUE)
head(gbsg)
```

create_fp_terms

*Helper to create overview table of fp terms***Description**

To be used in `fit_mfp()`.

Usage

```
create_fp_terms(
  fp_powers,
  acdx,
  df,
  select,
  alpha,
  criterion,
  zero,
  catzero,
  spike,
  spike_decision
)
```

Arguments

<code>fp_powers</code>	powers of the created FP terms.
<code>acdx</code>	a logical vector of length <code>nvars</code> indicating which continuous variables should undergo the approximate cumulative distribution (ACD) transformation.
<code>df</code>	a numeric vector of length <code>nvars</code> of degrees of freedom.
<code>select</code>	a numeric vector of length <code>nvars</code> indicating significance levels for backward elimination.
<code>alpha</code>	a numeric vector of length <code>nvars</code> indicating significance levels for tests between FP models of different degrees.
<code>criterion</code>	a character string defining the criterion used to select variables and FP models of different degrees.
<code>zero</code>	A logical vector indicating, by position, which columns of <code>x</code> should treat non-positive values (zero or negative) as zero before transformation. Must be the same length and in the same order as the columns of <code>x</code> .

catzero	A logical vector similar to zero, indicating which columns of x should treat nonpositive values as zero and also have a binary indicator automatically created and included in the model. Must match the length and order of the columns in x . See <code>mfp2</code> for details.
spike	A logical vector indicating which columns of x contain a spike at zero. The length and order of <code>spike</code> must match those of the columns in x .
spike_decision	Integer vector indicating the modeling decision for spike-at-zero variables.

Value

Dataframe with overview of all fp terms. Each row represents a variable, with rownames giving the name of the variable. Variables with acd transformation are prefixed by `A_` by the `print` and `summary` methods. The dataframe comprises the following columns:

- `df_initial`: initial degrees of freedom.
- `select`: significance level used for backward elimination (or criterion name if not "pvalue").
- `alpha`: significance level for FP terms (or criterion name if not "pvalue").
- `acd`: logical, whether an ACD transformation was applied.
- `zero`: logical, indicates whether only the positive values of the variable are transformed (i.e., whether the FP function is applied exclusively to values greater than zero).
- `catzero`: logical, whether a binary variable for zero values was created.
- `spike`: logical, indicates presence of a spike-at-zero variable.
- `spike_decision`: integer code describing how the spike-at-zero variable is modeled.
- `selected`: logical, whether the FP term is included in the final model.
- `df_final`: final estimated degrees of freedom for the variable.
- `power1`, `power2`, ...: final estimated FP powers (as many columns as needed).

deviance_gaussian	<i>Deviance computations as used in mfp in stata</i>
-------------------	--

Description

Deviance computations as used in `mfp` in `stata`

Usage

```
deviance_gaussian(rss, weights, n)
```

Arguments

<code>rss</code>	residual sum of squares.
<code>weights</code>	numeric vector of weights used in computation of <code>rss</code> .
<code>n</code>	number of observations used to compute <code>rss</code> .

Details

Note that this is not the usual formula of deviance used in R, but uses the formula found here <https://www.stata.com/manuals/rfp.pdf>.

It can be applied for normal error models, but should not be used for other kinds of glms.

Value

A numeric value representing the deviance of a Gaussian model.

ensure_length	<i>Helper function to ensure vectors have a specified length</i>
---------------	--

Description

Used to make sure dimensions of matrix rows match.

Usage

```
ensure_length(x, size, fill = NA)
```

Arguments

x	input vector or matrix.
size	length or size of x which is desired.
fill	value to fill in if x is not of desired length or size.

find_best_fp_cycle	<i>Helper to run cycles of the mfp algorithm</i>
--------------------	--

Description

This function estimates the best FP functions for all predictors in the current cycle. To be used in `fit_mfp()`.

Usage

```
find_best_fp_cycle(
  x,
  y,
  powers_current,
  df,
  weights,
  offset,
  family,
```

```

family_string,
criterion,
select,
alpha,
keep,
powers,
method,
strata,
verbose,
ftest,
control,
rownames,
nocenter,
zero,
catzero,
spike,
spike_decision,
acd_parameter,
acdx,
prev_adj_params
)

```

Arguments

<code>x</code>	an input matrix of dimensions <code>nobs</code> x <code>nvars</code> . Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
<code>y</code>	a vector for the response variable or a <code>Surv</code> object.
<code>powers_current</code>	a list of length equal to the number of variables, indicating the fp powers to be used in the current step for all variables (except <code>xi</code>).
<code>df</code>	a numeric vector of length <code>nvars</code> of degrees of freedom.
<code>weights</code>	a vector of observation weights of length <code>nobs</code> .
<code>offset</code>	a vector of length <code>nobs</code> of offsets.
<code>family</code>	Either a character string specifying the model family (e.g., "gaussian", "binomial", "poisson", "cox") or a function that returns a GLM family object, such as <code>stats::gaussian(link = "identity")</code> or <code>stats::binomial(link = "logit")</code> . For Cox models, only the character string "cox" is allowed.
<code>family_string</code>	A character string representing the selected family, e.g., "gaussian".
<code>criterion</code>	a character string defining the criterion used to select variables and FP models of different degrees.
<code>select</code>	a numeric vector of length <code>nvars</code> indicating significance levels for backward elimination.
<code>alpha</code>	a numeric vector of length <code>nvars</code> indicating significance levels for tests between FP models of different degrees.
<code>keep</code>	a character vector with names of variables to be kept in the model.

powers	a named list of numeric values that sets the permitted FP powers for each co-variate.
method	a character string specifying the method for tie handling in Cox regression model.
strata	a factor of all possible combinations of stratification variables. Returned from survival::strata() .
verbose	Logical; if TRUE, additional information will be printed during model fitting steps. Useful for understanding internal processing. Default is FALSE.
ftest	a logical indicating the use of the F-test for Gaussian models.
control	a list with parameters for model fit. See survival::coxph() or stats::glm() for details.
rownames	passed to survival::coxph.fit() .
nocenter	a numeric vector with a list of values for fitting Cox models. See survival::coxph() for details.
zero	A logical vector indicating, by position, which columns of x should treat non-positive values (zero or negative) as zero before transformation. Must be the same length and in the same order as the columns of x .
catzero	A named list of binary indicator variables of length $\text{ncol}(x)$ for nonpositive values, created when specific variables are passed to the <code>catzero</code> argument of <code>fit_mfp</code> . If an element of the list is NULL, it indicates that the corresponding variable was not specified by the user in the <code>catzero</code> argument of <code>fit_mfp</code> . Here, <code>catzero</code> is a list of binary variables, not a named logical vector as in <code>fit_mfp</code> .
spike	A logical vector indicating which columns of x contain a spike at zero. The length and order of <code>spike</code> must match those of the columns in x .
spike_decision	Named vector indicating how spike-at-zero (SAZ) variables are handled. Each element corresponds to a variable and encodes the selected strategy: 1 = include FP for positive values plus binary SAZ, 2 = treat as continuous FP only, 3 = include binary SAZ only.
acd_parameter	Named list of ACD parameters produced by <code>fit_acd()</code> , with length equal to $\text{ncol}(x)$. Each list element corresponds to a variable; if an element is NULL, the variable was not specified in the <code>acdx</code> argument of <code>fit_mfp</code> .
acdx	a logical vector of length <code>nvars</code> indicating which continuous variables should undergo the approximate cumulative distribution (ACD) transformation.
prev_adj_params	Named list used to store previously computed adjustment variable transformations. This is updated at each step and reused in the next cycle to avoid recomputation.

Details

A cycle is defined as a complete pass through all the predictors in the input matrix x , while a step is defined as the assessment of a single predictor. This algorithm is described in Sauerbrei et al. (2006) and given in detail in Royston and Sauerbrei (2008), in particular chapter 6.

Briefly, a cycle works as follows: it takes as input the data matrix along with a set of current best fp powers for each variable. In each step, the fp powers of a single covariate are assessed, while adjusting for other covariates. Adjustment variables are transformed using their current fp powers (this is done in `transform_data_step()` and the fp powers of the variable of interest are tested using the closed test procedure (conducted in `find_best_fp_step()`). Some of the adjustment variables may have their fp power set to NA, which means they were not selected from the working model and are not used in that step. The results from all steps are returned, completing a cycle.

Note that in each cycle every variable is evaluated. This includes variables that may have been eliminated in previous cycles. They will re-enter each new cycle for potential inclusion in the working model or to be re-evaluated for elimination.

The current adjustment set is always given through the current fp powers, which are updated in each step (denoted as `powers_current`).

If `catzero` variables are supplied, the algorithm will automatically create the corresponding binary variables and include them in the model. Additionally, each binary variable and its associated continuous variable will be treated as one predictor, and they will be tested jointly for inclusion in the model.

Value

A list with updated components `powers_current` (current FP powers for all variables), `spike_decision` (updated spike-at-zero decisions), and `prev_adj_params` (adjustment variable transformations to be used in the next cycle).

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.

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.

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.

<code>find_best_fp_step</code>	<i>Function to estimate the best FP functions for a single variable</i>
--------------------------------	---

Description

See `mfp2()` for a brief summary on the notation used here and `fit_mfp()` for an overview of the fitting procedure.

Usage

```
find_best_fp_step(
  x,
  y,
  xi,
  weights,
  offset,
  df,
  powers_current,
  family,
  family_string,
  criterion,
  select,
  alpha,
  keep,
  powers,
  method,
  strata,
  nocenter,
  acdx,
  ftest,
  control,
  rownames,
  zero,
  catzero,
  spike,
  spike_decision,
  acd_parameter,
  prev_adj_params,
  verbose
)
```

Arguments

<code>x</code>	an input matrix of dimensions <code>nobs</code> x <code>nvars</code> . Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
<code>y</code>	a vector for the response variable or a <code>Surv</code> object.
<code>xi</code>	a character string indicating the name of the current variable of interest, for which the best fractional polynomial transformation is to be estimated in the current step.
<code>weights</code>	a vector of observation weights of length <code>nobs</code> .
<code>offset</code>	a vector of length <code>nobs</code> of offsets.
<code>df</code>	a numeric vector indicating the maximum degrees of freedom for the variable of interest <code>xi</code> .
<code>powers_current</code>	a list of length equal to the number of variables, indicating the fp powers to be used in the current step for all variables (except <code>xi</code>).

family	Either a character string naming the family (e.g., "gaussian", "binomial", "cox") or a function that returns a GLM family object (e.g., stats::gaussian). For Cox models, only a character string "cox" is allowed.
family_string	A character string representing the selected family, e.g., "gaussian".
criterion	a character string defining the criterion used to select variables and FP models of different degrees.
select	a numeric value indicating the significance level for backward elimination of x_i .
alpha	a numeric value indicating the significance level for tests between FP models of different degrees for x_i .
keep	a character vector with names of variables to be kept in the model.
powers	a named list of numeric values that sets the permitted FP powers for each co-variate.
method	a character string specifying the method for tie handling in Cox regression.
strata	a factor of all possible combinations of stratification variables. Returned from survival::strata() .
nocenter	a numeric vector with a list of values for fitting Cox models. See survival::coxph() for details.
acdx	a logical vector of length nvars indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
fctest	a logical indicating the use of the F-test for Gaussian models.
control	a list with parameters for model fit.
rownames	a parameter for Cox models.
zero	A named logical vector indicating, which columns of x should treat nonpositive values (zero or negative) as zero before transformation. Must be the same length as the columns of x .
catzero	A named list of binary indicator variables of length $\text{ncol}(x)$ for nonpositive values, created when specific variables are passed to the catzero argument of <code>fit_mfp</code> . If an element of the list is NULL, it indicates that the corresponding variable was not specified by the user in the catzero argument of <code>fit_mfp</code> . Here, catzero is a list of binary variables, not a named logical vector as in <code>fit_mfp</code> .
spike	A logical vector indicating which columns of x contain a spike at zero. The length and order of spike must match those of the columns in x .
spike_decision	Named vector indicating how spike-at-zero (SAZ) variables are handled. Each element corresponds to a variable and encodes the selected strategy: 1 = include FP for positive values plus binary SAZ, 2 = treat as continuous FP only, 3 = include binary SAZ only.
acd_parameter	Named list of ACD parameters produced by <code>fit_acd()</code> , with length equal to $\text{ncol}(x)$. Each list element corresponds to a variable; if an element is NULL, the variable was not specified in the acdx argument of <code>fit_mfp</code> .
prev_adj_params	Named list storing adjustment variable transformations from previous steps for each variable.
verbose	a logical; run in verbose mode.

Details

The function selection procedure (FSP) is used if the p-value criterion is chosen, whereas the criteria AIC and BIC select the model with the smallest AIC and BIC, respectively.

It uses transformations for all other variables to assess the FP form of the current variable of interest. This function covers three main use cases:

- the linear case ($df = 1$) to test between null and linear models (see `select_linear()`). This step differs from the mfp case because linear models only use 1 df, while estimation of (every) fp power adds another df. This is also the case applied for categorical variables for which df are set to 1.
- the case that an acd transformation is requested (`acdx` is TRUE for `xi`) for the variable of interest (see `find_best_fpm_step()`).
- the (usual) case of the normal mfp algorithm to assess non-linear functional forms (see `find_best_fpm_step()`).

Note that these cases do not encompass the setting that a variable is not selected, because the evaluation is done for each variable in each cycle. A variable which was de-selected in earlier cycles may be added to the working model again. Also see `find_best_fp_cycle()`.

The adjustment in each step uses the current fp powers given in `powers_current` for all other variables to determine the adjustment set and transformations in the working model.

Note that the algorithm starts by setting all $df = 1$, and higher fps are evaluated in turn starting from the first step in the first cycle.

Value

A numeric vector indicating the best powers for `xi`. Entries can be NA if variable is to be removed from the working model. Note that this vector may include up to two NA entries when ACD transformation is requested, but otherwise is either a vector with all numeric entries, or a single NA.

Functional form selection

There are 3 criteria to decide for the current best functional form of a continuous variable.

The first option for `criterion = "pvalue"` is the function selection procedure as outlined in e.g. Chapters 4 and 6 of Royston and Sauerbrei (2008), also abbreviated as "RA2". It is a closed testing procedure and is implemented in `select_ra2()` and extended for ACD transformation in `select_ra2_acd()` according to Royston and Sauerbrei (2016).

For the other criteria `aic` and `bic` all FP models up to the desired degree are fitted and the model with the lowest value for the information criteria is chosen as the final one. This is implemented in `select_ic()`.

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.

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

find_best_fp1_for_acd *Function to fit univariable FP1 models for acd transformation*

Description

To be used in fit_acd().

Usage

```
find_best_fp1_for_acd(x, y, powers, zero)
```

Arguments

x	a numeric vector.
y	normal cdf of rank transform of x.
powers	a vector of allowed FP powers. The default value is NULL, meaning that the set $S = (-2, -1, -0.5, 0, 0.5, 1, 2, 3)$ is used.
zero	Logical indicating whether only positive values of the variable should be transformed, with nonpositive values (zero or negative) set to zero. If TRUE, transformation is applied only to positive values; nonpositive values are replaced with zero before transformation. If FALSE (default), all values are shifted (if needed) to ensure positivity before transformation.

Value

The best FP power with smallest deviance and the fitted model.

find_best_fpm_step *Function to find the best FP functions of given degree for a single variable*

Description

Handles the FP1 and the higher order FP cases. For parameter definitions, see find_best_fp_step().

Usage

```
find_best_fpm_step(
  x,
  xi,
  degree,
  y,
  powers_current,
  powers,
  acdx,
```

```

    family,
    family_string,
    zero,
    catzero,
    spike_decision,
    acd_parameter,
    prev_adj_params,
    ...
)

```

Arguments

<code>x</code>	an input matrix of dimensions <code>nobs</code> x <code>nvars</code> . Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
<code>xi</code>	a character string indicating the name of the current variable of interest, for which the best fractional polynomial transformation is to be estimated in the current step.
<code>degree</code>	degrees of freedom for fp transformation of <code>xi</code> .
<code>y</code>	a vector for the response variable or a <code>Surv</code> object.
<code>powers_current</code>	a list of length equal to the number of variables, indicating the fp powers to be used in the current step for all variables (except <code>xi</code>).
<code>powers</code>	a named list of numeric values that sets the permitted FP powers for each covariate.
<code>acdx</code>	a logical vector of length <code>nvars</code> indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
<code>family</code>	Either a character string naming the family (e.g., "gaussian", "binomial", "cox") or a function that returns a GLM family object (e.g., <code>stats::gaussian</code>). For Cox models, only a character string "cox" is allowed.
<code>family_string</code>	A character string representing the selected family, e.g., "gaussian".
<code>zero</code>	A named logical vector indicating, which columns of <code>x</code> should treat nonpositive values (zero or negative) as zero before transformation. Must be the same length as the columns of <code>x</code> .
<code>catzero</code>	A named list of binary indicator variables of length <code>ncol(x)</code> for nonpositive values, created when specific variables are passed to the <code>catzero</code> argument of <code>fit_mfp</code> . If an element of the list is <code>NULL</code> , it indicates that the corresponding variable was not specified by the user in the <code>catzero</code> argument of <code>fit_mfp</code> . Here, <code>catzero</code> is a list of binary variables, not a named logical vector as in <code>fit_mfp</code> .
<code>spike_decision</code>	Named vector indicating how spike-at-zero (SAZ) variables are handled. Each element corresponds to a variable and encodes the selected strategy: 1 = include FP for positive values plus binary SAZ, 2 = treat as continuous FP only, 3 = include binary SAZ only.
<code>acd_parameter</code>	Named list of ACD parameters produced by <code>fit_acd()</code> , with length equal to <code>ncol(x)</code> . Each list element corresponds to a variable; if an element is <code>NULL</code> , the variable was not specified in the <code>acdx</code> argument of <code>fit_mfp</code> .

```
prev_adj_params      Named list storing adjustment variable transformations from previous steps for
                      each variable.
...                  parameters passed to fit_model().
```

Details

The "best" model is determined by the highest likelihood (or smallest deviance by our definition as minus twice the log-likelihood). This is also the case for the use of information criteria, as all models investigated in this function have the same df, so the penalization term is equal for all models and only their likelihoods differ.

Note that the estimation of each fp power adds a degree of freedom. Thus, all fp1s have 2 df, all fp2s have 4 df and so on.

In the case that degree = 1, the linear model (fp power of 1) is NOT returned, as it is not considered to be a fractional polynomial in this algorithm. A linear model has only one df, whereas the same function regarded as fp would have 2 fp.

Value

A list with several components:

- `acd`: logical indicating if an ACD transformation was applied for `xi`.
- `powers`: fp powers investigated in step.
- `power_best`: the best power found. `power_best` will always be a two-column matrix when an ACD transformation is used, otherwise the number of columns will depend on degree.
- `metrics`: a matrix with performance indices for all models investigated. Same number of rows as, and indexed by, `powers`.
- `model_best`: row index of best model in `metrics`.
- `zero`: Logical indicating whether a zero transformation was applied to `xi`. In this case, non-positive values of `xi` were set to zero before transformation, and only positive values were transformed.
- `catzero`: Logical indicating whether a combination of a zero transformation and a binary indicator variable was applied to `xi`. This means that nonpositive values of `xi` were set to zero, only positive values were transformed, and an additional binary variable was created to indicate whether `xi` was positive or nonpositive.

ACD transformation

This function also handles the case of ACD transformations if `acdx` is set to TRUE for `xi`. In this case, if degree = 1, then 7 models are assessed (like for the non-acd case it excludes the linear case), and if degree = 2, then 64 models are assessed (unlike the 36 models for non-acd transformation). Other settings for degree are currently not supported when used with ACD transformations.

find_scale_factor	<i>Function that calculates an integer used to scale predictor</i>
-------------------	--

Description

Function that calculates an integer used to scale predictor

Usage

```
find_scale_factor(x)
```

Arguments

x	a numeric vector already shifted to positive values (see <code>find_shift_factor()</code>). This function requires at least 2 distinct values to work. #’ @examples x = 1:1000 find_scale_factor(x)
---	---

Details

For details on why scaling is useful see the corresponding section in the documentation of `mfp2()`.

The determination of the scaling factor is independent (i.e. not affected by) shifts in the input data, as it only depends on the range of the input data.

Note that the estimation of powers is unaffected by scaling, the same powers are found for scaled input data. In extreme cases scaling is necessary to preserve accuracy, see Royston and Sauerbrei (2008). This formula uses the scaling formula from Section 4.11.1 of Royston and Sauerbrei (2008). Further information can also be found in the Stata manual for `mfp` at <https://www.stata.com/manuals/rfp.pdf>.

Value

An integer that can be used to scale `x` to a reasonable range. For binary variables 1 is returned.

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.

find_shift_factor	<i>Function that calculates a value used to shift predictor</i>
-------------------	---

Description

Function that calculates a value used to shift predictor

Usage

```
find_shift_factor(x)
```

Arguments

x a numeric vector.

Details

For details on why shifting is necessary see the corresponding section in the documentation of `mfp2()`.

This function implements the formula in Section 4.7 of Royston and Sauerbrei (2008).

Value

A numeric value that can be used to shift x to positive values. If all values are positive, or if x is binary then 0 is returned.

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.

Examples

```
x = 1:1000  
find_shift_factor(x)
```

fit_acd

*Function to estimate approximate cumulative distribution (ACD)***Description**

Fits ACD transformation as outlined in Royston (2014). The ACD transformation smoothly maps the observed distribution of a continuous covariate x onto one scale, namely, that of an approximate uniform distribution on the interval $(0, 1)$.

Usage

```
fit_acd(x, powers = NULL, shift = 0, scale = 1, zero = FALSE)
```

Arguments

<code>x</code>	a numeric vector.
<code>powers</code>	a vector of allowed FP powers. The default value is <code>NULL</code> , meaning that the set $S = (-2, -1, -0.5, 0, 0.5, 1, 2, 3)$ is used.
<code>shift</code>	a numeric that is used to shift the values of x to positive values. The default value is 0, meaning no shifting is conducted. If <code>NULL</code> , then the program will estimate an appropriate shift automatically (see <code>find_shift_factor()</code>).
<code>scale</code>	a numeric used to scale x . The default value is 1, meaning no scaling is conducted. If <code>NULL</code> , then the program will estimate an appropriate scaling factor automatically (see <code>find_scale_factor()</code>).
<code>zero</code>	Logical indicating whether only positive values of the variable should be transformed, with nonpositive values (zero or negative) set to zero. If <code>TRUE</code> , transformation is applied only to positive values; nonpositive values are replaced with zero before transformation. If <code>FALSE</code> (default), all values are shifted (if needed) to ensure positivity before transformation.

Details

Briefly, the estimation works as follows. First, the input data are shifted to positive values and scaled as requested. Then

$$z = \Phi^{-1}\left(\frac{\text{rank}(x) - 0.5}{n}\right)$$

is computed, where n is the number of elements in x , with ties in the ranks handled as averages. To approximate z , an FP1 model (least squares) is used, i.e. $E(z) = \beta_0 + \beta_1(x)^p$, where p is chosen such that it provides the best fitting model among all possible FP1 models. The ACD transformation is then given as

$$\text{acd}(x) = \Phi(\hat{z}),$$

where the fitted values of the estimated model are used. If the relationship between a response Y and $\text{acd}(x)$ is linear, say, $E(Y) = \beta_0 + \beta_1 \text{acd}(x)$, the relationship between Y and x is nonlinear and is typically sigmoid in shape. The parameters β_0 and $\beta_0 + \beta_1$ in such a model are interpreted as the expected values of Y at the minimum and maximum of x , that is, at $\text{acd}(x) = 0$ and 1, respectively. The parameter β_1 represents the range of predictions of $E(Y)$ across the whole observed distribution of x (Royston 2014).

Value

A list is returned with components

- `acd`: the `acd` transformed input data.
- `beta0`: intercept of estimated model.
- `beta1`: coefficient of estimated model.
- `power`: estimated power.
- `shift`: shift value used for computations.
- `scale`: scaling factor used for computations.

References

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

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

Examples

```
set.seed(42)
x = apply_shift_scale(rnorm(100))
y = rnorm(100)
fit_acd(x, y)
```

`fit_cox`

Function that fits Cox proportional hazards models

Description

Function that fits Cox proportional hazards models

Usage

```
fit_cox(
  x,
  y,
  strata,
  weights,
  offset,
  control,
  method,
  rownames,
  nocenter,
  fast = TRUE
)
```

Arguments

x	a matrix of predictors excluding intercept with nobs observations.
y	a Surv object.
strata, control, rownames, nocenter	passed to survival::coxph.fit() .
weights	a numeric vector of length nobs of 'prior weights' to be used in the fitting process.
offset	a numeric vector of length nobs of a priori known component to be included in the linear predictor during fitting.
method	a character string specifying the method for tie handling. See survival::coxph() .
fast	a logical which determines how the model is fitted. The default TRUE uses fast fitting routines (i.e. survival::coxph.fit()), while FALSE uses the normal fitting routines (survival::coxph()) (used for the final output of mfp2).

Value

A list with the following components:

- logl: the log likelihood of the fitted model.
- coefficients: regression coefficients.
- df: number of parameters (degrees of freedom).
- sse: residual sum of squares (not used).
- fit: the fitted model object.

fit_glm

Function that fits generalized linear models

Description

Function that fits generalized linear models

Usage

```
fit_glm(x, y, family, weights, offset, fast = TRUE)
```

Arguments

x	a matrix of predictors with nobs observations.
y	a vector for the outcome variable.
family	a family function e.g. stats::gaussian() .
weights	a numeric vector of length nobs of 'prior weights' to be used in the fitting process. see stats::glm() for details.

offset	a numeric vector of length nobs of of a priori known component to be included in the linear predictor during fitting.
fast	a logical which determines how the model is fitted. The default TRUE uses fast fitting routines (i.e. <code>stats::glm.fit()</code>), while FALSE uses the normal fitting routines (<code>stats::glm()</code>) (used for the final output of <code>mfp2</code>). The difference is mainly due to the fact that normal fitting routines have to handle <code>data.frames</code> , which is a lot slower than using the model matrix and outcome vectors directly.

Value

A list with the following components:

- `logl`: the log likelihood of the fitted model.
- `coefficients`: regression coefficients.
- `df`: number of parameters (degrees of freedom).
- `sse`: residual sum of squares.
- `fit`: the fitted model object.

<code>fit_linear_step</code>	<i>Function to fit linear model for variable of interest</i>
------------------------------	--

Description

"Linear" model here refers to a model that includes the variable of interest `xi` with an FP (fractional polynomial) power of 1. Note that `xi` may be ACD-transformed if indicated by `acdx[xi]`. If the variable was passed through the `catzero` argument in `mfp2()`, both the continuous variable and its corresponding binary indicator will be included in the model as linear terms. For parameter definitions, see `find_best_fp_step`. All parameters captured by `...` are passed to `fit_model`.

Usage

```
fit_linear_step(
  x,
  xi,
  y,
  powers_current,
  powers,
  acdx,
  family,
  family_string,
  zero,
  catzero,
  spike_decision,
  acd_parameter,
  prev_adj_params,
  ...
)
```

Arguments

<code>x</code>	an input matrix of dimensions <code>nobs</code> x <code>nvars</code> . Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
<code>xi</code>	a character string indicating the name of the current variable of interest, for which the best fractional polynomial transformation is to be estimated in the current step.
<code>y</code>	a vector for the response variable or a <code>Surv</code> object.
<code>powers_current</code>	a list of length equal to the number of variables, indicating the fp powers to be used in the current step for all variables (except <code>xi</code>).
<code>powers</code>	a named list of numeric values that sets the permitted FP powers for each co-variate.
<code>acdx</code>	a logical vector of length <code>nvars</code> indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
<code>family</code>	Either a character string naming the family (e.g., "gaussian", "binomial", "cox") or a function that returns a GLM family object (e.g., <code>stats::gaussian</code>). For Cox models, only a character string "cox" is allowed.
<code>family_string</code>	A character string representing the selected family, e.g., "gaussian".
<code>zero</code>	A named logical vector indicating, which columns of <code>x</code> should treat nonpositive values (zero or negative) as zero before transformation. Must be the same length as the columns of <code>x</code> .
<code>catzero</code>	A named list of binary indicator variables of length <code>ncol(x)</code> for nonpositive values, created when specific variables are passed to the <code>catzero</code> argument of <code>fit_mfp</code> . If an element of the list is <code>NULL</code> , it indicates that the corresponding variable was not specified by the user in the <code>catzero</code> argument of <code>fit_mfp</code> . Here, <code>catzero</code> is a list of binary variables, not a named logical vector as in <code>fit_mfp</code> .
<code>spike_decision</code>	Named vector indicating how spike-at-zero (SAZ) variables are handled. Each element corresponds to a variable and encodes the selected strategy: 1 = include FP for positive values plus binary SAZ, 2 = treat as continuous FP only, 3 = include binary SAZ only.
<code>acd_parameter</code>	Named list of ACD parameters produced by <code>fit_acd()</code> , with length equal to <code>ncol(x)</code> . Each list element corresponds to a variable; if an element is <code>NULL</code> , the variable was not specified in the <code>acdx</code> argument of <code>fit_mfp</code> .
<code>prev_adj_params</code>	Named list storing adjustment variable transformations from previous steps for each variable.
<code>...</code>	Parameters passed to <code>fit_model()</code> .

Value

A list with two entries:

- `powers`: FP power(s) of `xi` (or its ACD transformation) in fitted model.
- `metrics`: A matrix with performance indices for fitted model.
- `prev_adj_params`: Previously adjusted parameters.

fit_model

*Function that fits models supported by mfp2***Description**

Fits generalized linear models and Cox proportional hazard models.

Usage

```
fit_model(
  x,
  y,
  family,
  weights = NULL,
  offset = NULL,
  method = NULL,
  strata = NULL,
  control = NULL,
  rownames = NULL,
  nocenter = NULL,
  fast = TRUE
)
```

Arguments

x	a matrix of predictors (excluding intercept) with column names. If column names are not provided they are set according to <code>colnames(x, do.NULL = FALSE)</code> .
y	a vector for the outcome variable for glms, and a <code>Surv</code> object for Cox models.
family	a character string specifying glm family to be used, or "cox" for Cox models. The default family is set to 'Gaussian'.
method	a character string specifying the method for tie handling. See survival::coxph() .
strata, control, weights, offset, rownames, nocenter	parameters for Cox or glm. See survival::coxph() or stats::glm() for details.
fast	passed to <code>fit_glm()</code> and <code>fit_cox()</code> .

@return A list with the following components:

- logl: the log likelihood of the fitted model.
- coefficients: regression coefficients.
- df: number of parameters (degrees of freedom).
- sse: residual sum of squares.
- fit: the object returned by the fitting procedure.

Details

Computations rely on `fit_glm()` and `fit_cox()`.

fit_null_step

*Function to fit a null model excluding variable of interest***Description**

"Null" model here refers to a model which does not include the variable of interest `xi`. For parameter definitions, see `find_best_fp_step()`. All parameters captured by `...` are passed on to `fit_model()`.

Usage

```
fit_null_step(
  x,
  xi,
  y,
  powers_current,
  powers,
  acdx,
  family,
  family_string,
  zero,
  catzero,
  spike_decision,
  acd_parameter,
  prev_adj_params,
  ...
)
```

Arguments

<code>x</code>	an input matrix of dimensions <code>nobs x nvars</code> . Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
<code>xi</code>	a character string indicating the name of the current variable of interest, for which the best fractional polynomial transformation is to be estimated in the current step.
<code>y</code>	a vector for the response variable or a <code>Surv</code> object.
<code>powers_current</code>	a list of length equal to the number of variables, indicating the fp powers to be used in the current step for all variables (except <code>xi</code>).
<code>powers</code>	a named list of numeric values that sets the permitted FP powers for each co-variate.
<code>acdx</code>	a logical vector of length <code>nvars</code> indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
<code>family</code>	Either a character string naming the family (e.g., "gaussian", "binomial", "cox") or a function that returns a GLM family object (e.g., <code>stats::gaussian</code>). For Cox models, only a character string "cox" is allowed.

family_string	A character string representing the selected family, e.g., "gaussian".
zero	A named logical vector indicating, which columns of <code>x</code> should treat nonpositive values (zero or negative) as zero before transformation. Must be the same length as the columns of <code>x</code> .
catzero	A named list of binary indicator variables of length <code>ncol(x)</code> for nonpositive values, created when specific variables are passed to the <code>catzero</code> argument of <code>fit_mfp</code> . If an element of the list is <code>NULL</code> , it indicates that the corresponding variable was not specified by the user in the <code>catzero</code> argument of <code>fit_mfp</code> . Here, <code>catzero</code> is a list of binary variables, not a named logical vector as in <code>fit_mfp</code> .
spike_decision	Named vector indicating how spike-at-zero (SAZ) variables are handled. Each element corresponds to a variable and encodes the selected strategy: 1 = include FP for positive values plus binary SAZ, 2 = treat as continuous FP only, 3 = include binary SAZ only.
acd_parameter	Named list of ACD parameters produced by <code>fit_acd()</code> , with length equal to <code>ncol(x)</code> . Each list element corresponds to a variable; if an element is <code>NULL</code> , the variable was not specified in the <code>acdx</code> argument of <code>fit_mfp</code> .
prev_adj_params	Named list storing adjustment variable transformations from previous steps for each variable.
...	Parameters passed to <code>fit_model()</code> .

Value

A list with two entries:

- `powers`: FP power(s) of `xi` in fitted model - in this case NA.
- `metrics`: A matrix with performance indices for fitted model.

fp	<i>Helper to assign attributes to a variable undergoing FP-transformation</i>
----	---

Description

Used in formula interface to `mfp2()`.

Usage

```
fp(
  x,
  df = 4,
  alpha = 0.05,
  select = 0.05,
  shift = NULL,
```

```

    scale = NULL,
    center = TRUE,
    acdx = FALSE,
    powers = NULL,
    zero = FALSE,
    catzero = FALSE,
    spike = FALSE
  )

  fp2(...)

```

Arguments

x a vector representing a continuous variable undergoing fp-transformation.

df, alpha, select, shift, scale, center, acdx, zero, catzero, spike
See `mfp2()` for details.

powers a vector of powers to be evaluated for x. Default is NULL and `powers = c(-2, -1, -0.5, 0, 0.5, 1, 2, 3)` will be used.

... used in alias `fp2` to pass arguments.

Value

The vector `x` with new attributes relevant for fp-transformation. All arguments passed to this function will be stored as attributes.

Functions

- `fp2()`: Alias for `fp()` - use in formula when both `mfp` and `mfp2` are loaded to avoid name shadowing.

Examples

```

xr = 1:10
fp(xr)
fp2(xr)

```

fracplot

Plot response functions from a fitted mfp2 object

Description

Produces partial predictor plots (or contrasts) with confidence intervals against selected covariates from a fitted ``mfp2`` model. If requested, component-plus-residual plots are also supported.

Usage

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

plot_mfp(...)
```

Arguments

<code>model</code>	A fitted <code>`mfp2`</code> model.
<code>terms</code>	Character vector with variable names to be plotted. If <code>NULL</code> , all fractional polynomial terms in the model are plotted.
<code>partial_only</code>	Logical. If <code>TRUE</code> , only the partial predictor (model component) is plotted. If <code>FALSE</code> (default), component-plus-residual plots are drawn. Only used if <code>type = "terms"</code> . See below for details.
<code>type</code>	Character, one of <code>"terms"</code> or <code>"contrasts"</code> . Passed to <code>predict.mfp2()</code> . <code>"terms"</code> plots partial predictors (with or without residuals), while <code>"contrasts"</code> plots contrasts relative to a reference value.
<code>ref</code>	Reference list passed to <code>predict.mfp2()</code> when <code>type = "contrasts"</code> . Ignored otherwise.
<code>terms_seq</code>	Character, one of <code>"data"</code> or <code>"equidistant"</code> . Passed to <code>predict.mfp2()</code> . <code>"data"</code> uses the observed values, <code>"equidistant"</code> creates a grid over the covariate range.
<code>alpha</code>	Confidence level for intervals. Passed to <code>predict.mfp2()</code> .
<code>color_points</code>	Character value. Color of points used when residuals are displayed.
<code>color_line</code>	Character value. Color of the line representing the partial predictor.
<code>color_line_spike</code>	Character value. Color of the point drawn at zero when the covariate includes a spike-at-zero component (<code>spike_decision = 1</code>).
<code>color_fill</code>	Character value. Fill color of the confidence interval ribbon.

shape	Numeric value. Shape of points used when residuals are displayed.
size_points	Numeric value. Size of points used when residuals are displayed.
size_points_spike	Numeric value. Size of the point drawn at zero when the covariate includes a spike-at-zero component (<code>spike_decision = 1</code>).
linetype	Character value. Line type for the partial predictor. See <code>ggplot2::geom_line()</code> for options.
linewidth	Numeric value. Width of the line representing the partial predictor.
alpha_fill	Numeric value between 0 and 1. Transparency of the confidence interval ribbon.
...	Further arguments passed from the alias <code>plot_mfp()</code> .

Details

Confidence intervals are based on the variance–covariance matrix of the final fitted model. They reflect uncertainty in the regression coefficients but not in the selection of fractional polynomial powers. Intervals may therefore be too narrow. A bootstrap approach (not yet implemented) is recommended for more realistic intervals (see Royston & Sauerbrei, 2008, Section 4.9.2).

Component-plus-residual plots are available if `type = "terms"`. Deviance residuals are used for generalized linear models, while martingale residuals are used for Cox regression. This matches the behavior of the Stata `mfp` program.

Spike-at-zero covariates are handled according to the `spike_decision` code:

- 1 – Include both the transformed FP function for positive values and the binary spike-at-zero indicator.
- 2 – Ignore the spike; treat the variable as continuous (usual FP plot).
- 3 – Show only the binary spike-at-zero indicator.

Plot behavior for each decision:

- If `spike_decision == 1`, the plot shows the FP function for positive values and includes the binary spike-at-zero indicator. The term $\hat{\beta}_0 + \hat{\beta}$ for observations equal to zero is also displayed with a vertical error bar. The plot title includes + z to indicate the presence of the spike-at-zero component. The FP power for the positive part is enclosed in parentheses. For example, $FP(0) + z$ indicates an FP power of 0 (log) for the positive values.
- If `spike_decision == 3`, the plot shows the binary indicator alone (z only in the title). Mean values at 0 and 1 are connected with a line, and a ribbon showing confidence intervals is displayed.
- If `spike_decision == 2` (or not specified), the covariate is plotted as a continuous FP function in the usual way. See `fracplot` for details on partial predictors

Value

A list of `ggplot2` plot objects, one for each term requested. Can be drawn as individual plots or faceted / combined easily using e.g. `patchwork::wrap_plots` and further customized.

Functions

- `plot_mfp()`: Alias for `fracplot`.

See Also

```
predict.mfp2()
```

Examples

```
# Gaussian response
data("prostate")
x = as.matrix(prostate[,2:8])
y = as.numeric(prostate$lpsa)
# default interface
fit = mfp2(x, y, verbose = FALSE)
fracplot(fit) # generate plots
```

gbsg

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

Description

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

Usage

```
data(gbsg)
```

Format

A dataset with 686 observations and 11 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 $\exp(-0.12 \cdot \text{nodes})$.

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).

generate_combinations_with_replacement

Helper function to generate combinations with replacement

Description

This helper generates combinations with replacement.

Usage

```
generate_combinations_with_replacement(x, k)
```

Arguments

x	Vector of elements to choose from.
k	Number of elements to choose.

Details

This function replicates the functionality of `arrangements::combinations(x, k, replace = TRUE)`. Unlike `utils::combn(x, k)`, which returns combinations without replacement, this function allows repeated elements, so combinations such as (0, 0) are included in the output.

Internally, the function uses a stars-and-bars indexing construction. It first applies `utils::combn()` to the sequence `1:(length(x) + k - 1)` and then shifts the resulting indices by `1:k`. This produces the indices needed to select nondecreasing combinations from the sorted input vector `x`.

This avoids generating all ordered tuples with `expand.grid()` and then removing duplicates. However, the number of combinations can still grow quickly with increasing `k`. In the MFP context, high FP degrees correspond to a large number of possible FP power combinations, and the subsequent model selection step may therefore be computationally intensive. A warning is issued for `k > 5`.

Value

A matrix with one row per combination and `k` columns.

generate_powers_fp

Function that generates a matrix of FP powers for any degree

Description

Function that generates a matrix of FP powers for any degree

Usage

```
generate_powers_fp(degree = NULL, powers = NULL)
```

```
generate_powers_acd(degree = NULL, powers = NULL)
```

Arguments

degree	The degree of fractional polynomial. For example, degree = 1 is FP1 and returns 8 powers; degree 2 is FP2 and returns 36 pairs of powers; degree 3 is FP3 and returns 120 triples of powers, and so on. If the ACD transformation is used, this degree is assumed to be 2.
powers	the set of allowed powers for the fractional polynomials. Default is NULL and the set $(-2, -1, -0.5, 0, 0.5, 1, 2, 3)$ is used.

Details

For FP powers, this function returns all combinations of the powers of length degree, that is all pairs in which each entry is taken from the set powers, but no pair is repeated (i.e. the order of the entries does not matter). Thus, for the default set of powers and degree 2, this function returns 36 combinations.

For ACD powers, this function simply returns all possible tuples of powers of length n. Thus, for the default set of powers, this function returns 8 possible powers, and for degree 2 it returns 64 pairs of powers. Higher degrees are not supported by the function. In case that degree = 0 or degree = 1, the first column of the matrix representing untransformed data are set to NA to indicate that the normal data do not play a role. Higher degrees than two are not supported.

Value

A matrix of powers with degree columns and rows depending on the degree. For ACD powers always a matrix with two columns. For normal fps each row will be sorted in increasing order (in alignment with how transform_vector_fp() processes the data).

Functions

- generate_powers_acd(): Function to generate acd powers.

Examples

```
powx <- c(-2, -1, -0.5, 0, 0.5, 1, 2, 3)
generate_powers_fp(degree = 2, powers = powx)
generate_powers_acd(degree = 2, powers = powx)
```

generate_transformations_fp

Function to generate all requested FP transformations for a single variable

Description

Function to generate all requested FP transformations for a single variable

Usage

```
generate_transformations_fp(x, degree, powers, zero, catzero = NULL)
```

```
generate_transformations_acd(
  x,
  degree,
  powers,
  zero,
  catzero = NULL,
  acd_parameter = NULL
)
```

Arguments

x	A numeric vector of length nob, assumed to have been shifted (except for variables with zero or catzero transformations) and scaled.
degree	A numeric value indicating the degree of fractional polynomials (FPs). For ACD transformation, this is assumed to be 2.
powers	A numeric vector specifying the set of allowed fractional polynomial (FP) powers to be used in the transformation.
zero	Logical indicating whether only positive values of the variable should be transformed, with nonpositive values (zero or negative) set to zero. If TRUE, transformation is applied only to positive values; nonpositive values are replaced with zero before transformation. If FALSE (default), all values are shifted (if needed) to ensure positivity before transformation.
catzero	A vector of binary values to be added to the transformed variables. Default is NULL, meaning no binary variables added.
acd_parameter	Optional named list of ACD parameters, generated by <code>fit_acd()</code> . If NULL, the function generates it.

Details

Any fractional polynomial (FP) transformation is defined by a vector of powers, such as (p1, p2) for degree 2. These correspond to the terms x^{p1} and x^{p2} . Therefore, all combinations of the values in powers are considered (see `generate_powers_fp`).

A special case arises when powers are repeated, i.e., $p1 = p2$. In such cases, the second term is multiplied by $\log(x)$, following the standard FP convention (see `transform_vector_fp`).

When the ACD transformation is requested, all pairs of powers of length 2 are evaluated, resulting in 64 unique combinations (see `generate_powers_acd`).

If `degree = 0` then these functions return the data unchanged for fp, or simply the acd transformation of the input variable, i.e. in both cases the power is set to 1 (linear).

If `degree = 0`, the function returns the data unchanged for an FP transformation, or applies only the ACD transformation to the input variable. In both cases, the power is set to 1 (linear).

When `catzero` is used, the transformed (or untransformed) continuous variable is combined with its corresponding binary indicator, representing whether the original value was positive or nonpositive.

Value

A list with two components:

- **data**: A list with length equal to the number of possible fractional polynomial (FP) transformations for the variable of interest. Each entry is a matrix with `nobs` rows. The number of columns equals the FP degree, unless `catzero = TRUE`, in which case an additional column is included for the binary indicator variable. For example, with `degree = 2`, `catzero = TRUE`, and `nobs = 10`, each entry is a 10×3 matrix. The FP-transformed values are not centered. If `degree = 0`, the list contains a single entry with one column (or two columns if `catzero = TRUE`), representing the linear transformation (and binary indicator, if applicable).
- **data**: the associated FP powers for each entry in `data`.
- **powers**: A matrix of FP powers corresponding to each entry in `data`. Each row contains the powers used for the associated transformation (e.g., two columns for `degree = 2`, one for `degree = 1`, and one for `degree = 0`).

Functions

- `generate_transformations_acd()`: Function to generate acd transformations.

`get_selected_variable_names`

Helper function to extract selected variables from fitted mfp2 object

Description

Simply extracts all variables for which not all powers are estimated to be NA. The names refer to the original names in the dataset and do not include transformations.

Usage

```
get_selected_variable_names(object)
```

Arguments

`object` fitted mfp2 object.

Value

Character vector of names, ordered as defined by `xorder` in `mfp2()`.

Examples

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

mfp2

Multivariable Fractional Polynomial Models with Extensions

Description

Selects the multivariable fractional polynomial (MFP) model that best predicts the outcome variable. It also supports approximate cumulative distribution (ACD) transformations for continuous variables, which enable modeling of sigmoid relationships between predictors x and the outcome y (Royston, 2014; Royston and Sauerbrei, 2016). In addition, it implements spike-at-zero (SAZ) modeling, a method for appropriately handling semi-continuous predictors with a non-negligible proportion of zero values (Becher et al., 2012). The function provides two interfaces for input data: one for supplying the data matrix x and the outcome y directly, where y may be a numeric vector for continuous or binary outcomes, or a `Surv` object for Cox proportional hazards models; and another for using a formula object together with a dataframe `data`. Both interfaces are equivalent in functionality.

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 = c("pvalue", "aic", "bic"),
  select = 0.05,
  alpha = 0.05,
  keep = NULL,
```

```

xorder = c("ascending", "descending", "original"),
powers = NULL,
ties = c("breslow", "efron", "exact"),
strata = NULL,
nocenter = NULL,
acdx = NULL,
ftest = FALSE,
control = NULL,
zero = NULL,
catzero = NULL,
spike = NULL,
min_prop = 0.05,
max_prop = 0.95,
verbose = TRUE,
...
)

## 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 = c("pvalue", "aic", "bic"),
  select = 0.05,
  alpha = 0.05,
  keep = NULL,
  xorder = c("ascending", "descending", "original"),
  powers = NULL,
  ties = c("breslow", "efron", "exact"),
  strata = NULL,
  nocenter = NULL,
  ftest = FALSE,
  control = NULL,
  min_prop = 0.05,
  max_prop = 0.95,
  verbose = TRUE,
  ...
)

```

Arguments

<code>x</code>	for <code>mfp2.default</code> : <code>x</code> is an input matrix of dimensions <code>nobs</code> x <code>nvars</code> . Each row is an observation vector.
<code>...</code>	Not used.
<code>y</code>	for <code>mfp2.default</code> : <code>y</code> is a vector for the response variable. For <code>family = "binomial"</code> it should be a vector with two levels (see <code>stats::glm()</code>). For <code>family = "cox"</code> it must be a <code>survival::Surv()</code> object containing 2 columns.
<code>weights</code>	a vector of observation weights of length <code>nobs</code> . Default is <code>NULL</code> which assigns a weight of 1 to each observation.
<code>offset</code>	a vector of length <code>nobs</code> that is included in the linear predictor. Useful for the poisson family (e.g. log of exposure time). Default is <code>NULL</code> which assigns an offset of 0 to each observation. If supplied, then values must also be supplied to the <code>predict()</code> function.
<code>cycles</code>	an integer, specifying the maximum number of iteration cycles. Default is 5.
<code>scale</code>	a numeric vector of length <code>nvars</code> or single numeric specifying scaling factors. If a single numeric, then the value will be replicated as necessary. The formula interface <code>mfp2.formula</code> only supports single numeric input to set a default value, individual values can be set using <code>fp</code> terms in the formula input. Default is <code>NULL</code> which lets the program estimate the scaling factors (see Details section). If scaling is not required set <code>scale = 1</code> to disable it. The final regression coefficients are expressed in the original scale of the data.
<code>shift</code>	a numeric vector of length <code>nvars</code> or a single numeric specifying shift terms. If a single numeric, then the value will be replicated as necessary. The formula interface <code>mfp2.formula</code> only supports single numeric input to set a default value, individual values can be set using <code>fp</code> terms in the formula input. Default is <code>NULL</code> which lets the program estimate the shifts (see Details section). If shifting is not required, set <code>shift = 0</code> to disable it.
<code>df</code>	a numeric vector of length <code>nvars</code> or a single numeric that sets the (default) degrees of freedom (<code>df</code>) for each predictor. If a single numeric, then the value will be replicated as necessary. The formula interface <code>mfp2.formula</code> only supports single numeric input to set a default value, individual values can be set using <code>fp</code> terms in the formula input. The <code>df</code> (not counting the intercept) are twice the degree of a fractional polynomial (FP). For example, an FP2 has 4 <code>df</code> , while FPM has $2*m$ <code>df</code> . The program overrides default <code>df</code> based on the number of distinct (unique) values for a variable as follows: 2-3 distinct values are assigned <code>df = 1</code> (linear), 4-5 distinct values are assigned <code>df = min(2, default)</code> and ≥ 6 distinct values are assigned <code>df = default</code> .
<code>center</code>	a logical determining whether variables are centered before final model fitting. The default <code>TRUE</code> implies mean centering, except for binary covariates, where the covariate is centered using the lower of the two distinct values of the covariate. See Details section below.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process. Default is <code>NULL</code> and all observations are used. See Details below.
<code>family</code>	Either a character string specifying the model family (e.g., "gaussian", "binomial", "poisson", "cox") or a function that returns a GLM family object,

	such as <code>stats::gaussian(link = "identity")</code> or <code>stats::binomial(link = "logit")</code> . For Cox models, only the character string "cox" is allowed.
criterion	a character string specifying the criterion used to select variables and FP models of different degrees. Default is to use p-values in which case the user can specify the nominal significance level (or use default level of 0.05) for variable and functional form selection (see <code>select</code> and <code>alpha</code> parameters below). If the user specifies the BIC (<code>bic</code>) or AIC (<code>aic</code>) criteria the program ignores the nominal significance levels and selects variables and functional forms using the chosen information criterion.
select	a numeric vector of length <code>nvars</code> or a single numeric that sets the nominal significance levels for variable selection on each predictor by backward elimination. If a single numeric, then the value will be replicated as necessary. The formula interface <code>mfp2.formula</code> only supports single numeric input to set a default value, individual values can be set using <code>fp</code> terms in the formula input. The default nominal significance level is 0.05 for all variables. Setting the nominal significance level to be 1 for certain variables forces them into the model, leaving all other variables to be selected.
alpha	a numeric vector of length <code>nvars</code> or a single numeric that sets the significance levels for testing between FP models of different degrees. If a single numeric, then the value will be replicated as necessary. The formula interface <code>mfp2.formula</code> only supports single numeric input to set a default value, individual values can be set using <code>fp</code> terms in the formula input. The default nominal significance level is 0.05 for all variables.
keep	a character vector with names of variables to be kept in the model. In case that <code>criterion = "pvalue"</code> , this is equivalent to setting the selection level for the variables in <code>keep</code> to 1. However, this option also keeps the specified variables in the model when using the BIC or AIC criteria.
xorder	a string determining the order of entry of the covariates into the model-selection algorithm (backfitting algorithm). The default is ascending, which enters them by ascending p-values, or decreasing order of significance in a multiple regression (i.e. most significant first). descending places them in reverse significance order, whereas <code>original</code> respects the original order in <code>x</code> .
powers	a named list of numeric values specifying the set of candidate FP powers for each covariate. The default is <code>NULL</code> , in which case every covariate is assigned <code>powers = c(-2, -1, -0.5, 0, 0.5, 1, 2, 3)</code> , where 0 denotes the natural logarithm. Powers are sorted before further processing. If some variables are not explicitly assigned powers, the default set is used. In the formula interface, powers can be specified either through the <code>powers</code> argument or within the <code>fp()</code> function. If both are provided for the same variable, the specification inside <code>fp()</code> takes precedence. Each variable must be assigned at least two candidate powers for model selection. To restrict a variable to a single transformation, pre-transform the variable and fit it with <code>df = 1</code> .
ties	a character string specifying the method for tie handling in Cox regression. If there are no tied death times all the methods are equivalent. Default is the Breslow method. This argument is used for Cox models only and has no effect on other model families. See <code>survival::coxph()</code> for details.

strata	a numeric vector or matrix of variables that define strata to be used for stratification in a Cox model. A new factor, whose levels are all possible combinations of the variables supplied will be created. Default is NULL and a Cox model without stratification would be fitted. See <code>survival::coxph()</code> for details.
nocenter	a numeric vector with a list of values for fitting Cox models. See <code>survival::coxph()</code> for details.
acdx	a character vector giving the names of continuous variables to undergo the approximate cumulative distribution (ACD) transformation. Using this also triggers the function-selection procedure for ACD (FSPA) to determine the best-fitting FP1(p1, p2) model (see Details). This argument is not available in the formula interface (<code>mfp2.formula</code>), where the user should instead use <code>fp()</code> . Within <code>fp()</code> terms, the ACD transformation is specified via the logical argument <code>acdx</code> rather than by listing variable names. The variable representing the ACD transformation of <code>x</code> is named <code>A_x</code> .
ftest	a logical indicating whether <code>mfp2</code> should use critical values from the F-distribution instead of the Chi-Square distribution for small-sample normal error models. This affects variable selection, functional form selection, and spike-at-zero testing when evaluating fractional polynomial terms. Default is <code>FALSE</code> , in which case the Chi-Square distribution is used. This argument applies only to Gaussian models.
control	a list of parameters controlling the fitting process, as returned by <code>stats::glm.control()</code> or <code>survival::coxph.control()</code> . Default is NULL, in which case <code>mfp2</code> uses the default settings for the specified model family.
zero	A character vector specifying the names of continuous variables for which non-positive values should be treated as zero. This enables fitting a fractional polynomial (FP) model using only the positive values of the covariate, while setting nonpositive values to zero during transformation. In <code>fp()</code> terms, <code>zero</code> is a logical value indicating whether the adjustment should be applied to the variable. See the Details section.
catzero	A character vector specifying the names of continuous variables for which non-positive values should be treated as zero, similar to <code>zero</code> . In addition, <code>mfp2</code> automatically creates a binary indicator variable and includes it in the model alongside the transformed variable. In <code>fp()</code> terms, <code>catzero</code> is a logical value indicating whether the adjustment and indicator should be applied. See the Details section.
spike	A character vector specifying the names of continuous variables to be assessed for a spike at zero. Supplying this triggers the spike-at-zero (SAZ) algorithm (see Details). This argument is not available in the formula interface (<code>mfp2.formula</code>), where <code>spike</code> is specified as a logical value within <code>fp()</code> terms for each variable.
min_prop	A numeric value between 0 and 1; the minimum proportion of zeros for which the spike-at-zero (SAZ) modeling is applied. Defaults to 0.05.
max_prop	A numeric value between 0 and 1; the maximum proportion of zeros for which SAZ modeling is applied. Defaults to 0.95. before calling this function, for example with
verbose	Logical specifying whether to print progress messages. Default is <code>FALSE</code> .

formula	for mfp2.formula: an object of class formula: a symbolic description of the model to be fitted. Special fp terms can be used to define fp-transformations. The details of model specification are given under ‘Details’.
data	for mfp2.data: a data.frame which contains all variables specified in formula.

Value

mfp2() returns an object of class inheriting from glm or copxh, depending on the family parameter. The function summary() (i.e. summary.mfp2()) can be used to obtain or print a summary of the results. The generic accessor function coef() can be used to extract the vector of coefficients from the fitted model object. The generic predict() can be used to obtain predictions from the fitted model object.

An object of class mfp2 is a list containing all entries as for glm or coxph, and in addition the following entries:

- convergence_mfp: logical value indicating convergence of mfp algorithm.
- fp_terms: a data.frame with information on fractional polynomial terms.
- transformations: a data.frame with information on shifting, scaling and centering for all variables.
- fp_powers: a list with all powers of fractional polynomial terms. Each entry of the list is named according to the transformation of the variable.
- acd: a vector with information for which variables the acd transformation was applied.
- x_original: the scaled and shifted input matrix but without transformations.
- y: the original outcome variable.
- x: the final transformed input matrix used to fit the final model.
- call_mfp: the call to the mfp2() function.
- family: the family stored as either character string or function.
- family_string: the family stored as character string.
- zero: named logical vector indicating, for each variable, whether only positive values were transformed.
- catzero: named logical vector indicating which columns in x treated nonpositive values as zero and included an additional binary indicator in the model.
- catzero_list: A list of binary variables created when catzero is set to TRUE. Returns NULL if catzero is FALSE.
- spike_decision: named numeric vector with values 1, 2, or 3 specifying spike-at-zero handling for each variable. Value 1 includes both the transformed variable and a binary indicator, 2 disables the spike and binary indicator, and 3 retains only the binary indicator.

The mfp2 object may contain further information depending on family.

Methods (by class)

- mfp2(default): Default method using input matrix x and outcome vector y.
- mfp2(formula): Provides formula interface for mfp2.

Brief summary of fractional polynomials (FPs)

Fractional polynomials (FPs) provide a flexible framework for modeling nonlinear relationships between a continuous predictor x and an outcome. In general, we denote an FP of degree m as $FPm(p_1, \dots, p_m)$, where $p_1, \dots, p_m$ are the selected powers and $m \geq 1$.

The most commonly used cases are:

- **FP1:** a single-term transformation, $FP1(p_1) = \beta_1 x^{p_1}$, representing the simplest FP model.
- **FP2:** a two-term transformation, $FP2(p_1, p_2) = \beta_1 x^{p_1} + \beta_2 x^{p_2}$, where $p_1 \neq p_2$, providing greater flexibility to capture nonlinear effects.

When $p_1 = p_2$ (repeated powers), the FP2 model is defined as

$$FP2(p_1, p_2) = \beta_1 x^{p_1} + \beta_2 x^{p_1} \log(x).$$

The powers p_1 and p_2 are usually chosen from a predefined set $S = \{-2, -1, -0.5, 0, 0.5, 1, 2, 3\}$, where a power of 0 indicates the natural logarithm. The best FP2 model is then selected using a closed testing procedure that evaluates all 36 pairs of powers (p_1, p_2) .

For further details, see Sauerbrei et al. (2006) and Royston and Sauerbrei (2008). For the effects of influential points on FP functions, see Sauerbrei et al. (2023).

Details on family option

`mfp2()` supports the family argument as used by `stats::glm()`. Families can be specified either as a character string or as a function returning a GLM family object (e.g., `stats::gaussian(link = "identity")` or `stats::binomial(link = "logit")`).

Only the following families are supported at the moment: "gaussian", "binomial", "poisson", and "cox".

Examples with character strings: `mfp2(..., family = "binomial")` fits a logistic regression model. `mfp2(..., family = "gaussian")` fits a linear regression model using ordinary least squares.

Examples with family functions and custom links: `mfp2(..., family = gaussian(link = "log"))` fits a linear model with a log link. `mfp2(..., family = binomial(link = "probit"))` fits a logistic regression model with a probit link.

For Cox proportional hazards models, the response should be a `Surv` object (created with `survival::Surv()`), and the family argument should be set to "cox". Only right-censored data are currently supported. Stratified Cox models can be specified using the `strata` argument, or by including `strata` terms in the model formula when using the formula interface `mfp2.formula`.

Details on shifting, scaling, centering

Fractional polynomials are defined only for positive variables due to the use of logarithms and other powers. Thus, `mfp2()` estimates shifts for each variables to ensure positivity or assumes that the variables are already positive when computing fractional powers of the input variables in case that shifting is disabled manually.

If the values of the variables are too large or too small, it is important to conduct variable scaling to reduce the chances of numerical underflow or overflow which can lead to inaccuracies and difficulties in estimating the model. Scaling can be done automatically or by directly specifying the

scaling values so that the magnitude of the x values are not too extreme. By default scaling factors are estimated by the program as follows.

After adjusting the location of x so that its minimum value is positive, creating x' , automatic scaling will divide each value of x' by 10^p where the exponent p is given by

$$p = \text{sign}(k) \times \text{floor}(|k|) \quad \text{where} \quad k = \log_{10}(\max(x') - \min(x'))$$

After the final FP powers are estimated, the program backscals x' to the original scale x , ensuring that the final regression coefficients are expressed in the original scale of the data. The FP transformation of x is centered on the mean of the observed values of x . For example, for the FP1 model $\beta_0 + \beta_1 x^p$, the actual model fitted by the software would be $\beta'_0 + \beta'_1 (x^p - \text{mean}(x^p))$. This approach ensures that the revised constant β'_0 or baseline hazard function in a Cox model retains a meaningful interpretation.

So in brief: shifting is required to make input values positive, scaling helps to bring the values to a reasonable range. Both operations are conducted before estimating the FP powers for an input variable. Centering, however, is done after estimating the FP functions for each variable.

Additionally, any variable marked as zero, catzero, or with degrees of freedom equal to 1 ($df = 1$) has its shift automatically set to zero. This ensures that variables for which transformations are unnecessary like linear are not artificially shifted.

Centering before estimating the FP powers may result in different powers and should be avoided. Also see `transform_vector_fp()` for some more details.

Details on the subset argument

Subsetting in `mfp2()` occurs after data pre-processing (shifting, scaling, or centering) but before model selection and fitting. Specifically, if the `subset` option is used and `scale`, `shift`, or `centering` parameters need to be estimated, `mfp2()` first estimates these parameters using the full dataset (without subsetting). The subset is then applied before performing model selection and fitting on the specified portion of the data.

Consequently, subsetting within `mfp2()` is not equivalent to subsetting the data prior to calling the function. It should not be used to implement tasks such as cross-validation or removal of NA values, which should be handled by the user beforehand. The `subset` argument is primarily useful when the same pre-processing should be applied to multiple subsets. For example, one might estimate separate models for women and men while using identical data pre-processing (e.g., centering or scaling) across both subsets. In this case, `subset` can restrict model selection to the chosen group while retaining consistent pre-processing parameters.

mfp2 formula interface

The `mfp2.formula()` constructs a model frame from the supplied formula and data, processes all predictor variables, and fits a multivariable fractional polynomial model by calling `mfp2.default()` internally.

The formula interface automatically handles several tasks:

- detection and transformation of variables wrapped in `fp()`;
- creation of dummy variables for factors using `model.matrix()`;
- synchronization of arguments such as `select` and `keep` with the expanded model matrix.

When the user specifies a variable name in `keep`, all corresponding dummy variables generated from that variable are automatically included and protected from exclusion during model selection.

Handling of Categorical Variables

The `mfp2.formula()` method expands categorical variables through `model.matrix`, ensuring that the model is fitted on a fully numeric design matrix. This affects how predictors are represented internally and how arguments such as `keep` and `select` are interpreted.

Unordered Factors:

Unordered categorical variables (created using `factor()`) are encoded using the default treatment contrasts (`contr.treatment`). The first level of the factor is treated as the reference category, and a dummy variable is created for each of the remaining levels. For example:

```
x <- factor(c("A", "B", "C"))
model.matrix(~ x)
# (Intercept) xB xC
```

When the argument `keep` includes the name of a categorical variable, all dummy variables derived from that factor (e.g., `xB`, `xC`) are automatically retained in the model. The corresponding entries in `select` are internally set to 1 to prevent their removal during variable selection.

Ordered Factors:

Ordered categorical variables (created using `ordered()`) are, by default, encoded using polynomial contrasts (`contr.poly`), which produce orthogonal contrasts representing trend effects across ordered levels (e.g., `x.L`, `x.Q`, `x.C`).

If the analysis requires dummy coding that reflects ordinal thresholds—for instance, comparing level A versus the rest, or levels A and B versus the rest—the user must define and assign an appropriate contrast matrix before calling `mfp2.formula()`. This can be done using the `contr.cumulative()` function provided in this package:

```
data$x <- ordered(data$x, levels = c("A", "B", "C", "D"))
contrasts(data$x) <- contr.cumulative(levels(data$x))
fit <- mfp2(y ~ x, data = data)
```

This approach generates cumulative dummy variables that preserve the ordinal nature of the variable while allowing for interpretable threshold-type effects. Users should ensure that the contrast specification reflects the intended interpretation prior to fitting the model.

The `mfp2.default()` Method:

The default method `mfp2.default()` does not perform any automatic expansion of factor variables. It assumes that the input design matrix `x` consists solely of numeric predictors. Therefore, when calling `mfp2.default()` directly, the user must create any required dummy or contrast-coded variables manually before model fitting.

Details on approximate cumulative distribution transformation

The approximate cumulative distribution (ACD) transformation is a method to model continuous covariates flexibly in regression models. Instead of including the raw variable X directly, a smooth function approximating its empirical cumulative distribution function (`ecdf`) is used.

Method

Let $x_1, \dots, x_n$ be a sample from the distribution of X . The ACD transformation proceeds in three steps:

1. **Inverse normal transformation of ranks:** Compute the rank of each x_i in the sample and transform it using the standard normal inverse CDF (probit):

$$z_i = \Phi^{-1}\left(\frac{\text{rank}(x_i) - 0.5}{n}\right),$$

where Φ^{-1} is the inverse standard normal CDF. This maps the empirical distribution of X to approximately standard normal values.

2. **Power-linear approximation:** Fit a one-term fractional polynomial regression of z_i on a shifted and powered version of X :

$$\hat{z}_i = \hat{\beta}_0 + \hat{\beta}_1(x_i + \text{shift})^p,$$

where p is the best-fitting power, and shift ensures all values are positive if necessary. Ordinary least squares is used to estimate $\hat{\beta}_0$ and $\hat{\beta}_1$. A power of 0 corresponds to a log transformation.

3. **Back-transformation to the (0,1) scale:** The fitted values $\hat{z}_i$ are transformed back to the interval (0,1) using the standard normal CDF:

$$\text{ACD}(x_i) = a_i = \Phi(\hat{z}_i) = \Phi(\hat{\beta}_0 + \hat{\beta}_1(x_i + \text{shift})^p),$$

producing a smooth approximation of the ecdf.

Interpretation

The ACD transformation maps X to an approximately uniform scale on (0,1). When a regression model is specified as $E(Y) = \beta_0 + \beta_1 \text{ACD}(X)$, the expected value of Y changes smoothly from β_0 at $\text{ACD}(X) \approx 0$ (corresponding to the minimum of X) to $\beta_0 + \beta_1$ at $\text{ACD}(X) \approx 1$ (corresponding to the maximum of X).

Because the mapping from X to $\text{ACD}(X)$ is S-shaped—changing slowly at the extremes and more rapidly in the central range—the resulting relationship between Y and X is typically nonlinear and sigmoid-shape. This sigmoid shape cannot be achieved by standard fractional polynomial (FP) functions.

Intuitively, the ACD transformation “stretches” the middle of X ’s distribution while compressing the tails. A linear effect of $\text{ACD}(X)$ in the regression thus produces slow changes in Y for extreme values of X and faster changes for central values, generating a characteristic sigmoid curve.

Details of the precise definition and some possible uses of the ACD transformation in a univariate context are given by Royston (2014).

FP1 with ACD component

Royston (2014) and Royston and Sauerbrei (2016) describe extending the FP2 family by replacing one FP1 term with an ACD-transformed term:

$$FP1(p_1, p_2) = \beta_1 x^{p_1} + \beta_2 \text{ACD}(x)^{p_2},$$

which allows modeling of sigmoid-like effects while maintaining flexibility similar to standard FP2 functions.

The powers p_1 and p_2 are chosen from a predefined set $S = \{-2, -1, -0.5, 0, 0.5, 1, 2, 3\}$, with 0 corresponding to a logarithmic transformation. All 64 combinations of (p_1, p_2) are considered during model selection.

Model simplification

After fitting, the chosen FP1+ACD function can be simplified via a function selection procedure (FSPA) that evaluates six nested sub-families:

- M1: FP1(p_1, p_2) (no simplification)
- M2: FP1($p_1, .$) (regular FP1 in x)
- M3: FP1($., p_2$) (FP1 in $ACD(x)$)
- M4: FP1(1, $.$) (linear in x)
- M5: FP1($., 1$) (linear in $ACD(x)$)
- M6: null (x omitted)

Closed test procedure to choose final model

Selection among these six sub-functions is performed by a closed test procedure known as the function-selection procedure FSPA. It maintains the family-wise type 1 error probability for selecting x at the value determined by the `select` parameter. To obtain a 'final' model, a structured sequence of up to five tests is carried out, the first at the significance level specified by the `select` parameter, and the remainder at the significance level provided by the `alpha` option. The sequence of tests is as follows:

- Test 1: Compare the deviances of models 6 and 1 on 4 d.f. If not significant then stop and omit x , otherwise continue to step 2.
- Test 2: Compare the deviances of models 4 and 1 on 3 d.f. If not significant then accept model 4 and stop. Otherwise, continue to step 3.
- Test 3: Compare the deviance of models 2 and 1 on 2 d.f. If not significant then accept model 2 and stop. Otherwise continue to step 4.
- Test 4: Compare the deviance of models 3 and 1 on 2 d.f. If significant then model 1 cannot be simplified; accept model 1 and stop. Otherwise continue to step 5.
- Test 5: Compare the deviances of models 5 and 3 on 1 d.f. If significant then model 3 cannot be simplified; accept model 3. Otherwise, accept model 5. End of procedure.

The result is the selection of one of the six models. For details see Royston and Sauerbrei (2016).

Information criteria

As an alternative criterion, the fit of models M1–M6 can be evaluated using AIC or BIC. The model with the smallest AIC or BIC is selected.

Details on spike-at-zero (SAZ) modeling

In epidemiological and clinical studies, a continuous covariate x may have a mixture of zeros and positive values, which is called a semi-continuous variable. There is a 'spike' at zero in an otherwise continuous distribution. Examples include occupational exposures (e.g., asbestos) or alcohol/tobacco consumption, where some individuals have no exposure while others have positive values. When $x = 0$ defines a distinct subpopulation, the outcome for these individuals can be modeled using a binary variable Z , while positive values are modeled with a continuous FP function or

ACD transformation. To select a model, a two-stage procedure has been proposed and implemented in the `mfp2()` function as described below.

Stage 1: Functional form selection

A binary indicator Z is created ($Z = 1$ if $x = 0$, 0 otherwise). FP transformations are applied only to positive values of x , so there is no need to shift x . The user must choose a significance level (select) if the criterion is "pvalue". If "aic" or "bic" is used, the model with the minimum information criterion is selected. The user must also choose the maximum complexity of the FP function for x (default is FP2).

Suppose FP2 is the maximum complexity and `criterion = "pvalue"`. The selection procedure proceeds sequentially as follows:

1. Compare the most complex model (best FP2 + Z) with the null model ($df = 5$: 4 from FP2, 1 from Z). If significant, continue; otherwise, choose the null model.
2. Compare best FP2 + Z versus linear + Z ($df = 3$). If significant, continue; otherwise, select linear + Z .
3. Compare best FP2 + Z versus best FP1 + Z ($df = 2$). If significant, retain best FP2 + Z ; otherwise, select best FP1 + Z .

This ensures that the functional form is selected sequentially, retaining only models that significantly improve fit, always including Z . Stage 1 mirrors the standard MFP function selection process, except that Z is forced into the model. To force the SAZ variable into the model, set `select = 1` or use the `keep` argument.

Stage 2: Component assessment

The selected model from Stage 1 is evaluated to decide which components are needed. If the null model is selected in Stage 1, Stage 2 is not executed. Otherwise, suppose the selected model is best FP2 + Z :

- Test 1: Compare best FP2 + Z versus best FP2 alone to assess the contribution of Z .
- Test 2: Compare best FP2 + Z versus Z alone to assess the contribution of the FP2 term.

Decision rules:

- If both tests are significant, retain both Z and FP2 in the final model, because each component adds information beyond the other.
- If Test 1 is significant but Test 2 is not, retain only Z , because the spike indicator improves the model while the functional form does not contribute additional information.
- If Test 2 is significant but Test 1 is not, retain only FP2, because the functional form improves the model while the spike indicator does not add further explanatory value.
- If neither test is significant, compare the deviances of the Z -only and FP2-only models. The final model is the one with the smaller deviance, since it provides the better fit to the data without unnecessary terms.

For `criterion = "aic"` or `criterion = "bic"`, the information criteria for the three models (FP2 + Z , FP2, and Z only) are compared, and the one with the minimum value is chosen. If Z is removed, the spike at zero plays no specific role, and a standard FP function is selected for the positive values of x . Conversely, if the FP function is removed, leaving only Z in the model, the covariate's effect is entirely captured by the binary indicator. It should be noted that the presence of Z in Stage 1 may influence the selection of FP powers; consequently, the final FP powers may differ from those obtained in a standard FP analysis where `zero = TRUE` and `spike = FALSE`.

Handling Extreme Proportions of Zeros in Spike-at-Zero Modeling

The spike-at-zero (SAZ) approach is meaningful only when the proportion of zeros in a covariate is moderate. Covariates with fewer than 5% zeros may not contain sufficient information to justify a separate binary indicator, particularly in small samples where estimation of separate FP terms could be unstable. Conversely, covariates with more than 95% zeros are dominated by the zero values, leaving the positive portion too sparse for reliable functional form estimation.

In the implementation, the `reset_spike` function automatically addresses such extreme cases. By default, if the proportion of zeros is below 5% or above 95%, the spike is reset to `FALSE`, and the variable is treated as a standard continuous covariate. This ensures that SAZ modeling is applied only when both the zero and positive portions of the covariate provide sufficient information. Users may override these thresholds using the `min_prop` and `max_prop` options.

Details on model specification using a formula

`mfp2` supports model specification through two interfaces. The first accepts the design matrix `x` together with an outcome object `y`, similar to `stats::glm.fit()` or `glmnet::glmnet()`. For generalized linear models this may be a vector, but in the case of Cox regression `y` must be a two-column survival object (time and event indicator) created using `survival::Surv()`. The second interface follows the formula style used by functions such as `stats::glm()` or `survival::coxph()`.

Both interfaces are equivalent in functionality; only the details of specification differ. In the standard interface all details regarding FP-transformations are given as vectors, while in the formula interface they are specified using the special `fp()` function. The following options can be set: degrees of freedom (`df`), nominal significance level for variable selection (`select`), nominal significance level for functional form selection (`alpha`), shift values (`shift`), scale values (`scale`), centering (`center`), ACD-transformation (`acd`), handling of nonpositive values (`zero`), automatic creation of binary indicators for nonpositive values (`catzero`), and assessment of a potential spike at zero (`spike`). When arguments are supplied both inside `fp()` and as global defaults in `mfp2()`, the specifications inside `fp()` take precedence. For example, in `mfp2(y ~ fp(x, df = 2), df = 4, data = data)`, the fractional polynomial for `x` is fitted with `df = 2`, and the global setting `df = 4` is ignored.

The formula may also contain strata terms to fit stratified Cox models, or an offset term to specify a model offset.

Note that for a formula using `.`, such as `y ~ .`, the `mfp2()` function may not fit a linear model, but may perform variable and functional form selection using FP-transformations, depending on the global default settings of `df`, `select` and `alpha` passed as arguments to `mfp2()`. For example, using `y ~ .` with default settings means that `mfp2()` will apply FP transformation with 4 df to all continuous variables and use `alpha` equal to 0.05 to select functional forms, along with the selection algorithm with a significance level of 0.05 for all variables.

Handling Nonpositive Values (`zero` and `catzero`)

The `zero` and `catzero` options provide mechanisms for handling covariates with nonpositive values in fractional polynomial (FP) models, especially when those values have a qualitatively different interpretation than positive ones.

The `zero` argument enables fitting FP models using only the positive values of a covariate, while treating nonpositive values (e.g., zero or negative) as exactly zero. In other words, only the positive values of x are transformed, and the nonpositive values are set to zero. This approach is useful when the relationship between the covariate and the outcome is expected to start above zero. For

example, in modeling the effect of cigarette consumption, nonsmokers (zero cigarettes) may be fundamentally different from smokers. Instead of applying a constant shift (e.g., adding 1) to all values before transformation, the `zero` argument allows the model to treat zero values as a separate baseline and apply FP transformations only to the positive values.

The `catzero` argument extends this idea by automatically creating a binary indicator for whether the covariate is positive. Specifically, for each covariate where `catzero = TRUE`, a binary variable `Z` is created such that:

$Z = 1$ if the covariate value is zero ($x = 0$) and $Z = 0$ if the covariate value is positive ($x > 0$)

This indicator is included in the model alongside the transformed version of the covariate. Both are treated as a single predictor during model selection (if applicable). This approach captures the potential difference between having a value of zero and having any positive value, while still allowing flexible modeling of the positive range.

The `spike` argument triggers the spike-at-zero (SAZ) algorithm. When `spike = TRUE`, the binary indicator created by `catzero` is formally evaluated together with the FP terms during model selection. When `spike = FALSE` but `catzero = TRUE`, the binary indicator is included in the model without SAZ testing.

This methodology is based on work by Royston and Sauerbrei (2008, Section 4.15) and is particularly relevant in epidemiological contexts where exposure may have a threshold effect.

Compatibility with mfp package

`mfp2` is an extension of the `mfp` package and can be used to reproduce the results from a model fitted by `mfp`. Since both packages implement the MFP algorithm, they use functions with the same names (e.g. `fp()`). Therefore, if you load both packages using a call to `library`, there will be namespace conflicts and only the functions from the package loaded last will work properly.

Convergence and Troubleshooting

Typically, `mfp2` requires two to five cycles to achieve convergence. Lack of convergence involves oscillation between two or more models and is extremely rare. If convergence problems occur, consider adjusting the nominal significance levels for variable selection (`select`) or functional form selection (`alpha`)

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.

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. *mfp2: 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

`summary.mfp2()`, `coef.mfp2()`, `predict.mfp2()`, `fp()` [model.matrix](#), [contr.treatment](#), [contr.poly](#)

Examples

```
# Gaussian model
data("prostate")
x = as.matrix(prostate[,2:8])
y = as.numeric(prostate$lpsa)
# default interface
fit1 = mfp2(x, y, verbose = FALSE)
fit1$fp_terms
fracplot(fit1) # generate plots
summary(fit1)
# formula interface
fit1b = mfp2(lpsa ~ fp(age) + fp(svi, df = 1) + fp(pgg45) + fp(cavol) + fp(weight) +
fp(bph) + fp(cp), data = prostate)

# logistic regression model
data("pima")
xx <- as.matrix(pima[, 2:9])
yy <- as.vector(pima$y)
fit2 <- mfp2(xx, yy, family = "binomial", verbose = FALSE)
fit2$fp_terms

# Cox regression model
data("gbsg")
# create dummy variable for grade using ordinal coding
gbsg <- create_dummy_variables(gbsg, var_ordinal = "grade", drop_variables = TRUE)
xd <- as.matrix(gbsg[, -c(1, 6, 10, 11)])
yd <- survival::Surv(gbsg$rectime, gbsg$censrec)
# fit mfp and keep hormon in the model
```

```
fit3 <- mfp2(xd, yd, family = "cox", keep = "hormon", verbose = FALSE)
fit3$fp_terms
```

name_transformed_variables

Helper function to name transformed variables

Description

Helper function to name transformed variables

Usage

```
name_transformed_variables(name, n_powers, acd = FALSE)
```

Arguments

name	character with name of variable being transformed.
n_powers	number of resulting variables from FP-transformation.
acd	logical indicating the use of ACD-transformation

Value

Character vector of names of length n_powers.

order_variables

Helper to order variables for mfp2 algorithm

Description

To be used in fit_mfp().

Usage

```
order_variables(xorder = "ascending", x = NULL, ...)
```

```
order_variables_by_significance(
  xorder,
  x,
  y,
  family,
  family_string,
  weights,
  offset,
```

```

    strata,
    method,
    control,
    nocenter
  )

```

Arguments

xorder	a string determining the order of entry of the covariates into the model-selection algorithm. The default is ascending, which enters them by ascending p-values, or decreasing order of significance in a multiple regression (i.e. most significant first). descending places them in reverse significance order, whereas original respects the original order in x.
x	a design matrix of dimension $n * p$ where n is the number of observations and p the number of predictors including intercept for glms, or excluding intercept for Cox models.
...	passed to <code>order_variables_by_significance</code> .
y	a vector of responses for glms, or a <code>Surv</code> object generated using the <code>survival::Surv()</code> function for Cox models.
family	a character string naming a family function supported by <code>glm()</code> or "cox" for Cox models.
family_string	A character string representing the selected family, e.g., "gaussian".
weights, offset	parameters for both <code>glm</code> and Cox models, see either <code>stats::glm()</code> or <code>survival::coxph()</code> depending on family.
strata, method, control, nocenter	Cox model specific parameters, see <code>survival::coxph()</code> .

Value

A vector of the variable names in x, ordered according to xorder.

Functions

- `order_variables_by_significance()`: Order by significance in regression model. The number of columns of x should be greater than 1 for Cox models.

pima

Pima Indians dataset used in the 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 9 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).

predict.mfp2

Predict Method for mfp2

Description

Obtains predictions from an mfp2 object.

Usage

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

Arguments

object	a fitted object of class mfp2.
newdata	optionally, a matrix with column names in which to look for variables with which to predict. If provided, the variables are internally shifted using the shifting values stored in object. See mfp2() for further details.
type	the type of prediction required. The default is on the scale of the linear predictors. See predict.glm() or predict.coxph() for details. In case type = "terms", see the Section on Terms prediction. In case type = "contrasts", see the Section on Contrasts.
terms	a character vector of variable names specifying for which variables term or contrast predictions are desired. Only used in case type = "terms" or type = "contrasts". If NULL (the default) then all selected variables in the final model will be used. In any case, only variables used in the final model are used, even if more variable names are passed.
terms_seq	a character string specifying how the range of variable values for term predictions are handled. The default equidistant computes the range of the data range and generates an equidistant sequence of 100 points from the minimum to the maximum values of shifted values to properly show the functional form estimated in the final model. The option data uses the observed data values directly, but these may not adequately reflect the functional form of the data, especially when extreme values or influential points are present.
alpha	significance level used for computing confidence intervals in terms prediction.
ref	a named list of reference values used when type = "contrasts". Note that any variable requested in terms, but not having an entry in this list (or if the entry is NULL) then the mean value of shifted data (or minimum for binary variables) will be used as reference. Values should be specified on the original scale of the variable since the program will internally scale it using the scaling factors obtained from find_scale_factor(). By default, this function uses the means (for continuous variables) and minimum (for binary variables) as reference values.
strata	stratum levels used for predictions.
newoffset	A vector of offsets used for predictions. This parameter is important when newdata is supplied. The offsets are directly added to the linear predictor without any transformations.
nseq	Integer specifying how many values to generate when terms_seq = "equidistant". Default is 100.
add_intercept	Logical indicating whether to include an intercept in the linear predictor when type = "terms". The default is TRUE.
...	further arguments passed to predict.glm() or predict.coxph().

Details

When newdata is supplied, it is first shifted using the factors obtained from the training data, then transformed using the selected fractional polynomial powers, and optionally centered if center = TRUE was used in the original mfp2() fit. If the shifting factors from the training data are not

large enough, some variables may remain non-positive, which can cause errors when non-linear transformations (e.g., logarithms) are applied; in such cases, a warning is issued.

After transformation (and centering), the prepared newdata is passed directly to `predict.glm()` or `predict.coxph()` depending on the model family, including proper handling of offsets. This replaces manual linear predictor computation in previous implementations and allows computation of `se.fit` when requested.

Terms predictions (`type = "terms"`) compute partial linear predictors for selected variables, accounting for fractional polynomial transformations and spike-at-zero indicators.

Contrasts (`type = "contrasts"`) compute differences relative to reference values. Reference values are shifted, transformed, and centered consistently with the model. If `ref = NULL`, the mean (continuous) or minimum (binary) of shifted values is used.

Value

For any type other than "terms" the output conforms to the output of `predict.glm()` or `predict.coxph()`.

If `type = "terms"` or `type = "contrasts"`, then a named list with entries for each variable requested in terms (excluding those not present in the final model). Each entry is a `data.frame` with the following columns:

- `variable`: variable values on original scale (without shifting).
- `variable_pre`: variable with pre-transformation applied, i.e. shifted, and centered as required.
- `value`: partial linear predictor or contrast (depending on type).
- `se`: standard error of partial linear predictor or contrast.
- `lower`: lower limit of confidence interval.
- `upper`: upper limit of confidence interval.

Terms prediction

If `type = "terms"`, this function computes the partial linear predictors for each variable included in the final model. Unlike `predict.glm()` and `predict.coxph()`, this function accounts for the fact that a single variable may be represented by multiple transformed terms.

For a variable modeled using a first-degree fractional polynomial (FP1), the partial predictor is given by $\hat{\eta}_j = \hat{\beta}_0 + x_j^* \hat{\beta}_j$, where x_j^* is the transformed variable (centered if `center = TRUE`).

If a spike-at-zero binary indicator is included (`catzero = TRUE`), the partial predictor becomes $\hat{\eta}_j = \hat{\beta}_0 + (x_j^*)^+ \hat{\beta}_j + z_j^* \hat{\beta}_{z_j}$, where $(x_j^*)^+ = \max(x_j^*, 0)$ denotes the positive-part transformation, z_j^* is the binary indicator for nonpositive values and $\hat{\beta}_{z_j}$ is its corresponding coefficient.

Explicitly: When $x_j = 0$, $\hat{\eta}_j = \hat{\beta}_0 + 0 \cdot \hat{\beta}_j + \hat{\beta}_{z_j} = \hat{\beta}_0 + \hat{\beta}_{z_j}$. When $x_j > 0$, $\hat{\eta}_j = \hat{\beta}_0 + x_j^* \hat{\beta}_j + 0 \cdot \hat{\beta}_{z_j} = \hat{\beta}_0 + x_j^* \hat{\beta}_j$.

If only `zero = TRUE` (and `catzero = FALSE`), the partial predictor becomes $\hat{\eta}_j = \hat{\beta}_0 + (x_j^*)^+ \hat{\beta}_j$.

For a second-degree fractional polynomial (FP2), the partial predictor takes the form $\hat{\eta}_j = \hat{\beta}_0 + x_{j1}^* \hat{\beta}_{j1} + x_{j2}^* \hat{\beta}_{j2}$, where x_{j1}^* and x_{j2}^* are the two transformed components of the original variable (centered if `center = TRUE`).

If `catzero = TRUE`, the FP2 partial predictor extends to $\hat{\eta}_j = \hat{\beta}_0 + (x_{j1}^*)^+ \hat{\beta}_{j1} + (x_{j2}^*)^+ \hat{\beta}_{j2} + z_j^* \hat{\beta}_{z_j}$.

If only `zero = TRUE` (and `catzero = FALSE`), the FP2 partial predictor extends to $\hat{\eta}_j = (x_{j1}^*)^+ \hat{\beta}_{j1} + (x_{j2}^*)^+ \hat{\beta}_{j2}$.

This functionality is particularly useful for visualizing the functional relationship of a continuous variable, or for assessing model fit when residuals are included. See also `fracplot()`.

Contrasts

If `type = "contrasts"`, this function computes contrasts relative to a specified reference value for the j th variable (e.g., `age = 50`). Let x_j denote the values of the j th variable in `newdata`, and x_j^{ref} the reference value. The contrast is defined as the difference between the partial linear predictor evaluated at the transformed (and centered, if `center = TRUE`) value x_j , and that evaluated at the transformed reference value $x_j^{*(\text{ref})}$, i.e., $f(x_j^*) - f(x_j^{*(\text{ref})})$.

For a first-degree fractional polynomial (FP1), the partial predictor is

$$\hat{f}(x_j^*) = \hat{\beta}_0 + x_j^* \hat{\beta}_j$$

. If a spike-at-zero binary indicator is included (`catzero = TRUE`), the partial predictor becomes

$$\hat{f}(x_j^*) = \hat{\beta}_0 + (x_j^*)^+ \hat{\beta}_j + z_j^* \hat{\beta}_{z_j}$$

, where z_j^* is the binary indicator for nonpositive values. The contrast is then computed as the difference between the partial predictor evaluated at x_j^* (and z_j^* if `catzero = TRUE`) and the partial predictor evaluated at the reference value $x_j^{*(\text{ref})}$ (and $z_j^{*(\text{ref})}$ if `catzero = TRUE`).

For a second-degree fractional polynomial (FP2), the partial predictor is

$$\hat{f}(x_j^*) = \hat{\beta}_0 + x_{j1}^* \hat{\beta}_{j1} + x_{j2}^* \hat{\beta}_{j2}$$

. If a spike-at-zero binary indicator is included (`catzero = TRUE`), the partial predictor becomes

$$\hat{f}(x_j^*) = \hat{\beta}_0 + (x_{j1}^*)^+ \hat{\beta}_{j1} + (x_{j2}^*)^+ \hat{\beta}_{j2} + z_j^* \hat{\beta}_{z_j}$$

. The contrast is then computed in the same conditional manner as for FP1.

Here, x_j^* , x_{j1}^* , x_{j2}^* , and z_j^* are the transformed (and centered if applicable) components, and the $\hat{\beta}$ terms are the corresponding model coefficients.

Reference values $x_j^{*(\text{ref})}$ (and $z_j^{*(\text{ref})}$ if `catzero = TRUE`) are shifted, transformed, and centered using the training data, ensuring full consistency with the fitted model.

If `ref = NULL`, the function uses the mean of the shifted x_j for continuous variables and the minimum (typically 0) for binary variables. For `catzero` variables, the reference binary indicator $z_j^{*(\text{ref})}$ is determined by whether the value is positive or zero.

The fitted partial predictors are centered at the reference point, meaning the contrast at that point is zero. Confidence intervals at the reference value have zero width.

This approach allows direct comparison of a variable's effect relative to a meaningful baseline, including the spike-at-zero effect only when it is present.

See Also

`mfp2()`, `stats::predict.glm()`, `survival::predict.coxph()`

Examples

```
# Gaussian model
data("prostate")
x = as.matrix(prostate[,2:8])
y = as.numeric(prostate$lpsa)
# default interface
fit1 = mfp2(x, y, verbose = FALSE)
predict(fit1) # make predictions

# Binomial model
data("pima")
x2 <- as.matrix(pima[,2:9])
y2 <- as.vector(pima$y)
fit2 <- mfp2(x2, y2, family = binomial(link = "logit"), verbose = FALSE)
predict(fit2, newdata = x2, type = "response") # make predictions on response scale
```

```
prepare_newdata_for_predict
```

Helper function to prepare newdata for predict function

Description

To be used in `predict.mfp2()`.

Usage

```
prepare_newdata_for_predict(
  object,
  newdata,
  strata = NULL,
  offset = NULL,
  apply_pre = TRUE,
  apply_center = TRUE,
  check_binary = TRUE,
  reset_zero = TRUE
)
```

Arguments

<code>object</code>	fitted <code>mfp2</code> model object.
<code>newdata</code>	dataset to be prepared for predictions. Its columns can be a subset of the columns used for fitting the model.
<code>strata, offset</code>	passed from <code>predict.mfp2()</code> .
<code>apply_pre</code>	logical indicating whether the fitted pre-transformation is applied or not.
<code>apply_center</code>	logical indicating whether the fitted centers are applied after transformation or not.

check_binary	passed to transform_vector_fp().
reset_zero	Logical. If TRUE (default), variables incorrectly flagged as zero = TRUE but containing only positive values are reset to FALSE. Parameter of transform_matrix()

Value

A dataframe of transformed newdata

print.mfp2	<i>Print method for objects of class mfp2</i>
------------	---

Description

Enhances printing by information on data processing and fractional polynomials.

Usage

```
## S3 method for class 'mfp2'
print(x, ...)
```

Arguments

x	mfp2 object to be printed.
...	passed to print methods of underlying model class. A useful option as the digits argument, indicating printed digits.

Value

Two dataframes: the first one contains preprocessing parameters (shifting, scaling, and centering), and the second one includes additional parameters such as df, select, and alpha passed through mfp2. It also returns a list of the final model fitted, which can be either a GLM or Cox model depending on the chosen family.

print_mfp_step	<i>Verbose printing of the function selection procedure (FSP)</i>
----------------	---

Description

This function prints intermediate results from the fractional polynomial function selection process, including details of the spike-at-zero algorithm when relevant. It is primarily called from within mfp_step during model selection.

Usage

```
print_mfp_step(xi, criterion, fit, stage2 = FALSE)

print_mfp_pvalue_step(xi, fit, criterion, spike = FALSE)

print_mfp_ic_step(xi, fit, criterion, spike = FALSE)
```

Arguments

xi	Character string. The name of the covariate currently being tested.
criterion	Character string. The selection criterion, either "pvalue" or "aic"/"bic" (depending on implementation in print_mfp_ic_step()).
fit	A list containing the intermediate fit for the current variable. Typically produced by mfp_step.
stage2	Logical. When FALSE (default), only Stage 1 results of the spike-at-zero (SAZ) procedure are printed, or the usual FSP output for xi when fit\$spike[xi] = FALSE. When TRUE, Stage 1 results are printed, and if fit\$spike[xi] = TRUE and the Stage 1 selected model is not "null", the Stage 2 spike-at-zero results are printed immediately after Stage 1.
spike	Logical. Indicates whether xi was treated as a spike-at-zero variable. Default is FALSE.

Details

- Printing is split into two parts:
- **Stage 1:** All candidate FP/ACD models for xi are listed with their powers, degrees of freedom, and criterion-specific statistics. If fit\$spike[xi] = TRUE, this corresponds to the first stage of the spike-at-zero (SAZ) algorithm.
 - **Stage 2:** If stage2 = TRUE and the Stage 1 selected model for xi is not "null", the results of the spike-at-zero are printed immediately after Stage 1. The chosen model is highlighted at the end of the table.

Functions

- print_mfp_pvalue_step(): Helper for verbose printing based on p-value.
- print_mfp_ic_step(): Helper for verbose printing based on information criterion.

prostate	<i>Prostate cancer dataset used in the Royston and Sauerbrei (2008) book.</i>
----------	---

Description

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

Usage

```
data(prostate)
```

Format

A dataset with 97 observations and 8 variables.

obsno Observation number.

age Age in years.

svi Seminal vessel invasion (yes/no).

pgg45 Percentage Gleason score 4 or 5.

cavol Cancer volume (mm).

weight Prostate weight (g).

bph Amount of benign prostatic hyperplasia (g).

cp Amount of capsular penetration (g).

lpsa Log PSA concentration (outcome variable).

reset_acd

Helper to reset acd transformation for variables with few values

Description

To be used in `fit_mfp()`. This function resets the `acdx` parameter (logical vector) of variables with less than 5 distinct values to FALSE.

Usage

```
reset_acd(x, acdx)
```

Arguments

<code>x</code>	a design matrix of dimension <code>nobs x nvars</code> where <code>nvars</code> is the number of predictors excluding an intercept.
<code>acdx</code>	a named logical vector of length <code>nvars</code> indicating which continuous variables should undergo the approximate cumulative distribution (ACD) transformation. May be ordered differently than the columns of <code>x</code> .

Value

Logical vector of same length as `acdx`.

`select_ic`*Function selection procedure based on information criteria*

Description

Used in `find_best_fp_step()` when `criterion = "aic"` or `"bic"`. For parameter explanations, see `find_best_fp_step()`. All parameters captured by `...` are passed on to `fit_model()`.

Usage

```
select_ic(  
  x,  
  xi,  
  keep,  
  degree,  
  acdx,  
  y,  
  powers_current,  
  powers,  
  criterion,  
  ftest,  
  select,  
  alpha,  
  family,  
  family_string,  
  zero,  
  catzero,  
  spike,  
  spike_decision,  
  acd_parameter,  
  prev_adj_params,  
  ...  
)
```

```
select_ic_acd(  
  x,  
  xi,  
  keep,  
  degree,  
  acdx,  
  y,  
  powers_current,  
  powers,  
  criterion,  
  ftest,  
  select,  
  alpha,
```

```

    family,
    family_string,
    zero,
    catzero,
    spike,
    spike_decision,
    acd_parameter,
    prev_adj_params,
    ...
)

```

Arguments

<code>x</code>	an input matrix of dimensions <code>nobs x nvars</code> . Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
<code>xi</code>	a character string indicating the name of the current variable of interest, for which the best fractional polynomial transformation is to be estimated in the current step.
<code>keep</code>	a character vector with names of variables to be kept in the model.
<code>degree</code>	integer > 0 giving the degree for the FP transformation.
<code>acdx</code>	a logical vector of length <code>nvars</code> indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
<code>y</code>	a vector for the response variable or a <code>Surv</code> object.
<code>powers_current</code>	a list of length equal to the number of variables, indicating the fp powers to be used in the current step for all variables (except <code>xi</code>).
<code>powers</code>	a named list of numeric values that sets the permitted FP powers for each co-variate.
<code>criterion</code>	a character string defining the criterion used to select variables and FP models of different degrees.
<code>ftest</code>	a logical indicating the use of the F-test for Gaussian models.
<code>select</code>	a numeric value indicating the significance level for backward elimination of <code>xi</code> .
<code>alpha</code>	a numeric value indicating the significance level for tests between FP models of different degrees for <code>xi</code> .
<code>family</code>	Either a character string naming the family (e.g., "gaussian", "binomial", "cox") or a function that returns a GLM family object (e.g., <code>stats::gaussian</code>). For Cox models, only a character string "cox" is allowed.
<code>family_string</code>	A character string representing the selected family, e.g., "gaussian".
<code>zero</code>	A named logical vector indicating, which columns of <code>x</code> should treat nonpositive values (zero or negative) as zero before transformation. Must be the same length as the columns of <code>x</code> .
<code>catzero</code>	A named list of binary indicator variables of length <code>ncol(x)</code> for nonpositive values, created when specific variables are passed to the <code>catzero</code> argument of <code>fit_mfp</code> . If an element of the list is <code>NULL</code> , it indicates that the corresponding

	variable was not specified by the user in the <code>catzero</code> argument of <code>fit_mfp</code> . Here, <code>catzero</code> is a list of binary variables, not a named logical vector as in <code>fit_mfp</code> .
<code>spike</code>	A logical vector indicating which columns of <code>x</code> contain a spike at zero. The length and order of <code>spike</code> must match those of the columns in <code>x</code> .
<code>spike_decision</code>	Named vector indicating how spike-at-zero (SAZ) variables are handled. Each element corresponds to a variable and encodes the selected strategy: 1 = include FP for positive values plus binary SAZ, 2 = treat as continuous FP only, 3 = include binary SAZ only.
<code>acd_parameter</code>	Named list of ACD parameters produced by <code>fit_acd()</code> , with length equal to <code>ncol(x)</code> . Each list element corresponds to a variable; if an element is <code>NULL</code> , the variable was not specified in the <code>acdx</code> argument of <code>fit_mfp</code> .
<code>prev_adj_params</code>	Named list storing adjustment variable transformations from previous steps for each variable.
<code>...</code>	passed to fitting functions.

Details

In case an information criterion is used to select the best model the selection procedure simply fits all relevant models and selects the best one according to the given criterion.

"Relevant" models for a given degree are the null model excluding the variable of interest, the linear model and all best FP models up to the specified degree.

In case an ACD transformation is requested, then the models assessed are the null model, the linear model in `x` and `A(x)`, the best FP1 models in `x` and `A(x)`, and the best FP1(`x`, `A(x)`) model.

Note that the "best" FPx model used in this function are given by the models using a FPx transformation for the variable of interest and having the highest likelihood of all such models given the current powers for all other variables, as outlined in Section 4.8 of Royston and Sauerbrei (2008). These best FPx models are computed in `find_best_fpm_step()`. Keep in mind that for a fixed number of degrees of freedom (i.e. fixed `m`), the model with the highest likelihood is the same as the model with the best information criterion of any kind since all the models share the same penalty term.

When a variable is forced into the model by including it in `keep`, then this function will not exclude it from the model (by setting its power to `NA`), but will only choose its functional form.

Value

A list with several components:

- `keep`: logical indicating if `xi` is forced into model.
- `acd`: logical indicating if an ACD transformation was applied for `xi`, i.e. `FALSE` in this case.
- `powers`: (best) fp powers investigated in step, indexing `metrics`. Ordered by increasing complexity, i.e. null, linear, FP1, FP2 and so on. For ACD transformation, it is null, linear, linear(., `A(x)`), FP1(`x`, .), FP1(., `A(x)`) and FP1(`x`, `A(x)`).
- `power_best`: a numeric vector with the best power found. The returned best power may be `NA`, indicating the variable has been removed from the model.

- **metrics**: a matrix with performance indices for all best models investigated. Same number of rows as, and indexed by, **powers**.
- **model_best**: row index of best model in **metrics**.
- **pvalue**: p-value for comparison of linear and null model, NA in this case..
- **statistic**: test statistic used, depends on **ftest**, NA in this case.
- **spike_decision**: Spike decision flag for **xi**.
- **prev_adj_params**: Previously adjusted parameters.

Functions

- **select_ic_acd()**: Function to select ACD based transformation.

See Also

`select_ra2()`

<code>select_linear</code>	<i>Helper function to select between null and linear term for a single variable</i>
----------------------------	---

Description

To be used in `find_best_fp_step()`. Only used if `df = 1` for a variable. Handles all criteria for selection. For parameter explanations, see `find_best_fp_step()`. All parameters captured by ... are passed to `fit_model()`.

Usage

```
select_linear(
  x,
  xi,
  keep,
  degree,
  acdx,
  y,
  powers_current,
  powers,
  criterion,
  ftest,
  select,
  alpha,
  family,
  family_string,
  zero,
  catzero,
  spike,
```

```

    spike_decision,
    acd_parameter,
    prev_adj_params,
    ...
)

```

Arguments

<code>x</code>	an input matrix of dimensions <code>nobs x nvars</code> . Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
<code>xi</code>	a character string indicating the name of the current variable of interest, for which the best fractional polynomial transformation is to be estimated in the current step.
<code>keep</code>	a character vector with names of variables to be kept in the model.
<code>degree</code>	not used.
<code>acdx</code>	a logical vector of length <code>nvars</code> indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
<code>y</code>	a vector for the response variable or a <code>Surv</code> object.
<code>powers_current</code>	a list of length equal to the number of variables, indicating the fp powers to be used in the current step for all variables (except <code>xi</code>).
<code>powers</code>	a named list of numeric values that sets the permitted FP powers for each covariate.
<code>criterion</code>	a character string defining the criterion used to select variables and FP models of different degrees.
<code>ftest</code>	a logical indicating the use of the F-test for Gaussian models.
<code>select</code>	a numeric value indicating the significance level for backward elimination of <code>xi</code> .
<code>alpha</code>	a numeric value indicating the significance level for tests between FP models of different degrees for <code>xi</code> .
<code>family</code>	Either a character string naming the family (e.g., "gaussian", "binomial", "cox") or a function that returns a GLM family object (e.g., <code>stats::gaussian</code>). For Cox models, only a character string "cox" is allowed.
<code>family_string</code>	A character string representing the selected family, e.g., "gaussian".
<code>zero</code>	A named logical vector indicating, which columns of <code>x</code> should treat nonpositive values (zero or negative) as zero before transformation. Must be the same length as the columns of <code>x</code> .
<code>catzero</code>	A named list of binary indicator variables of length <code>ncol(x)</code> for nonpositive values, created when specific variables are passed to the <code>catzero</code> argument of <code>fit_mfp</code> . If an element of the list is <code>NULL</code> , it indicates that the corresponding variable was not specified by the user in the <code>catzero</code> argument of <code>fit_mfp</code> . Here, <code>catzero</code> is a list of binary variables, not a named logical vector as in <code>fit_mfp</code> .
<code>spike</code>	A logical vector indicating which columns of <code>x</code> contain a spike at zero. The length and order of <code>spike</code> must match those of the columns in <code>x</code> .

spike_decision	Named vector indicating how spike-at-zero (SAZ) variables are handled. Each element corresponds to a variable and encodes the selected strategy: 1 = include FP for positive values plus binary SAZ, 2 = treat as continuous FP only, 3 = include binary SAZ only.
acd_parameter	Named list of ACD parameters produced by <code>fit_acd()</code> , with length equal to <code>ncol(x)</code> . Each list element corresponds to a variable; if an element is <code>NULL</code> , the variable was not specified in the <code>acdx</code> argument of <code>fit_mfp</code> .
prev_adj_params	Named list storing adjustment variable transformations from previous steps for each variable.
...	passed to fitting functions.

Details

This function assesses a single variable of interest `xi` regarding its functional form in the current working model as indicated by `powers_current`, with the choice between excluding `xi` ("null model") and including a linear term ("linear fp") for `xi`.

Note that this function handles an ACD transformation for `xi` as well.

When a variable is forced into the model by including it in `keep`, then this function will not exclude it from the model (by setting its power to `NA`), but will only choose the linear model.

Value

A list with several components:

- `keep`: logical indicating if `xi` is forced into model.
- `acd`: logical indicating if an ACD transformation was applied for `xi`.
- `powers`: fp powers investigated in step, indexing `metrics`.
- `power_best`: a numeric vector with the best power found. The returned best power may be `NA`, indicating the variable has been removed from the model.
- `metrics`: a matrix with performance indices for all models investigated. Same number of rows as, and indexed by, `powers`.
- `model_best`: row index of best model in `metrics`.
- `pvalue`: p-value for comparison of linear and null model.
- `statistic`: test statistic used, depends on `ftest`.
- `zero`: Logical indicating whether a zero transformation was applied to `xi`. In this case, non-positive values of `xi` were set to zero before transformation, and only positive values were transformed.
- `catzero`: Logical indicating whether a combination of a zero transformation and a binary indicator variable was applied to `xi`. This means that nonpositive values of `xi` were set to zero, only positive values were transformed, and an additional binary variable was created to indicate whether `xi` was positive or nonpositive.
- `spike_decision`: Spike decision flag for `xi`.
- `prev_adj_params`: Previously adjusted parameters.

select_ra2

*Function selection procedure based on closed testing procedure***Description**

Used in `find_best_fp_step()` when `criterion = "pvalue"`. For parameter explanations, see `find_best_fp_step()`. All parameters captured by `...` are passed to `fit_model()`.

Usage

```
select_ra2(
  x,
  xi,
  keep,
  degree,
  acdx,
  y,
  powers_current,
  powers,
  criterion,
  ftest,
  select,
  alpha,
  family,
  family_string,
  zero,
  catzero,
  spike,
  spike_decision,
  acd_parameter,
  prev_adj_params,
  ...
)
```

Arguments

<code>x</code>	an input matrix of dimensions <code>nobs x nvars</code> . Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
<code>xi</code>	a character string indicating the name of the current variable of interest, for which the best fractional polynomial transformation is to be estimated in the current step.
<code>keep</code>	a character vector with names of variables to be kept in the model.
<code>degree</code>	integer > 0 giving the degree for the FP transformation.
<code>acdx</code>	a logical vector of length <code>nvars</code> indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.

<code>y</code>	a vector for the response variable or a <code>Surv</code> object.
<code>powers_current</code>	a list of length equal to the number of variables, indicating the fp powers to be used in the current step for all variables (except <code>xi</code>).
<code>powers</code>	a named list of numeric values that sets the permitted FP powers for each covariate.
<code>criterion</code>	a character string defining the criterion used to select variables and FP models of different degrees.
<code>ftest</code>	a logical indicating the use of the F-test for Gaussian models.
<code>select</code>	a numeric value indicating the significance level for backward elimination of <code>xi</code> .
<code>alpha</code>	a numeric value indicating the significance level for tests between FP models of different degrees for <code>xi</code> .
<code>family</code>	Either a character string naming the family (e.g., "gaussian", "binomial", "cox") or a function that returns a GLM family object (e.g., <code>stats::gaussian</code>). For Cox models, only a character string "cox" is allowed.
<code>family_string</code>	A character string representing the selected family, e.g., "gaussian".
<code>zero</code>	A named logical vector indicating, which columns of <code>x</code> should treat nonpositive values (zero or negative) as zero before transformation. Must be the same length as the columns of <code>x</code> .
<code>catzero</code>	A named list of binary indicator variables of length <code>ncol(x)</code> for nonpositive values, created when specific variables are passed to the <code>catzero</code> argument of <code>fit_mfp</code> . If an element of the list is <code>NULL</code> , it indicates that the corresponding variable was not specified by the user in the <code>catzero</code> argument of <code>fit_mfp</code> . Here, <code>catzero</code> is a list of binary variables, not a named logical vector as in <code>fit_mfp</code> .
<code>spike</code>	A logical vector indicating which columns of <code>x</code> contain a spike at zero. The length and order of <code>spike</code> must match those of the columns in <code>x</code> .
<code>spike_decision</code>	Named vector indicating how spike-at-zero (SAZ) variables are handled. Each element corresponds to a variable and encodes the selected strategy: 1 = include FP for positive values plus binary SAZ, 2 = treat as continuous FP only, 3 = include binary SAZ only.
<code>acd_parameter</code>	Named list of ACD parameters produced by <code>fit_acd()</code> , with length equal to <code>ncol(x)</code> . Each list element corresponds to a variable; if an element is <code>NULL</code> , the variable was not specified in the <code>acdx</code> argument of <code>fit_mfp</code> .
<code>prev_adj_params</code>	Named list storing adjustment variable transformations from previous steps for each variable.
<code>...</code>	passed to fitting functions <code>fit_model()</code> .

Details

In case `criterion = "pvalue"` the function selection procedure as outlined in Chapters 4 and 6 of Royston and Sauerbrei (2008) is used.

- *Step 1:* Test the best FP_m function against a null model at the significance level specified by `select`, using $2m$ degrees of freedom. If the test is not significant, the variable is excluded. Otherwise, proceed to Step 2.

- *Step 2*: Test the best FP_m function against a linear model at the significance level specified by α , using $2m-1$ degrees of freedom. If the test is not significant, select the linear model. Otherwise, proceed to Step 3.
- *Step 3*: Test the best FP_m function against the best FP_1 model at the significance level specified by α , using $2m-2$ degrees of freedom. If the test is not significant, retain the best FP_1 model. Otherwise, repeat this step by comparing FP_m to all remaining lower-order FP models, down to FP_{m-1} , which is tested with 2 degrees of freedom. If the final test is not significant, retain the best FP_{m-1} model; otherwise, retain the best FP_m model.

Note that the "best" FP_x model used in each step refers to the model that applies an FP_x transformation to the variable of interest and achieves the highest likelihood among all such models, given the current power transformations for all other variables. This procedure is described in Section 4.8 of Royston and Sauerbrei (2008). The best FP_x models are computed by `find_best_fpm_step`.

When a variable is forced into the model by including it in the `keep` argument of `mfp2()`, this function will not exclude it (i.e., will not set its power to NA), but will instead select its functional form.

Value

A list with several components:

- `keep`: logical indicating if x_i is forced into model.
- `acd`: logical indicating if an ACD transformation was applied for x_i , i.e. FALSE in this case.
- `powers`: (best) fp powers investigated in step, indexing metrics. Always starts with highest power, then null, then linear, then FP in increasing degree (e.g. FP_2 , null, linear, FP_1).
- `power_best`: a numeric vector with the best power found. The returned best power may be NA, indicating the variable has been removed from the model.
- `metrics`: a matrix with performance indices for all models investigated. Same number of rows as, and indexed by, `powers`.
- `model_best`: row index of best model in `metrics`.
- `pvalue`: p-value for comparison of linear and null model.
- `statistic`: test statistic used, depends on `ftest`.
- `spike_decision`: Spike decision flag for x_i .
- `prev_adj_params`: Previously adjusted parameters.

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.

See Also

`select_ra2_acd()`

select_ra2_acd	<i>Function selection procedure for ACD based on closed testing procedure</i>
----------------	---

Description

Used in `find_best_fp_step()` when `criterion = "pvalue"` and an ACD transformation is requested for `xi`. For parameter explanations, see `find_best_fp_step()`. All parameters captured by `...` are passed on to `fit_model()`.

Usage

```
select_ra2_acd(
  x,
  xi,
  keep,
  degree,
  acdx,
  y,
  powers_current,
  powers,
  criterion,
  ftest,
  select,
  alpha,
  family,
  family_string,
  zero,
  catzero,
  spike,
  spike_decision,
  acd_parameter,
  prev_adj_params,
  ...
)
```

Arguments

<code>x</code>	an input matrix of dimensions <code>nobs x nvars</code> . Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
<code>xi</code>	a character string indicating the name of the current variable of interest, for which the best fractional polynomial transformation is to be estimated in the current step.
<code>keep</code>	a character vector with names of variables to be kept in the model.
<code>degree</code>	integer > 0 giving the degree for the FP transformation.

<code>acdx</code>	a logical vector of length <code>nvars</code> indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
<code>y</code>	a vector for the response variable or a <code>Surv</code> object.
<code>powers_current</code>	a list of length equal to the number of variables, indicating the fp powers to be used in the current step for all variables (except <code>xi</code>).
<code>powers</code>	a named list of numeric values that sets the permitted FP powers for each covariate.
<code>criterion</code>	a character string defining the criterion used to select variables and FP models of different degrees.
<code>ftest</code>	a logical indicating the use of the F-test for Gaussian models.
<code>select</code>	a numeric value indicating the significance level for backward elimination of <code>xi</code> .
<code>alpha</code>	a numeric value indicating the significance level for tests between FP models of different degrees for <code>xi</code> .
<code>family</code>	Either a character string naming the family (e.g., "gaussian", "binomial", "cox") or a function that returns a GLM family object (e.g., <code>stats::gaussian</code>). For Cox models, only a character string "cox" is allowed.
<code>family_string</code>	A character string representing the selected family, e.g., "gaussian".
<code>zero</code>	A named logical vector indicating, which columns of <code>x</code> should treat nonpositive values (zero or negative) as zero before transformation. Must be the same length as the columns of <code>x</code> .
<code>catzero</code>	A named list of binary indicator variables of length <code>ncol(x)</code> for nonpositive values, created when specific variables are passed to the <code>catzero</code> argument of <code>fit_mfp</code> . If an element of the list is <code>NULL</code> , it indicates that the corresponding variable was not specified by the user in the <code>catzero</code> argument of <code>fit_mfp</code> . Here, <code>catzero</code> is a list of binary variables, not a named logical vector as in <code>fit_mfp</code> .
<code>spike</code>	A logical vector indicating which columns of <code>x</code> contain a spike at zero. The length and order of <code>spike</code> must match those of the columns in <code>x</code> .
<code>spike_decision</code>	Named vector indicating how spike-at-zero (SAZ) variables are handled. Each element corresponds to a variable and encodes the selected strategy: 1 = include FP for positive values plus binary SAZ, 2 = treat as continuous FP only, 3 = include binary SAZ only.
<code>acd_parameter</code>	Named list of ACD parameters produced by <code>fit_acd()</code> , with length equal to <code>ncol(x)</code> . Each list element corresponds to a variable; if an element is <code>NULL</code> , the variable was not specified in the <code>acdx</code> argument of <code>fit_mfp</code> .
<code>prev_adj_params</code>	Named list storing adjustment variable transformations from previous steps for each variable.
<code>...</code>	passed to fitting functions.

Details

This function extends the algorithm used in `select_ra2()` to allow the usage of ACD transformations. The implementation follows the description in Royston and Sauerbrei (2016). The procedure is outlined in detail in the corresponding section in the documentation of `mfp2()`.

When a variable is forced into the model by including it in `keep`, then this function will not exclude it from the model (by setting its power to NA), but will only choose its functional form.

Value

A list with several components:

- `keep`: logical indicating if `xi` is forced into model.
- `acd`: logical indicating if an ACD transformation was applied for `xi`, i.e. FALSE in this case.
- `powers`: (best) fp powers investigated in step, indexing `metrics`. Ordering: `FP1(x, A(x))`, `null`, `linear`, `FP1(x, .)`, `linear(., A(x))`, `FP1(., A(x))`.
- `power_best`: a numeric vector with the best power found. The returned best power may be NA, indicating the variable has been removed from the model.
- `metrics`: a matrix with performance indices for all models investigated. Same number of rows as, and indexed by, `powers`.
- `model_best`: row index of best model in `metrics`.
- `pvalue`: p-value for comparison of linear and null model.
- `statistic`: test statistic used, depends on `ftest`.
- `spike_decision`: Spike decision flag for `xi`.
- `prev_adj_params`: Previously adjusted parameters.

References

Royston, P. and Sauerbrei, W., 2016. *mfp2: Extension of mfp using the ACD covariate transformation for enhanced parametric multivariable modeling*. *The Stata Journal*, 16(1), pp.72-87.

See Also

`select_ra2()`

summary.mfp2

Summarizing mfp2 model fits

Description

This function is a method for the generic `base::summary()` function for objects of class `mfp2`.

Usage

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

Arguments

`object` an object of class `mfp2`, usually, a result of a call to `mfp2()`.

`...` further arguments passed to the summary functions for `glm()` (`stats::summary.glm()`, i.e. families supported by `glm()`) or `coxph()` (`survival::summary.coxph()`, if `object$family = "cox"`).

Value

An object returned from `stats::summary.glm()` or `survival::summary.coxph()`, depending on the family parameter of `object`.

See Also

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

<code>transform_matrix</code>	<i>Transform each column of matrix using final FP powers or ACD transformation</i>
-------------------------------	--

Description

This function applies FP and/or ACD transformations to the columns of a matrix, optionally centers the transformed variables, and can generate binary indicators for semi-continuous variables (`catzero`). Spike-at-zero variables are supported via the `spike` and `spike_decision` arguments.

Usage

```
transform_matrix(
  x,
  power_list,
  center,
  acdx,
  keep_x_order = FALSE,
  acd_parameter_list = NULL,
  check_binary = TRUE,
  zero = NULL,
  catzero = NULL,
  spike = NULL,
  spike_decision = NULL,
  reset_zero = TRUE
)
```

Arguments

<code>x</code>	a matrix with all continuous variables shifted and scaled.
<code>power_list</code>	a named list of FP powers to be applied to the columns of <code>x</code> . Only variables named in this list are transformed.
<code>center</code>	a named logical vector specifying whether the columns in <code>x</code> should be centered. Centering will occur after transformations and will be done separately for each individual column of the transformed data matrix.
<code>acdx</code>	a named logical vector specifying the use of acd transformation.
<code>keep_x_order</code>	a logical indicating whether the order of columns should be kept as in the input matrix <code>x</code> , or if the columns should be ordered according to <code>power_list</code> . The default is FALSE, since the ordering by <code>power_list</code> reflects the <code>xorder</code> argument in <code>mfp2()</code> .
<code>acd_parameter_list</code>	A named list of ACD parameters. Only required when transformation are to be applied to new data. Entries must correspond to the entries where <code>acdx</code> is set to TRUE. Each components is to be passed to <code>transform_vector_acd()</code> . The default value NULL indicates that the parameters for the acd transformations are to be estimated.
<code>check_binary</code>	passed to <code>transform_vector_fp()</code> .
<code>zero</code>	A named logical vector specifying, for each variable, whether only positive values should be transformed. If TRUE, the transformation is applied only to positive values; nonpositive values (zero or negative) are replaced with zero. If FALSE, all values are used (after shifting, if necessary). Variable names must match the corresponding column names in <code>x</code> . The default is NULL, meaning no variables are treated as zero-transformed.
<code>catzero</code>	A named logical vector similar to <code>zero</code> , indicating which columns in <code>x</code> should treat nonpositive values as zero and additionally have a binary indicator created and included in the model. The vector must have names matching the column names in <code>x</code> . The default is NULL, meaning no categorical-zero variables are used.
<code>spike</code>	A named logical vector indicating which variables are subject to spike-at-zero handling. Default is NULL which is equivalent to setting all FALSE.
<code>spike_decision</code>	Named numeric vector with values 1, 2, or 3 specifying spike-at-zero handling for each variable. Default is NULL, meaning no spike-at-zero processing is applied. Value 1 includes both the transformed variable and binary indicator, 2 disables the spike/binary indicator, and 3 keeps only the binary indicator.
<code>reset_zero</code>	Logical. If TRUE (default), variables incorrectly flagged as <code>zero = TRUE</code> but containing only positive values are reset to FALSE.

Details

Transformations are applied via `transform_vector_fp()` and `transform_vector_acd()`. The resulting `x_transformed` matrix contains transformed variables, optionally centered. Binary indicators are appended for variables where `catzero = TRUE`.

The interaction of `spike`, `spike_decision`, and `catzero` determines whether binary indicators are included:

- If spike = TRUE:
 - spike_decision = 1: both FP/ACD transformation and binary (if catzero = TRUE)
 - spike_decision = 2: FP/ACD only; binary suppressed, regardless of catzero
 - spike_decision = 3: binary only (FP/ACD removed)
- If spike = FALSE:
 - Binary inclusion depends solely on catzero
 - FP/ACD always included if powers are specified

This distinction allows spike-at-zero variables to suppress or isolate the binary indicator based on the decision rule, while leaving non-spike variables controlled only by catzero

Centering:

- For continuous variables, the mean of the transformed values is subtracted.
- For binary variables, centering uses the minimum (typically 0).
- If zero = TRUE, the mean is computed only over the positive transformed values, while the original zero entries are left at zero. This preserves the spike-at-zero alignment on the original scale.
- Columns corresponding to catzero binary indicators are left as 0/1 in the returned matrix; their centering value is 0 in the current implementation, so the centering step does not change those columns. If you prefer mean-centering of the binary indicators (subtracting their sample proportion), supply explicit centers or post-process the returned matrix.

Column names: Transformed variable names are based on original names. FP terms are suffixed with `.i` to indicate the power index. Variables transformed using ACD are prefixed with `"A_"`. Binary indicators created for semi-continuous variables (catzero) are suffixed with `"_bin"`.

Value

If all elements of `power_list` are NA, returns NULL. Otherwise, returns with the following components:

- `x_transformed`: matrix of transformed (and optionally centered) variables. The number of columns may differ from the input matrix due to higher-order FP transformations. Binary variables are also added if catzero is not NULL.
- `centers`: values used for centering, if center = TRUE (typically all variables are centered, or none of them).
- `acd_parameter`: a named list of estimated ACD parameters, which may be empty if no ACD transformation is applied.
- `x_trafo`: list of transformed variables before centering and adding binary variables when catzero = TRUE. This is important for the spike-at-zero algorithm.
- `zero_expanded`: logical vector indicating, for each transformed column, whether zero-specific centering was applied (this may be TRUE even when no spike-at-zero processing is used). This is important for centering transformed variables via `center_matrix`.

Examples

```
x = matrix(1:100, nrow = 10)
colnames(x) = paste0("x", 1:ncol(x))
powx = setNames(replicate(ncol(x), c(1,2), simplify = FALSE), colnames(x))
center = setNames(rep(FALSE, ncol(x)), colnames(x))
acdx = setNames(rep(FALSE, ncol(x)), colnames(x))
transform_matrix(x, powx, center, acdx)
```

transform_vector_fp	<i>Functions to transform a variable using fractional polynomial powers or acd</i>
---------------------	--

Description

These functions generate fractional polynomials for a variable similar to fracgen in Stata. transform_vector_acd generates the acd transformation for a variable.

Usage

```
transform_vector_fp(
  x,
  power = 1,
  scale = 1,
  shift = 0,
  name = NULL,
  zero = FALSE,
  check_binary = TRUE
)

transform_vector_acd(
  x,
  power = c(1, 1),
  shift = 0,
  powers = NULL,
  scale = 1,
  acd_parameter = NULL,
  name = NULL,
  zero = FALSE
)
```

Arguments

x	a vector of a predictor variable.
power	a numeric vector indicating the FP power. Default is 1 (linear). Must be a vector of length 2 for acd transformation. Ignores NA, unless an ACD transformation is applied in which case power must be a numeric vector of length 2, and NA indicated which parts are used for the final FP.

scale	scaling factor for x of interest. Must be a positive integer or NULL. Default is 1, meaning no scaling is applied. If NULL, then scaling factors are automatically estimated by the program.
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 $x \leq 0$.
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 positive values of the variable should be transformed, with nonpositive values (zero or negative) set to zero. If TRUE, transformation is applied only to positive values; nonpositive values are replaced with zero before transformation. If FALSE (default), all values are shifted (if needed) to ensure positivity before transformation.
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 be changed when this function is to be used to transform data for predictions. See Details.
powers	passed to fit_acd().
acd_parameter	a list usually returned by fit_acd(). In particular, it must have components that define beta0, beta1, power, shift and scale which are to be applied when using the acd transformation in new data.

Details

The fp transformation generally transforms x as follows. For each p_i in $\text{power} = (p_1, p_2, \dots, p_n)$ it creates a variable x^{p_i} and returns the collection of variables as a matrix. It may process the data using shifting and scaling as desired. Centering has to be done after the data is transformed using these functions, if desired.

A special case are repeated powers, i.e. when some $p_i = p_j$. In this case, the fp transformations are given by x^{p_i} and $x^{p_i} * \log(x)$. In case more than 2 powers are repeated they are repeatedly multiplied with $\log(x)$ terms, e.g. $p_i = p_j = p_k$ leads to x^{p_i} , $x^{p_i} * \log(x)$ and $x^{p_i} * \log(x)^2$.

Note that the powers p_i 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 $\text{power} = c(1, 1, 2)$ and $\text{power} = c(1, 2, 1)$. 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` is set to FALSE. This is usually not necessary, the only special case to set it to FALSE is when a single value is to be transformed during prediction (e.g. to transform a reference value). When this is done, binary variables are still returned unchanged, but a single value from a continuous variable will be transformed as desired by the fitted transformations. 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. In case an acd transformation is applied, the output is a list with two entries. The first acd is the matrix of transformed variables, the acd term is returned as the last column of the matrix (i.e. in case that the power for the normal data is NA, then it is the only column in the matrix). The second entry acd_parameter returns a list of estimated parameters for the ACD transformation, or simply the input acd_parameter if it was not NULL.

Functions

- transform_vector_acd(): Function to generate acd transformation.

Data processing

An important note on data processing. Variables are shifted and scaled before being transformed by any powers. That is to ensure positive values and reasonable scales. Note that scaling does not change the estimated powers, see also find_scale_factor().

However, they may be centered after transformation. This is not done by these functions. That is to ensure that the correlation between variables stay intact, as centering before transformation would affect them. This is described in Sauerbrei et al (2006), as well as in the Stata manual of mfp. Also, centering is not recommended, and should only be done for the final model if desired.

If a variable is specified in the zero or catzero arguments, nonpositive values (zero or negative) are not shifted. Instead, they are replaced with zero, and transformation is applied only to the positive values. This approach is useful in cases where nonpositive values have a qualitatively different interpretation (e.g., nonsmokers in smoking data) and should not be transformed in the same way as positive values.

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)
transform_vector_acd(z)
```

transform_vector_power

Simple function to transform vector by a single power

Description

Simple function to transform vector by a single power

Usage

```
transform_vector_power(x, power = 1, zero = FALSE)
```

Arguments

x	a vector of a predictor variable.
power	single power.
zero	Logical indicating whether only positive values of the variable should be transformed, with nonpositive values (zero or negative) set to zero. If TRUE, transformation is applied only to positive values; nonpositive values are replaced with zero before transformation.

Value

A vector of transformed values if power is not equal to 1

Index

- * **data**
 - art, [5](#)
 - gbsg, [40](#)
 - pima, [61](#)
 - prostate, [68](#)
- apply_acd, [3](#)
- apply_shift_scale, [4](#)
- art, [5](#)
- assign_df, [5](#)
- backscale_matrix, [6](#)
- base::summary(), [81](#)
- calculate_f_test, [6](#)
- calculate_lr_test, [8](#)
- calculate_model_metrics, [8](#)
- calculate_number_fp_powers, [9](#)
- calculate_standard_error, [10](#)
- center_matrix, [11](#)
- coef.mfp2, [12](#)
- contr.cumulative, [12](#)
- contr.poly, [53](#), [59](#)
- contr.treatment, [53](#), [59](#)
- convert_powers_list_to_matrix, [13](#)
- create_dummy_variables, [14](#)
- create_fp_terms, [15](#)
- deviance_gaussian, [16](#)
- ensure_length, [17](#)
- find_best_fp1_for_acd, [24](#)
- find_best_fp_cycle, [17](#)
- find_best_fp_step, [20](#)
- find_best_fpm_step, [24](#)
- find_scale_factor, [27](#)
- find_shift_factor, [28](#)
- fit_acd, [29](#)
- fit_cox, [30](#)
- fit_glm, [31](#)
- fit_linear_step, [32](#)
- fit_model, [34](#)
- fit_null_step, [35](#)
- fp, [36](#)
- fp2(fp), [36](#)
- fracplot, [37](#)
- gbsg, [40](#)
- generate_combinations_with_replacement,
[41](#)
- generate_powers_acd
(generate_powers_fp), [41](#)
- generate_powers_fp, [41](#)
- generate_transformations_acd
(generate_transformations_fp),
[42](#)
- generate_transformations_fp, [42](#)
- get_selected_variable_names, [44](#)
- ggplot2::geom_line(), [39](#)
- mfp2, [45](#)
- mfp2(), [37](#)
- model.matrix, [53](#), [59](#)
- name_transformed_variables, [60](#)
- order_variables, [60](#)
- order_variables_by_significance
(order_variables), [60](#)
- pima, [61](#)
- plot_mfp(fracplot), [37](#)
- predict.mfp2, [62](#)
- prepare_newdata_for_predict, [66](#)
- print.mfp2, [67](#)
- print_mfp_ic_step(print_mfp_step), [67](#)
- print_mfp_pvalue_step(print_mfp_step),
[67](#)
- print_mfp_step, [67](#)
- prostate, [68](#)

reset_acd, [69](#)

select_ic, [70](#)

select_ic_acd(select_ic), [70](#)

select_linear, [73](#)

select_ra2, [76](#)

select_ra2_acd, [79](#)

stats::anova(), [8](#)

stats::coef(), [12](#)

stats::glm(), [19](#), [31](#), [32](#), [34](#), [47](#), [61](#), [82](#)

stats::glm.control(), [49](#)

stats::glm.fit(), [32](#)

stats::predict.glm(), [65](#)

stats::summary.glm(), [82](#)

summary.mfp2, [81](#)

survival::coxph(), [19](#), [22](#), [31](#), [34](#), [61](#), [82](#)

survival::coxph.control(), [49](#)

survival::coxph.fit(), [31](#)

survival::predict.coxph(), [65](#)

survival::strata(), [19](#), [22](#)

survival::summary.coxph(), [82](#)

survival::Surv(), [47](#), [61](#)

transform_matrix, [82](#)

transform_vector_acd
 (transform_vector_fp), [85](#)

transform_vector_fp, [85](#)

transform_vector_power, [87](#)