Package 'mfp2'

November 14, 2023

Type Package

Title Multivariable Fractional Polynomial Models with Extensions

Version 1.0.0

Date 2023-11-13

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 for this algorithm are Royston and Altman (1994)<doi:10.2307/2986270> and Sauerbrei and Royston (2008, ISBN:978-0-470-02842-1). In addition, it can model a 'sigmoid' relationship between variable x and an outcome variable y using the approximate cumulative distribution transformation proposed by Royston (2014) <doi:10.1177/1536867X1401400206>. 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

NeedsCompilation yes

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

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

RoxygenNote 7.2.3

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

Repository CRAN

Date/Publication 2023-11-14 18:00:03 UTC

R topics documented:

`apply_acd` *Function to apply Approximate Cumulative Distribution (ACD)*

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, ...)
```

Arguments

`x`	a numeric vector.
`beta0, beta1`	each a numeric value, representing the coefficients of the FP1 model for the ACD transformation.
`power`	a numeric value, estimated power to be used in the FP1 model for the ACD transformation.
`shift`	a numeric value that is used to shift the values of `x` to positive values.
`scale`	a numeric value used to scale `x`.
`...`	not used.

Value

The transformed input vector `x`.

apply_shift_scale *Shift and scale vector x*

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 *Artificial dataset with continuous response*

Description

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

Usage

```
data(art)
```

Format

The dataset has 250 observations and 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` *Helper to assign degrees of freedom*

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)
```

calculate_df *Helper to calculates the final degrees of freedom for the selected model*

Description

To be used in fit_mfp().

Usage

```
calculate_df(p)
```

Arguments

p	power of a variable.

Details

An example calculation: if p is the power(s) and p = c(1,2), then df = 4 but if x = NA then df = 0.

Value

returns numeric value denoting the number of degrees of freedom (df).

calculate_f_test *Function to compute F-statistic and p-value from deviances*

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 deviances[1] - 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:

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

where D refers to deviances of two models 1 and 2. $d1$ is the number of additional parameters used in in model 2 as compared to model 1, i.e. `dfs_resid[1] - dfs_resid[2]`. $d2$ is the number of residual degrees of freedom minus the number of estimated powers for model 2, i.e. `dfs_resid[2]`. #' The p-value then results from the use of a F-distribution with (d1, d2) 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 at 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 `deviance[1]` to `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` *Function to calculate p-values for likelihood-ratio test*

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 `logl[1] / logl[2]`.
`dfs`	a numeric vector with degrees of freedom.

Details

Uses Wilk's theorem that -2log(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(likelihood reduced model / likelihood full model), and then use a Chi-square distribution with df_full - 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 `logl[1]` / `logl[2]`.

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

`calculate_model_metrics` *Function to compute model metrics to be used within* `mfp2`

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 `fit_model()` 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 `-2logL + 2(df + df_additional)`.
- `bic`: Bayesian information criterion, defined as `-2logL + 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 Anaylsis 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 Anaylsis based on Fractional Polynomials for Modelling Continuous Variables. John Wiley & Sons.*

center_matrix *Simple function to center data*

Description

Simple function to center data

Usage

```
center_matrix(mat, centers = 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).

Details

Centering is done by means for continuous variables (i.e. more than 2 distinct values), and the minimum for binary variables.

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)
center_matrix(mat)
```

`coef.mfp2` *Extract coefficients from object of class* `mfp2`

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 `mfp2`, usually, a result of a call to `mfp2()`.
`...`	not used.

Value

Named numeric vector of coefficients extracted from the model `object`.

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

`data`	A dataframe containing the ordinal variable.
`var_ordinal`	Names of ordinal variables in the data for which dummy variables should be created.
`var_nominal`	Names of nominal variables in the data for which dummy variables should be created.
`drop_variables`	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)
```

Arguments

fp_powers	powers of the created FP terms.
acdx	a logical vector of length nvars indicating which continuous variables should undergo the approximate cumulative distribution (ACD) transformation.
df	a numeric vector of length nvars of degrees of freedom.
select	a numeric vector of length nvars indicating significance levels for backward elimination.
alpha	a numeric vector of length nvars indicating significance levels for tests between FP models of different degrees.
criterion	a character string defining the criterion used to select variables and FP models of different degrees.

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 for backward elimination.
- `alpha`: significance level for fractional polyomial terms.
- `selected`: logical value encoding presence in the model.
- `df_final`: final estimated degrees of freedom.
- `acd`: logical value encoding use of ACD transformation.
- `powerN`: one or more columns with the final estimated fp powers (numbered 1 to N).

---

deviance_gaussian *Deviance computations as used in mfp in stata*

Description

Deviance computations as used in mfp in stata

Usage

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

Arguments

rss	residual sum of squares.
weights	numeric vector of weights used in computation of `rss`.
n	number of observations used to compute `rss`.

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` *Helper function to ensure vectors have a specified length*

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_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)
```

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.

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, ...)
```

Arguments

`x`	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
`xi`	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.
`degree`	degrees of freedom for fp transformation of `xi`.
`y`	a vector for the response variable or a `Surv` object.
`powers_current`	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 `xi`).
`powers`	a named list of numeric values that sets the permitted FP powers for each covariate.
`acdx`	a logical vector of length nvars indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
`...`	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`.

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_best_fp_cycle` *Helper to run cycles of the mfp algorithm*

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,
  criterion,
  select,
  alpha,
  keep,
  powers,
  method,
  strata,
  verbose,
  ftest,
  control,
```

```
  rownames,
  nocenter,
  acdx
)
```

Arguments

`x`	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
`y`	a vector for the response variable or a `Surv` object.
`powers_current`	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 `xi`).
`df`	a numeric vector of length nvars of degrees of freedom.
`weights`	a vector of observation weights of length nobs.
`offset`	a vector of length nobs of offsets.
`family`	a character string representing a family object.
`criterion`	a character string defining the criterion used to select variables and FP models of different degrees.
`select`	a numeric vector of length nvars indicating significance levels for backward elimination.
`alpha`	a numeric vector of length nvars indicating significance levels for tests between FP models of different degrees.
`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 covariate.
`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`	a logical; run in verbose mode.
`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.
`acdx`	a logical vector of length nvars indicating which continuous variables should undergo the approximate cumulative distribution (ACD) transformation.

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`).

Value

current FP powers

References

Royston, P. and Sauerbrei, W., 2008. *Multivariable Model - Building: A Pragmatic Approach to Regression Anaylsis 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.*

`find_best_fp_step` *Function to estimate the best FP functions for a single variable*

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,
  criterion,
  select,
  alpha,
  keep,
  powers,
  method,
  strata,
  nocenter,
  acdx,
  ftest,
  control,
  rownames,
  verbose
)
```

Arguments

x	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
y	a vector for the response variable or a Surv object.
xi	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.
weights	a vector of observation weights of length nobs.
offset	a vector of length nobs of offsets.
df	a numeric vector indicating the maximum degrees of freedom for the variable of interest xi.
powers_current	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 xi).
family	a character string representing a family object.
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 xi.
alpha	a numeric value indicating the significance level for tests between FP models of different degrees for xi.
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 covariate.
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.
ftest	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.
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 Anaylsis based on Fractional Polynomials for Modelling Continuous Variables. John Wiley & Sons.*

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

`find_scale_factor` *Function that calculates an integer used to scale predictor*

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 `find_shift_factor()`). 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 Anaylsis based on Fractional Polynomials for Modelling Continuous Variables. John Wiley & Sons.*

`find_shift_factor` *Function that calculates a value used to shift predictor*

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 Anaylsis 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)
```

Arguments

`x`	a numeric vector.
`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.
`shift`	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 `NULL`, then the program will estimate an appropriate shift automatically (see `find_shift_factor()`).
`scale`	a numeric used to scale `x`. The default value is 1, meaning no scaling is conducted. If `NULL`, then the program will estimate an appropriate scaling factor automatically (see `find_scale_factor()`).

Details

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

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

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

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

where the fitted values of the estimated model are used. If the relationship between a response Y and acd(x) is linear, say, $E(Y) = \beta_0 + \beta_1 acd(x)$, 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 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). *mfpa: 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 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 FALSEuses 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. stats::glm.fit()), while FALSE uses the normal fitting routines (stats::glm()) (used for the final output of mfp2). The difference is mainly due to the fact that normal fitting routines have to handle data.frames, 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.

---

fit_linear_step *Function to fit linear model for variable of interest*

Description

"Linear" model here refers to a model which includes the variable of interest xi with a fp power of 1. Note that xi may be ACD transformed if indicated by acdx[xi]. For parameter definitions, see find_best_fp_step(). All parameters captured by ... are passed on to fit_model().

Usage

```
fit_linear_step(x, xi, y, powers_current, powers, acdx, ...)
```

Arguments

x	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
xi	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.
y	a vector for the response variable or a Surv object.
powers_current	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 xi).
powers	a named list of numeric values that sets the permitted FP powers for each covariate.
acdx	a logical vector of length nvars indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
...	passed to fit_model().

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.

`fit_mfp` *Function for fitting a model using the MFP or MFPA algorithm*

Description

This function is not exported and is intended to be called from the `mfp2()` function. While most parameters are explained in the documentation of `mfp2()`, their form may differ in this function. Note that this function does not check its arguments and expects that its input has been prepared in `mfp2()` function.

Usage

```
fit_mfp(
  x,
  y,
  weights,
  offset,
  cycles,
  scale,
  shift,
  df,
  center,
  family,
  criterion,
  select,
  alpha,
  keep,
  xorder,
  powers,
  method,
  strata,
  nocenter,
  acdx,
  ftest,
  control,
  verbose
)
```

Arguments

`x`	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
`y`	a vector for the response variable or a `Surv` object.
`weights`	a vector of observation weights of length nobs.
`offset`	a vector of length nobs of offsets.
`cycles`	an integer representing the maximum number of iteration cycles during which FP powers for all predictor are updated.
`scale`	a numeric vector of length nvars of scaling factors. Not applied, but re-ordered to conform to `xorder`.
`shift`	a numeric vector of length nvars of shifts. Not applied, but re-ordered to conform to `xorder`.
`df`	a numeric vector of length nvars of degrees of freedom.
`center`	a logical vector of length nvars indicating if variables are to be centered.
`family`	a character string representing a family object.
`criterion`	a character string defining the criterion used to select variables and FP models of different degrees.
`select`	a numeric vector of length nvars indicating significance levels for backward elimination.
`alpha`	a numeric vector of length nvars indicating significance levels for tests between FP models of different degrees.
`keep`	a character vector with names of variables to be kept in the model.
`xorder`	a string determining the order of entry of the covariates into the model-selection algorithm.
`powers`	a named list of numeric values that sets the permitted FP powers for each covariate.
`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()`.
`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 which continuous variables should undergo the approximate cumulative distribution (ACD) transformation.
`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.
`verbose`	a logical; run in verbose mode.

Value

See `mfp2()` for details on the returned object.

Algorithm

- Step 1: order variables according to `xorder`. This step may involve fitting a regression model to determine order of significance.
- Step 2: input data pre-processing. Setting initial powers for fractional polynomial terms, checking if acd transformation is required and allowed. Note that the initial powers of all variables are always set to 1, and higher FPs are only evaluated in turn for each variables in the first cycle of the algorithm. See e.g. Sauerbrei and Royston (1999).
- Step 3: run mfp algorithm cycles. See `find_best_fp_cycle()` for more details.
- Step 4: fit final model using estimated powers.

References

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.*

See Also

`mfp2()`, `find_best_fp_cycle()`

`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 = "gaussian",
  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 `colnames(x, do.NULL = FALSE)`.
`y`	a vector for the outcome variable for glms, and a `Surv` object for Cox models.
`family`	a character strong 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 `fit_glm()` and `fit_cox()`. @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 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, ...)
```

Arguments

`x`	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
`xi`	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.
`y`	a vector for the response variable or a `Surv` object.

`powers_current`	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 `xi`).
`powers`	a named list of numeric values that sets the permitted FP powers for each covariate.
`acdx`	a logical vector of length nvars indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
`...`	passed to `fit_model()`.

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 — *Helper to assign attributes to a variable undergoing FP-transformation*

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
)

fp2(...)
```

Arguments

`x`	a vector representing a continuous variable undergoing fp-transformation.
`df, alpha, select, shift, scale, center, acdx`	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

Plots the partial linear predictors with confidence limits against the selected covariate(s) of interest.

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 = "#000000",
  color_fill = "#000000",
  shape = 1,
  size_points = 1,
  linetype = "solid",
  linewidth = 1,
  alpha_fill = 0.1
)

plot_mfp(...)
```

Arguments

- `model` — fitted `mfp2` model.
- `terms` — character vector with variable names to be plotted.
- `partial_only` — a logical value indicating whether only the partial predictor (component) is drawn (`TRUE`), or also component-plus-residual (`FALSE`, the default). Only used if `type = "terms"`. See below for details.
- `type, ref, terms_seq` — arguments of `predict.mfp2()`. Only `type = "terms"` and `type = "contrasts"` are supported by this function.
- `alpha` — `alpha` argument of `predict.mfp2()`.
- `color_line, linetype, linewidth` — `ggplot2` properties of line for partial predictor.
- `color_fill, alpha_fill` — `ggplot2` properties of ribbon for confidence interval.
- `shape, size_points, color_points` — `ggplot2` properties of drawn data points.
- `...` — used in alias `plot_mfp` to pass arguments.

Details

The confidence limits of the partial linear predictors or contrasts are obtained from the variance–covariance matrix of the final fitted model, which takes into account the uncertainty in estimating the model parameters but not the FP powers. This can lead to narrow confidence intervals. A simple way to obtain more realistic confidence intervals within the FP is by using bootstrap, which is not currently implemented. See Royston and Sauerbrei (2008) chapter 4.9.2 for guidance on conducting bootstrapping within the FP class.

The component-plus-residual, is the partial linear predictor plus residuals, where deviance residuals are used in generalized linear regression models, while martingale residuals are used in Cox models, as done in Stata mfp program. This kind of plot is only available if `type = "terms"`.

Value

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

Functions

- `plot_mfp()`: Alias for fracplot.

See Also

`predict.mfp2()`

Examples

```
# Gaussian
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*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 very simple 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 is replicating the functionality from `arrangements::combinations` with `replace = TRUE`. Note that base R function `utils::combn` only returns combinations without replacement, thus pairs like (0, 0) are not in the output.

Note that this function is extremely inefficient and only intended to be used with small use cases, i.e. small k. This is typically the case in the context of MFP, but a warning is given if this is not the case since the algorithm may take a while to compute the combinations, and even longer to do model selection.

Value

A m x k matrix, where m is the number of combinations.

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)

generate_transformations_acd(x, degree, powers)
```

Arguments

x	a numeric vector of length nobs assumed to have been shifted and scaled.
degree	numeric indicating the degree of FPs. Assumed to be 2 for acd transformation.
powers	a vector of allowed FP powers.

Details

Any FP transformation is given by a vector of powers, e.g. (p1, p2) for degree 2. These correspond to powers x^p1 and x^p2. Thus, we only need to consider combinations of all values in `powers`, since order of the entries does not matter. See `generate_powers_fp()`. A special case are repeated powers, i.e. p1 = p2. In this case, the repeated entries are multiplied by log(x) (see `transform_vector_fp()`).

When the ACD transformation is requested, then all pairs of length 2 are considered, i.e. 64. 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).

Value

A list with two entries:

- `data`: a list with length equal to the number of possible FPs for the variable of interest. Each entry is a matrix with degree many columns, and nobs observations comprising the FP transformed input variable. For example, for degree = 2 and nobs = 10, each entry is a 10 x 2 matrix. Values are not centered. If `degree = 0`, the single entry has a single column.
- `powers`: the associated FP powers for each entry in data.

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 has the ability to model a sigmoid relationship between x and an outcome variable y using the approximate cumulative distribution (ACD) transformation proposed by Royston (2014). This function provides two interfaces for input data: one for inputting data matrix x and outcome vector y directly and the other for using a formula object together with a dataframe data. Both interfaces are equivalent in terms of 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 = c("gaussian", "poisson", "binomial", "cox"),
  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,
  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 = c("gaussian", "poisson", "binomial", "cox"),
  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,
  verbose = TRUE,
  ...
)
```

Arguments

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

`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 `select` and `alpha` parameters below). If the user specifies the BIC (`bic`) or AIC (`aic`) 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 nvars 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 `mfp2.formula` only supports single numeric input to set a default value, individual values can be set using `fp` 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 nvars 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 `mfp2.formula` only supports single numeric input to set a default value, individual values can be set using `fp` 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 `criterion = "pvalue"`, this is equivalent to setting the selection level for the variables in `keep` 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. 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`.

`powers` a named list of numeric values that sets the permitted FP powers for each covariate. The default is NULL, and each covariate is assigned `powers = c(-2, -1, -0.5, 0, 0.5, 1, 2, 3)`, where 0 means the natural logarithm. Powers are sorted before further processing in the program. If some variables are not assigned powers, the default powers will be assigned. The formula interface offers two options for supplying powers: through the 'powers' argument and the 'fp()' function. So, if the user supplies powers in both options for a certain variable, the powers supplied through 'fp()' will be given preference.For the algorithm to select the powers, each variable must have a minimum of two powers. If the users wants to use one power, they should first transform their variables before using `mfp2()` function and specify appropriate df

`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 `survival::coxph()` 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 survival::coxph() for details.
nocenter	a numeric vector with a list of values for fitting Cox models. See survival::coxph() for details.
acdx	a numeric vector of names of continuous variables to undergo the approximate cumulative distribution (ACD) transformation. It also invokes the function-selection procedure to determine the best-fitting FP1(p1, p2) model (see Details section). Not present in the formula interface mfp2.formula and to be set using fp terms in the formula input. The variable representing the ACD transformation of x is named A_x.
ftest	a logical; for normal error models with small samples, critical points from the F-distribution can be used instead of Chi-Square distribution. Default FALSE uses the latter. This argument is used for Gaussian models only and has no effect for other model families.
control	a list object with parameters controlling model fit details. Returned by either stats::glm.control() or survival::coxph.control(). Default is NULL to use default parameters for the given model class.
verbose	a logical; run in verbose mode.
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.formula: 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_string: the family stored as character string.

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 FPs

In the following we denote fractional polynomials for a variable x by increasing complexity as either FP1 or FP2. In this example, $FP2(p1, p2)$ for $p1 \neq p2$ is the most flexible FP transformation, where

$$FP2(p1, p2) = \beta_1 x^{p1} + \beta_2 x^{p2}.$$

When $p1 = p2$ (repeated powers), the FP2 model is given by

$$FP2(p1, p2) = \beta_1 x^{p1} + \beta_2 x^{p1} log(x).$$

The powers $p1$ and $p2$ are usually chosen from a predefined set of powers $S = (-2, -1, -0.5, 0, 0.5, 1, 2, 3)$ where the power of 0 indicates the natural logarithm. The best FP2 is then estimated by using a closed testing procedure that seeks the best combination from all 36 pairs of powers $(p1, p2)$. Functions that only involve a single power of the variable are denoted as FP1, i.e.

$$FP1(p1) = \beta_1 x^{p1}.$$

For details see e.g. Sauerbrei et al (2006).

Details on `family` option

`mfp2()` supports the family object as used by `stats::glm()`. The built in families are specified via a character string. `mfp2(..., family = "binomial")` fits a logistic regression model, while `mfp2(..., family = "gaussian")` fits a linear regression (ordinary least squares) model.

For Cox models, the response should preferably be a `Surv` object, created by the `survival::Surv()` function, and the `family = "cox"`. Only right-censored data are currently supported. To fit stratified Cox models, the `strata` option can be used, or alternatively `strata` terms can be included 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

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

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 $\beta'_0 + \beta'_1(x'^p - mean(x'^p))$. 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. 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

Note that subsetting occurs after data pre-processing (shifting and scaling), but before model selection and fitting. In detail, when the option `subset` is used and scale, shift or centering values are to be estimated, then `mfp2()` first estimates these parameters using the full dataset (no subsetting). It then conduct subsetting before proceeding to perform model selection and fitting on the specified subset of the data.

Therefore, subsetting in `mfp2()` is not equivalent to subsetting the data before passing it to `mfp2()`, and thus cannot be used to implement, for example, cross-validation or to remove `NA`. These tasks should be done by the caller beforehand. However, it does allow to use the same data pre-processing for different subsets of the data. An example use case is when separate models are to be estimated for women and men in the dataset, but a common data pre-processing should be applied. In this case the `subset` option can be used to restrict model selection to either women or men, but the data processing (e.g. shifting factors) will be shared between the two models.

Details on approximate cumulative distribution transformation

The approximate cumulative distribution (ACD) transformation (Royston 2014) converts each predictor, x, smoothly to an approximation, $acd(x)$, of its empirical cumulative distribution function. This is done by smoothing a probit transformation of the scaled ranks of x. $acd(x)$ could be used instead of x as a covariate. This has the advantage of providing sigmoid curves, something that regular FP functions cannot achieve. Details of the precise definition and some possible uses of the ACD transformation in a univariate context are given by Royston (2014). Royston and Sauerbrei (2016) describes how one could go further and replace FP2 functions with a pair of FP1 functions, one in x and the other in $acd(x)$.

This alternative class of four-parameter functions provides about the same flexibility as the standard FP2 family, but the ACD component offers the additional possibility of sigmoid functions. Royston (2014) discusses how the extended class of functions known as $FP1(p1, p2)$, namely

$$FP1(p1, p2) = \beta_1 x^{p1} + \beta_2 acd(x)^{p2}$$

can be fitted optimally by seeking the best combination of all 64 pairs of powers (p1, p2). The optimisation is invoked by use of the `acdx` parameter. Royston (2014) also described simplification of the chosen function through model reduction by applying significance testing to six sub-families of functions,M1-M6, giving models M1 (most complex) through M6 (null):

- M1: FP1(p1, p2) (no simplification)
- M2: FP1(p1, .) (regular FP1 function of x)
- M3: FP1(., p2) (regular FP1 function of $acd(x)$)
- M4: FP1(1, .) (linear function of x)
- M5: FP1(., 1) (linear function of $acd(x)$)
- M6: Null (x omitted entirely)

Selection among these six sub-functions is performed by a closed test procedure known as the function-selection pocedure 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.

Details on model specification using a `formula`

`mfp2` supports model specifications using two different interfaces: one which allows passing of the data matrix `x` and outcome vector `y` directly (as done in e.g. `stats::glm.fit()` or `glmnet`) and another which conforms to the formula interface used by many commonly used R modelling functions such as `stats::glm()` or `survival::coxph()`.

Both interfaces are equivalent in terms of possible fitted models, only the details of specification differ. In the standard interface all details regarding FP-transformations are given as vectors. In the formula interface all details are specified using special `fp()` function. These support the specification of 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`) and the ACD-transformation (`acd`). Values specified through `fp()` function override the values specified as defaults and passed to the `mfp2()` function.

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 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.

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 four cycles to achieve convergence. Lack of convergence involves oscillation between two or more models and is extremely rare. If the model does not converge, you can try changing the nominal significance levels for variable (select) or function selection (alpha).

References

Royston, P. and Sauerbrei, W., 2008. *Multivariable Model - Building: A Pragmatic Approach to Regression Anaylsis 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. *mfpa: Extension of mfp using the ACD covariate transformation for enhanced parametric multivariable modeling. The Stata Journal, 16(1), pp.72-87.*

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

See Also

summary.mfp2(), coef.mfp2(), predict.mfp2(), fp()

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,
  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 `order_variables_by_significance`.
`y`	a vector of responses for glms, or a `Surv` object generated using the `survival::Surv()` function for Cox models.
`family`	a character string naming a family function supported by `glm()` or "cox" for Cox models.
`weights, offset`	parameters for both glm and Cox models, see either `stats::glm()` or `survival::coxph()` depending on family.
`strata, method, control, nocenter`	Cox model specific parameters, see `survival::coxph()`.

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 *Fits*

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,
  ...
)
```

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. See mfp2() for 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 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 (or minimum for binary variables) will be used as reference. Values are 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 minima (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 will be directly added to the linear predictor without any transformations.

... — further arguments passed to predict.glm() or predict.coxph().

Details

To prepare the newdata for prediction, this function applies any necessary shifting and scaling based on the factors obtained from the training data. It is important to note that if the shifting factors are not sufficiently large as estimated from the training data, variables in newdata may end up with negative values, which can cause prediction errors if non-linear functional forms are used. A warning is given in this case by the function. The next step involves transforming the data using the selected fractional polynomial (FP) powers. If necessary, centering of variables is conducted. Once the transformation (and centering) is complete, the transformed data is passed to either predict.glm() or predict.coxph(), depending on the chosen family of models and when type is not terms and contrasts.

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.
- variable_pre: variable with pre-transformation applied, i.e. shifted, scaled 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

This function allows to compute the partial linear predictors for each variable selected into the final model if type = "terms". Note that the results returned from this function are different from those of predict.glm() and predict.coxph() since these functions do not take into account that a single variable can be represented by multiple terms. This functionality is useful to assess model fit, since it also allows to draw data points based on residuals.

Contrasts

This functions allows to compute contrasts with reference to a specified variable value if `type = "contrasts"`. In this case, the fitted partial predictors will be centered at the reference value (i.e. 0), and also confidence intervals will have width 0 at that point.

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
```

---

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
)
```

Arguments

`object`	fitted `mfp2` model object.
`newdata`	dataset to be prepared for predictions. Its columns can be a subset of the columns used for fitting the model.
`strata, offset`	passed from `predict.mfp2()`.

`apply_pre`	logical indicating wether the fitted pre-transformation is applied or not.
`apply_center`	logical indicating whether the fitted centers are applied after transformation or not.
`check_binary`	passed to `transform_vector_fp()`.

Value

A dataframe of transformed newdata

`print.mfp2` *Print method for objects of class* `mfp2`

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` *Function for verbose printing of function selection procedure (FSP)*

Description

Function for verbose printing of function selection procedure (FSP)

Usage

```
print_mfp_step(xi, criterion, fit)

print_mfp_pvalue_step(xi, fit, criterion)

print_mfp_ic_step(xi, fit, criterion)
```

Arguments

`xi`	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.
`criterion`	a character string defining the criterion used to select variables and FP models of different degrees.
`fit`	intermediary model fit in `mfp_step`.

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

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

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

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,
  ...
)

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

Arguments

x	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
xi	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.
keep	a character vector with names of variables to be kept in the model.
degree	integer > 0 giving the degree for the FP transformation.
acdx	a logical vector of length nvars indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
y	a vector for the response variable or a Surv object.
powers_current	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 xi).
powers	a named list of numeric values that sets the permitted FP powers for each covariate.
criterion	a character string defining the criterion used to select variables and FP models of different degrees.
ftest	a logical indicating the use of the F-test for Gaussian models.
select	a numeric value indicating the significance level for backward elimination of xi.
alpha	a numeric value indicating the significance level for tests between FP models of different degrees for xi.
...	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.

Functions

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

See Also

`select_ra2()`

select_linear — *Helper to select between null and linear term for a single variable*

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 on to fit_model().

Usage

```
select_linear(
  x,
  xi,
  keep,
  degree,
  acdx,
  y,
  powers_current,
  powers,
  criterion,
  ftest,
  select,
  alpha,
  ...
)
```

Arguments

x	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
xi	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.
keep	a character vector with names of variables to be kept in the model.
degree	not used.
acdx	a logical vector of length nvars indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
y	a vector for the response variable or a Surv object.
powers_current	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 xi).
powers	a named list of numeric values that sets the permitted FP powers for each covariate.

criterion	a character string defining the criterion used to select variables and FP models of different degrees.
ftest	a logical indicating the use of the F-test for Gaussian models.
select	a numeric value indicating the significance level for backward elimination of xi.
alpha	a numeric value indicating the significance level for tests between FP models of different degrees for xi.
...	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 a 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.

---

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 on to fit_model().

Usage

```
select_ra2(
  x,
  xi,
  keep,
  degree,
  acdx,
  y,
  powers_current,
  powers,
  criterion,
  ftest,
  select,
  alpha,
  ...
)
```

Arguments

x	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
xi	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.
keep	a character vector with names of variables to be kept in the model.
degree	integer > 0 giving the degree for the FP transformation.
acdx	a logical vector of length nvars indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
y	a vector for the response variable or a Surv object.
powers_current	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 xi).
powers	a named list of numeric values that sets the permitted FP powers for each covariate.
criterion	a character string defining the criterion used to select variables and FP models of different degrees.
ftest	a logical indicating the use of the F-test for Gaussian models.
select	a numeric value indicating the significance level for backward elimination of xi.
alpha	a numeric value indicating the significance level for tests between FP models of different degrees for xi.
...	passed to fitting functions.

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 FPm function against a null model at level select with 2m df. If not significant, the variable is excluded. Otherwise continue with step 2.
- *Step 2*: test the best FPm versus a linear model at level alpha with 2m - 1 df. If not significant, use a linear model. Otherwise continue with step 3.
- *Step 3*: test the best FPm versus the best FP1 at level alpha with 2m - 2 df. If not significant, use the best FP1 model. Otherwise, repeat this step for all remaining higher order FPs until FPm-1, which is tested at level alpha with 2 df against FPm. If the final test is not significant, use a FPm-1 model, otherwise use FPm.

Note that the "best" FPx model used in each step is given by the model 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().

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. Always starts with highest power, then null, then linear, then FP in increasing degree (e.g. FP2, null, linear, FP1).
- 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.

References

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

See Also

select_ra2_acd()

select_ra2_acd	*Function selection procedure for ACD based on closed testing procedure*

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,
  ...
)
```

Arguments

`x`	an input matrix of dimensions nobs x nvars. Does not contain intercept, but columns are already expanded into dummy variables as necessary. Data are assumed to be shifted and scaled.
`xi`	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.
`keep`	a character vector with names of variables to be kept in the model.
`degree`	integer > 0 giving the degree for the FP transformation.
`acdx`	a logical vector of length nvars indicating continuous variables to undergo the approximate cumulative distribution (ACD) transformation.
`y`	a vector for the response variable or a `Surv` object.
`powers_current`	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 `xi`).
`powers`	a named list of numeric values that sets the permitted FP powers for each covariate.

criterion	a character string defining the criterion used to select variables and FP models of different degrees.
ftest	a logical indicating the use of the F-test for Gaussian models.
select	a numeric value indicating the significance level for backward elimination of xi.
alpha	a numeric value indicating the significance level for tests between FP models of different degrees for xi.
...	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.

References

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

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()

transform_data_step	*Function to extract and transform adjustment variables*

Description

Function to extract and transform adjustment variables

Usage

```
transform_data_step(x, xi, powers_current, df, powers, acdx)
```

Arguments

x	a matrix of predictors that includes the variable of interest xi. It is assumed that continuous variables have already been shifted and scaled.
xi	name of the continuous predictor for which the FP function will be estimated. There are no binary or two-level variables allowed. All variables except xi are referred to as "adjustment variables".
powers_current	a named list of FP powers of all variables of interest, including xi. Note that these powers are updated during backfitting or MFP cycles.
df	a numeric vector of degrees of freedom for xi.
powers	a set of allowed FP powers.
acdx	a logical vector indicating the use of acd transformation.

Details

After extracting the adjustment variables this function, using their corresponding FP powers stored in powers_current, transforms them. This is necessary When evaluating x of interest, as we must account for other variables, which can be transformed or untransformed, depending on the individual powers. It's worth noting that some powers can be NA, indicating that the variable has been left out of the adjustment variables. It also returns the FP data, which is dependent on the degrees of freedom. For example, df = 2 is equivalent to FP degree one, resulting in the generation of 8 variables. If acdx for the current variables of interest is set to TRUE, however, 64 variables are generated.

When df = 1, this function returns data unchanged, i.e. a "linear" transformation with power equal to 1. In case acdx[xi] = TRUE, the acd transformation is applied.

Value

A list containing the following elements:

- powers_fp: fp powers used for data_fp.
- data_fp: a list with all possible fp transformations for xi, see the data component of the output of generate_transformations_fp() and generate_transformations_acd().
- powers_adj: fp powers for adjustment variables in data_adj.
- data_adj: adjustment data, i.e. transformed input data for adjustment variables.

---

transform_matrix — *Function to transform each column of matrix using final FP powers or acd*

Description

Function to transform each column of matrix using final FP powers or acd

Usage

```
transform_matrix(
  x,
  power_list,
  center,
  acdx,
  keep_x_order = FALSE,
  acd_parameter_list = NULL,
  check_binary = TRUE
)
```

Arguments

- `x` a matrix with all continuous variables shifted and scaled.
- `power_list` a named list of FP powers to be applied to the columns of `x`. Only variables named in this list are transformed.
- `center` a named logical vector specifying whether the columns in `x` should be centered. Centering will occur after transformations and will be done separately for each individual column of the transformed data matrix.
- `acdx` a named logical vector specifying the use of acd transformation.
- `keep_x_order` a logical indicating whether the order of columns should be kept as in the input matrix `x`, of if the columns should be ordered according to `power_list`. The default is `FALSE`, since the ordering by `power_list` reflects the `xorder` argument in `mfp2()`.
- `acd_parameter_list` a named list. Only required when transformation are to be applied to new data. Entries must correspond to the entries where `acdx` is set to `TRUE`. Each components is to be passed to `transform_vector_acd()`. The default value `NULL` indicates that the parameters for the acd transformations are to be estimated.
- `check_binary` passed to `transform_vector_fp()`.

Details

For details on the transformations see `transform_vector_fp()` and `transform_vector_acd()`.

Value

If all elements of `power_list` are `NA` then this function returns `NULL`. Otherwise a list with three entries: the first `x_transformed` is a matrix with transformed variables as named in `power_list`. The number of columns may possibly be different to the input matrix due to higher order FP transformations. The second entry `centers` stores the values used to center the variables if for any variable `center = TRUE` (note that usually all variables are centered, or none of them). The third entry `acd_parameter` stores a named list of estimated `acd_parameters`. May be empty if no ACD transformation is applied.

Column names

Generally the original variable names are suffixed with ".i", where i enumerates the powers for a given variable in power_list. If a term uses an acd transformation, then the variable is prefixed with A_.

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 *Functions to transform a variable using fractional polynomial powers or acd*

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,
  check_binary = TRUE
)

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

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` <= 0.
`name`	character used to define names for the output matrix. Default is `NULL`, meaning the output will have unnamed columns.
`check_binary`	a logical indicating whether or not input `x` is checked if it is a binary variable (i.e. has only two distinct values). The default `TRUE` usually only needs to changed when this function is to be used to transform data for predictions. See Details.
`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 pi in `power` = (p1, p2, ..., pn) it creates a variable x^pi 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 pi = pj. In this case, the fp transformations are given by x^pi and x^pi * log(x). In case more than 2 powers are repeated they are repeatedly multiplied with log(x) terms, e.g. pi = pj = pk leads to x^pi, x^pi * log(x) and x^pi * log(x)^2.

Note that the powers pi are assumed to be sorted. That is, this function sorts them, then proceeds to compute the transformation. For example, the output will be the same for `power = c(1, 1, 2)` and `power = c(1, 2, 1)`. This is done to make sense of repeated powers and to uniquely define FPs. In case an ACD transformation is used, there is a specific order in which powers are processed, which is always the same (but not necessarily sorted). Thus, throughout the whole package powers will always be given and processed in either sorted, or ACD specific order and the columns of the matrix returned by this function will always align with the powers used throughout this package.

Binary variables are not transformed, unless `check_binary` 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.

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)
```

Arguments

`x`	a vector of a predictor variable.
`power`	single power.

Value

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

Index