Package: disttree (via r-universe)

September 16, 2024

Title Trees and Forests for Distributional Regression

Date 2021-03-12

Version 0.2-0

Description Infrastructure for fitting distributional regression trees and forests based on maximum-likelihood estimation of parameters for specified distribution families, for example from the GAMLSS family.

Depends R (>= 3.1.0), partykit (>= 1.2-5), gamlss.dist

Imports graphics, stats, utils, Formula, sandwich, survival

Suggests crch, gamlss.cens, lmtest, mvtnorm, countreg, tinytest, RainTyrol

LazyData yes

License GPL-2 | GPL-3

RoxygenNote 7.1.1

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

RemoteUrl https://github.com/r-forge/partykit

RemoteRef HEAD

RemoteSha 4309c9bb2890ddf1e743e937b93a0281f28e2e17

Contents

distfamily *Preparation of family object of class* disttree.family *as employed in* distfit, disttree, *and* distforest

Description

The function distfamily prepares the required family object that is employed within distfit to estimate the parameters of the specified distribution family.

Usage

```
distfamily(family, bd = NULL, censpoint = NULL)
```

Arguments

family	can be one of the following: gamlss.family object, gamlss.family function, character string with the name of a gamlss.family object, function generating a family object with the required information about the distribution, character string with the name of a function generating a family object with the required information about the distribution, list with the required information about the distribution, character string with the name of a distribution for which a family generating function is provided in disttree
bd	optional argument for binomial distributions specifying the binomial denominator
censpoint	censoring point for a censored gamlss.family object

Details

The function distfamily is applied within distfit, disttree, and distforest. It generates a family object of class disttree.family. If family is a gamlss.family object the function distfamily_gamlss is called within distfamily.

Value

distfamily returns a family object of class disttree.family in form of a list with the following components:

family.name	character string with the name of the specified distribution family
ddist	density function of the specified distribution family.
sdist	score function (1st partial derivatives) of the specified distribution family.
hdist	hessian function (2nd partial derivatives) of the specified distribution family.
pdist	distribution function of the specified distribution family.
qdist	quantile function of the specified distribution family.
rdist	random generation function of the specified distribution family.
link	character strings of the applied link functions.

linkfun	link functions.
linkinv	inverse link functions.
linkinvdr	derivative of the inverse link functions.
startfun	function generating the starting values for the employed optimization.
mle	logical. Indicates whether a closed form solution exists (TRUE) for the maximum-likelihood optimization or whether a numerical optimization should be employed to estimate parameters (FALSE).
gamlssobj	logical. Indicates whether the family has been obtained from a gamlss.family object.
censored	logical. Indicates whether the specified distribution family is censored.
censpoint	numeric. Censoring point (only if censored and gamlssobj),
censtype	character. Type of censoring ("left", "right") (only if censored and gamlssobj).

References

Stasinopoulos DM, Rigby RA (2007). Generalized Additive Models for Location Scale and Shape (GAMLSS) in R, *Journal of Statistical Software*, **23**(7), 1-46. doi:10.18637/jss.v023.i07

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th Edition. Springer-Verlag, New York.

See Also

disttree.family, gamlss.family

Examples

```
library(disttree)
family <- distfamily(family = NO())
```

---

distfit *Maximum-Likelihood Fitting of Parametric Distributions*

Description

The function distfit carries out maximum-likelihood estimation of parameters for a specified distribution family, for example from the GAMLSS family (for generalized additive models for location, scale, and shape). The parameters can be transformed through link functions but do not depend on further covariates (i.e., are constant across observations).

Usage

```
distfit(y, family = NO(), weights = NULL, start = NULL, start.eta = NULL,
          vcov = TRUE, type.hessian =  c("checklist", "analytic", "numeric"),
          method = "L-BFGS-B", estfun = TRUE, optim.control = list(), ...)
```

Arguments

y	numeric vector of the response
family	specification of the response distribution. Either a gamlss.family object, a list generating function or a family list.
weights	optional numeric vector of case weights.
start	starting values for the distribution parameters handed over to optim
start.eta	starting values for the distribution parameters on the link scale handed over to optim.
vcov	logical. Specifies whether or not a variance-covariance matrix should be calculated and returned.
type.hessian	Can either be 'checklist', 'analytic' or 'numeric' to decide how the hessian matrix should be calculated in the fitting process in distfit. For 'checklist' it is checked whether a function 'hdist' is given in the family list. If so, 'type.hessian' is set to 'analytic', otherwise to 'numeric'.
method	Optimization which should be applied in optim
estfun	logical. Should the matrix of observation-wise score contributions (or empirical estimating functions) be returned?
optim.control	A list with optim control parameters.
...	further arguments passed to optim.

Details

The function distfit fits distributions, similar to fitdistr from **MASS** (Venables and Ripley 2002) but based on GAMLSS families (Stasinopoulos and Rigby 2007).

Provides analytical gradients and hessian, can be plugged into mob.

The resulting object of class distfit comes with a set of standard methods to generic functions including coef, estfun, vcov, predict and logLik.

Value

distfit returns an object of class distfit which is a list with the following components:

npar	number of parameter
y	numeric vector of the response
ny	number of observations
weights	numeric vector of case weights handed over as input argument
family	employed distribution family list of class disttree.family
start	used starting values in optim that were handed over as input argument
starteta	starting values on the link scale used in optim
opt	list returned by optim
converged	logical. TRUE if optim returns convergence = 0 and FALSE else.
par	fitted distribution parameters (on parameter scale)

eta	fitted distribution parameters (on link scale)
hess	hessian matrix
vcov	variance-covariance matrix
loglik	value of the maximized log-likelihood function
call	function call
estfun	matrix with the scores for the estimated parameters. Each line represents an observation and each column a parameter.
ddist	density function with the estimated distribution parameters already plugged in
pdist	probability function with the estimated distribution parameters already plugged in
qdist	quantile function with the estimated distribution parameters already plugged in
rdist	random number generating function with the estimated distribution parameters already plugged in
method	optimization method applied in optim

References

Stasinopoulos DM, Rigby RA (2007). Generalized Additive Models for Location Scale and Shape (GAMLSS) in R, *Journal of Statistical Software*, **23**(7), 1-46. doi:10.18637/jss.v023.i07

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th Edition. Springer-Verlag, New York.

See Also

gamlss.family, optim

Examples

```
## simulate artifical negative binomial data
set.seed(0)
y <- rnbinom(1000, size = 1, mu = 2)

## simple distfit
df <- distfit(y, family = NBI)
```

---

distforest *Distributional Regression Forests*

Description

Forests based on maximum-likelihood estimation of parameters for specified distribution families, for example from the GAMLSS family (for generalized additive models for location, scale, and shape).

Usage

```
distforest(formula, data, subset, na.action = na.pass, weights,
            offset, cluster, family = NO(), strata,
            control = disttree_control(teststat = "quad", testtype = "Univ",
            mincriterion = 0, saveinfo = FALSE, minsplit = 20, minbucket = 7,
            splittry = 2, ...),
            ntree = 500L, fit.par = FALSE,
            perturb = list(replace = FALSE, fraction = 0.632),
            mtry = ceiling(sqrt(nvar)), applyfun = NULL, cores = NULL,
            trace = FALSE, ...)
## S3 method for class 'distforest'
predict(object, newdata = NULL,
        type = c("parameter", "response", "weights", "node"),
        OOB = FALSE, scale = TRUE, ...)
```

Arguments

formula	a symbolic description of the model to be fit. This should be of type `y ~ x1 + x2` where `y` should be the response variable and `x1` and `x2` are used as partitioning variables.
data	a data frame containing the variables in the model.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain missing value.
weights	an optional vector of weights to be used in the fitting process. Non-negative integer valued weights are allowed as well as non-negative real weights. Observations are sampled (with or without replacement) according to probabilities `weights / sum(weights)`. The fraction of observations to be sampled (without replacement) is computed based on the sum of the weights if all weights are integer-valued and based on the number of weights greater zero else. Alternatively, `weights` can be a double matrix defining case weights for all `ncol(weights)` trees in the forest directly. This requires more storage but gives the user more control.
offset	an optional vector of offset values.
cluster	an optional factor indicating independent clusters. Highly experimental, use at your own risk.
family	specification of the response distribution. Either a `gamlss.family` object, a list generating function or a family list.
strata	an optional factor for stratified sampling.
control	a list with control parameters, see `disttree_control`. The default values that are not set within the call of `distforest` correspond to those of the default values used by `disttree` from the `disttree` package. `saveinfo = FALSE` leads to less memory hungry representations of trees. Note that arguments `mtry`, `cores` and `applyfun` in `disttree_control` are ignored for `distforest`, because they are already set.

ntree	number of trees to grow for the forest.
fit.par	logical. if TRUE, fitted and predicted values and predicted parameters are calculated for the learning data (together with loglikelihood)
perturb	a list with arguments replace and fraction determining which type of resampling with replace = TRUE referring to the n-out-of-n bootstrap and replace = FALSE to sample splitting. fraction is the number of observations to draw without replacement.
mtry	number of input variables randomly sampled as candidates at each node for random forest like algorithms. Bagging, as special case of a random forest without random input variable sampling, can be performed by setting mtry either equal to Inf or manually equal to the number of input variables.
applyfun	an optional lapply-style function with arguments function(X, FUN, ...). It is used for computing the variable selection criterion. The default is to use the basic lapply function unless the cores argument is specified (see below).
cores	numeric. If set to an integer the applyfun is set to mclapply with the desired number of cores.
trace	a logical indicating if a progress bar shall be printed while the forest grows.
object	an object as returned by distforest
newdata	an optional data frame containing test data.
type	a character string denoting the type of predicted value returned. For "parameter" the predicted distributional parameters are returned and for "response" the expectation is returned. "weights" returns an integer vector of prediction weights. For type = "node", a list of terminal node ids for each of the trees in the forest is returned.
OOB	a logical defining out-of-bag predictions (only if newdata = NULL).
scale	a logical indicating scaling of the nearest neighbor weights by the sum of weights in the corresponding terminal node of each tree. In the simple regression forest, predicting the conditional mean by nearest neighbor weights will be equivalent to (but slower!) the aggregation of means.
...	arguments to be used to form the default control argument if it is not supplied directly.

Details

Distributional regression forests are an application of model-based recursive partitioning (implemented in mob, ctree and cforest) to parametric model fits based on the GAMLSS family of distributions.

Distributional regression trees, see disttree, are fitted to each of the ntree perturbed samples of the learning sample. Most of the hyper parameters in disttree_control regulate the construction of the distributional regression trees.

Hyper parameters you might want to change are:

1. The number of randomly preselected variables mtry, which is fixed to the square root of the number of input variables.

2. The number of trees ntree. Use more trees if you have more variables.

3. The depth of the trees, regulated by `mincriterion`. Usually unstopped and unpruned trees are used in random forests. To grow large trees, set `mincriterion` to a small value.

The aggregation scheme works by averaging observation weights extracted from each of the `ntree` trees and NOT by averaging predictions directly as in `randomForest`. See Schlosser et al. (2019), Hothorn et al. (2004), and Meinshausen (2006) for a description.

Predictions can be computed using `predict`. For observations with zero weights, predictions are computed from the fitted tree when `newdata = NULL`.

Value

An object of class `distforest`.

References

Breiman L (2001). Random Forests. *Machine Learning*, **45**(1), 5–32.

Hothorn T, Lausen B, Benner A, Radespiel-Troeger M (2004). Bagging Survival Trees. *Statistics in Medicine*, **23**(1), 77–91.

Hothorn T, B\"uhlmann P, Dudoit S, Molinaro A, Van der Laan MJ (2006a). Survival Ensembles. *Biostatistics*, **7**(3), 355–373.

Hothorn T, Hornik K, Zeileis A (2006b). Unbiased Recursive Partitioning: A Conditional Inference Framework. *Journal of Computational and Graphical Statistics*, **15**(3), 651–674.

Hothorn T, Zeileis A (2015). partykit: A Modular Toolkit for Recursive Partytioning in R. *Journal of Machine Learning Research*, **16**, 3905–3909.

Meinshausen N (2006). Quantile Regression Forests. *Journal of Machine Learning Research*, **7**, 983–999.

Schlosser L, Hothorn T, Stauffer R, Zeileis A (2019). Distributional Regression Forests for Probabilistic Precipitation Forecasting in Complex Terrain. *arXiv 1804.02921*, arXiv.org E-Print Archive. http://arxiv.org/abs/1804.02921v3

Strobl C, Boulesteix AL, Zeileis A, Hothorn T (2007). Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution. *BMC Bioinformatics*, **8**, 25. http://www.biomedcentral.com/1471-2105/8/25

Strobl C, Malley J, Tutz G (2009). An Introduction to Recursive Partitioning: Rationale, Application, and Characteristics of Classification and Regression Trees, Bagging, and Random Forests. *Psychological Methods*, **14**(4), 323–348.

Examples

```
## basic example: distributional regression forest for cars data
df <- distforest(dist ~ speed, data = cars)

## prediction of fitted mean and visualization
nd <- data.frame(speed = 4:25)
nd$mean  <- predict(df, newdata = nd, type = "response")[["(fitted.response)"]]
plot(dist ~ speed, data = cars)
lines(mean ~ speed, data = nd)

## Not run:
```

```
## Rain Example
data("RainIbk", package = "crch")
RainIbk$sqrtensmean <-
  apply(sqrt(RainIbk[,grep('^rainfc',names(RainIbk))]), 1, mean)
RainIbk$sqrtenssd <-
  apply(sqrt(RainIbk[,grep('^rainfc',names(RainIbk))]), 1, sd)
RainIbk$rain <- sqrt(RainIbk$rain)
f.rain <- as.formula(paste("rain ~ ", paste(names(RainIbk)[-grep("rain$", names(RainIbk))],
  collapse= "+")))

dt.rain <- disttree(f.rain, data = RainIbk, family = NO())
df.rain <- distforest(f.rain, data = RainIbk, family = NO(), ntree = 10)
df_vi.rain <- varimp(df.rain)

## Bodyfat Example
data("bodyfat", package = "TH.data")
bodyfat$DEXfat <- sqrt(bodyfat$DEXfat)

f.fat <- as.formula(paste("DEXfat ~ ", paste(names(bodyfat)[-grep("DEXfat", names(bodyfat))],
  collapse= "+")))
df.fat <- distforest(f.fat, data = bodyfat, family = NO(), ntree = 10)
df.fat_vi <- varimp(df.fat)

## End(Not run)
```

`disttree` *Distributional Regression Tree*

Description

Trees based on maximum-likelihood estimation of parameters for specified distribution families, for example from the GAMLSS family (for generalized additive models for location, scale, and shape).

Usage

```
disttree(formula, data, subset, na.action = na.pass, weights, offset,
           cluster, family = NO(), control = disttree_control(...),
           converged = NULL, scores = NULL, doFit = TRUE, ...)
```

Arguments

`formula`	a symbolic description of the model to be fit. This should be of type `y ~ x1 + x2` where `y` should be the response variable and `x1` and `x2` are used as partitioning variables.
`data`	an optional data frame containing the variables in the model.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain missing value.

weights	optional numeric vector of case weights.
offset	an optional vector of offset values.
cluster	an optional factor indicating independent clusters. Highly experimental, use at your own risk.
family	specification of the response distribution. Either a gamlss.family object, a list generating function or a family list.
control	control arguments passed to extree_fit via disttree_control.
converged	an optional function for checking user-defined criteria before splits are implemented.
scores	an optional named list of scores to be attached to ordered factors.
doFit	a logical indicating if the tree shall be grown (TRUE) or not (FALSE).
...	arguments to be used to form the default control argument if it is not supplied directly.

Details

Distributional regression trees are an application of model-based recursive partitioning and unbiased recursive partitioning (implemented in extree_fit) to parametric model fits based on the GAMLSS family of distributions.

Value

An object of S3 class disttree inheriting from class modelparty.

See Also

mob, ctree, extree_fit, distfit

Examples

```
tr <- disttree(dist ~ speed, data = cars)
print(tr)

plot(tr)
plot(as.constparty(tr))
```

---

disttree.family *Family List Generating Functions*

Description

The functions dist_gaussian, dist_crch, dist_exponential, dist_weibull, dist_gamma and dist_poisson generate a distribution family object of class disttree.family with all the required elements to fit a distribution in distfit.

Complete distribution family lists are provided for example by dist_list_normal and dist_list_cens_normal.

Usage

```
dist_gaussian()
dist_crch(dist = c("gaussian", "logistic"), truncated = FALSE,
          type = c("left", "right", "interval"), censpoint = 0)
dist_exponential()
dist_weibull()
dist_gamma()
dist_poisson()
```

Arguments

dist	character. Either a gaussian ('gaussian') or a logistic ('logistic') distribution can be selected.
truncated	logical. If TRUE truncated family list is generated with 'censpoint' interpreted as truncation points, If FALSE censored family list is generated. Default is FALSE
type	character. Type of censoring can be selectes ('left', 'right' or 'interval')
censpoint	numeric. Censoring point can be set (per default set to 0).

Details

The functions dist_gaussian, dist_crch, dist_exponential, dist_weibull, dist_gamma and dist_poisson generate a distribution family list with all the required elements to fit a distribution in distfit. These lists include a density function, a score function, a hessian function, starting values, link functions and inverse link functions.

Complete distribution family lists are provided for example by dist_list_normal and dist_list_cens_normal for the normal and censored normal distribution respectively.

Value

These functions return a family of class disttree.family with functions of the corresponding distribution family as required by distfit, disttree, and distforest.

See Also

distfamily

Examples

```
## get the family list for a Gaussian distribution family
dist_gaussian()
```

disttree_control *Auxiliary Function for Controlling* disttree *Fitting*

Description

Auxiliary function for disttree fitting. Specifies a list of control values for fitting a distributional regression tree or forest. These disttree specific control values are set in addition to the control values of ctree_control and can vary from its default values.

Usage

```
disttree_control(type.tree = NULL, type.hessian = c("checklist",
                 "analytic", "numeric"), decorrelate = c("none", "opg",
                 "vcov"), method = "L-BFGS-B", optim.control = list(),
                 lower = -Inf, upper = Inf, minsplit = NULL, minbucket =
                 NULL, splittry = 1L, splitflavour = c("ctree",
                 "exhaustive"), testflavour = c("ctree", "mfluc",
                 "guide"), terminal = "object", model = TRUE, inner = "object",
                restart = TRUE, breakties = FALSE, parm = NULL, dfsplit = TRUE,
           vcov = c("opg", "info", "sandwich"), ordinal = c("chisq", "max", "L2"),
                 ytype = c("vector", "data.frame", "matrix"), trim = 0.1,
              guide_interaction = FALSE, interaction = FALSE, guide_parm = NULL,
            guide_testtype = c("max", "sum", "coin"), guide_decorrelate = "vcov",
                 xgroups = NULL, ygroups = NULL, weighted.scores = FALSE, ...)
```

Arguments

type.tree	NULL or character specifying which type of tree should be fitted: Either based on model-based recursive partitioning type.tree="mob" or unbiased recursive partitioning type.tree="ctree".
type.hessian	Can either be "checklist", "analytic" or "numeric" to decide how the hessian matrix should be calculated in the fitting process in distfit. For "checklist" it is checked whether a function "hdist" is given in the family list. If so, "type.hessian" is set to "analytic", otherwise to "numeric".
decorrelate	specification of the type of decorrelation for the empirical estimating functions (or scores) either "none" or "opg" (for the outer product of gradients) or "vcov" (for the variance-covariance matrix, assuming this is an estimate of the Fisher information).
method	optimization method passed to optim.
optim.control	a list with further arguments to be passed to 'fn and 'gr' in optim.
lower, upper	bounds on the variables for the "L-BFGS-B" method, or bounds in which to search for method "Brent" passed to optim.
minsplit, minbucket	integer. The minimum number of observations in a node. If NULL, the default is to use 10 times the number of parameters to be estimated (divided by the number of responses per observation if that is greater than 1).

`splittry` number of variables that are inspected for admissible splits if the best split doesn't meet the sample size constraints. FIXME: (ML) set to 1L, mob default.

`splitflavour` use exhaustive search (`mob`) over splits instead of maximally selected statistics (`ctree`). This feature may change.

`testflavour` employ permutation tests (`ctree`) or M-fluctuation tests (`mfluc`).

`terminal` character. Specification of which additional information ("estfun", "object", or both) should be stored in each terminal node. If NULL, no additional information is stored. Note that the information slot 'object' contains a slot 'estfun' as well. FIXME: (LS) Should estfun always be returned within object?

`model` logical. Should the full model frame be stored in the resulting object?

`inner` character. Specification of which additional information ("estfun", "object", or both) should be stored in each inner node. If NULL, no additional information is stored. Note that the information slot 'object' contains a slot 'estfun' as well. FIXME: (LS) Should estfun always be returned within object?

`restart` logical. When determining the optimal split point in a numerical variable: Should model estimation be restarted with NULL starting values for each split? The default is TRUE. If FALSE, then the parameter estimates from the previous split point are used as starting values for the next split point (because in practice the difference are often not huge). (Note that in that case a for loop is used instead of the applyfun for fitting models across sample splits.)

`breakties` logical. If M-fluctuation tests are applied, should ties in numeric variables be broken randomly for computing the associated parameter instability test?

`parm` numeric or character. Number or name of model parameters included in the parameter instability tests if M-fluctuation tests are applied (by default all parameters are included). FIXME: (LS) is it really applied?

`dfsplit` logical or numeric. as.integer(dfsplit) is the degrees of freedom per selected split employed when computing information criteria etc. FIXME: (LS) is it really applied?

`vcov` character indicating which type of covariance matrix estimator should be employed in the parameter instability tests if M-fluctuation tests are applied. The default is the outer product of gradients ("opg"). Alternatively, vcov = "info" employs the information matrix and vcov = "sandwich" the sandwich matrix (both of which are only sensible for maximum likelihood estimation).

`ordinal` character indicating which type of parameter instability test should be employed for ordinal partitioning variables (i.e., ordered factors) if M-fluctuation tests are applied. This can be "chisq", "max", or "L2". If "chisq" then the variable is treated as unordered and a chi-squared test is performed. If "L2", then a maxLM-type test as for numeric variables is carried out but correcting for ties. This requires simulation of p-values via catL2BB and requires some computation time. For "max" a weighted double maximum test is used that computes p-values via pmvnorm.

`ytype` character. For type.tree "mob": Specification of how mob should preprocess y variable. Possible choice are: "vector", i.e., only one variable; "matrix", i.e., the model matrix of all variables; "data.frame", i.e., a data frame of all variables., FIXME: (LS) handle multidim. response?

`trim`	numeric. This specifies the trimming in the parameter instability test for the numerical variables if M-fluctuation tests are applied. If smaller than 1, it is interpreted as the fraction relative to the current node size.
`guide_interaction`	logical. Should interaction tests be evaluated as well?
`interaction`	Add description
`guide_parm`	a vector of indices of the parameters (incl. intercept) for which estfun should be considered in chi-squared tests.
`guide_testtype`	character specifying whether a maximal selection ("max"), the summed up test statistic ("sum"), or COIN ("coin") should be employed.
`guide_decorrelate`	Add description
`xgroups`	integer. Number of categories for split variables to be employed in chi-squared tests (optionally breaks can be handed over).
`ygroups`	integer. Number of categories for scores to be employed in chi-squared tests (optionally breaks can be handed over).
`weighted.scores`	logical. Should scores be weighted in GUIDE
`...`	additional `ctree_control` arguments.

Value

A list with components named as the arguments.

See Also

`ctree_control`, `disttree`, `extree_fit`

`RainAxams` *Observations and covariates for station Axams*

Description

Observations of precipitation sums and weather forecasts of a set of meteorological quantities from an ensemble prediction system for one specific site. This site is Axams located in the Eastern European Alps (11.28E 47.23N, 890 meters a.m.s.l.).

Usage

```
data("RainAxams")
```

Format

A `data.frame` consisting of the station's name, observation day and year, power transformed observations of daily precipitation sums and the corresponding meteorological ensemble predictions for station Axams. The base variables of the numerical ensemble predictions are listed below. For each of them variations such as ensemble mean/standard deviation/minimum/maximum are included in the dataset. All "power transformed" values use the same power parameter `p=1/1.6`.

station `character`. Name of the observation station.

robs `numeric`. Observed total precipitation (power transformed).

year `integer`. Year in which the observation was taken.

day `integer`. Day for which the observation was taken.

tppow_mean, tppow_sprd, tppow_min, tppow_max, tppow_mean0612, tppow_mean1218, tppow_mean1824, tppow_ `numeric`. Predicted total precipitation (power transformed).

capepow_mean, capepow_sprd, capepow_min, capepow_max, capepow_mean0612, capepow_mean1218, capepow_m `numeric`. Predicted convective available potential energy (power transformed).

dswrf_mean_mean, dswrf_mean_min, dswrf_mean_max, dswrf_sprd_mean, dswrf_sprd_min, dswrf_sprd_max `numeric`. Predicted downwards shortwave radiation flux ("sunshine").

msl_diff, msl_mean_mean, msl_mean_min, msl_mean_max, msl_sprd_mean, msl_sprd_min, msl_sprd_max `numeric`. Predicted mean sea level pressure.

pwat_mean_mean, pwat_mean_min, pwat_mean_max, pwat_sprd_mean, pwat_sprd_min, pwat_sprd_max `numeric`. Predicted precipitable water.

tcolc_mean_mean, tcolc_mean_min, tcolc_mean_max, tcolc_sprd_mean, tcolc_sprd_min, tcolc_sprd_max `numeric`. Predicted total column-integrated condensate.

tmax_mean_mean, tmax_mean_min, tmax_mean_max, tmax_sprd_mean, tmax_sprd_min, tmax_sprd_max `numeric`. Predicted 2m maximum temperature.

t500_mean_mean, t500_mean_min, t500_mean_max, t500_sprd_mean, t500_sprd_min, t500_sprd_max `numeric`. Predicted temperature on 500 hPa.

t700_mean_mean, t700_mean_min, t700_mean_max, t700_sprd_mean, t700_sprd_min, t700_sprd_max `numeric`. Predicted temperature on 700 hPa.

t850_mean_mean, t850_mean_min, t850_mean_max, t850_sprd_mean, t850_sprd_min, t850_sprd_max `numeric`. Predicted temperature on 850 hPa.

tdiff500850_mean, tdiff500850_min, tdiff500850_max `numeric`. Predicted temperature difference 500 hPa to 850 hPa.

tdiff700850_mean, tdiff700850_min, tdiff700850_max `numeric`. Predicted temperature difference 700 hPa to 850 hPa.

tdiff500700_mean, tdiff500700_min, tdiff500700_max `numeric`. Predicted temperature difference 500 hPa to 700 hPa.

Details

The site is maintained by the hydrographical service Tyrol and provides daily precipitation sums reported at 06~UTC. Before published, the observations have been quality-controlled by the maintainer.

The forecast data is based on the second-generation global ensemble reforecast dataset and consists of range of different meteorological quantities for day one (forecast horizon +6 to +30 hours ahead). The forecasts have been bi-linearly interpolated to the station location.

References

Hamill T M, Bates G T, Whitaker J S, Murray D R, Fiorino M, Galarneau Jr. T J, Zhu Y, Lapenta W (2013). NOAA's Second-Generation Global Medium-Range Ensemble Reforecast Dataset. *Bulletin of the American Meteorological Society*, **94**(10), 1553–1565. doi:10.1175/BAMSD1200014.1

BMLFUW (2016). Bundesministerium f\"ur Land und Forstwirtschaft, Umwelt und Wasserwirtschaft (BMLFUW), Abteilung IV/4 – Wasserhaushalt. Available at http://ehyd.gv.at. Accessed: 2016–02–29.

Examples

```
data("RainAxams")
head(RainAxams)
colnames(RainAxams)
```

Index