Package 'CAST'

October 12, 2022

Type Package

Title 'caret' Applications for Spatial-Temporal Models

Version 0.7.0

Author Hanna Meyer [cre, aut],
Carles Milà [aut],
Marvin Ludwig [aut],
Chris Reudenbach [ctb],
Thomas Nauss [ctb],
Edzer Pebesma [ctb]

Maintainer Hanna Meyer <hanna.meyer@uni-muenster.de>

Description Supporting functionality to run 'caret' with spatial or spatial-temporal data. 'caret' is a frequently used package for model training and prediction using machine learning. CAST includes functions to improve spatial or spatial-temporal modelling tasks using 'caret'. It includes the newly suggested 'Nearest neighbor distance matching' cross-validation to estimate the performance of spatial prediction models and allows for spatial variable selection to selects suitable predictor variables in view to their contribution to the spatial model performance. CAST further includes functionality to estimate the (spatial) area of applicability of prediction models. Methods are described in Meyer et al. (2018) <doi:10.1016/j.envsoft.2017.12.001>; Meyer et al. (2019) <doi:10.1016/j.ecolmodel.2019.108
210X.13650>; Milà et al. (2022) <doi:10.1111/2041-
210X.13851>; Meyer and Pebesma (2022) <doi:10.1038/s41467-022-29838-9>.

License GPL (>= 2)

URL https://github.com/HannaMeyer/CAST,
https://hannameyer.github.io/CAST/

Encoding UTF-8

Depends R (>= 4.1.0)

Imports caret, stats, utils, ggplot2, graphics, reshape, FNN, plyr, zoo, methods, grDevices, data.table, lattice

Suggests doParallel, randomForest, lubridate, raster, sp, knitr, mapview, rmarkdown, sf, scales, parallel, latticeExtra, virtualspecies, gridExtra, viridis, rgeos, stars, scam, terra, rnaturalearth, rgdal

RoxygenNote 7.2.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-24 09:32:44 UTC

R topics documented:

---

aoa *Area of Applicability*

Description

This function estimates the Dissimilarity Index (DI) and the derived Area of Applicability (AOA) of spatial prediction models by considering the distance of new data (i.e. a Raster Stack of spatial predictors used in the models) in the predictor variable space to the data used for model training. Predictors can be weighted based on the internal variable importance of the machine learning algorithm used for model training. The AOA is derived by applying a threshold on the DI which is the (outlier-removed) maximum DI of the cross-validated training data.

Usage

```
aoa(
  newdata,
  model = NA,
  trainDI = NA,
  cl = NULL,
  train = NULL,
  weight = NA,
  variables = "all",
```

```
  CVtest = NULL,
  CVtrain = NULL
)
```

Arguments

newdata	A RasterStack, RasterBrick, stars object, SpatRaster or data.frame containing the data the model was meant to make predictions for.
model	A train object created with caret used to extract weights from (based on variable importance) as well as cross-validation folds. See examples for the case that no model is available or for models trained via e.g. mlr3.
trainDI	A trainDI object. Optional if trainDI was calculated beforehand.
cl	A cluster object e.g. created with doParallel. Optional. Should only be used if newdata is large.
train	A data.frame containing the data used for model training. Optional. Only required when no model is given
weight	A data.frame containing weights for each variable. Optional. Only required if no model is given.
variables	character vector of predictor variables. if "all" then all variables of the model are used or if no model is given then of the train dataset.
CVtest	list or vector. Either a list where each element contains the data points used for testing during the cross validation iteration (i.e. held back data). Or a vector that contains the ID of the fold for each training point. Only required if no model is given.
CVtrain	list. Each element contains the data points used for training during the cross validation iteration (i.e. held back data). Only required if no model is given and only required if CVtrain is not the opposite of CVtest (i.e. if a data point is not used for testing, it is used for training). Relevant if some data points are excluded, e.g. when using nndm.

Details

The Dissimilarity Index (DI) and the corresponding Area of Applicability (AOA) are calculated. If variables are factors, dummy variables are created prior to weighting and distance calculation.

Interpretation of results: If a location is very similar to the properties of the training data it will have a low distance in the predictor variable space (DI towards 0) while locations that are very different in their properties will have a high DI. See Meyer and Pebesma (2021) for the full documentation of the methodology.

Value

An object of class aoa containing:

parameters	object of class trainDI. see trainDI
DI	raster or data frame. Dissimilarity index of newdata
AOA	raster or data frame. Area of Applicability of newdata. AOA has values 0 (outside AOA) and 1 (inside AOA)

Note

If classification models are used, currently the variable importance can only be automatically retrieved if models were trained via train(predictors,response) and not via the formula-interface. Will be fixed.

Author(s)

Hanna Meyer

References

Meyer, H., Pebesma, E. (2021): Predicting into unknown space? Estimating the area of applicability of spatial prediction models. Methods in Ecology and Evolution 12: 1620-1633. doi: 10.1111/2041210X.13650

See Also

calibrate_aoa, trainDI

Examples

```
## Not run:
library(sf)
library(raster)
library(caret)
library(viridis)
library(latticeExtra)

# prepare sample data:
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
dat <- aggregate(dat[,c("VW","Easting","Northing")],by=list(as.character(dat$SOURCEID)),mean)
pts <- st_as_sf(dat,coords=c("Easting","Northing"))
pts$ID <- 1:nrow(pts)
set.seed(100)
pts <- pts[1:30,]
studyArea <- stack(system.file("extdata","predictors_2012-03-25.grd",package="CAST"))[[1:8]]
trainDat <- extract(studyArea,pts,df=TRUE)
trainDat <- merge(trainDat,pts,by.x="ID",by.y="ID")

# visualize data spatially:
spplot(scale(studyArea))
plot(studyArea$DEM)
plot(pts[,1],add=TRUE,col="black")

# train a model:
set.seed(100)
variables <- c("DEM","NDRE.Sd","TWI")
model <- train(trainDat[,which(names(trainDat)%in%variables)],
trainDat$VW, method="rf", importance=TRUE, tuneLength=1,
trControl=trainControl(method="cv",number=5,savePredictions=T))
print(model) #note that this is a quite poor prediction model
```

```
prediction <- predict(studyArea,model)
plot(varImp(model,scale=FALSE))

#...then calculate the AOA of the trained model for the study area:
AOA <- aoa(studyArea,model)
plot(AOA)
spplot(AOA$DI, col.regions=viridis(100),main="Dissimilarity Index")
#plot predictions for the AOA only:
spplot(prediction, col.regions=viridis(100),main="prediction for the AOA")+
spplot(AOA$AOA,col.regions=c("grey","transparent"))

####
# Calculating the AOA might be time consuming. Consider running it in parallel:
####
library(doParallel)
library(parallel)
cl <- makeCluster(4)
registerDoParallel(cl)
AOA <- aoa(studyArea,model,cl=cl)

####
#The AOA can also be calculated without a trained model.
#All variables are weighted equally in this case:
####
AOA <- aoa(studyArea,train=trainDat,variables=variables)
spplot(AOA$DI, col.regions=viridis(100),main="Dissimilarity Index")
spplot(AOA$AOA,main="Area of Applicability")


####
# The AOA can also be used for models trained via mlr3 (parameters have to be assigned manually):
####

library(mlr3)
library(mlr3learners)
library(mlr3spatial)
library(mlr3spatiotempcv)
library(mlr3extralearners)

# initiate and train model:
train_df <- trainDat[, c("DEM","NDRE.Sd","TWI", "VW")]
backend <- as_data_backend(train_df)
task <- as_task_regr(backend, target = "VW")
lrn <- lrn("regr.randomForest", importance = "mse")
lrn$train(task)

# cross-validation folds
rsmp_cv <- rsmp("cv", folds = 5L)$instantiate(task)

## predict:
prediction <- predict(studyArea,lrn$model)

### Estimate AOA
```

```
AOA <- aoa(studyArea,
           train = as.data.frame(task$data()),
           variables = task$feature_names,
           weight = data.frame(t(lrn$importance())),
           CVtest = rsmp_cv$instance[order(row_id)]$fold)


## End(Not run)
```

---

bss *Best subset feature selection*

Description

Evaluate all combinations of predictors during model training

Usage

```
bss(
  predictors,
  response,
  method = "rf",
  metric = ifelse(is.factor(response), "Accuracy", "RMSE"),
  maximize = ifelse(metric == "RMSE", FALSE, TRUE),
  globalval = FALSE,
  trControl = caret::trainControl(),
  tuneLength = 3,
  tuneGrid = NULL,
  seed = 100,
  verbose = TRUE,
  ...
)
```

Arguments

predictors	see train
response	see train
method	see train
metric	see train
maximize	see train
globalval	Logical. Should models be evaluated based on 'global' performance? See global_validation
trControl	see train
tuneLength	see train
tuneGrid	see train

seed	A random number
verbose	Logical. Should information about the progress be printed?
...	arguments passed to the classification or regression routine (such as randomForest).

Details

bss is an alternative to ffs and ideal if the training set is small. Models are iteratively fitted using all different combinations of predictor variables. Hence, 2^X models are calculated. Don't try running bss on very large datasets because the computation time is much higher compared to ffs.

The internal cross validation can be run in parallel. See information on parallel processing of carets train functions for details.

Value

A list of class train. Beside of the usual train content the object contains the vector "selectedvars" and "selectedvars_perf" that give the best variables selected as well as their corresponding performance. It also contains "perf_all" that gives the performance of all model runs.

Note

This variable selection is particularly suitable for spatial cross validations where variable selection MUST be based on the performance of the model for predicting new spatial units. Note that bss is very slow since all combinations of variables are tested. A more time efficient alternative is the forward feature selection (ffs) (ffs).

Author(s)

Hanna Meyer

See Also

train,ffs, trainControl,CreateSpacetimeFolds, nndm

Examples

```
## Not run:
data(iris)
bssmodel <- bss(iris[,1:4],iris$Species)
bssmodel$perf_all

## End(Not run)
```

`calibrate_aoa` *Calibrate the AOA based on the relationship between the DI and the prediction error*

Description

Performance metrics are calculated for moving windows of DI values of cross-validated training data

Usage

```
calibrate_aoa(
  AOA,
  model,
  window.size = 5,
  calib = "scam",
  multiCV = FALSE,
  length.out = 10,
  maskAOA = TRUE,
  showPlot = TRUE,
  k = 6,
  m = 2
)
```

Arguments

`AOA`	the result of `aoa`
`model`	the model used to get the AOA
`window.size`	Numeric. Size of the moving window. See `rollapply`.
`calib`	Character. Function to model the DI~performance relationship. Currently lm and scam are supported
`multiCV`	Logical. Re-run model fitting and validation with different CV strategies. See details.
`length.out`	Numeric. Only used if multiCV=TRUE. Number of cross-validation folds. See details.
`maskAOA`	Logical. Should areas outside the AOA set to NA?
`showPlot`	Logical.
`k`	Numeric. See mgcv::s
`m`	Numeric. See mgcv::s

Details

If multiCV=TRUE the model is re-fitted and validated by length.out new cross-validations where the cross-validation folds are defined by clusters in the predictor space, ranging from three clusters to LOOCV. Hence, a large range of DI values is created during cross-validation. If the AOA threshold based on the calibration data from multiple CV is larger than the original AOA threshold (which is likely if extrapolation situations are created during CV), the AOA is updated accordingly. See Meyer and Pebesma (2021) for the full documentation of the methodology.

Value

A list of length 2 with the elements "AOA": rasterStack which contains the original DI and the AOA (which might be updated if new test data indicate this option), as well as the expected performance based on the relationship. Data used for calibration are stored in the attributes. The second element is a plot showing the relationship.

Author(s)

Hanna Meyer

References

Meyer, H., Pebesma, E. (2021): Predicting into unknown space? Estimating the area of applicability of spatial prediction models. doi: 10.1111/2041210X.13650

See Also

aoa

Examples

```
## Not run:
library(sf)
library(raster)
library(caret)
library(viridis)
library(latticeExtra)

# prepare sample data:
library(sf)
library(raster)
library(caret)
# prepare sample data:
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
dat <- aggregate(dat[,c("VW","Easting","Northing")],by=list(as.character(dat$SOURCEID)),mean)
pts <- st_as_sf(dat,coords=c("Easting","Northing"))
pts$ID <- 1:nrow(pts)
studyArea <- stack(system.file("extdata","predictors_2012-03-25.grd",package="CAST"))[[1:8]]
dat <- extract(studyArea,pts,df=TRUE)
trainDat <- merge(dat,pts,by.x="ID",by.y="ID")

# train a model:
```

```
variables <- c("DEM","NDRE.Sd","TWI")
set.seed(100)
model <- train(trainDat[,which(names(trainDat)%in%variables)],
  trainDat$VW,method="rf",importance=TRUE,tuneLength=1,
  trControl=trainControl(method="cv",number=5,savePredictions=TRUE))

#...then calculate the AOA of the trained model for the study area:
AOA <- aoa(studyArea,model)

AOA_new <- calibrate_aoa(AOA,model)
plot(AOA_new$AOA$expected_RMSE)
# attributes(AOA_new$AOA) # data used for calibration

## End(Not run)
```

CAST *'caret' Applications for Spatial-Temporal Models*

Description

Supporting functionality to run 'caret' with spatial or spatial-temporal data. 'caret' is a frequently used package for model training and prediction using machine learning. CAST includes functions to improve spatial-temporal modelling tasks using 'caret'. It includes the newly suggested 'Nearest neighbor distance matching' cross-validation to estimate the performance of spatial prediction models and allows for spatial variable selection to selects suitable predictor variables in view to their contribution to the spatial model performance. CAST further includes functionality to estimate the (spatial) area of applicability of prediction models by analysing the similarity between new data and training data. Methods are described in Meyer et al. (2018); Meyer et al. (2019); Meyer and Pebesma (2021); Milà et al. (2022); Meyer and Pebesma (2022).

Details

'caret' Applications for Spatio-Temporal models

Author(s)

Hanna Meyer, Carles Milà, Marvin Ludwig

References

- Milà, C., Mateu, J., Pebesma, E., Meyer, H. (2022): Nearest Neighbour Distance Matching Leave-One-Out Cross-Validation for map validation. Methods in Ecology and Evolution 00, 1– 13.
- Meyer, H., Pebesma, E. (2022): Machine learning-based global maps of ecological variables and the challenge of assessing them. Nature Communications. 13.
- Meyer, H., Pebesma, E. (2021): Predicting into unknown space? Estimating the area of applicability of spatial prediction models. Methods in Ecology and Evolution. 12, 1620– 1633.

- Meyer, H., Reudenbach, C., Wöllauer, S., Nauss, T. (2019): Importance of spatial predictor variable selection in machine learning applications - Moving from data reproduction to spatial prediction. Ecological Modelling. 411, 108815.
- Meyer, H., Reudenbach, C., Hengl, T., Katurji, M., Nauß, T. (2018): Improving performance of spatio-temporal machine learning models using forward feature selection and target-oriented validation. Environmental Modelling & Software 101: 1-9.

`CreateSpacetimeFolds` *Create Space-time Folds*

Description

Create spatial, temporal or spatio-temporal Folds for cross validation based on pre-defined groups

Usage

```
CreateSpacetimeFolds(
  x,
  spacevar = NA,
  timevar = NA,
  k = 10,
  class = NA,
  seed = sample(1:1000, 1)
)
```

Arguments

`x`	data.frame containing spatio-temporal data
`spacevar`	Character indicating which column of x identifies the spatial units (e.g. ID of weather stations)
`timevar`	Character indicating which column of x identifies the temporal units (e.g. the day of the year)
`k`	numeric. Number of folds. If spacevar or timevar is NA and a leave one location out or leave one time step out cv should be performed, set k to the number of unique spatial or temporal units.
`class`	Character indicating which column of x identifies a class unit (e.g. land cover)
`seed`	numeric. See ?seed

Details

The function creates train and test sets by taking (spatial and/or temporal) groups into account. In contrast to `nndm`, it requires that the groups are already defined (e.g. spatial clusters or blocks or temporal units). Using "class" is helpful in the case that data are clustered in space and are categorical. E.g This is the case for land cover classifications when training data come as training polygons. In this case the data should be split in a way that entire polygons are held back (spacevar="polygonID") but at the same time the distribution of classes should be similar in each fold (class="LUC").

Value

A list that contains a list for model training and a list for model validation that can directly be used as "index" and "indexOut" in caret's trainControl function

Note

Standard k-fold cross-validation can lead to considerable misinterpretation in spatial-temporal modelling tasks. This function can be used to prepare a Leave-Location-Out, Leave-Time-Out or Leave-Location-and-Time-Out cross-validation as target-oriented validation strategies for spatial-temporal prediction tasks. See Meyer et al. (2018) for further information.

Author(s)

Hanna Meyer

References

Meyer, H., Reudenbach, C., Hengl, T., Katurji, M., Nauß, T. (2018): Improving performance of spatio-temporal machine learning models using forward feature selection and target-oriented validation. Environmental Modelling & Software 101: 1-9.

See Also

trainControl,ffs, nndm

Examples

```
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
### Prepare for 10-fold Leave-Location-and-Time-Out cross validation
indices <- CreateSpacetimeFolds(dat,"SOURCEID","Date")
str(indices)
### Prepare for 10-fold Leave-Location-Out cross validation
indices <- CreateSpacetimeFolds(dat,spacevar="SOURCEID")
str(indices)
### Prepare for leave-One-Location-Out cross validation
indices <- CreateSpacetimeFolds(dat,spacevar="SOURCEID",
    k=length(unique(dat$SOURCEID)))
str(indices)
```

---

ffs *Forward feature selection*

Description

A simple forward feature selection algorithm

Usage

```
ffs(
  predictors,
  response,
  method = "rf",
  metric = ifelse(is.factor(response), "Accuracy", "RMSE"),
  maximize = ifelse(metric == "RMSE", FALSE, TRUE),
  globalval = FALSE,
  withinSE = FALSE,
  minVar = 2,
  trControl = caret::trainControl(),
  tuneLength = 3,
  tuneGrid = NULL,
  seed = sample(1:1000, 1),
  verbose = TRUE,
  ...
)
```

Arguments

predictors	see train
response	see train
method	see train
metric	see train
maximize	see train
globalval	Logical. Should models be evaluated based on 'global' performance? See global_validation
withinSE	Logical Models are only selected if they are better than the currently best models Standard error
minVar	Numeric. Number of variables to combine for the first selection. See Details.
trControl	see train
tuneLength	see train
tuneGrid	see train
seed	A random number used for model training
verbose	Logical. Should information about the progress be printed?
...	arguments passed to the classification or regression routine (such as randomForest).

Details

Models with two predictors are first trained using all possible pairs of predictor variables. The best model of these initial models is kept. On the basis of this best model the predictor variables are iteratively increased and each of the remaining variables is tested for its improvement of the currently best model. The process stops if none of the remaining variables increases the model performance when added to the current best model.

The internal cross validation can be run in parallel. See information on parallel processing of carets train functions for details.

Using withinSE will favour models with less variables and probably shorten the calculation time

Per Default, the ffs starts with all possible 2-pair combinations. minVar allows to start the selection with more than 2 variables, e.g. minVar=3 starts the ffs testing all combinations of 3 (instead of 2) variables first and then increasing the number. This is important for e.g. neural networks that often cannot make sense of only two variables. It is also relevant if it is assumed that the optimal variables can only be found if more than 2 are considered at the same time.

Value

A list of class train. Beside of the usual train content the object contains the vector "selectedvars" and "selectedvars_perf" that give the order of the best variables selected as well as their corresponding performance (starting from the first two variables). It also contains "perf_all" that gives the performance of all model runs.

Note

This variable selection is particularly suitable for spatial cross validations where variable selection MUST be based on the performance of the model for predicting new spatial units. See Meyer et al. (2018) and Meyer et al. (2019) for further details.

Author(s)

Hanna Meyer

References

- Gasch, C.K., Hengl, T., Gräler, B., Meyer, H., Magney, T., Brown, D.J. (2015): Spatio-temporal interpolation of soil water, temperature, and electrical conductivity in 3D+T: the Cook Agronomy Farm data set. Spatial Statistics 14: 70-90.
- Meyer, H., Reudenbach, C., Hengl, T., Katurji, M., Nauß, T. (2018): Improving performance of spatio-temporal machine learning models using forward feature selection and target-oriented validation. Environmental Modelling & Software 101: 1-9. doi: 10.1016/j.envsoft.2017.12.001
- Meyer, H., Reudenbach, C., Wöllauer, S., Nauss, T. (2019): Importance of spatial predictor variable selection in machine learning applications - Moving from data reproduction to spatial prediction. Ecological Modelling. 411, 108815. doi: 10.1016/j.ecolmodel.2019.108815

See Also

train,bss, trainControl,CreateSpacetimeFolds,nndm

Examples

```
## Not run:
data(iris)
ffsmodel <- ffs(iris[,1:4],iris$Species)
ffsmodel$selectedvars
ffsmodel$selectedvars_perf
```

```
## End(Not run)

# or perform model with target-oriented validation (LLO CV)
#the example is described in Gasch et al. (2015). The ffs approach for this dataset is described in
#Meyer et al. (2018). Due to high computation time needed, only a small and thus not robust example
#is shown here.

## Not run:
#run the model on three cores:
library(doParallel)
cl <- makeCluster(3)
registerDoParallel(cl)

#load and prepare dataset:
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
trainDat <- dat[dat$altitude==-0.3&year(dat$Date)==2012&week(dat$Date)%in%c(13:14),]

#visualize dataset:
ggplot(data = trainDat, aes(x=Date, y=VW)) + geom_line(aes(colour=SOURCEID))

#create folds for Leave Location Out Cross Validation:
set.seed(10)
indices <- CreateSpacetimeFolds(trainDat,spacevar = "SOURCEID",k=3)
ctrl <- trainControl(method="cv",index = indices$index)

#define potential predictors:
predictors <- c("DEM","TWI","BLD","Precip_cum","cday","MaxT_wrcc",
"Precip_wrcc","NDRE.M","Bt","MinT_wrcc","Northing","Easting")

#run ffs model with Leave Location out CV
set.seed(10)
ffsmodel <- ffs(trainDat[,predictors],trainDat$VW,method="rf",
tuneLength=1,trControl=ctrl)
ffsmodel

#compare to model without ffs:
model <- train(trainDat[,predictors],trainDat$VW,method="rf",
tuneLength=1, trControl=ctrl)
model
stopCluster(cl)

## End(Not run)
```

`global_validation` *Evaluate 'global' cross-validation*

Description

Calculate validation metric using all held back predictions at once

Usage

```
global_validation(model)
```

Arguments

model	an object of class train

Details

Relevant when folds are not representative for the entire area of interest. In this case, metrics like R2 are not meaningful since it doesn't reflect the general ability of the model to explain the entire gradient of the response. Comparable to LOOCV, predictions from all held back folds are used here together to calculate validation statistics.

Value

regression (postResample) or classification (confusionMatrix) statistics

Author(s)

Hanna Meyer

See Also

CreateSpacetimeFolds

Examples

```
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
dat <- dat[sample(1:nrow(dat),500),]
indices <- CreateSpacetimeFolds(dat,"SOURCEID","Date")
ctrl <- caret::trainControl(method="cv",index = indices$index,savePredictions="final")
model <- caret::train(dat[,c("DEM","TWI","BLD")],dat$VW, method="rf", trControl=ctrl, ntree=10)
global_validation(model)
```

---

nndm *Nearest Neighbour Distance Matching (NNDM) algorithm*

Description

This function implements the *NNDM* algorithm and returns the necessary indices to perform a NNDM LOO CV for map validation.

Usage

```
nndm(
  tpoints,
  modeldomain = NULL,
  ppoints = NULL,
  samplesize = 1000,
  sampling = "regular",
  phi = "max",
  min_train = 0
)
```

Arguments

tpoints	sf or sfc point object. Contains the training points samples.
modeldomain	raster or sf object defining the prediction area (see Details).
ppoints	sf or sfc point object. Contains the target prediction points. Optional. Alternative to modeldomain (see Details).
samplesize	numeric. How many sampled of the modeldomain should be sampled? Only required if modeldomain is used instead of ppoints
sampling	character. How to draw samples from the modeldomain? See spsample. Use sampling = "Fibonacci" for global applications." Only required if modeldomain is used instead of ppoints
phi	Numeric. Estimate of the landscape autocorrelation range in the same units as the tpoints and ppoints for projected CRS, in meters for geographic CRS. Per default (phi="max"), the size of the prediction area is used. See Details
min_train	Numeric between 0 and 1. Minimum proportion of training data that must be used in each CV fold. Defaults to 0 (i.e. no restrictions).

Details

Details of the method can be found in Milà et al. (2022). Euclidean distances are used for projected and non-defined CRS, great circle distances are used for geographic CRS (units in meters). Specifying phi allows limiting distance matching to the area where this is assumed to be relevant due to spatial autocorrelation. Distances are only matched up to phi. Beyond that range, all data points are used for training, without exclusions. When phi is set to "max", nearest neighbor distance matching is performed for the entire prediction area.

The modeldomain is a sf polygon or a raster that defines the prediction area. The function takes a regular point sample (amount defined by samplesize) from the spatial extent. As an alternative use ppoints instead of modeldomain, if you alre<dy have a representative sample from your prediction area.

Value

An object of class *nndm* consisting of a list of six elements: indx_train, indx_test, and indx_exclude (indices of the observations to use as training/test/excluded data in each NNDM LOO CV iteration), Gij (distances for multitype G function construction between prediction and target points), Gj (distances for G function construction during LOO CV), Gjstar (distances for modified G function

during NNDM LOO CV), phi (landscape autocorrelation range). indx_train and indx_test can directly be used as "index" and "indexOut" in caret's `trainControl` function or used to initiate a custom validation strategy in mlr3.

Note

NNDM is a variation of LOOCV and therefore may take a long time for large training data sets. You may need to consider alternatives following the ideas of Milà et al. (2022) for large data sets.

Author(s)

Carles Milà

References

- Milà, C., Mateu, J., Pebesma, E., Meyer, H. (2022): Nearest Neighbour Distance Matching Leave-One-Out Cross-Validation for map validation. Methods in Ecology and Evolution 00, 1– 13.
- Meyer, H., Pebesma, E. (2022): Machine learning-based global maps of ecological variables and the challenge of assessing them. Nature Communications. 13.

See Also

`plot_geodist`

Examples

```
library(sf)

# Simulate 100 random training and test points in a 100x100 square
set.seed(123)
poly <- list(matrix(c(0,0,0,100,100,100,100,0,0,0), ncol=2, byrow=TRUE))
sample_poly <- sf::st_polygon(poly)
train_points <- sf::st_sample(sample_poly, 100, type = "random")
pred_points <- sf::st_sample(sample_poly, 100, type = "random")

# Run NNDM.
nndm_pred <- nndm(train_points, ppoints=pred_points)
nndm_pred
plot(nndm_pred)

# ...or run NNDM with a known autocorrelation range.
# Here, the autocorrelation range (phi) is known to be 10.
nndm_pred <- nndm(train_points, ppoints=pred_points, phi = 10)
nndm_pred
plot(nndm_pred)


#########################################################################
# Example 2: Real- world example; using a modeldomain instead of previously
# sampled prediction locations
```

```
#############################################################################
## Not run:
library(raster)

### prepare sample data:
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
dat <- aggregate(dat[,c("DEM","TWI", "NDRE.M", "Easting", "Northing","VW")],
   by=list(as.character(dat$SOURCEID)),mean)
pts <- dat[,-1]
pts <- st_as_sf(pts,coords=c("Easting","Northing"))
st_crs(pts) <- 26911
studyArea <- raster::stack(system.file("extdata","predictors_2012-03-25.grd",package="CAST"))

nndm_folds <- nndm(pts, modeldomain= studyArea)

#use for cross-validation:
library(caret)
ctrl <- trainControl(method="cv",
   index=nndm_folds$indx_train,
   indexOut=nndm_folds$indx_test,
   savePredictions='final')
model_nndm <- train(dat[,c("DEM","TWI", "NDRE.M")],
   dat$VW,
   method="rf",
   trControl = ctrl)
model_nndm
global_validation(model_nndm)

## End(Not run)
```

---

plot *Plot CAST classes*

Description

Generic plot function for trainDI and aoa

Usage

```
## S3 method for class 'trainDI'
plot(x, ...)

## S3 method for class 'aoa'
plot(x, samplesize = 1000, ...)

## S3 method for class 'nndm'
plot(x, ...)
```

Arguments

x	An object of type *nndm*.
...	other arguments.
samplesize	numeric. How many prediction samples should be plotted?

Author(s)

Marvin Ludwig, Hanna Meyer

Carles Milà

---

plot_ffs *Plot results of a Forward feature selection or best subset selection*

Description

A plotting function for a forward feature selection result. Each point is the mean performance of a model run. Error bars represent the standard errors from cross validation. Marked points show the best model from each number of variables until a further variable could not improve the results. If type=="selected", the contribution of the selected variables to the model performance is shown.

Usage

```
plot_ffs(
  ffs_model,
  plotType = "all",
  palette = rainbow,
  reverse = FALSE,
  marker = "black",
  size = 1.5,
  lwd = 0.5,
  pch = 21,
  ...
)
```

Arguments

ffs_model	Result of a forward feature selection see ffs
plotType	character. Either "all" or "selected"
palette	A color palette
reverse	Character. Should the palette be reversed?
marker	Character. Color to mark the best models
size	Numeric. Size of the points
lwd	Numeric. Width of the error bars
pch	Numeric. Type of point marking the best models
...	Further arguments for base plot if type="selected"

Author(s)

Marvin Ludwig and Hanna Meyer

See Also

ffs, bss

Examples

```
## Not run:
data(iris)
ffsmodel <- ffs(iris[,1:4],iris$Species)
plot_ffs(ffsmodel)
#plot performance of selected variables only:
plot_ffs(ffsmodel,plotType="selected")

## End(Not run)
```

---

plot_geodist *Plot euclidean nearest neighbor distances in geographic space or feature space*

Description

Density plot of nearest neighbor distances in geographic space or feature space between training data as well as between training data and prediction locations. Optional, the nearest neighbor distances between training data and test data or between training data and CV iterations is shown. The plot can be used to check the suitability of a chosen CV method to be representative to estimate map accuracy. Alternatively distances can also be calculated in the multivariate feature space.

Usage

```
plot_geodist(
  x,
  modeldomain,
  type = "geo",
  cvfolds = NULL,
  cvtrain = NULL,
  testdata = NULL,
  samplesize = 2000,
  sampling = "regular",
  variables = NULL,
  unit = "m",
  stat = "density",
  showPlot = TRUE
)
```

Arguments

x	object of class sf, training data locations
modeldomain	raster or sf object defining the prediction area (see Details)
type	"geo" or "feature". Should the distance be computed in geographic space or in the normalized multivariate predictor space (see Details)
cvfolds	optional. List of row indices of x that are held back in each CV iteration. See e.g. ?createFolds or ?createSpaceTimeFolds
cvtrain	optional. List of row indices of x to fit the model to in each CV iteration. If cvtrain is null but cvfolds is not, all samples but those included in cvfolds are used as training data
testdata	optional. object of class sf: Data used for independent validation
samplesize	numeric. How many prediction samples should be used?
sampling	character. How to draw prediction samples? See spsample. Use sampling = "Fibonacci" for global applications.
variables	character vector defining the predictor variables used if type="feature. If not provided all variables included in modeldomain are used.
unit	character. Only if type=="geo" and only applied to the plot. Supported: "m" or "km".
stat	"density" for density plot or "ecdf" for cumulative plot.
showPlot	logical

Details

The modeldomain is a sf polygon or a raster that defines the prediction area. The function takes a regular point sample (amount defined by samplesize) from the spatial extent. If type = "feature", the argument modeldomain (and if provided then also the testdata) has to include predictors. Predictor values for x are optional if modeldomain is a raster. If not provided they are extracted from the modeldomain rasterStack.

Value

A list including the plot and the corresponding data.frame containing the distances. Unit of returned geographic distances is meters.

Note

See Meyer and Pebesma (2022) for an application of this plotting function

Author(s)

Hanna Meyer, Edzer Pebesma, Marvin Ludwig

See Also

nndm

Examples

```
## Not run:
library(sf)
library(raster)
library(caret)

############ prepare sample data:
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
dat <- aggregate(dat[,c("DEM","TWI", "NDRE.M", "Easting", "Northing")],
  by=list(as.character(dat$SOURCEID)),mean)
pts <- st_as_sf(dat,coords=c("Easting","Northing"))
st_crs(pts) <- 26911
pts_train <- pts[1:29,]
pts_test <- pts[30:42,]
studyArea <- raster::stack(system.file("extdata","predictors_2012-03-25.grd",package="CAST"))
studyArea <- studyArea[[c("DEM","TWI", "NDRE.M", "NDRE.Sd", "Bt")]]

############ Distance between training data and new data:
dist <- plot_geodist(pts_train,studyArea)

############ Distance between training data, new data and test data:
#mapview(pts_train,col.regions="blue")+mapview(pts_test,col.regions="red")
dist <- plot_geodist(pts_train,studyArea,testdata=pts_test)

############ Distance between training data, new data and CV folds:
folds <- createFolds(1:nrow(pts_train),k=3,returnTrain=FALSE)
dist <- plot_geodist(x=pts_train, modeldomain=studyArea, cvfolds=folds)

## or use nndm to define folds
nndm_pred <- nndm(pts_train, studyArea)
dist <- plot_geodist(x=pts_train, modeldomain=studyArea,
    cvfolds=nndm_pred$indx_test, cvtrain=nndm_pred$indx_train)

############ Distances in the feature space:
plot_geodist(x=pts_train, modeldomain=studyArea,
    type = "feature",variables=c("DEM","TWI", "NDRE.M"))

dist <- plot_geodist(x=pts_train, modeldomain=studyArea, cvfolds = folds, testdata = pts_test,
    type = "feature",variables=c("DEM","TWI", "NDRE.M"))

############ Example for a random global dataset
############ (refer to figure in Meyer and Pebesma 2022)
library(sf)
library(rnaturalearth)
library(ggplot2)

### Define prediction area (here: global):
ee <- st_crs("+proj=eqearth")
co <- ne_countries(returnclass = "sf")
co.ee <- st_transform(co, ee)

### Simulate a spatial random sample
```

```
### (alternatively replace pts_random by a real sampling dataset (see Meyer and Pebesma 2022):
sf_use_s2(FALSE)
pts_random <- st_sample(co, 2000)

### See points on the map:
ggplot() + geom_sf(data = co.ee, fill="#00BFC4",col="#00BFC4") +
     geom_sf(data = pts_random, color = "#F8766D",size=0.5, shape=3) +
     guides(fill = FALSE, col = FALSE) +
     labs(x = NULL, y = NULL)

### plot distances:
dist <- plot_geodist(pts_random,co,showPlot=FALSE)
dist$plot+scale_x_log10(labels=round)

## End(Not run)
```

`print` *Print CAST classes*

Description

Generic print function for trainDI and aoa

Usage

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

show.trainDI(x, ...)

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

show.aoa(x, ...)

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

show.nndm(x, ...)
```

Arguments

`x`	An object of type *nndm*.
`...`	other arguments.

trainDI *Calculate Dissimilarity Index of training data*

Description

This function estimates the Dissimilarity Index (DI) of within the training data set used for a prediction model. Predictors can be weighted based on the internal variable importance of the machine learning algorithm used for model training.

Usage

```
trainDI(
  model = NA,
  train = NULL,
  variables = "all",
  weight = NA,
  CVtest = NULL,
  CVtrain = NULL
)
```

Arguments

model	A train object created with caret used to extract weights from (based on variable importance) as well as cross-validation folds
train	A data.frame containing the data used for model training. Only required when no model is given
variables	character vector of predictor variables. if "all" then all variables of the model are used or if no model is given then of the train dataset.
weight	A data.frame containing weights for each variable. Only required if no model is given.
CVtest	list or vector. Either a list where each element contains the data points used for testing during the cross validation iteration (i.e. held back data). Or a vector that contains the ID of the fold for each training point. Only required if no model is given.
CVtrain	list. Each element contains the data points used for training during the cross validation iteration (i.e. held back data). Only required if no model is given and only required if CVtrain is not the opposite of CVtest (i.e. if a data point is not used for testing, it is used for training). Relevant if some data points are excluded, e.g. when using nndm.

Value

A list of class trainDI containing:

train	A data frame containing the training data
weight	A data frame with weights based on the variable importance.

variables	Names of the used variables
catvars	Which variables are categorial
scaleparam	Scaling parameters. Output from scale
trainDist_avrg	A data frame with the average eucildean distance of each training point to every other point
trainDist_avrgmean	The mean of trainDist_avrg. Used for normalizing the DI
trainDI	Dissimilarity Index of the training data
threshold	The DI threshold used for inside/outside AOA
lower_threshold	The lower DI threshold. Currently unused.

Note

This function is called within aoa to estimate the DI and AOA of new data. However, it may also be used on its own if only the DI of training data is of interest, or to facilitate a parallelization of aoa by avoiding a repeated calculation of the DI within the training data.

Author(s)

Hanna Meyer, Marvin Ludwig

References

Meyer, H., Pebesma, E. (2021): Predicting into unknown space? Estimating the area of applicability of spatial prediction models. doi: 10.1111/2041210X.13650

See Also

aoa

Examples

```
## Not run:
library(sf)
library(raster)
library(caret)
library(viridis)
library(latticeExtra)
library(ggplot2)

# prepare sample data:
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
dat <- aggregate(dat[,c("VW","Easting","Northing")],by=list(as.character(dat$SOURCEID)),mean)
pts <- st_as_sf(dat,coords=c("Easting","Northing"))
pts$ID <- 1:nrow(pts)
set.seed(100)
pts <- pts[1:30,]
studyArea <- stack(system.file("extdata","predictors_2012-03-25.grd",package="CAST"))[[1:8]]
```

```
trainDat <- extract(studyArea,pts,df=TRUE)
trainDat <- merge(trainDat,pts,by.x="ID",by.y="ID")

# visualize data spatially:
spplot(scale(studyArea))
plot(studyArea$DEM)
plot(pts[,1],add=TRUE,col="black")

# train a model:
set.seed(100)
variables <- c("DEM","NDRE.Sd","TWI")
model <- train(trainDat[,which(names(trainDat)%in%variables)],
trainDat$VW, method="rf", importance=TRUE, tuneLength=1,
trControl=trainControl(method="cv",number=5,savePredictions=T))
print(model) #note that this is a quite poor prediction model
prediction <- predict(studyArea,model)
plot(varImp(model,scale=FALSE))

#...then calculate the DI of the trained model:
DI = trainDI(model=model)
plot(DI)

# the DI can now be used to compute the AOA:
AOA = aoa(studyArea, model = model, trainDI = DI)
print(AOA)
plot(AOA)

## End(Not run)
```

Index