Package ‘meteo’

October 14, 2023

Version 2.0-2

Date 2023-10-09

Type Package

Encoding UTF-8

Title RFSI and Spatio-Temporal Geostatistical Interpolation for Meteorological and Other Environmental Variables

Author Milan Kilibarda [aut] (<https://orcid.org/0000-0002-2930-3596>), Aleksandar Sekulić [aut, cre] (<https://orcid.org/0000-0002-5515-2779>), Tomislav Hengl [ctb], Edzer Pebesma [ctb], Benedikt Graeler [ctb]

Maintainer Aleksandar Sekulić <asekulic@grf.bg.ac.rs>

Depends R (>= 4.0.0)

Description RFSI and spatio-temporal geostatistical interpolation for meteorological and other environmental variables. Global spatio-temporal models calculated using publicly available data are stored in package.

License GPL (>= 2.0) | file LICENCE

URL https://www.r-pkg.org/pkg/meteo, https://r-forge.r-project.org/projects/meteo/, https://github.com/AleksandarSekulic/Rmeteo

LazyLoad yes

Imports methods, utils, stats, DescTools, caret, data.table, dplyr, sp, spacetime, gstat, foreach, parallel, snowfall, doParallel, plyr, units, nabor, CAST, ranger, sf, sftime, raster, terra

BugReports https://github.com/AleksandarSekulic/Rmeteo/issues

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-10-14 12:50:11 UTC

R topics documented:

`acc.metric.fun` *Accuracy metrics calculation*

Description

Calculates classification and regression accuracy metrics for given coresponding observation and prediction vectors.

Usage

```
acc.metric.fun(obs, pred, acc.m)
```

Arguments

`obs`	`numeric` or `factor` `vector`; Observations.
`pred`	`numeric` or `factor` `vector`; Predictions.
`acc.m`	`character`; Accuracy metric. Possible values for regression: "ME", "MAE", "NMAE", "RMSE", "NRMSE", "R2", "CCC". Possible values for classification: "Accuracy", "Kappa", "AccuracyLower", "AccuracyUpper", "AccuracyNull", "AccuracyPValue", "McnemarPValue".

Value

Accuracy metric value.

Author(s)

Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

Sekulić, A., Kilibarda, M., Heuvelink, G. B., Nikolić, M. & Bajat, B. Random Forest Spatial Interpolation.Remote. Sens. 12, 1687, https://doi.org/10.3390/rs12101687 (2020).

See Also

`acc.metric.fun` `rfsi` `pred.rfsi` `tune.rfsi` `cv.rfsi` `pred.strk` `cv.strk`

Examples

```
library(sp)
library(sf)
library(CAST)
library(ranger)
library(plyr)
library(meteo)

# preparing data
demo(meuse, echo=FALSE)
meuse <- meuse[complete.cases(meuse@data),]
data = st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
fm.RFSI <- as.formula("zinc ~ dist + soil + ffreq")

# making tgrid
n.obs <- 1:3
min.node.size <- 2:10
sample.fraction <- seq(1, 0.632, -0.05) # 0.632 without / 1 with replacement
splitrule <- "variance"
ntree <- 250 # 500
```

```
mtry <- 3:(2+2*max(n.obs))
tgrid = expand.grid(min.node.size=min.node.size, num.trees=ntree,
                    mtry=mtry, n.obs=n.obs, sample.fraction=sample.fraction)

## Not run:
# do cross-validation
rfsi_cv <- cv.rfsi(formula=fm.RFSI, # without nearest obs
                   data = data,
                   zero.tol=0,
                   tgrid = tgrid, # combinations for tuning
              tgrid.n = 5, # number of randomly selected combinations from tgrid for tuning
                   tune.type = "LLO", # Leave-Location-Out CV
                   k = 5, # number of folds
                   seed = 42,
                   acc.metric = "RMSE", # R2, CCC, MAE
                   output.format = "data.frame",
                   cpus=detectCores()-1,
                   progress=1,
                   importance = "impurity")
summary(rfsi_cv)

# accuracy metric calculation
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "R2")
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "RMSE")
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "NRMSE")
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "MAE")
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "NMAE")
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "CCC")

## End(Not run)
```

cv.rfsi — *Nested k-fold cross-validation for Random Forest Spatial Interpolation (RFSI)*

Description

Function for nested k-fold cross-validation function for Random Forest Spatial Interpolation (RFSI) (Sekulić et al. 2020). It is based on rfsi, pred.rfsi, and tune.rfsi functions. Currently, only spatial (leave-location-out) cross-validation is implemented. Temporal and spatio-temporal cross-validation will be implemented in the future.

Usage

```
cv.rfsi(formula,
         data,
         data.staid.x.y.z = NULL,
         use.idw = FALSE,
         s.crs = NA,
         p.crs = NA,
         tgrid,
         tgrid.n=10,
         tune.type = "LLO",
         k = 5,
         seed=42,
         out.folds,
         in.folds,
         acc.metric,
         output.format = "data.frame",
         cpus = detectCores()-1,
         progress = 1,
         soil3d = FALSE,
         no.obs = 'increase',
         ...)
```

Arguments

`formula`	formula; Formula for specifying target variable and covariates (without nearest observations and distances to them). If `z~1`, an RFSI model using only nearest obsevrations and distances to them as covariates will be cross-validated.
`data`	sf-class, sftime-class, SpatVector-class or data.frame; Contains target variable (observations) and covariates used for making an RFSI model. If data.frame object, it should have next columns: station ID (staid), longitude (x), latitude (y), 3rd component - time, depth, ... (z) of the observation, observation value (obs) and covariates (cov1, cov2, ...). If covariates are missing, the RFSI model using only nearest obsevrations and distances to them as covariates (`formula=z~1`) will be cross-validated.
`data.staid.x.y.z`	numeric or character vector; Positions or names of the station ID (staid), longitude (x), latitude (y) and 3rd component (z) columns in data.frame object (e.g. c(1,2,3,4)). If `data` is sf-class, sftime-class, or SpatVector-class object, `data.staid.x.y.z` is used to point staid and z position. Set z position to NA (e.g. c(1,2,3,NA)) or ommit it (e.g. c(1,2,3)) for spatial interpolation. Default is NULL.
`use.idw`	boolean; IDW prediction as covariate - will IDW predictions from `n.obs` nearest observations be calculated and tuned (see function near.obs). Default is FALSE.
`s.crs`	st_crs or crs; Source CRS of `data`. If `data` contains crs, `s.crs` will be overwritten. Default is NA.
`p.crs`	st_crs or crs; Projection CRS for `data` reprojection. If NA, `s.crs` will be used for distance calculation. Note that observations should be in projection for finding nearest observations based on Eucleadean distances (see function near.obs). Default is NA.
`tgrid`	data.frame; Possible tuning parameters for nested folds. The column names are same as the tuning parameters. Possible tuning parameters are: `n.obs`, `num.trees`, `mtry`, `min.node.size`, `sample.fraction`, `splirule`, `idw.p`, and `depth.range`.

`tgrid.n`	numeric; Number of randomly chosen `tgrid` combinations for nested tuning of RFSI. If larger than `tgrid`, will be set to `length(tgrid)`
`tune.type`	character; Type of nested cross-validation: leave-location-out ("LLO"), leave-time-out ("LTO") - TO DO, and leave-location-time-out ("LLTO") - TO DO. Default is "LLO".
`k`	numeric; Number of random outer and inner folds (i.e. for cross-validation and nested tuning) that will be created with CreateSpacetimeFolds function. Default is 5.
`seed`	numeric; Random seed that will be used to generate outer and inner folds with CreateSpacetimeFolds function.
`out.folds`	numeric or character vector or value; Showing outer folds column (if value) or rows (vector) of `data` observations used for cross-validation. If missing, will be created with CreateSpacetimeFolds function.
`in.folds`	numeric or character vector or value; Showing innner folds column (if value) or rows (vector) of `data` observations used for cross-validation. If missing, will be created with CreateSpacetimeFolds function.
`acc.metric`	character; Accuracy metric that will be used as a criteria for choosing an optimal RFSI model in nested tuning. Possible values for regression: "ME", "MAE", "NMAE", "RMSE" (default), "NRMSE", "R2", "CCC". Possible values for classification: "Accuracy","Kappa" (default), "AccuracyLower", "AccuracyUpper", "AccuracyNull", "AccuracyPValue", "McnemarPValue".
`output.format`	character; Format of the output, data.frame (default), sf-class, sftime-class, or SpatVector-class.
`cpus`	numeric; Number of processing units. Default is detectCores()-1.
`progress`	numeric; If progress bar is shown. 0 is no progress bar, 1 is outer folds results, 2 is + innner folds results, 3 is + prediction progress bar. Default is 1.
`soil3d`	logical; If 3D soil modellig is performed and near.obs.soil function is used for finding n nearest observations and distances to them. In this case, z position of the `data.staid.x.y.z` points to the depth column.
`no.obs`	character; Possible values are `increase` (default) and `exactly`. If set to `increase`, in case if there is no `n.obs` observations in `depth.range` for a specific location, the `depth.range` is increased (multiplied by 2, 3, ...) until the number of observations are larger or equal to `n.obs`. If set to `exactly`, the function will raise an error when it come to the first location with no `n.obs` observations in specified `depth.range` (see function near.obs.soil).
`...`	Further arguments passed to ranger.

Value

A data.frame, sf-class, sftime-class, or SpatVector-class object (depends on `output.format` argument), with columns:

`obs`	Observations.
`pred`	Predictions from cross-validation.
`folds`	Folds used for cross-validation.

Author(s)

Aleksandar Sekulic <asekulic@grf.bg.ac.rs>

References

Sekulić, A., Kilibarda, M., Heuvelink, G. B., Nikolić, M. & Bajat, B. Random Forest Spatial Interpolation. Remote. Sens. 12, 1687, https://doi.org/10.3390/rs12101687 (2020).

See Also

near.obs rfsi pred.rfsi tune.rfsi

Examples

```
library(CAST)
library(doParallel)
library(ranger)
library(sp)
library(sf)
library(terra)
library(meteo)

# prepare data
demo(meuse, echo=FALSE)
meuse <- meuse[complete.cases(meuse@data),]
data = st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
# data = terra::vect(meuse)
# data.frame
# data <- as.data.frame(meuse)
# data$id = 1:nrow(data)
# data.staid.x.y.z <- c(15,"x","y",NA)
fm.RFSI <- as.formula("zinc ~ dist + soil + ffreq")

# making tgrid
n.obs <- 1:6
min.node.size <- 2:10
sample.fraction <- seq(1, 0.632, -0.05) # 0.632 without / 1 with replacement
splitrule <- "variance"
ntree <- 250 # 500
mtry <- 3:(2+2*max(n.obs))
tgrid = expand.grid(min.node.size=min.node.size, num.trees=ntree,
                    mtry=mtry, n.obs=n.obs, sample.fraction=sample.fraction)
## Not run:
# Cross-validation of RFSI
rfsi_cv <- cv.rfsi(formula=fm.RFSI, # without nearest obs
                   data = data,
             # data.staid.x.y.z = c("id", "x", "y", NA), # only if class(data) == data.frame
                   # s.crs=NA,
                   # t.crs=NA,
                   tgrid = tgrid, # combinations for tuning
              tgrid.n = 2, # number of randomly selected combinations from tgrid for tuning
                   tune.type = "LLO", # Leave-Location-Out CV
                   k = 5, # number of folds
                   seed = 42,
                   acc.metric = "RMSE", # R2, CCC, MAE
                   output.format = "sf", # "data.frame", # "SpatVector",
                   cpus=detectCores()-1,
                   progress=1,
                   importance = "impurity") # ranger parameter

summary(rfsi_cv)
rfsi_cv$dif <- rfsi_cv$obs - rfsi_cv$pred
# plot(rfsi_cv["dif"])
# spplot(rfsi_cv[, , "obs"])
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "R2")
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "RMSE")
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "MAE")
acc.metric.fun(rfsi_cv$obs, rfsi_cv$pred, "CCC")

## End(Not run)
```

cv.strk *k-fold cross-validation for spatio-temporal regression kriging*

Description

k-fold cross-validation function for spatio-temporal regression kriging based on pred.strk. Currently, only spatial (leave-location-out) cross-validation is implemented. Temporal and spatio-temporal cross-validation will be implemented in the future.

Usage

```
cv.strk(data,
        obs.col=1,
        data.staid.x.y.z = NULL,
        crs = NA,
        zero.tol=0,
        reg.coef,
        vgm.model,
        sp.nmax=20,
        time.nmax=2,
        type = "LLO",
        k = 5,
        seed = 42,
        folds,
        refit = TRUE,
        output.format = "STFDF",
        parallel.processing = FALSE,
        pp.type = "snowfall",
        cpus=detectCores()-1,
        progress=TRUE,
        ...)
```

Arguments

`data`	STFDF-class, STSDF-class, STIDF-class, sf-class, sftime-class, SpatVector-class or data.frame; Contains target variable (observations) and covariates in space and time used to perform STRK cross validation. If data.frame object, it should have next columns: station ID (staid), longitude (x), latitude (y), 3rd component - time, depth, ... (z) of the observation, observation value (obs), and covariates (cov1, cov2, ...). Covariate names should be the same as in the `reg.coef` (see below). If covariates are missing, then spatio-temporal ordinary kriging cross validation is performed.
`obs.col`	numeric or character; Column name or number showing position of the observation column in the `data`. Default is 1.
`data.staid.x.y.z`	numeric or character vector; Positions or names of the station ID (staid), longitude (x), latitude (y) and 3rd component - time, depth (z) columns in data.frame object (e.g. c(1,2,3,4)). If `data` is sf-class, sftime-class, or SpatVector-class object, `data.staid.x.y.z` is used to point staid and z position. If `data` is STFDF-class, STSDF-class, STIDF-class object, `data.staid.x.y.z` is used to point only staid position. Default is NULL.
`crs`	st_crs or crs; Source CRS of `data`. If `data` contains crs, `crs` will not be used. Default is NA.
`zero.tol`	numeric; A distance value below (or equal to) which locations are considered as duplicates. Default is 0. See rm.dupl. Duplicates are removed to avoid singular covariance matrices in kriging.
`reg.coef`	numeric; Vector of named linear regression coefficients. Names of the coefficients (e.g. "Intercept", "temp_geo", "modis", "dem", "twi") will be used to match appropriate covariates from `data`. Coefficients for metorological variables (temperature, precipitation, etc.) can be taken from data(tregcoef) or can be specified by the user.
`vgm.model`	StVariogramModel list; Spatio-temporal variogram of regression residuals (or observations if spatio-temporal ordinary kriging). See vgmST. Spatio-temporal variogram model on residuals for metorological variables (temperature, precipitation, etc.) can be taken from data(tvgms) or can be specified by the user as a vgmST object.
`sp.nmax`	numeric; A number of spatially nearest observations that should be used for kriging predictions. If `tiling` is TRUE (see below), then is a number of spatially nearest observations that should be used for each tile. Deafult is 20.
`time.nmax`	numeric; A number of temporally nearest observations that should be used for kriging predictions Deafult is 2.
`type`	character; Type of cross-validation: leave-location-out ("LLO"), leave-time-out ("LTO"), and leave-location-time-out ("LLTO"). Default is "LLO". "LTO" and "LLTO" are not implemented yet. Will be in the future.
`k`	numeric; Number of random folds that will be created with CreateSpacetimeFolds function. Default is 5.
`seed`	numeric; Random seed that will be used to generate outer and inner folds with CreateSpacetimeFolds function.

`folds`	numeric or character vector or value; Showing folds column (if value) or rows (vector) of `data` observations used for cross-validation. If missing, will be created with CreateSpacetimeFolds function.
`refit`	logical; If refit of linear regression trend and spatio-teporal variogram should be performed. Spatio-teporal variogram is fit using `vgm.model` as desired spatio-temporal model for fit.StVariogram function. Default is TRUE.
`output.format`	character; Format of the output, STFDF-class (default), STSDF-class, STIDF-class, data.frame, sf-class, sftime-class, or SpatVector-class.
`parallel.processing`	logical; If parallel processing is performed. Default is FALSE.
`pp.type`	character; Type (R package) of parallel processing, "snowfall" (default) or "doParallel".
`cpus`	numeric; Number of processing units. Default is detectCores()-1.
`progress`	logical; If progress bar is shown. Default is TRUE.
`...`	Further arguments passed to krigeST or pred.strk.

Value

A STFDF-class (default), STSDF-class, STIDF-class, data.frame, sf-class, sftime-class, or SpatVector-class object (depends on `output.format` argument), with columns:

`obs`	Observations.
`pred`	Predictions from cross-validation.
`folds`	Folds used for cross-validation.

For accuracy metrics see acc.metric.fun function.

Author(s)

Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>, Milan Kilibarda <`kili@grf.bg.ac.rs`>

References

Kilibarda, M., T. Hengl, G. B. M. Heuvelink, B. Graeler, E. Pebesma, M. Percec Tadic, and B. Bajat (2014), Spatio-temporal interpolation of daily temperatures for global land areas at 1 km resolution, J. Geophys. Res. Atmos., 119, 2294-2313, doi:10.1002/2013JD020803.

See Also

`acc.metric.fun` `pred.strk` `tregcoef` `tvgms` `regdata` `meteo2STFDF` `tgeom2STFDF`

Examples

```
library(sp)
library(spacetime)
library(gstat)
library(plyr)
library(CAST)
library(doParallel)
library(ranger)
# preparing data
data(dtempc)
data(stations)
data(regdata) # covariates, made by mete2STFDF function

regdata@sp@proj4string <- CRS('+proj=longlat +datum=WGS84')
data(tvgms) # ST variogram models
data(tregcoef) # MLR coefficients

lonmin=18 ;lonmax=22.5 ; latmin=40 ;latmax=46
serbia = point.in.polygon(stations$lon, stations$lat, c(lonmin,lonmax,lonmax,lonmin),
                          c(latmin,latmin,latmax,latmax))
st = stations[ serbia!=0, ] # stations in Serbia approx.
crs = CRS('+proj=longlat +datum=WGS84')

# create STFDF
stfdf <- meteo2STFDF(obs = dtempc,
                     stations = st,
                     crs = crs)

# Cross-validation for mean temperature for days "2011-07-05" and "2011-07-06"
# global model is used for regression and variogram

# Overlay observations with covariates
time <- index(stfdf@time)
covariates.df <- as.data.frame(regdata)
names_covar <- names(tregcoef[[1]])[-1]
for (covar in names_covar){
  nrowsp <- length(stfdf@sp)
  regdata@sp=as(regdata@sp,'SpatialPixelsDataFrame')
  ov <- sapply(time, function(i)
    if (covar %in% names(regdata@data)) {
      if (as.Date(i) %in% as.Date(index(regdata@time))) {
        over(stfdf@sp, as(regdata[, i, covar], 'SpatialPixelsDataFrame'))[, covar]
      } else {
        rep(NA, length(stfdf@sp))
      }
    } else {
      over(stfdf@sp, as(regdata@sp[covar], 'SpatialPixelsDataFrame'))[, covar]
    }
  )
  ov <- as.vector(ov)
  if (all(is.na(ov))) {
    stop(paste('There is no overlay of data with covariates!', sep = ""))
  }
  stfdf@data[covar] <- ov
}

# Remove stations out of covariates
for (covar in names_covar){
  # count NAs per stations
  numNA <- apply(matrix(stfdf@data[,covar],
                        nrow=nrowsp,byrow= FALSE), MARGIN=1,
                 FUN=function(x) sum(is.na(x)))
  rem <- numNA != length(time)
  stfdf <-  stfdf[rem,drop= FALSE]
}

# Remove dates out of covariates
rm.days <- c()
for (t in 1:length(time)) {
  if(sum(complete.cases(stfdf[, t]@data)) == 0) {
    rm.days <- c(rm.days, t)
  }
}
if(!is.null(rm.days)){
  stfdf <- stfdf[,-rm.days]
}

### Example with STFDF and without parallel processing and without refitting of variogram
results <- cv.strk(data = stfdf,
                   obs.col = 1, # "tempc"
                   data.staid.x.y.z = c(1,NA,NA,NA),
                   reg.coef = tregcoef[[1]],
                   vgm.model = tvgms[[1]],
                   sp.nmax = 20,
                   time.nmax = 2,
                   type = "LLO",
                   k = 5,
                   seed = 42,
                   refit = FALSE,
                   progress = TRUE
)

# stplot(results[,,"pred"])
summary(results)
# accuracy
acc.metric.fun(results@data$obs, results@data$pred, "R2")
acc.metric.fun(results@data$obs, results@data$pred, "RMSE")
acc.metric.fun(results@data$obs, results@data$pred, "MAE")
acc.metric.fun(results@data$obs, results@data$pred, "CCC")
```

`data.prepare` *Prepare data*

Description

Function for data preparation for RFSI and STRK functions. It transforms data to a data.frame.

Usage

```
data.prepare(data,
             data.staid.x.y.z=NULL,
             obs.col=NULL,
             s.crs=NA
)
```

Arguments

`data`	sf-class, sftime-class, SpatVector-class, SpatRaster-class or data.frame; Contains target variable (observations) and covariates. If data.frame object, it should have next columns: station ID (staid), longitude (x), latitude (y), 3rd component - time, depth, ... (z) of the observation, and observation value (obs).
`data.staid.x.y.z`	numeric or character vector; Positions or names of the station ID (staid), longitude (x), latitude (y) and 3rd component (z) columns in data.frame object (e.g. c(1,2,3,4)). If `data` is sf-class, sftime-class, or SpatVector-class object, `data.staid.x.y.z` is used to point staid and z position. Set z position to NA (e.g. c(1,2,3,NA)) or ommit it (e.g. c(1,2,3)) for spatial interpolation. Default is NULL.
`obs.col`	numeric or character; Column name or number showing position of the observation column in the `data`. Default is 1.
`s.crs`	st_crs or crs; Source CRS of `data`. If `data` contains crs, `s.crs` will not be used. Default is NA.

Value

A list with the following elements:

`data.df`	A data.frame obtained from `data`.
`data.staid.x.y.z`	Positions of the station ID (staid), longitude (x), latitude (y) and 3rd component (z) columns in data.frame object (e.g. c(1,2,3,4)).
`s.crs`	Source CRS of `data`.
`obs.col`	Column number showing position of the observation column in the `data`.

Author(s)

Aleksandar Sekulic `<asekulic@grf.bg.ac.rs>`

References

Sekulić, A., Kilibarda, M., Heuvelink, G. B., Nikolić, M. & Bajat, B. Random Forest Spatial Interpolation. Remote. Sens. 12, 1687, https://doi.org/10.3390/rs12101687 (2020).

See Also

`near.obs` `rfsi` `tune.rfsi` `cv.rfsi`

Examples

```
library(sf)
library(meteo)
library(sp)

# prepare data
demo(meuse, echo=FALSE)
meuse <- meuse[complete.cases(meuse@data),]
data = st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
data.df <- data.prepare(data,
                        obs.col="zinc")
str(data.df)
```

---

dem_twi_srb   *Digital Elevation Model (DEM) and Topographic Wetness Index (TWI) for Serbia*

Description

Digital Elevation Model (DEM) and Topographic Wetness Index (TWI) for Serbia in SpatRaster format.

Usage

```
data(dem_twi_srb)
```

Format

The `dem_twi_srb` contains the following layers:

`dem`	Digital Elevation Model (DEM) in meters
`twi`	Topographic Wetness Index (TWI)

Author(s)

Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

Examples

```
library(terra)
## load data
data(dem_twi_srb)
terra::unwrap(dem_twi_srb)
```

`dprec` *Daily precipitation amount in mm for July 2011*

Description

Sample data set showing values of merged daily precipitation amount measurements from the Global Surface Summary of Day data (GSOD) with European Climate Assessment & Data set (ECA&D) for the month July 2011.

Usage

```
data(dprec)
```

Format

The `dprec` contains the following columns:

`staid` character; station ID from GSOD or ECA&D data set

`time` Date; day of the measurement

`prec` numeric; daily precipitation amount in mm

Note

The data summaries provided here are based on data exchanged under the World Meteorological Organization (WMO) World Weather Watch Program. To prepare a point map, merge with the `stations` table containing stations' coordinates.

Author(s)

Milan Kilibarda and Tomislav Hengl

References

- Global Surface Summary of the day data (`ftp://ftp.ncdc.noaa.gov/pub/data/gsod/`)
- European Climate Assessment & Dataset (`https://www.ecad.eu/dailydata/predefinedseries.php`)

Examples

```
## load data
data(dprec)
str(dprec)
```

dslp — *Mean sea level pressure in hPa for July 2011*

Description

Sample data set showing values of merged mean sea level pressure measurements from the Global Surface Summary of Day data (GSOD) with European Climate Assessment & Data set (ECA&D) for the month July 2011.

Usage

```
data(dslp)
```

Format

The `dslp` contains the following columns:

`staid`	character; station ID from GSOD or ECA&D data set
`time`	Date; day of the measurement
`slp`	numeric; mean sea level pressure amount in hPa

Note

The data summaries provided here are based on data exchanged under the World Meteorological Organization (WMO) World Weather Watch Program. To prepare a point map, merge with the `stations` table containing stations' coordinates.

Author(s)

Milan Kilibarda and Tomislav Hengl

References

- Global Surface Summary of the day data (`ftp://ftp.ncdc.noaa.gov/pub/data/gsod/`)
- European Climate Assessment & Dataset (`https://www.ecad.eu/dailydata/predefinedseries.php`)

Examples

```
## load data
data(dslp)
str(dslp)
```

dsndp *Daily snow depth in cm for July 2011*

Description

Sample data set showing values of merged daily snow depth measurements from the Global Surface Summary of Day data (GSOD) with European Climate Assessment & Data set (ECA&D) for the month July 2011.

Usage

```
data(dsndp)
```

Format

The dsndp contains the following columns:

staid	character; station ID from GSOD or ECA&D data set
time	Date; day of the measurement
sndp	numeric; daily snow depth in cm

Note

The data summaries provided here are based on data exchanged under the World Meteorological Organization (WMO) World Weather Watch Program. To prepare a point map, merge with the stations table containing stations' coordinates.

Author(s)

Milan Kilibarda and Tomislav Hengl

References

- Global Surface Summary of the day data (ftp://ftp.ncdc.noaa.gov/pub/data/gsod/)
- European Climate Assessment & Dataset (https://www.ecad.eu/dailydata/predefinedseries.php)

Examples

```
## load data
data(dsndp)
str(dsndp)
```

dtempc — *Mean daily temperature in degrees Celsius for July 2011*

Description

Sample data set showing values of merged mean daily temperature measurements from the Global Surface Summary of Day data (GSOD) with European Climate Assessment & Dataset (ECA&D) data for the month July 2011.

Usage

```
data(dtempc)
```

Format

The `dtempc` contains the following columns:

`staid` character; station ID from GSOD or ECA&D dataset

`time` Date; day of the measurement

`tempc` numeric; mean daily temperature in degrees Celsius

Note

The data summaries provided here are based on data exchanged under the World Meteorological Organization (WMO) World Weather Watch Program. To prepare a point map, merge with the `stations` table containing stations' coordinates.

Author(s)

Milan Kilibarda and Tomislav Hengl

References

- Global Surface Summary of the day data (`ftp://ftp.ncdc.noaa.gov/pub/data/gsod/`)
- European Climate Assessment & Dataset (`https://www.ecad.eu/dailydata/predefinedseries.php`)

Examples

```
## load data
data(dtempc)
str(dtempc)
```

`dtempc_ogimet` *Mean daily temperature in degrees Celsius for the year 2019 for Serbia*

Description

Sample data set of mean daily temperature measurements from the OGIMET service for the year 2019 for Serbian territory.

Usage

```
data(dtempc_ogimet)
```

Format

The `dtempc_ogimet` contains the following columns:

`staid`	character; station ID from OGIMET
`name`	character; station name
`lon`	numeric; Longitude
`lat`	numeric; Latitude
`elevation`	numeric; Hight
`time`	Date; day of the measurement
`tmean`	numeric; mean daily temperature in degrees Celsius
`dem`	numeric; Digital Elevation Model (DEM) in meters
`twi`	numeric; Topographic Wetness Index (TWI)
`cdate`	numeric; Cumulative day from 1960
`doy`	numeric; Day of year
`gtt`	numeric; Geometrical temperature trend

Author(s)

Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

- OGIMET service (`https://www.ogimet.com/`)

Examples

```
## load data
data(dtempc_ogimet)
str(dtempc)
```

`dtemp_maxc` *Maximum daily temperature in degrees Celsius for July 2011*

Description

Sample data set showing values of merged maximum daily temperature measurements from the Global Surface Summary of Day data (GSOD) with European Climate Assessment & Dataset (ECA&D) data for the month July 2011.

Usage

```
data(dtemp_maxc)
```

Format

The `dtemp_maxc` contains the following columns:

`staid` character; station ID from GSOD or ECA&D dataset

`time` Date; day of the measurement

`temp_minc` numeric; maximum daily temperature in degrees Celsius

Note

The data summaries provided here are based on data exchanged under the World Meteorological Organization (WMO) World Weather Watch Program. To prepare a point map, merge with the `stations` table containing stations' coordinates.

Author(s)

Milan Kilibarda and Tomislav Hengl

References

- Global Surface Summary of the day data (`ftp://ftp.ncdc.noaa.gov/pub/data/gsod/`)
- European Climate Assessment & Dataset (`https://www.ecad.eu/dailydata/predefinedseries.php`)

Examples

```
## load data
data(dtemp_maxc)
str(dtemp_maxc)
```

`dtemp_minc` *Minimum daily temperature in degrees Celsius for July 2011*

Description

Sample data set showing values of merged minimum daily temperature measurements from the Global Surface Summary of Day data (GSOD) with European Climate Assessment & Data set (ECA&D) for the month July 2011.

Usage

```
data(dtemp_minc)
```

Format

The `dtemp_minc` contains the following columns:

`staid` character; station ID from GSOD or ECA&D data set

`time` Date; day of the measurement

`temp_minc` numeric; minimum daily temperature in degrees Celsius

Note

The data summaries provided here are based on data exchanged under the World Meteorological Organization (WMO) World Weather Watch Program. To prepare a point map, merge with the `stations` table containing stations' coordinates.

Author(s)

Milan Kilibarda and Tomislav Hengl

References

- Global Surface Summary of the day data (`ftp://ftp.ncdc.noaa.gov/pub/data/gsod/`)
- European Climate Assessment & Dataset (`https://www.ecad.eu/dailydata/predefinedseries.php`)

Examples

```
## load data
data(dtemp_minc)
str(dtemp_minc)
```

`dwdsp` *Daily mean wind speed in m/s for July 2011*

Description

Sample data set showing values of merged daily mean wind speed measurements from the Global Surface Summary of Day data (GSOD) with European Climate Assessment & Data set (ECA&D) for the month July 2011.

Usage

```
data(dwdsp)
```

Format

The `dwdsp` contains the following columns:

`staid`	character; station ID from GSOD or ECA&D data set
`time`	Date; day of the measurement
`wdsp`	numeric; daily mean wind speed in m/s

Note

The data summaries provided here are based on data exchanged under the World Meteorological Organization (WMO) World Weather Watch Program. To prepare a point map, merge with the `stations` table containing stations' coordinates.

Author(s)

Milan Kilibarda and Tomislav Hengl

References

- Global Surface Summary of the day data (`ftp://ftp.ncdc.noaa.gov/pub/data/gsod/`)
- European Climate Assessment & Dataset (`https://www.ecad.eu/dailydata/predefinedseries.php`)

Examples

```
## load data
data(dwdsp)
str(dwdsp)
```

`get_meteo` *Get daily meteorological data for specific location and dates.*

Description

The function gives back daily metorological data for specific location and dates for Europe for 1991-2020 period (from MeteoEurope1km dataset).

Usage

```
get_meteo(loc,
          dates,
          var = "tmean",
          source = "MeteoEurope1km")
```

Arguments

loc	`sf`, `SpatVector`, data.frame, `matrix`, `numeric` or `integer`; Locations in WGS84 (EPSG:4326). If data.frame or `matrix` columns are lon/lat.
dates	`Date` or `character`; Date(s).
var	`character`; Daily meteorological variable. Possible values are: `"tmean"` (default), `"tmax"`, `"tmin"`, `"prcp"`, or `"slp"`.
source	`character`; Data source. Possible values are: `"MeteoEurope1km"` (default).

Value

data.frame object with daily meteorological values for specific locations (rows) and specific dates (columns). First column is location ID.

Author(s)

Aleksandar Sekulić <asekulic@grf.bg.ac.rs>

See Also

`pred.strk`

Examples

```
library(terra)

loc <- c(21, 45)
# loc <- as.data.frame(rbind(c(21, 45),
#                            c(21,45.5),
#                            c(21.5,45),
#                            c(21.5,45.5)))

dates <- as.Date("2020-12-25")
```

```
# dates <- seq(as.Date("2020-12-25"), as.Date("2020-12-31"), by="day")

tmean <- get_meteo(loc,
                   dates,
                   var = "tmean", # "tmax" "tmin" "prcp" "slp"
                   source = "MeteoEurope1km")
```

meteo2STFDF *Create an object of STFDF-class class from two data frames (observation and stations)*

Description

The function creates an object of STFDF-class class, spatio-temporal data with full space-time grid, from two data frames (observation and stations). Observations data frame minimum contains station ID column, time column (day of observation) and measured variable column. Stations data frame contains at least station ID column, longitude (or x) and latitude (or y) column.

Usage

```
meteo2STFDF(obs,
            stations,
            obs.staid.time = c(1, 2),
            stations.staid.lon.lat = c(1, 2, 3),
            crs=CRS(as.character(NA)),
            delta=NULL)
```

Arguments

`obs`	data.frame; observations data frame minimum contains station ID column, time column (day of observation) and measured variable column. It can contain additional variables (columns).
`stations`	data.frame; Stations data frame contains at least station ID column, longitude (or x) and latitude (or y) column.It can contain additional variables (columns).
`obs.staid.time`	numeric; records the column positions where in `obs` (observation) data frame the station ID and time values are stored.
`stations.staid.lon.lat`	numeric; records the column positions where in `stations` data frame the station ID, longitude (x) and latitude (y) values are stored.
`crs`	CRS; coordinate reference system (see CRS-class) of `stations` coordinates
`delta`	time; time interval to end points in seconds

Value

STFDF-class object

Note

The function is intended for conversion of meteorological data to STFDF-class object, but can be used for similar spatio temporal data stored in two separated tables.

Author(s)

Milan Kilibarda <kili@grf.bg.ac.rs>, Aleksandar Sekulic <asekulic@grf.bg.ac.rs>

See Also

tgeom2STFDF, pred.strk

Examples

```
# prepare data
# load observation - data.frame of mean temperatures
data(dtempc)
# str(dtempc)
data(stations)
# str(stations)
lonmin=18 ;lonmax=22.5 ; latmin=40 ;latmax=46
library(sp)
library(spacetime)
serbia = point.in.polygon(stations$lon, stations$lat, c(lonmin,lonmax,lonmax,lonmin),
                          c(latmin,latmin,latmax,latmax))
st= stations[ serbia!=0, ] # stations in Serbia approx.
# create STFDF
temp <- meteo2STFDF(dtempc,st, crs= CRS('+proj=longlat +datum=WGS84'))
# str(temp)
```

`near.obs` *Finds n nearest observations from given locations.*

Description

The function finds n nearest observations from given locations and creates an object of data.frame class. First n columns are Euclidean distances to n nearest locations and next n columns are observations at n nearest stations, and rows are given locations. Further more it can calculate averages in circles with different radiuses, can find nearest observation in quadrants (directions) and calculate IDW predictions from nearest observations. It is based on knn function of package nabor.

Usage

```
near.obs(locations,
          locations.x.y = c(1,2),
          observations,
          observations.x.y = c(1,2),
          obs.col = 3,
          n.obs = 10,
          rm.dupl = TRUE,
          avg = FALSE,
          increment,
          range,
          quadrant = FALSE,
          idw=FALSE,
          idw.p=2)
```

Arguments

`locations`	data.frame with x and y coordinates columns, or sf-class, SpatVector-class or SpatRaster-class object; Locations (FROM) for which n nearest observations are found and distances are calculated.
`locations.x.y`	numeric or character vector; Positions or names of the x and y columns in `locations` if data.frame. Default is c(1,2).
`observations`	data.frame with x, y and observation columns, or sf-class or SpatVector-class object with an observation column; Observations (TO).
`observations.x.y`	numeric or character vector; Positions or names of the x and y columns in `observations` if data.frame. Default is c(1,2).
`obs.col`	numeric or character; Column name or number showing position of the observation column in the `observations`. Default is 3.
`n.obs`	numeric; Number of nearest observations to be found. Note that it cannot be larger than number of obsevrations. Default is 10.
`rm.dupl`	boolean; Remove spatial duplicates - will the spatial duplicates (nearest observations where Euclidean distance is 0) be removed from the result. Default is TRUE.
`avg`	boolean; Averages in circles - will averages in circles with different radiuses be calculated. Default is FALSE.
`increment`	numeric; Increment of radiuses for calculation of averages in circles with different radiuses. Units depends on CRS.
`range`	numeric; Maximum radius for calculation of averages in circles with different radiuses. Units depends on CRS.
`quadrant`	boolean; Nearest observations in quadrants - will nearest observation in quadrants be calculated. Default is FALSE.
`idw`	boolean; IDW prediction as predictor - will IDW predictions from `n.obs` nearest observations be calculated. Default is FALSE.
`idw.p`	numeric; Exponent parameter for IDW weights. Default is 2.

Value

data.frame object. Rows represents specific locations. First `n.obs` columns are Euclidean distances to `n.obs` nearest observations. Next `n.obs` columns are observations at `n.obs` nearest stations.

The following columns are averages in circles with different radiuses if avg is set to TRUE. The following columns are nearest observation in quadrants if direct is set to TRUE. The following columns are IDW prediction from nearest observation if idw is set to TRUE.

Note

The function can be used in any case if it is needed to find n nearest observations from given locations and distances to them.

Author(s)

Aleksandar Sekulic <asekulic@grf.bg.ac.rs>

References

Sekulić, A., Kilibarda, M., Heuvelink, G. B., Nikolić, M. & Bajat, B. Random Forest Spatial Interpolation. Remote. Sens. 12, 1687, https://doi.org/10.3390/rs12101687 (2020).

See Also

knn rfsi pred.rfsi tune.rfsi cv.rfsi

Examples

```
library(sp)
library(sf)
library(terra)
library(meteo)
# prepare data
# load observation - data.frame of mean temperatures
demo(meuse, echo=FALSE)
meuse <- meuse[complete.cases(meuse@data),]
locations = terra::rast(meuse.grid)
observations = st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
# find 5 nearest observations and distances to them (remove duplicates)
nearest_obs <- near.obs(locations = locations, # from
                        observations = observations, # to
                        obs.col = "zinc",
                        n.obs = 5, # number of nearest observations
                        rm.dupl = TRUE)
str(nearest_obs)
summary(nearest_obs)
```

near.obs.soil *Finds n nearest observations from given locations for soil mapping.*

Description

The function finds n nearest observations from given locations and at specific depth range from location min depth and creates an object of data.frame class. First n columns are Euclidean distances to n nearest locations and next n columns are observations at n nearest stations, and rows are given locations. It is based on knn function of package nabor.

Usage

```
near.obs.soil(locations,
              locations.x.y.md = c(1,2,3),
              observations,
              observations.x.y.md = c(1,2,3),
              obs.col = 4,
              n.obs = 5,
              depth.range = 0.1,
              no.obs = 'increase',
              parallel.processing = TRUE,
              pp.type = "doParallel", # "snowfall"
              cpus = detectCores()-1)
```

Arguments

`locations`	data.frame with x and y coordinates and mid depth columns, or sf-class, SpatVector-class or SpatRaster-class object; Locations (FROM) for which n nearest observations are found and distances are calculated.
`locations.x.y.md`	numeric or character vector; Positions or names of the x, y, and mid depth columns in `locations` if data.frame. Default is c(1,2,3).
`observations`	data.frame with x, y, mid depth and observation columns, or sf-class or SpatVector-class object with mid depth and observation columns; Observations (TO).
`observations.x.y.md`	numeric or character vector; positions or names of the x, y, and mid depth columns in `observations` if data.frame. Default is c(1,2,3).
`obs.col`	numeric or character; Column name or number showing position of the observation column in the `observations`. Default is 4.
`n.obs`	numeric; Number of nearest observations to be found. Note that it cannot be larger than number of obsevrations. Default is 5.
`depth.range`	numeric; Depth range for location mid depth in which to search for nearest observations. It's in the mid depth units. Default is 0.1.

`no.obs`	character; Possible values are `increase` (default) and `exactly`. If set to `increase`, in case if there is no `n.obs` observations in `depth.range` for a specific location, the `depth.range` is increased (multiplied by 2, 3, ...) until the number of observations are larger or equal to `n.obs`. If set to `exactly`, the function will raise an error when it come to the first location with no `n.obs` observations in specified `depth.range`.
`parallel.processing`	logical; If parallel processing is performed. Default is FALSE.
`pp.type`	character; Type (R package) used for parallel processing, "doParallel" (default) or "snowfall".
`cpus`	numeric; Number of processing units. Default is detectCores()-1.

Value

data.frame object. Rows represents specific locations. First `n.obs` columns are Euclidean distances to `n.obs` nearest observations. Next `n.obs` columns are observations at `n.obs` nearest stations.

Note

The function is intended for soil mapping applications.

Author(s)

Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>, Anatol Helfenstein <`anatol.helfenstein@wur.nl`>

See Also

`knn` `near.obs` `rfsi` `pred.rfsi` `tune.rfsi` `cv.rfsi`

Examples

```
library(sp)
library(sf)
library(meteo)
# prepare data
# load observation - data.frame of mean temperatures
demo(meuse, echo=FALSE)
meuse <- meuse[complete.cases(meuse@data),]
locations = st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
locations = # terra::rast(meuse.grid)
observations = st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
# find 5 nearest observations and distances to them (remove duplicates)
nearest_obs <- near.obs.soil(locations = locations, # from
                             locations.x.y.md =  c("x","y","dist"),
                             observations = observations, # to
                             observations.x.y.md= c("x","y","dist"),
                             obs.col = "zinc",
                             n.obs = 5) # number of nearest observations
str(nearest_obs)
summary(nearest_obs)
```

nlmodis20110704 *MODIS LST 8 day images image for the Netherlands ('2011-07-04')*

Description

The original 8 day MODIS LST images were also converted from Kelvin to degrees Celsius using the formula indicated in the MODIS user's manual.SpatialGridDataFrame.

Usage

```
data(nlmodis20110704)
```

Author(s)

Milan Kilibarda <kili@grf.bg.ac.rs>

References

Wan, Z., Y. Zhang, Q. Zhang, and Z.-L. Li (2004), Quality assessment and validation of the MODIS global land surface temperature, Int. J. Remote Sens., 25(1), 261-274

Examples

```
data(nlmodis20110704)
# spplot(nlmodis20110704)
```

nlmodis20110712 *MODIS LST 8 day images image for the Netherlands ('2011-07-12')*

Description

The original 8 day MODIS LST images were also converted from Kelvin to degrees Celsius using the formula indicated in the MODIS user's manual.

Usage

```
data(nlmodis20110712)
```

Author(s)

Milan Kilibarda <kili@grf.bg.ac.rs>

References

Wan, Z., Y. Zhang, Q. Zhang, and Z.-L. Li (2004), Quality assessment and validation of the MODIS global land surface temperature, Int. J. Remote Sens., 25(1), 261-274

Examples

```
data(nlmodis20110712)
# spplot(nlmodis20110712)
```

NLpol *The Netherlands border polygon from WCAB*

Description

SpatialGridDataFrame from World Country Admin Boundary Shapefile

Usage

```
data(NLpol)
```

Examples

```
data(NLpol)
##  plot(NLpol) ...
```

---

pred.rfsi *Random Forest Spatial Interpolation (RFSI) prediction*

Description

Function for spatial/spatio-temporal prediction based on Random Forest Spatial Interpolation (RFSI) model (Sekulić et al. 2020).

Usage

```
pred.rfsi(model,
          data,
          obs.col=1,
          data.staid.x.y.z = NULL,
          newdata,
          newdata.staid.x.y.z = NULL,
          z.value = NULL,
          s.crs = NA,
          newdata.s.crs=NA,
          p.crs = NA,
          output.format = "data.frame",
          cpus = detectCores()-1,
          progress = TRUE,
          soil3d = FALSE, # soil RFSI
          depth.range = 0.1, # in units of depth
          no.obs = 'increase',
          ...)
```

Arguments

`model`	ranger; An RFSI model made by rfsi function.
`data`	sf-class, sftime-class, SpatVector-class or data.frame; Contains target variable (observations) and covariates used for RFSI prediction. If data.frame object, it should have next columns: station ID (staid), longitude (x), latitude (y), 3rd component - time, depth, ... (z) of the observation, and observation value (obs).
`obs.col`	numeric or character; Column name or number showing position of the observation column in the `data`. Default is 1.
`data.staid.x.y.z`	numeric or character vector; Positions or names of the station ID (staid), longitude (x), latitude (y) and 3rd component (z) columns in data.frame object (e.g. c(1,2,3,4)). If `data` is sf-class, sftime-class, or SpatVector-class object, `data.staid.x.y.z` is used to point staid and z position. Set z position to NA (e.g. c(1,2,3,NA)) or ommit it (e.g. c(1,2,3)) for spatial interpolation. Default is NULL.
`newdata`	sf-class, sftime-class, SpatVector-class, SpatRaster-class or data.frame; Contains prediction locations and covariates used for RFSI prediction. If data.frame object, it should have next columns: prediction location ID (staid), longitude (x), latitude (y), 3rd component - time, depth, ... (z), and covariates (cov1, cov2, ...). Covariate names have to be the same as in the `model`.
`newdata.staid.x.y.z`	numeric or character vector; Positions or names of the prediction location ID (staid), longitude (x), latitude (y) and 3rd component (z) columns in data.frame `newdata` object (e.g. c(1,2,3,4)). If `newdata` is sf-class, sftime-class, SpatVector-class or SpatRaster-class object, `newdata.staid.x.y.z` is used to point staid and z position. Set z position to NA (e.g. c(1,2,3,NA)) or ommit it (e.g. c(1,2,3)) for spatial interpolation. Default is NULL.
`z.value`	vector; A vector of 3rd component - time, depth, ... (z) values if `newdata` is SpatRaster-class.
`s.crs`	st_crs or crs; Source CRS of `data`. If `data` contains crs, `s.crs` will not be used. Default is NA.
`newdata.s.crs`	st_crs or crs; Source CRS of `newdata`. If `newdata` contains crs, `newdata.s.crs` will not be used. Default is NA.
`p.crs`	st_crs or crs; Projection CRS for `data` reprojection. If NA, `s.crs` will be used for distance calculation. Note that observations should be in projection for finding nearest observations based on Eucleadean distances (see function near.obs). Default is NA.
`output.format`	character; Format of the output, data.frame (default), sf-class, sftime-class, SpatVector-class, or SpatRaster-class.
`cpus`	numeric; Number of processing units. Default is detectCores()-1.
`progress`	logical; If progress bar is shown. Default is TRUE.
`soil3d`	logical; If 3D soil modellig is performed and near.obs.soil function is used for finding n nearest observations and distances to them. In this case, z position of the `data.staid.x.y.z` points to the depth column.

`depth.range`	numeric; Depth range for location mid depth in which to search for nearest observations (see function near.obs.soil). It's in the mid depth units. Default is 0.1.
`no.obs`	character; Possible values are `increase` (default) and `exactly`. If set to `increase`, in case if there is no `n.obs` observations in `depth.range` for a specific location, the `depth.range` is increased (multiplied by 2, 3, ...) until the number of observations are larger or equal to `n.obs`. If set to `exactly`, the function will raise an error when it come to the first location with no `n.obs` observations in specified `depth.range` (see function near.obs.soil).
`...`	Further arguments passed to predict.ranger function, such as `type = "quantile"` and `quantiles = c(0.1,0.5,0.9)` for quantile regression, etc.

Value

A data.frame, sf-class, sftime-class, SpatVector-class, or SpatRaster-class object (depends on `output.format` argument) with prediction - `pred` or `quantile..X.X` (quantile regression) columns.

Author(s)

Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

Sekulić, A., Kilibarda, M., Heuvelink, G. B., Nikolić, M. & Bajat, B. Random Forest Spatial Interpolation. Remote. Sens. 12, 1687, https://doi.org/10.3390/rs12101687 (2020).

See Also

`near.obs` `rfsi` `tune.rfsi` `cv.rfsi`

Examples

```
library(ranger)
library(sp)
library(sf)
library(terra)
library(meteo)

# prepare data
demo(meuse, echo=FALSE)
meuse <- meuse[complete.cases(meuse@data),]
data = st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
# data = terra::vect(meuse)
# data.frame
# data <- as.data.frame(meuse)
# data$id = 1:nrow(data)
# data.staid.x.y.z <- c("id","x","y",NA)
fm.RFSI <- as.formula("zinc ~ dist + soil + ffreq")

# fit the RFSI model
rfsi_model <- rfsi(formula = fm.RFSI,
                   data = data, # meuse.df (use data.staid.x.y.z)
                # data.staid.x.y.z = data.staid.x.y.z, # only if class(data) == data.frame
                   n.obs = 5, # number of nearest observations
               # s.crs = st_crs(data), # nedded only if the coordinates are lon/lat (WGS84)
               # p.crs = st_crs(data), # nedded only if the coordinates are lon/lat (WGS84)
                   cpus = detectCores()-1,
                   progress = TRUE,
                   # ranger parameters
                   importance = "impurity",
                   seed = 42,
                   num.trees = 250,
                   mtry = 5,
                   splitrule = "variance",
                   min.node.size = 5,
                   sample.fraction = 0.95,
                   quantreg = FALSE)
                   # quantreg = TRUE) # for quantile regression

rfsi_model
# OOB prediction error (MSE):       47758.14
# R squared (OOB):                  0.6435869
sort(rfsi_model$variable.importance)
sum("obs" == substr(rfsi_model$forest$independent.variable.names, 1, 3))

# Make RFSI prediction
# data.frame
# newdata <- as.data.frame(meuse.grid)
# newdata$id <- 1:nrow(newdata)
# newdata <- meuse.grid
newdata <- terra::rast(meuse.grid)
class(newdata)

# prediction
rfsi_prediction <- pred.rfsi(model = rfsi_model,
                             data = data, # meuse.df (use data.staid.x.y.z)
                             obs.col = "zinc",
                             # data.staid.x.y.z = data.staid.x.y.z, # data.frame
                            newdata = newdata, # meuse.grid.df (use newdata.staid.x.y.z)
                             # newdata.staid.x.y.z = c("id", "x", "y", NA), # data.frame
                             output.format = "SpatRaster", # "sf", # "SpatVector",
                             zero.tol = 0,
                         # s.crs = st_crs(data), # meuse@proj4string, # NA # st_crs(data)
                             # newdata.s.crs = st_crs(data), # meuse@proj4string, # NA
                             # p.crs = st_crs(data), # meuse@proj4string, # NA
                             cpus = 1, # detectCores()-1,
                             progress = TRUE,
                             # type = "quantiles", # for quantile regression
                             # quantiles = c(0.1, 0.5, 0.9) # for quantile regression
)
class(rfsi_prediction)
names(rfsi_prediction)
# head(rfsi_prediction)
# plot(rfsi_prediction)
# plot(rfsi_prediction['pred'])
# plot(rfsi_prediction['quantile..0.1'])
# plot(rfsi_prediction['quantile..0.5'])
# plot(rfsi_prediction['quantile..0.9'])
```

---

pred.strk *Spatio-temporal regression kriging prediction*

Description

Function for spatio-temporal regression kriging prediction based on krigeST.

Usage

```
pred.strk(data,
          obs.col=1,
          data.staid.x.y.z = NULL,
          newdata,
          newdata.staid.x.y.z = NULL,
          z.value = NULL,
          crs = NA,
          zero.tol=0,
          reg.coef,
          vgm.model,
          sp.nmax=20,
          time.nmax=2,
          by='time',
          tiling= FALSE,
          ntiles=64,
          output.format = "STFDF",
          parallel.processing = FALSE,
          pp.type = "snowfall",
          cpus=detectCores()-1,
          computeVar=FALSE,
          progress=TRUE,
          ...)
```

Arguments

`data`	STFDF-class, STSDF-class, STIDF-class, sf-class, sftime-class, SpatVector-class or data.frame; Contains target variable (observations) and covariates in space and time used to perform STRK. If data.frame object, it should have next columns: station ID (staid), longitude (x), latitude (y), 3rd component - time, depth, ... (z) of the observation, observation value (obs), and covariates (cov1, cov2, ...). Covariate names should be the same as in the `reg.coef` (see below). If covariates are missing, overlay with `newdata` is tried. If overlay with `newdata` is not possible, then spatio-temporal ordinary kriging is performed.

`obs.col`	numeric or character; Column name or number showing position of the observation column in the `data`. Default is 1.
`data.staid.x.y.z`	numeric or character vector; Positions or names of the station ID (staid), longitude (x), latitude (y) and 3rd component - time, depth (z) columns in data.frame object (e.g. c(1,2,3,4)). If `data` is sf-class, sftime-class, or SpatVector-class object, `data.staid.x.y.z` is used to point staid and z position. Default is NULL.
`newdata`	STFDF-class, STSDF-class, STIDF-class, sf-class, sftime-class, SpatVector-class, SpatRaster-class or data.frame; Contains prediction locations and covariates used for STRK prediction. If data.frame object, it should have next columns: prediction location ID (staid), longitude (x), latitude (y), 3rd component - time, depth, ... (z), and covariates (cov1, cov2, ...). Covariate names have to be the same as in the `reg.coef` (see below).
`newdata.staid.x.y.z`	numeric or character vector; Positions or names of the prediction location ID (staid), longitude (x), latitude (y) and 3rd component (z) columns in data.frame `newdata` object (e.g. c(1,2,3,4)). If `newdata` is sf-class, sftime-class, SpatVector-class or SpatRaster-class object, `newdata.staid.x.y.z` is used to point staid and z position. Default is NULL.
`z.value`	vector; A vector of 3rd component - time, depth, ... (z) values if `newdata` is SpatRaster-class.
`crs`	st_crs or crs; Source CRS of `data` and `newdata`. If `data` or `newdata` contains crs, `crs` will not be used. Default is NA.
`zero.tol`	numeric; A distance value below (or equal to) which locations are considered as duplicates. Default is 0. See rm.dupl. Duplicates are removed to avoid singular covariance matrices in kriging.
`reg.coef`	numeric; Vector of named linear regression coefficients. Names of the coefficients (e.g. "Intercept", "temp_geo", "modis", "dem", "twi") will be used to match appropriate covariates from `data`. Coefficients for metorological variables (temperature, precipitation, etc.) can be taken from data(tregcoef) or can be specified by the user.
`vgm.model`	StVariogramModel list; Spatio-temporal variogram of regression residuals (or observations if spatio-temporal ordinary kriging). See vgmST. Spatio-temporal variogram model on residuals for metorological variables (temperature, precipitation, etc.) can be taken from data(tvgms) or can be specified by the user as a vgmST object.
`sp.nmax`	numeric; A number of spatially nearest observations that should be used for kriging predictions. If `tiling` is TRUE (see below), then is a number of spatially nearest observations that should be used for each tile. Deafult is 20.
`time.nmax`	numeric; A number of temporally nearest observations that should be used for kriging predictions Deafult is 2.
`by`	cahracter; Will foreach loop by time (default) or station. If station is set, `sp.nmax` will be used for each station prediction.
`tiling`	logical; Should simplified local kriging be used. Default is FALSE. If TRUE, area is divided in tiles and kriging calculation is done for each tile separately.

	Number of observation used per tile is defined with `sp.nmax` and `time.nmax`. If `FALSE`, temporal local kriging will be applied defined.
`ntiles`	numeric; A number of tiles for tilling. Default is 64. Ideally, each tile should contain less observations than `sp.nmax` and observations fall in neighboring tiles.
`output.format`	character; Format of the output, STFDF-class (default), STSDF-class, STIDF-class, data.frame, sf-class, sftime-class, SpatVector-class, or SpatRaster-class.
`parallel.processing`	logical; If parallel processing is performed. Default is FALSE.
`pp.type`	character; Type (`R` package) for parallel processing, "snowfall" (default) or "doParallel".
`cpus`	numeric; Number of processing units. Default is detectCores()-1.
`computeVar`	logical; If kriging variance is computed. Default is FALSE.
`progress`	logical; If progress bar is shown. Default is TRUE.
`...`	Further arguments passed to krigeST function.

Value

A STFDF-class, STSDF-class, STIDF-class, data.frame, sf-class, sftime-class, SpatVector-class, or SpatRaster-class object (depends on `output.format` argument), with columns (elements):

`pred`	Predictions.
`tlm`	Trend.
`var`	Kriging variance, if `computeVar=TRUE`.

Author(s)

Milan Kilibarda <`kili@grf.bg.ac.rs`>, Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

Kilibarda, M., T. Hengl, G. B. M. Heuvelink, B. Graeler, E. Pebesma, M. Percec Tadic, and B. Bajat (2014), Spatio-temporal interpolation of daily temperatures for global land areas at 1 km resolution, J. Geophys. Res. Atmos., 119, 2294-2313, doi:10.1002/2013JD020803.

See Also

`tregcoef` `tvgms` `regdata` `meteo2STFDF` `tgeom2STFDF`

Examples

```
library(sp)
library(spacetime)
library(sf)
library(gstat)
library(plyr)
# prepare data
# load observation - data.frame of mean temperatures
# preparing data
data(dtempc)
data(stations)
data(regdata) # covariates, made by mete2STFDF function
regdata@sp@proj4string <- CRS('+proj=longlat +datum=WGS84')

lonmin=18 ;lonmax=22.5 ; latmin=40 ;latmax=46
serbia = point.in.polygon(stations$lon, stations$lat, c(lonmin,lonmax,lonmax,lonmin),
                          c(latmin,latmin,latmax,latmax))
st = stations[ serbia!=0, ] # stations in Serbia approx.
obs.staid.time = c("staid", "time")
stations.staid.lon.lat = c(1,2,3)
crs = CRS('+proj=longlat +datum=WGS84')
delta = NULL

## Not run:
# create STFDF
stfdf <- meteo2STFDF(obs = dtempc,
                     stations = st,
                     crs = crs)

# Calculate prediction of mean temperatures for "2011-07-05" and "2011-07-06"
# global model is used for regression and variogram
# load precalculated variograms
data(tvgms) # ST variogram models
data(tregcoef) # MLR coefficients

### Example with STFDF and without parallel processing
results <- pred.strk(data = stfdf, # observations
                     newdata = regdata, # prediction locations with covariates
                     # newdata = regdata[,2,drop=FALSE], # for one day only
               output.format = "STFDF", # data.frame | sf | sftime | SpatVector | SpatRaster
                     reg.coef = tregcoef[[1]], # MLR coefficients
                     vgm.model = tvgms[[1]], # STRK variogram model
                     sp.nmax = 20,
                     time.nmax = 2,
                     computeVar=TRUE
)
class(results)
# plot prediction
# results@sp=as(results@sp,'SpatialPixelsDataFrame')
# stplot(results[,,"pred", drop= FALSE], col.regions=bpy.colors())
# stplot(results[,,"var", drop= FALSE], col.regions=bpy.colors())

## End(Not run)

## Not run:
### Example with data.frames and parallel processing - SpatRaster output
library(terra)
library(doParallel)
# create data.frame
stfdf.df <- join(dtempc, st)
summary(stfdf.df)
```

```
regdata.df <- as.data.frame(regdata)
results <- pred.strk(data = stfdf.df,
                     obs.col = 3,
                     data.staid.x.y.z = c(1,4,5,2),
                     newdata = regdata.df,
                     newdata.staid.x.y.z = c(3,1,2,4),
                     crs = CRS("EPSG:4326"),
               output.format = "SpatRaster", # STFDF |data.frame | sf | sftime | SpatVector
                     reg.coef = tregcoef[[1]],
                     vgm.model = tvgms[[1]],
                     sp.nmax = 20,
                     time.nmax = 2,
                     parallel.processing = TRUE,
                     pp.type = "doParallel", # "snowfall"
                     cpus = 1, # detectCores()-1,
                     computeVar = TRUE,
                     progress = TRUE
)
# plot prediction
# plot(results$`2011-07-06`[["pred"]])
# plot(results$`2011-07-06`[["var"]])

## End(Not run)
```

regdata *Dynamic and static covariates for spatio-temporal regression kriging*

Description

Dynamic and static covariates for spatio-temporal regression kriging of STFDF-class. The `regdata` contains geometrical temperature trend, MODIS LST 8-day splined at daily resolution, elevation and topographic wetness index.

Usage

```
data(regdata)
```

Format

The `regdata` contains the following dynamic and static covariates:

- `regdata$temp_geo` numeric; geometrical temperature trend for mean temperature, calculated with `tgeom2STFDF` ; from 2011-07-05 to 2011-07-09, in degree Celsius
- `regdata$modis` numeric; MODIS LST 8-day splined at daily resolution, missing pixels are filtered by spatial splines and 8-day values are splined at daily level; from 2011-07-05 to 2011-07-09, in degree Celsius
- `regdata@sp$dem` numeric; elevation data obtained from Worldgrids (depricated)
- `regdata@sp$twi` numeric; SAGA Topographic Wetness Index (TWI) from Worldgrids (depricated)

Author(s)

Milan Kilibarda <`kili@grf.bg.ac.rs`>

Examples

```
data(regdata)
str(regdata)
library(sp) # spplot
library(spacetime) # stplot

# stplot(regdata[,,'modis']) # plot modis data
# spplot(regdata@sp,zcol='twi', col.regions = bpy.colors() ) # plot TWI
# spplot(regdata@sp,zcol='dem', col.regions = bpy.colors() ) # plot dem
```

`rfillspgaps` *Close gaps of a grid or raster Layer data*

Description

The function close gaps of a raster data by using IDW.

Usage

```
rfillspgaps(rasterLayer,
            maskPol=NULL,
            nmax =50,
            zcol=1,
            ...)
```

Arguments

`rasterLayer`	SpatRaster-class, RasterLayer-class, SpatialGrid-class or SpatialPixels-class; Raster that contains NAs.
`maskPol`	sf-class, SpatVector-class, SpatialPolygons or SpatialPolygonsDataFrame; Area of interest to spatially fill `rasterLayer` missing values and to mask `rasterLayer`.
`nmax`	see krige, idw function.
`zcol`	`integer` or `character`; variable column name or number showing position of a variable in `rasterLayer` to be interpolated.
`...`	arguments passed to krige, idw function.

Value

raster object with NA replaced using IDW in `rasterLayer` format.

Author(s)

Milan Kilibarda <`kili@grf.bg.ac.rs`>

References

Kilibarda, M., T. Hengl, G. B. M. Heuvelink, B. Graeler, E. Pebesma, M. Percec Tadic, and B. Bajat (2014), Spatio-temporal interpolation of daily temperatures for global land areas at 1 km resolution, J. Geophys. Res. Atmos., 119, 2294-2313, doi:10.1002/2013JD020803;

Kilibarda M., M. Percec Tadic, T. Hengl, J. Lukovic, B. Bajat - Spatial Statistics (2015), Global geographic and feature space coverage of temperature data in the context of spatio-temporal interpolation, doi:10.1016/j.spasta.2015.04.005.

See Also

`rfilltimegaps` `pred.strk`

Examples

```
library(terra)
data(nlmodis20110712)
data(NLpol)

nlmodis20110712 <- terra::rast(nlmodis20110712)
# SpaVector
NLpol = vect(NLpol)
crs(NLpol) <- "epsg:4326"
# # sf
# NLpol <- st_as_sf(NLpol) #, crs = st_crs(4326))

# plot(nlmodis20110712)
# fill spatial gaps
r=rfillspgaps(nlmodis20110712,NLpol)
# plot(r)
```

`rfilltimegaps` *Disaggregation in the time dimension through the use of splines for each pixel*

Description

The function creates an object of STFDF-class class, spatio-temporal data with full space-time grid, from another STFDF-class and fills attribute data for missing values in time using splines.

Usage

```
rfilltimegaps(stfdf,
               tunits="day",
               attrname=1,
               ...)
```

Arguments

`stfdf`	STFDF-class; object with time information of minimum length 2, and gap in time dimension.
`tunits`	`character`; increment of the sequence used to generete time infromation for temporal gap. See 'Details'.
`attrname`	`integer` or `character`; varible from STFDF-class to be splined in time.
`...`	arguments passed to splinefun, function spline.

Details

`tunits` can be specified in several ways:

- A number, taken to be in seconds
- A object of class difftime
- A character string, containing one of `"sec"`, `"min"`, `"hour"`, `"day"`, `"DSTday"`, `"week"`, `"month"`, `"quarter"` or `"year"`. This can optionally be preceded by a (positive or negative) integer and a space, or followed by `"s"`

The difference between `"day"` and `"DSTday"` is that the former ignores changes to/from daylight savings time and the latter takes the same clock time each day. (`"week"` ignores DST (it is a period of 144 hours), but `"7 DSTdays"`) can be used as an alternative. `"month"` and `"year"` allow for DST.)

Value

STFDF-class object with filled temporal gaps.

Author(s)

Milan Kilibarda `<kili@grf.bg.ac.rs>`

References

Kilibarda, M., T. Hengl, G. B. M. Heuvelink, B. Graeler, E. Pebesma, M. Percec Tadic, and B. Bajat (2014), Spatio-temporal interpolation of daily temperatures for global land areas at 1 km resolution, J. Geophys. Res. Atmos., 119, 2294-2313, doi:10.1002/2013JD020803;

Kilibarda M., M. Percec Tadic, T. Hengl, J. Lukovic, B. Bajat - Spatial Statistics (2015), Global geographic and feature space coverage of temperature data in the context of spatio-temporal interpolation, doi:10.1016/j.spasta.2015.04.005.

See Also

`rfillspgaps` `pred.strk`

Examples

```
data(nlmodis20110704)
data(nlmodis20110712)
data(NLpol)

# fill spatial gaps
library(raster)
NLpol@proj4string <- nlmodis20110704@proj4string

## Not run:
nlmodis20110704 <- rfillspgaps(nlmodis20110704,NLpol)
nlmodis20110712 <- rfillspgaps(nlmodis20110712,NLpol)

nlmodis20110704 <- as(nlmodis20110704,"SpatialPixelsDataFrame")
names(nlmodis20110704)='m1'
nlmodis20110712 <- as(nlmodis20110712,"SpatialPixelsDataFrame")
names(nlmodis20110712)='m2'

nlmodis20110704@data <- cbind(nlmodis20110704@data, nlmodis20110712@data)

df<-reshape(nlmodis20110704@data , varying=list(1:2), v.names="modis",direction="long",
          times=as.Date(c('2011-07-04','2011-07-12')), ids=1:dim(nlmodis20110704)[1])

library(spacetime)
stMODIS<- STFDF(as( nlmodis20110704, "SpatialPixels"),
                time= as.Date(c('2011-07-04','2011-07-12')),
                data.frame(modis=df[,'modis']))

# stplot(stMODIS, col.regions=bpy.colors())
stMODIS <- rfilltimegaps(stMODIS)
# stplot(stMODIS, col.regions=bpy.colors())

## End(Not run)
```

`rfsi` *Random Forest Spatial Interpolation (RFSI) model*

Description

Function for creation of Random Forest Spatial Interpolation (RFSI) model (Sekulić et al. 2020). Besides environmental covariates, RFSI uses additional spatial covariates: (1) observations at n nearest locations and (2) distances to them, in order to include spatial context into the random forest.

Usage

```
rfsi(formula,
     data,
     data.staid.x.y.z = NULL,
     n.obs = 5,
     avg = FALSE,
     increment = 10000,
     range = 50000,
     quadrant = FALSE,
     use.idw = FALSE,
     idw.p = 2,
     s.crs = NA,
     p.crs = NA,
     cpus = detectCores()-1,
     progress = TRUE,
     soil3d = FALSE,
     depth.range = 0.1,
     no.obs = 'increase',
     ...)
```

Arguments

`formula`	formula; Formula for specifying target variable and covariates (without nearest observations and distances to them). If `z~1`, an RFSI model using only nearest obsevrations and distances to them as covariates will be made.
`data`	sf-class, sftime-class, SpatVector-class or data.frame; Contains target variable (observations) and covariates used for making an RFSI model. If data.frame object, it should have next columns: station ID (staid), longitude (x), latitude (y), 3rd component - time, depth, ... (z) of the observation, observation value (obs) and covariates (cov1, cov2, ...). If covariates are missing, the RFSI model using only nearest obsevrations and distances to them as covariates (`formula=z~1`) will be made.
`data.staid.x.y.z`	numeric or character vector; Positions or names of the station ID (staid), longitude (x), latitude (y) and 3rd component (z) columns in data.frame object (e.g. c(1,2,3,4)). If `data` is sf-class, sftime-class, or SpatVector-class object, `data.staid.x.y.z` is used to point staid and z position. Set z position to NA (e.g. c(1,2,3,NA)) or ommit it (e.g. c(1,2,3)) for spatial interpolation. Default is NULL.
`n.obs`	numeric; Number of nearest observations to be used as covariates in RFSI model (see function near.obs). Note that it cannot be larger than number of obsevrations. Default is 5.
`avg`	boolean; Averages in circles covariate - will averages in circles with different radiuses be calculated (see function near.obs). Default is FALSE.
`increment`	numeric; Increment of radiuses for calculation of averages in circles with different radiuses (see function near.obs). Units depends on CRS.
`range`	numeric; Maximum radius for calculation of averages in circles with different radiuses (see function near.obs). Units depends on CRS.
`quadrant`	boolean; Nearest observations in quadrants covariate - will nearest observation in quadrants be calculated (see function near.obs). Default is FALSE.

`use.idw`	boolean; IDW prediction as covariate - will IDW predictions from `n.obs` nearest observations be calculated (see function near.obs). Default is FALSE.
`idw.p`	numeric; Exponent parameter for IDW weights (see function near.obs). Default is 2.
`s.crs`	st_crs or crs; Source CRS of `data`. If `data` contains crs, `s.crs` will be overwritten. Default is NA.
`p.crs`	st_crs or crs; Projection CRS for `data` reprojection. If NA, `s.crs` will be used for distance calculation. Note that observations should be in projection for finding nearest observations based on Eucleadean distances (see function near.obs). Default is NA.
`cpus`	numeric; Number of processing units. Default is detectCores()-1.
`progress`	logical; If progress bar is shown. Default is TRUE.
`soil3d`	logical; If 3D soil modellig is performed and near.obs.soil function is used for finding n nearest observations and distances to them. In this case, z position of the `data.staid.x.y.z` points to the depth column.
`depth.range`	numeric; Depth range for location mid depth in which to search for nearest observations (see function near.obs.soil). It's in the mid depth units. Default is 0.1.
`no.obs`	character; Possible values are `increase` (default) and `exactly`. If set to `increase`, in case if there is no `n.obs` observations in `depth.range` for a specific location, the `depth.range` is increased (multiplied by 2, 3, ...) until the number of observations are larger or equal to `n.obs`. If set to `exactly`, the function will raise an error when it come to the first location with no `n.obs` observations in specified `depth.range` (see function near.obs.soil).
`...`	Further arguments passed to ranger, such as `quantreg`, `importance`, etc.

Value

RFSI model of class ranger.

Note

Observations should be in projection for finding nearest observations based on Eucleadean distances (see function near.obs). If crs is not specified in the `data` object or through the `s.crs` parameter, the coordinates will be used as they are in projection. Use `s.crs` and `p.crs` if the coordinates of the `data` object are in lon/lat (WGS84).

Author(s)

Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

Sekulić, A., Kilibarda, M., Heuvelink, G. B., Nikolić, M. & Bajat, B. Random Forest Spatial Interpolation. Remote. Sens. 12, 1687, https://doi.org/10.3390/rs12101687 (2020).

See Also

near.obs pred.rfsi tune.rfsi cv.rfsi

Examples

```
library(ranger)
library(sp)
library(sf)
library(terra)
library(meteo)
# prepare data
demo(meuse, echo=FALSE)
meuse <- meuse[complete.cases(meuse@data),]
data = st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
# data = terra::vect(meuse)
# data.frame
# data <- as.data.frame(meuse)
# data$id = 1:nrow(data)
# data.staid.x.y.z <- c(15,"x","y",NA)
fm.RFSI <- as.formula("zinc ~ dist + soil + ffreq")

# fit the RFSI model
rfsi_model <- rfsi(formula = fm.RFSI,
                   data = data, # meuse.df (use data.staid.x.y.z)
                # data.staid.x.y.z = data.staid.x.y.z, # only if class(data) == data.frame
                   n.obs = 5, # number of nearest observations
               # s.crs = st_crs(data), # nedded only if the coordinates are lon/lat (WGS84)
               # p.crs = st_crs(data), # nedded only if the coordinates are lon/lat (WGS84)
                   cpus = detectCores()-1,
                   progress = TRUE,
                   # ranger parameters
                   importance = "impurity",
                   seed = 42,
                   num.trees = 250,
                   mtry = 5,
                   splitrule = "variance",
                   min.node.size = 5,
                   sample.fraction = 0.95,
                   quantreg = FALSE)

rfsi_model
# OOB prediction error (MSE):       47758.14
# R squared (OOB):                  0.6435869
sort(rfsi_model$variable.importance)
sum("obs" == substr(rfsi_model$forest$independent.variable.names, 1, 3))
```

---

rm.dupl *Find point pairs with equal spatial coordinates from STFDF-class object.*

Description

This function finds point pairs with equal spatial coordinates from STFDF-class object and remove locations with less observations.

Usage

```
rm.dupl(obj, zcol = 1, zero.tol = 0)
```

Arguments

`obj`	STFDF-class object
`zcol`	variable column name, or column number, from obj@data
`zero.tol`	distance values less than or equal to this threshold value are considered as duplicates; units are those of the coordinates for projected data or unknown projection, or km if coordinates are defined to be longitute/latitude

Value

STFDF-class object with removed duplicate locations. Stations with less observation is removed, if number of observation is the same for two stations the first is removed.

Author(s)

Milan Kilibarda <`kili@grf.bg.ac.rs`>

See Also

`tgeom2STFDF`, `pred.strk`

Examples

```
library(sp)
# load observation - data frame
data(dtempc)
# load stations - data frame
data(stations)

str(dtempc)
str(stations)

## Not run:
# create STFDF - from 2 data frames
temp <- meteo2STFDF(dtempc,
                    stations,
                    crs = CRS('+proj=longlat +datum=WGS84'))

nrow(temp@sp) # number of stations before removing dupl.

temp2 <-rm.dupl(temp, zcol = 1, zero.tol = 50) # 50 km
nrow(temp2@sp) # number of stations after
```

```
## End(Not run)
```

`stations` *Data frame containing stations' information*

Description

Data frame containing stations' information of merged daily observations from the Global Surface Summary of Day (GSOD) with European Climate Assessment & Data set (ECA&D) for the month July 2011.

Usage

```
data(stations)
```

Format

The `stations` contains the following columns:

`staid` character; station ID from GSOD or ECA&D data set

`lon` numeric; longitude coordinate

`lat` numeric; longitude coordinate

`elev_1m` numeric; elevation derived from station metadata in m

`data_source` Factor; data source, GSOD or ECA&D

`station_name` character; station name

Author(s)

Milan Kilibarda and Tomislav Hengl

References

- Global Surface Summary of the day data (ftp://ftp.ncdc.noaa.gov/pub/data/gsod/)
- European Climate Assessment & Dataset (https://www.ecad.eu/dailydata/predefinedseries.php)

Examples

```
## load data:
data(stations)
str(stations)
library(sp)
coordinates(stations) <-~ lon +lat
stations@proj4string <-CRS('+proj=longlat +datum=WGS84')
# plot(stations)
```

`stations_ogimet` *Data frame containing stations' information from the OGIMET service for Serbian territory*

Description

Data frame containing stations' information of daily observations from the OGIMET service for the year 2019 for Serbian territory.

Usage

```
data(stations_ogimet)
```

Format

The `dtempc_ogimet` contains the following columns:

`staid` character; station ID from OGIMET

`name` character; station name

`lon` numeric; Longitude

`lat` numeric; Latitude

`elevation` numeric; Hight

`dem` numeric; Digital Elevation Model (DEM) in meters

`twi` numeric; Topographic Wetness Index (TWI)

Author(s)

Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

- OGIMET service (`https://www.ogimet.com/`)

Examples

```
## load data:
data(stations_ogimet)
str(stations)
library(sp)
coordinates(stations) <-~ lon +lat
stations@proj4string <-CRS('+proj=longlat +datum=WGS84')
# plot(stations)
```

`temp_geom` *Calculate geometrical temperature trend*

Description

Calculate geometrical temperature trend for mean, minimum or maximum temperature.

Usage

```
temp_geom(day,
          fi,
          variable="mean",
          ab = NULL)
```

Arguments

`day`	integer; Day of the year (from 1 to 366). Single value or vector of days of the year (only if `fi` is single value).
`fi`	numeric; Latitude. Single value or vector of latitudes (only if `day` is single value).
`variable`	character; Geometrical temperature trend calculated for mean, minimum or maximum temperature; Possible values are `'mean'`, `'min'` or `'max'`. `'mean'` is default.
`ab`	Predefined coefficients to be used instead of incorporated.

Value

A numerical single value or `vector` with calculated geometrical temperature trend.

Author(s)

Milan Kilibarda <`kili@grf.bg.ac.rs`>, Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

Kilibarda, M., T. Hengl, G. B. M. Heuvelink, B. Graeler, E. Pebesma, M. Percec Tadic, and B. Bajat (2014), Spatio-temporal interpolation of daily temperatures for global land areas at 1 km resolution, J. Geophys. Res. Atmos., 119, 2294-2313, doi:10.1002/2013JD020803.

Examples

```
tgeom <- temp_geom(day = 1,
         fi = 45.33,
         variable="mean")

tgeom_vect <- temp_geom(day = 1:365,
         fi = 45.33,
         variable="mean")

tgeom_vect2 <- temp_geom(day = 1,
         fi = seq(35, 45, 0.5),
         variable="mean")
```

tgeom2STFDF — *Calculate geometrical temperature trend*

Description

Calculate geometrical temperature trend for mean, minimum or maximum temperature.

Usage

```
tgeom2STFDF(grid,
            time,
            variable = "mean",
            ab=NULL)
```

Arguments

`grid`	Spatial-class (Points, Grid or Pixels), sf-class, SpatVector-class, SpatRaster-class; Locations for which gtt is calculated with associated coordinate reference systems (CRS-class). If CRS is not defined longitude latitude is assumed.
`time`	date or datetime; Object holding time information, reasonably it is day (calendar date), or vector of days.
`variable`	character; Geometrical temperature trend calculated for mean, minimum or maximum temperature; Possible values are `'mean'`, `'min'` or `'max'`. `'mean'` is default.
`ab`	Predefined coefficients to be used instead of incorporated.

Value

STFDF-class object with calculated `temp_geo` geometrical temperature trend. The calculated values are stored in `obj@data` slot.

Author(s)

Milan Kilibarda <`kili@grf.bg.ac.rs`>, Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

Kilibarda, M., T. Hengl, G. B. M. Heuvelink, B. Graeler, E. Pebesma, M. Percec Tadic, and B. Bajat (2014), Spatio-temporal interpolation of daily temperatures for global land areas at 1 km resolution, J. Geophys. Res. Atmos., 119, 2294-2313, doi:10.1002/2013JD020803.

Examples

```
library(sp)
library(spacetime)
## create one point from lon lat
pos <- SpatialPoints(coords = cbind(19.22,45.33))
## temp_geom for 1st Jan 2011
tg1 <- tgeom2STFDF(pos,as.POSIXct("2011-01-01") )
tg1

## temp_geom for the 2011 at pos location
tg365<- tgeom2STFDF(pos,time = seq(as.POSIXct("2011-01-01"), as.POSIXct("2011-12-31"),
                    by="day") )
# stplot(tg365, mode='ts')

data(regdata)
## DEM and TWI data for Serbia at 1 km resolution
# str(regdata@sp)
# spplot(regdata@sp, zcol='dem', col.regions=bpy.colors() )

## temp_geom for Serbia 1st and 2nd Jully 2011
tgSrb<- tgeom2STFDF(regdata@sp,time = seq(as.POSIXct("2011-07-01"),
                    as.POSIXct("2011-07-02"), by="day") )

## temp_geom for "2011-07-01" , "2011-07-02"
# stplot(tgSrb, col.regions = bpy.colors() )
```

`tiling` *Tiling raster or Spatial-class Grid or Pixels object*

Description

Tiling raster or Spatial-class Grid or Pixels (data frame) object to smaller parts with optional overlap.

Usage

```
tiling(rast,
       tilesize=500,
       overlapping=50,
       aspoints= NA,
       asfiles=FALSE,
       tilename="tile",
       tiles_folder=paste(getwd(),'tiles',sep='/'),
       parallel.processing=FALSE,
       cpus=6,
       ...)
```

Arguments

`rast`	SpatRaster, SpatialPixels* object, SpatialGrid* object or file path to raster object stored on the disk (can be read via rast), for more details see SpatRaster. The resolution of the raster should be the same in x and y direction.
`tilesize`	integer or vector; tile size in number of cells. Can be a vector of tilesize in x and y direction. Total number of tile cells is `tilesize[1]` x `tilesize[2]`.
`overlapping`	integer or vector; overlapping in number of cells. Can be a vector of overlapping in x and y direction.
`aspoints`	character; Posiible values are `sf`, `terra` `sp`. If specified, tiles are returned in form of points as sf-class, SpatVector or SpatialPointsDataFrame.
`asfiles`	boolean; if TRUE tiles are stored on local drive as raster objects.
`tilename`	character; prefix given to file names
`tiles_folder`	character; destination folder where tiles will be stored. If doesn't exist, the folder will be created.
`parallel.processing`	boolean; if TRUE parralel processing is performed via snowfall-calculation, sfLapply function.
`cpus`	integer; number of proccesing units.
`...`	character; additional arguments for for writing files, see writeRaster.

Value

The list of tiles in SpatRaster-class format or in sf-class, SpatVector or SpatialPointsDataFrame format if `aspoints=TRUE`.

Author(s)

Milan Kilibarda <`kili@grf.bg.ac.rs`>

See Also

`pred.strk`

Examples

```
library(sp)
demo(meuse, echo=FALSE)
rast <- terra::rast(meuse.grid[, "dist"])

# tiling dem in tiles 250x250 with 25 cells overlap
tiles = tiling(rast,
               tilesize=20,
               overlapping=5,
               aspoints=TRUE)
# number of tiles
length(tiles)
```

```
# plot(rast)
# plot(tiles[[1]] , pch='-' ,col ='green', add=T)
# plot(tiles[[2]], pch='.', add=T)

str(tiles[[1]])
```

tregcoef — *Multiple linear regression coefficients for global and local daily air temperatures*

Description

Multiple linear regression coefficients for mean, minimum, maximum daily temperature on geometric temperature trend (GTT), MODIS LST, digital elevation model (DEM) and topographic wetness index (TWI). The models are computed from GSOD, ECA&D, GHCN-Daily or local meteorological stations.

Usage

```
data(tregcoef)
```

Format

A list of 9 multiple linear regression coefficients for daily air temperatures.

`tmeanGSODECAD` Multiple linear regression coefficients of global mean daily temperature on GTT, MODIS LST, DEM and TWI. Data used: GSOD and ECA&D

`tmeanGSODECAD_noMODIS` Multiple linear regression coefficients of global mean daily temperature on GTT, DEM, and TWI. Data used: GSOD and ECA&D

`tminGSODECAD` Multiple linear regression coefficients of global minimum daily temperature on GTT, MODIS LST, DEM, and TWI. Data used: GSOD and ECA&D

`tminGHCND` Multiple linear regression coefficients of global minimum daily temperature on GTT, MODIS LST, DEM, and TWI. Data used: GHCN-Daily

`tminGSODECAD_noMODIS` Multiple linear regression coefficients of global minimum daily temperature on GTT, DEM, and TWI. Data used: GSOD and ECA&D

`tmaxGSODECAD` Multiple linear regression coefficients of global maximum daily temperature on GTT, MODIS LST, DEM, and TWI. Data used: GSOD and ECA&D

`tmaxGHCND` Multiple linear regression coefficients of global maximum daily temperature on GTT, MODIS LST, DEM, and TWI. Data used: GHCN-Daily

`tmaxGSODECAD_noMODIS` Multiple linear regression coefficients of global maximum daily temperature on GTT, DEM, and TWI. Data used: GSOD and ECA&D

`tmeanHR` Multiple linear regression coefficients of Croatian mean daily temperature on GTT, DEM, and TWI. Data used: Croatian mean daily temperature dataset for the year 2008 (Croatian national meteorological network)

Author(s)

Milan Kilibarda <kili@grf.bg.ac.rs>, Aleksandar Sekulic <asekulic@grf.bg.ac.rs>

References

Kilibarda, M., T. Hengl, G. B. M. Heuvelink, B. Graeler, E. Pebesma, M. Percec Tadic, and B. Bajat (2014), Spatio-temporal interpolation of daily temperatures for global land areas at 1 km resolution, J. Geophys. Res. Atmos., 119, 2294-2313, doi:10.1002/2013JD020803.

Examples

```
data(tregcoef)
tregcoef[[1]] # model for mean daily temp.
```

`tune.rfsi` *Tuning of Random Forest Spatial Interpolation (RFSI) model*

Description

Function for tuning of Random Forest Spatial Interpolation (RFSI) model using k-fold leave-location-out cross-validation (Sekulić et al. 2020).

Usage

```
tune.rfsi(formula,
          data,
          data.staid.x.y.z = NULL,
          use.idw = FALSE,
          s.crs = NA,
          p.crs = NA,
          tgrid,
          tgrid.n=10,
          tune.type = "LLO",
          k = 5,
          seed=42,
          folds,
          acc.metric,
          fit.final.model=TRUE,
          cpus = detectCores()-1,
          progress = TRUE,
          soil3d = FALSE,
          no.obs = 'increase',
          ...)
```

Arguments

`formula`	formula; Formula for specifying target variable and covariates (without nearest observations and distances to them). If `z~1`, an RFSI model using only nearest obsevrations and distances to them as covariates will be tuned.
`data`	sf-class, sftime-class, SpatVector-class or data.frame; Contains target variable (observations) and covariates used for making an RFSI model. If data.frame object, it should have next columns: station ID (staid), longitude (x), latitude (y), 3rd component - time, depth, ... (z) of the observation, observation value (obs) and covariates (cov1, cov2, ...). If covariates are missing, the RFSI model using only nearest obsevrations and distances to them as covariates (`formula=z~1`) will be tuned.
`data.staid.x.y.z`	numeric or character vector; Positions or names of the station ID (staid), longitude (x), latitude (y) and 3rd component (z) columns in data.frame object (e.g. c(1,2,3,4)). If `data` is sf-class, sftime-class, or SpatVector-class object, `data.staid.x.y.z` is used to point staid and z position. Set z position to NA (e.g. c(1,2,3,NA)) or ommit it (e.g. c(1,2,3)) for spatial interpolation. Default is NULL.
`use.idw`	boolean; IDW prediction as covariate - will IDW predictions from `n.obs` nearest observations be calculated and tuned (see function near.obs). Default is FALSE.
`s.crs`	st_crs or crs; Source CRS of `data`. If `data` contains crs, `s.crs` will be overwritten. Default is NA.
`p.crs`	st_crs or crs; Projection CRS for `data` reprojection. If NA, `s.crs` will be used for distance calculation. Note that observations should be in projection for finding nearest observations based on Eucleadean distances (see function near.obs). Default is NA.
`tgrid`	data.frame; Possible tuning parameters. The column names are same as the tuning parameters. Possible tuning parameters are: `n.obs`, `num.trees`, `mtry`, `min.node.size`, `sample.fraction`, `splirule`, `idw.p`, and `depth.range`.
`tgrid.n`	numeric; Number of randomly chosen `tgrid` combinations for tuning of RFSI. If larger than `tgrid`, will be set to `length(tgrid)`
`tune.type`	character; Type of cross-validation: leave-location-out ("LLO"), leave-time-out ("LTO") - TO DO, and leave-location-time-out ("LLTO") - TO DO. Default is "LLO".
`k`	numeric; Number of random folds that will be created with CreateSpacetimeFolds function if `folds` is column. Default is 5.
`seed`	numeric; Random seed that will be used to generate folds with CreateSpacetimeFolds function.
`folds`	numeric or character vector or value; Showing folds column (if value) or rows (vector) of `data` observations used for cross-validation. If missing, will be created with CreateSpacetimeFolds function.
`acc.metric`	character; Accuracy metric that will be used as a criteria for choosing an optimal RFSI model. Possible values for regression: "ME", "MAE", "NMAE", "RMSE" (default), "NRMSE", "R2", "CCC". Possible values for classification: "Accuracy","Kappa" (default), "AccuracyLower", "AccuracyUpper", "AccuracyNull", "AccuracyPValue", "McnemarPValue".

`fit.final.model`	boolean; Fit the final RFSI model. Defailt is TRUE.
`cpus`	numeric; Number of processing units. Default is detectCores()-1.
`progress`	logical; If progress bar is shown. Default is TRUE.
`soil3d`	logical; If 3D soil modellig is performed and near.obs.soil function is used for finding n nearest observations and distances to them. In this case, z position of the `data.staid.x.y.z` points to the depth column.
`no.obs`	character; Possible values are `increase` (default) and `exactly`. If set to `increase`, in case if there is no `n.obs` observations in `depth.range` for a specific location, the `depth.range` is increased (multiplied by 2, 3, ...) until the number of observations are larger or equal to `n.obs`. If set to `exactly`, the function will raise an error when it come to the first location with no `n.obs` observations in specified `depth.range` (see function near.obs.soil).
`...`	Further arguments passed to ranger.

Value

A list with elements:

`combinations`	data.frame; All tuned parameter combinations with chosen accuracy metric value.
`tuned.parameters`	numeric vector; Tuned parameters with chosen accuracy metric value.
`final.model`	ranger; Final RFSI model (if `fit.final.model=TRUE`).

Author(s)

Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

Sekulić, A., Kilibarda, M., Heuvelink, G. B., Nikolić, M. & Bajat, B. Random Forest Spatial Interpolation. Remote. Sens. 12, 1687, https://doi.org/10.3390/rs12101687 (2020).

See Also

`near.obs` `rfsi` `pred.rfsi` `cv.rfsi`

Examples

```
library(CAST)
library(doParallel)
library(ranger)
library(sp)
library(sf)
library(terra)
library(meteo)

# prepare data
demo(meuse, echo=FALSE)
```

```
meuse <- meuse[complete.cases(meuse@data),]
data = st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")
# data = terra::vect(meuse)
# data.frame
# data <- as.data.frame(meuse)
# data$id = 1:nrow(data)
# data.staid.x.y.z <- c(15,"x","y",NA)
fm.RFSI <- as.formula("zinc ~ dist + soil + ffreq")

# making tgrid
n.obs <- 1:6
min.node.size <- 2:10
sample.fraction <- seq(1, 0.632, -0.05) # 0.632 without / 1 with replacement
splitrule <- "variance"
ntree <- 250 # 500
mtry <- 3:(2+2*max(n.obs))
tgrid = expand.grid(min.node.size=min.node.size, num.trees=ntree,
                    mtry=mtry, n.obs=n.obs, sample.fraction=sample.fraction)

## Not run:
# Tune RFSI model
rfsi_tuned <- tune.rfsi(formula = fm.RFSI,
                        data = data,
                        # data.staid.x.y.z = data.staid.x.y.z, # data.frame
                        # s.crs = st_crs(data),
                        # p.crs = st_crs(data),
                        tgrid = tgrid, # combinations for tuning
                      tgrid.n = 20, # number of randomly selected combinations from tgrid
                        tune.type = "LLO", # Leave-Location-Out CV
                        k = 5, # number of folds
                        seed = 42,
                        acc.metric = "RMSE", # R2, CCC, MAE
                        fit.final.model = TRUE,
                        cpus = detectCores()-1,
                        progress = TRUE,
                        importance = "impurity") # ranger parameter

rfsi_tuned$combinations
rfsi_tuned$tuned.parameters
# min.node.size num.trees mtry n.obs sample.fraction     RMSE
# 3701             3       250    6     5            0.75 222.6752
rfsi_tuned$final.model
# OOB prediction error (MSE):       46666.51
# R squared (OOB):                  0.6517336

## End(Not run)
```

tvgms *Spatio-temporal variogram models for global and local daily air temperatures*

Description

Variograms of residuals from multiple linear regression of mean, minimum, maximum daily temperatures on geometric temperature trend (GTT), MODIS LST, digital elevation model (DEM) and topographic wetness index (TWI). The models is computed from GSOD, ECA&D, GHCN-Daily or local meteorological stations. The obtained global or local models for mean, minimum, and maximum temperature can be used to produce gridded images of daily temperatures at high spatial and temporal resolution.

Usage

```
data(tvgms)
```

Format

A list of 9 space-time sum-metric models for daily air temperatures, units: space km, time days.

`tmeanGSODECAD` Variogram for residuals from multiple linear regression of global mean daily temperature on GTT, MODIS LST, DEM, and TWI. D used: GSOD and ECA&D

`tmeanGSODECAD_noMODIS` Variogram for residuals from multiple linear regression of global mean daily temperature on GTT, DEM, and TWI. Data used: GSOD and ECA&D

`tminGSODECAD` Variogram for residuals from multiple linear regression of global minimum daily temperature on GTT, MODIS LST, DEM, and TWI. Data used: GSOD and ECA&D

`tminGHCND` Variogram for residuals from multiple linear regression of global minimum daily temperature on GTT, MODIS LST, DEM, and TWI. Data used: GHCN-Daily

`tminGSODECAD_noMODIS` Variogram for residuals from multiple linear regression of global minimum daily temperature on GTT, DEM, and TWI. Data used: GSOD and ECA&D

`tmaxGSODECAD` Variogram for residuals from multiple linear regression of global maximum daily temperature on GTT, MODIS LST, DEM, and TWI. Data used: GSOD and ECA&D

`tmaxGHCND` Variogram for residuals from multiple linear regression of global maximum daily temperature on GTT, MODIS LST, DEM, and TWI. Data used: GHCN-Daily

`tmaxGSODECAD_noMODIS` Variogram for residuals from multiple linear regression of global maximum daily temperature on GTT, DEM, and TWI. Data used: GSOD and ECA&D

`tmeanHR` Variogram for residuals from multiple linear regression of Croatian mean daily temperature on GTT, DEM, and TWI. Data used: Croatian mean daily temperature dataset for the year 2008 (Croatian national meteorological network)

Author(s)

Milan Kilibarda <`kili@grf.bg.ac.rs`>, Aleksandar Sekulic <`asekulic@grf.bg.ac.rs`>

References

Kilibarda, M., T. Hengl, G. B. M. Heuvelink, B. Graeler, E. Pebesma, M. Percec Tadic, and B. Bajat (2014), Spatio-temporal interpolation of daily temperatures for global land areas at 1 km resolution, J. Geophys. Res. Atmos., 119, 2294-2313, doi:10.1002/2013JD020803.

Examples

```
data(tvgms)
tvgms[[1]] # model for mean daily temp.
```

Index