

# Package: meteo (via r-universe)

November 1, 2024

**Version** 2.0-3

**Date** 2024-04-16

**Type** Package

**Encoding** UTF-8

**Title** RFSI & STRK Interpolation for Meteo and 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](mailto:asekulic@grf.bg.ac.rs)>

**Depends** R (>= 4.0.0)

**Description** Random Forest Spatial Interpolation (RFSI, Sekulić et al. (2020) <[doi:10.3390/rs12101687](https://doi.org/10.3390/rs12101687)>) and spatio-temporal geostatistical (spatio-temporal regression Kriging (STRK)) interpolation for meteorological (Kilibarda et al. (2014) <[doi:10.1002/2013JD020803](https://doi.org/10.1002/2013JD020803)>, Sekulić et al. (2020) <[doi:10.1007/s00704-019-03077-3](https://doi.org/10.1007/s00704-019-03077-3)>) and other environmental variables. Contains global spatio-temporal models calculated using publicly available data.

**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,  
jsonlite

**BugReports** <https://github.com/AleksandarSekulic/Rmeteo/issues>

**RoxygenNote** 7.2.3

**Repository** <https://aleksandarsekulic.r-universe.dev>

**RemoteUrl** <https://github.com/aleksandarsekulic/rmeteo>

**RemoteRef** HEAD

**RemoteSha** a2f47787dd55e738d11d6b77d4b4ebf9e0e2cfa5

## Contents

acc.metric.fun . . . . .	3
cv.rfsi . . . . .	4
cv.strk . . . . .	8
data.prepare . . . . .	13
dem_twi_srb . . . . .	14
dprec . . . . .	15
dslp . . . . .	16
dsndp . . . . .	17
dtempc . . . . .	18
dtempc_ogimet . . . . .	19
dtemp_maxc . . . . .	20
dtemp_minc . . . . .	21
dwdsp . . . . .	22
get_coordinates . . . . .	23
get_meteo . . . . .	23
meteo2STFDF . . . . .	25
near.obs . . . . .	27
near.obs.soil . . . . .	29
nlmodis20110704 . . . . .	31
nlmodis20110712 . . . . .	31
NLpol . . . . .	32
pred.rfsi . . . . .	32
pred.strk . . . . .	36
regdata . . . . .	40
rfillspgaps . . . . .	41
rfilltimegaps . . . . .	43
rfsi . . . . .	45
rm.dupl . . . . .	48
stations . . . . .	49
stations_ogimet . . . . .	50
temp_geom . . . . .	51
tgeom2STFDF . . . . .	52
tiling . . . . .	54
tregcoef . . . . .	55
tune.rfsi . . . . .	57
tvgms . . . . .	60

**Index**

**62**

---

acc.metric.fun	<i>Accuracy metrics calculation</i>
----------------	-------------------------------------

---

### Description

Calculates classification and regression accuracy metrics for given corresponding 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)

# 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=2, # 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")

```

---

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 $z \sim 1$ , an RFSI model using only nearest observations and distances to them as covariates will be cross-validated.
data	<a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> or <a href="#">data.frame</a> ; Contains target variable (observations) and covariates used for making an RFSI model. If <a href="#">data.frame</a> 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 observations and distances to them as covariates (formula= $z \sim 1$ ) 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 <a href="#">data.frame</a> object (e.g. <code>c(1,2,3,4)</code> ). If data is <a href="#">sf-class</a> , <a href="#">sftime-class</a> , or <a href="#">SpatVector-class</a> object, <code>data.staid.x.y.z</code> is used to point staid and z position. Set z position to NA (e.g. <code>c(1,2,3,NA)</code> ) or omit it (e.g. <code>c(1,2,3)</code> ) 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 <a href="#">near.obs</a> ). Default is FALSE.
s.crs	<a href="#">st_crs</a> or <a href="#">crs</a> ; Source CRS of data. If data contains crs, s.crs will be overwritten. Default is NA.
p.crs	<a href="#">st_crs</a> or <a href="#">crs</a> ; Projection CRS for data reprojection. If NA, s.crs will be used for distance calculation. Note that observations should be in projection for find-

	ing nearest observations based on Euclidean distances (see function <a href="#">near.obs</a> ). Default is NA.
<code>tgrid</code>	<code>data.frame</code> ; Possible tuning parameters for nested folds. The column names are same as the tuning parameters. Possible tuning parameters are: <code>n.obs</code> , <code>num.trees</code> , <code>mtry</code> , <code>min.node.size</code> , <code>sample.fraction</code> , <code>splirule</code> , <code>idw.p</code> , and <code>depth.range</code> .
<code>tgrid.n</code>	numeric; Number of randomly chosen <code>tgrid</code> combinations for nested tuning of RFSI. If larger than <code>tgrid</code> , will be set to <code>length(tgrid)</code>
<code>tune.type</code>	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".
<code>k</code>	numeric; Number of random outer and inner folds (i.e. for cross-validation and nested tuning) that will be created with <a href="#">CreateSpacetimeFolds</a> function. Default is 5.
<code>seed</code>	numeric; Random seed that will be used to generate outer and inner folds with <a href="#">CreateSpacetimeFolds</a> function.
<code>out.folds</code>	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 <a href="#">CreateSpacetimeFolds</a> function.
<code>in.folds</code>	numeric or character vector or value; Showing inner folds column (if value) or rows (vector) of data observations used for cross-validation. If missing, will be created with <a href="#">CreateSpacetimeFolds</a> function.
<code>acc.metric</code>	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".
<code>output.format</code>	character; Format of the output, <code>data.frame</code> (default), <code>sf-class</code> , <code>sftime-class</code> , or <code>SpatVector-class</code> .
<code>cpus</code>	numeric; Number of processing units. Default is <code>detectCores()-1</code> .
<code>progress</code>	numeric; If progress bar is shown. 0 is no progress bar, 1 is outer folds results, 2 is + inner folds results, 3 is + prediction progress bar. Default is 1.
<code>soil3d</code>	logical; If 3D soil modelling is performed and <a href="#">near.obs.soil</a> function is used for finding <code>n</code> nearest observations and distances to them. In this case, <code>z</code> position of the <code>data.staid.x.y.z</code> points to the depth column.
<code>no.obs</code>	character; Possible values are <code>increase</code> (default) and <code>exactly</code> . If set to <code>increase</code> , in case if there is no <code>n.obs</code> observations in <code>depth.range</code> for a specific location, the <code>depth.range</code> is increased (multiplied by 2, 3, ...) until the number of observations are larger or equal to <code>n.obs</code> . If set to <code>exactly</code> , the function will raise an error when it come to the first location with no <code>n.obs</code> observations in specified <code>depth.range</code> (see function <a href="#">near.obs.soil</a> ).
<code>...</code>	Further arguments passed to <a href="#">ranger</a> .

**Value**

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

<code>obs</code>	Observations.
<code>pred</code>	Predictions from cross-validation.
<code>folds</code>	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")
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)

# Cross-validation of RFSI
rfsi_cv <- cv.rfsi(formula=fm.RFSI, # without nearest obs
                  data = data,
                  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=2, # detectCores()-1,
progress=1,
importance = "impurity") # ranger parameter

summary(rfsi_cv)
rfsi_cv$dif <- rfsi_cv$obs - rfsi_cv$pred
plot(rfsi_cv["dif"])
plot(rfsi_cv[, , "obs"])
acc.metric.fun(rfsi_cv$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")

```

---

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	<a href="#">STFDF-class</a> , <a href="#">STSDF-class</a> , <a href="#">STIDF-class</a> , <a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> or <a href="#">data.frame</a> ; Contains target variable (observations) and covariates in space and time used to perform STRK cross validation. If <a href="#">data.frame</a> 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 <code>reg.coef</code> (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 <a href="#">data.frame</a> object (e.g. <code>c(1,2,3,4)</code> ). If data is <a href="#">sf-class</a> , <a href="#">sftime-class</a> , or <a href="#">SpatVector-class</a> object, <code>data.staid.x.y.z</code> is used to point staid and z position. If data is <a href="#">STFDF-class</a> , <a href="#">STSDF-class</a> , <a href="#">STIDF-class</a> object, <code>data.staid.x.y.z</code> is used to point only staid position. Default is NULL.
crs	<a href="#">st_crs</a> or <a href="#">crs</a> ; Source CRS of data. If data contains <code>crs</code> , <code>crs</code> 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 <a href="#">rm.dupl</a> . 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 meteorological variables (temperature, precipitation, etc.) can be taken from <code>data(tregcoef)</code> or can be specified by the user.
vgm.model	<a href="#">StVariogramModel</a> list; Spatio-temporal variogram of regression residuals (or observations if spatio-temporal ordinary kriging). See <a href="#">vgmST</a> . Spatio-temporal variogram model on residuals for meteorological variables (temperature, precipitation, etc.) can be taken from <code>data(tvgms)</code> or can be specified by the user as a <a href="#">vgmST</a> object.
sp.nmax	numeric; A number of spatially nearest observations that should be used for kriging predictions. If <code>tiling</code> is TRUE (see below), then is a number of spatially nearest observations that should be used for each tile. Default is 20.
time.nmax	numeric; A number of temporally nearest observations that should be used for kriging predictions. Default 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.

<code>k</code>	numeric; Number of random folds that will be created with <a href="#">CreateSpacetimeFolds</a> function. Default is 5.
<code>seed</code>	numeric; Random seed that will be used to generate outer and inner folds with <a href="#">CreateSpacetimeFolds</a> function.
<code>folds</code>	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 <a href="#">CreateSpacetimeFolds</a> function.
<code>refit</code>	logical; If refit of linear regression trend and spatio-temporal variogram should be performed. Spatio-temporal variogram is fit using <code>vgm.model</code> as desired spatio-temporal model for <a href="#">fit.StVariogram</a> function. Default is TRUE.
<code>output.format</code>	character; Format of the output, <a href="#">STFDF-class</a> (default), <a href="#">STSDF-class</a> , <a href="#">STIDF-class</a> , <a href="#">data.frame</a> , <a href="#">sf-class</a> , <a href="#">sftime-class</a> , or <a href="#">SpatVector-class</a> .
<code>parallel.processing</code>	logical; If parallel processing is performed. Default is FALSE.
<code>pp.type</code>	character; Type (R package) of parallel processing, "snowfall" (default) or "doParallel".
<code>cpus</code>	numeric; Number of processing units. Default is <code>detectCores()-1</code> .
<code>progress</code>	logical; If progress bar is shown. Default is TRUE.
<code>...</code>	Further arguments passed to <a href="#">krigeST</a> or <a href="#">pred.strk</a> .

### 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:

<code>obs</code>	Observations.
<code>pred</code>	Predictions from cross-validation.
<code>folds</code>	Folds used for cross-validation.

For accuracy metrics see [acc.metric.fun](#) function.

### Author(s)

Aleksandar Sekulic <[asekulic@grf.bg.ac.rs](mailto:asekulic@grf.bg.ac.rs)>, Milan Kilibarda <[kili@grf.bg.ac.rs](mailto: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 STDFD 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	<i>Prepare data</i>
--------------	---------------------

---

### 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	<a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> , <a href="#">SpatRaster-class</a> or <a href="#">data.frame</a> ; Contains target variable (observations) and covariates. If <a href="#">data.frame</a> 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 <a href="#">data.frame</a> object (e.g. <code>c(1,2,3,4)</code> ). If data is <a href="#">sf-class</a> , <a href="#">sftime-class</a> , or <a href="#">SpatVector-class</a> object, <code>data.staid.x.y.z</code> is used to point staid and z position. Set z position to NA (e.g. <code>c(1,2,3,NA)</code> ) or omit it (e.g. <code>c(1,2,3)</code> ) 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	<a href="#">st_crs</a> or <a href="#">crs</a> ; Source CRS of data. If data contains <code>crs</code> , <code>s.crs</code> will not be used. Default is NA.

### Value

A list with the following elements:

data.df	A <a href="#">data.frame</a> obtained from data.
data.staid.x.y.z	Positions of the station ID (staid), longitude (x), latitude (y) and 3rd component (z) columns in <a href="#">data.frame</a> object (e.g. <code>c(1,2,3,4)</code> ).
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](mailto: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](mailto: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/g sod/>)
- 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	<i>Mean daily temperature in degrees Celsius for the year 2019 for Serbia</i>
---------------	---

---

**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; Height  
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_coordinates	<i>Get lon/lat coordinates for a specific location name.</i>
-----------------	--

---

**Description**

The function gives back lon/lat coordinates for a specific location name using Nominatim (<https://nominatim.org/>) service.

**Usage**

```
get_coordinates(location_name = "Belgrade")
```

**Arguments**

location\_name character; Location name (e.g. "Belgrade").

**Value**

numeric vector with lon/lat values for a specific location name.

**Author(s)**

Aleksandar Sekulić <[asekulic@grf.bg.ac.rs](mailto:asekulic@grf.bg.ac.rs)>

**See Also**

[get\\_meteo](#)

**Examples**

```
coords <- get_coordinates("Belgrade")
str(coords)
```

---

get_meteo	<i>Get daily, monthly, or annual; aggregated or long-term means meteorological data for specific location(s) and date(s).</i>
-----------	---

---

**Description**

The function gives back daily, monthly, or annual and aggregated or long-term means meteorological data for specific location(s) and date(s) for entire World for 1961-2020 period from DailyMeteo portal (<https://app.dailymeteo.com/>).

**Usage**

```
get_meteo(loc,
          var = "tmean",
          agg_level = "agg",
          time_scale = "day",
          time,
          from,
          to,
          api_key)
```

**Arguments**

loc	sf, SpatVector, <a href="#">data.frame</a> , matrix, numeric or integer; Location(s) in WGS84 (EPSG:4326). If <a href="#">data.frame</a> or matrix columns are lon/lat. See <a href="#">get_coordinates</a> .
var	character; Meteorological variable. Possible values are: "tmean" (mean temperature, default), "tmax" (maximum temperature), "tmin" (minimum temperature), "prcp" (total precipitation), or "slp" (sea-level pressure).
agg_level	character; Aggregation level. Possible values are: "agg" (aggregated, default) or "ltm" (long-term mean).
time_scale	character; Time scale. Possible values are: "day" (daily, default), "mon" (monthly), or "ann" (annual).
time	character; Vector of time references. If specified, ignore from and to. If time_scale="day" then format is 'YYYY-MM-DD'. If time_scale="mon" then the format is 'YYYY-MM'. If time_scale="ann" then format is 'YYYY' (e.g. if time_scale="day" then c('2020-01-01', '2020-01-02'); if time_scale="mon" then c('2020-01', '2020-02'); if time_scale="ann" then c('2020', '2019')).
from	character; Time reference from. Format is the same as for the parameter time
to	character; Time reference to. Format is the same as for the parameter time
api_key	character; API key. Check DailyMeteo portal ( <a href="https://app.dailymeteo.com/">https://app.dailymeteo.com/</a> )

**Value**

[data.frame](#) object with columns loc (location index from loc argument), timestamp (time reference), and value (meteorological value).

**Author(s)**

Aleksandar Sekulić <[asekulic@grf.bg.ac.rs](mailto:asekulic@grf.bg.ac.rs)>

**See Also**

[get\\_coordinates](#)



## Examples

```
## Not run:
loc <- get_coordinates("Belgrade")
loc <- rbind(loc, get_coordinates("Kopaonik"))
loc
api_key <- "" # get API key from DailyMeteo portal (https://app.dailymeteo.com/)
result <- get_meteo(loc = loc,
                    var = "tmean",
                    agg_level = "agg",
                    time_scale = "day",
                    from = "2020-01-01",
                    to = "2020-01-02", # or use time = c("2020-01-01", "2020-01-02"),
                    api_key = api_key)

# result
#   loc timestamp value
# 1  1 2020-01-01  0.7
# 2  1 2020-01-02  1.0
# 3  2 2020-01-01 -9.2
# 4  2 2020-01-02 -8.6
# 5  3 2020-01-01 -9.2
# 6  3 2020-01-02 -8.6

## End(Not run)
```

---

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 <a href="#">CRS-class</a> ) 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	<i>Finds n nearest observations from given locations.</i>
----------	---

---

### 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 <code>sf-class</code> , <code>SpatVector-class</code> or <code>SpatRaster-class</code> 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 <code>sf-class</code> or <code>SpatVector-class</code> 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 observations. 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	<i>Finds n nearest observations from given locations for soil mapping.</i>
---------------	--

---

### 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 <code>sf-class</code> , <code>SpatVector-class</code> or <code>SpatRaster-class</code> 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 <code>sf-class</code> or <code>SpatVector-class</code> 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 observations. 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

```

library(sp)
data(nlmodis20110704)

splot(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**

```
library(sp)
data(nlmodis20110712)

splot(nlmodis20110712)
```

---

NLpol

*The Netherlands border polygon from WCAB*

---

**Description**

[SpatialGridDataFrame](#) from World Country Admin Boundary Shapefile

**Usage**

```
data(NLpol)
```

**Examples**

```
library(sp)
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 <a href="#">rfsi</a> function.
data	<a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> or <a href="#">data.frame</a> ; Contains target variable (observations) and covariates used for RFSI prediction. If <a href="#">data.frame</a> 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 <a href="#">data.frame</a> object (e.g. <code>c(1,2,3,4)</code> ). If data is <a href="#">sf-class</a> , <a href="#">sftime-class</a> , or <a href="#">SpatVector-class</a> object, <code>data.staid.x.y.z</code> is used to point staid and z position. Set z position to NA (e.g. <code>c(1,2,3,NA)</code> ) or omit it (e.g. <code>c(1,2,3)</code> ) for spatial interpolation. Default is NULL.
newdata	<a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> , <a href="#">SpatRaster-class</a> or <a href="#">data.frame</a> ; Contains prediction locations and covariates used for RFSI prediction. If <a href="#">data.frame</a> 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 <a href="#">data.frame</a> newdata object (e.g. <code>c(1,2,3,4)</code> ). If newdata is <a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> or <a href="#">SpatRaster-class</a> object, <code>newdata.staid.x.y.z</code> is used to point staid and z position. Set z position to NA (e.g. <code>c(1,2,3,NA)</code> ) or omit it (e.g. <code>c(1,2,3)</code> ) for spatial interpolation. Default is NULL.

z.value	vector; A vector of 3rd component - time, depth, ... (z) values if newdata is <a href="#">SpatRaster-class</a> .
s.crs	<a href="#">st_crs</a> or <a href="#">crs</a> ; Source CRS of data. If data contains crs, s.crs will not be used. Default is NA.
newdata.s.crs	<a href="#">st_crs</a> or <a href="#">crs</a> ; Source CRS of newdata. If newdata contains crs, newdata.s.crs will not be used. Default is NA.
p.crs	<a href="#">st_crs</a> or <a href="#">crs</a> ; 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 Euclidean distances (see function <a href="#">near.obs</a> ). Default is NA.
output.format	character; Format of the output, <a href="#">data.frame</a> (default), <a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> , or <a href="#">SpatRaster-class</a> .
cpus	numeric; Number of processing units. Default is <code>detectCores()-1</code> .
progress	logical; If progress bar is shown. Default is TRUE.
soil3d	logical; If 3D soil modelling is performed and <a href="#">near.obs.soil</a> function is used for finding n nearest observations and distances to them. In this case, z position of the <code>data.staid.x.y.z</code> points to the depth column.
depth.range	numeric; Depth range for location mid depth in which to search for nearest observations (see function <a href="#">near.obs.soil</a> ). It's in the mid depth units. Default is 0.1.
no.obs	character; Possible values are <code>increase</code> (default) and <code>exactly</code> . If set to <code>increase</code> , in case if there is no n.obs observations in <code>depth.range</code> for a specific location, the <code>depth.range</code> is increased (multiplied by 2, 3, ...) until the number of observations are larger or equal to n.obs. If set to <code>exactly</code> , the function will raise an error when it come to the first location with no n.obs observations in specified <code>depth.range</code> (see function <a href="#">near.obs.soil</a> ).
...	Further arguments passed to <a href="#">predict.ranger</a> function, such as <code>type = "quantile"</code> and <code>quantiles = c(0.1, 0.5, 0.9)</code> 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](mailto: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")
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)
                  n.obs = 5, # number of nearest observations
                  cpus = 2, # 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
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",
                             newdata = newdata, # meuse.grid.df (use newdata.staid.x.y.z)
                             output.format = "SpatRaster", # "sf", # "SpatVector",
                             zero.tol = 0,
                             cpus = 2, # 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'])

```

---

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 <code>data.frame</code> object (e.g. <code>c(1,2,3,4)</code> ). If data is <code>sf-class</code> , <code>sftime-class</code> , or <code>SpatVector-class</code> object, <code>data.staid.x.y.z</code> is used to point staid and z position. Default is NULL.
newdata	<code>STFDF-class</code> , <code>STSDF-class</code> , <code>STIDF-class</code> , <code>sf-class</code> , <code>sftime-class</code> , <code>SpatVector-class</code> , <code>SpatRaster-class</code> or <code>data.frame</code> ; Contains prediction locations and covariates used for STRK prediction. If <code>data.frame</code> 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 <code>reg.coef</code> (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 <code>data.frame</code> newdata object (e.g. <code>c(1,2,3,4)</code> ). If newdata is <code>sf-class</code> , <code>sftime-class</code> , <code>SpatVector-class</code> or <code>SpatRaster-class</code> object, <code>newdata.staid.x.y.z</code> 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 <code>SpatRaster-class</code> .
crs	<code>st_crs</code> or <code>crs</code> ; Source CRS of data and newdata. If data or newdata contains <code>crs</code> , <code>crs</code> 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 <code>rm.dupl</code> . 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 meteorological variables (temperature, precipitation, etc.) can be taken from <code>data(tregcoef)</code> or can be specified by the user.
vgm.model	<code>StVariogramModel</code> list; Spatio-temporal variogram of regression residuals (or observations if spatio-temporal ordinary kriging). See <code>vgmST</code> . Spatio-temporal variogram model on residuals for meteorological variables (temperature, precipitation, etc.) can be taken from <code>data(tvgms)</code> or can be specified by the user as a <code>vgmST</code> object.
sp.nmax	numeric; A number of spatially nearest observations that should be used for kriging predictions. If <code>tiling</code> is TRUE (see below), then is a number of spatially nearest observations that should be used for each tile. Default is 20.
time.nmax	numeric; A number of temporally nearest observations that should be used for kriging predictions. Default is 2.
by	character; Will foreach loop by time (default) or station. If station is set, <code>sp.nmax</code> 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 <code>sp.nmax</code> and <code>time.nmax</code> . 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 <code>sp.nmax</code> and observations fall in neighboring tiles.
output.format	character; Format of the output, <a href="#">STFDF-class</a> (default), <a href="#">STSDF-class</a> , <a href="#">STIDF-class</a> , <a href="#">data.frame</a> , <a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> , or <a href="#">SpatRaster-class</a> .
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 <code>detectCores()-1</code> .
computeVar	logical; If kriging variance is computed. Default is FALSE.
progress	logical; If progress bar is shown. Default is TRUE.
...	Further arguments passed to <a href="#">krigeST</a> 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.
t1m	Trend.
var	Kriging variance, if <code>computeVar=TRUE</code> .

**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

# 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 = tvgs[[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())

# 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 = 2, # detectCores()-1,
                    computeVar = TRUE,
                    progress = TRUE
                )
# plot prediction
plot(results$`2011-07-06`[["pred"]])
plot(results$`2011-07-06`[["var"]])

```

---

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 (deprecated)



```
regdata@sp$twi numeric; SAGA Topographic Wetness Index (TWI) from Worldgrids (depre-
  cated)
```

### 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	<a href="#">SpatRaster-class</a> , <a href="#">RasterLayer-class</a> , <a href="#">SpatialGrid-class</a> or <a href="#">SpatialPixels-class</a> ; Raster that contains NAs.
maskPol	<a href="#">sf-class</a> , <a href="#">SpatVector-class</a> , <a href="#">SpatialPolygons</a> or <a href="#">SpatialPolygonsDataFrame</a> ; Area of interest to spatially fill rasterLayer missing values and to mask rasterLayer.
nmax	see <a href="#">krige</a> , <a href="#">idw</a> function.
zcol	integer or character; variable column name or number showing position of a variable in rasterLayer to be interpolated.
...	arguments passed to <a href="#">krige</a> , <a href="#">idw</a> 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	<i>Disaggregation in the time dimension through the use of splines for each pixel</i>
---------------	---

---

## 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	<a href="#">STFDF-class</a> ; object with time information of minimum length 2, and gap in time dimension.
tunits	character; increment of the sequence used to generate time information for temporal gap. See 'Details'.
attrname	integer or character; variable from <a href="#">STFDF-class</a> to be splined in time.
...	arguments passed to <a href="#">splinefun</a> , 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

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())
```

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 $z \sim 1$ , an RFSI model using only nearest observations and distances to them as covariates will be made.
data	<a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> or <a href="#">data.frame</a> ; Contains target variable (observations) and covariates used for making an RFSI model. If <a href="#">data.frame</a> 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 observations and distances to them as covariates ( $formula = z \sim 1$ ) 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 <a href="#">data.frame</a> object (e.g. <code>c(1,2,3,4)</code> ). If data is <a href="#">sf-class</a> , <a href="#">sftime-class</a> , or <a href="#">SpatVector-class</a> object,

	<code>data.staid.x.y.z</code> is used to point <code>staid</code> and <code>z</code> position. Set <code>z</code> position to NA (e.g. <code>c(1,2,3,NA)</code> ) or omit it (e.g. <code>c(1,2,3)</code> ) for spatial interpolation. Default is NULL.
<code>n.obs</code>	numeric; Number of nearest observations to be used as covariates in RFSI model (see function <code>near.obs</code> ). Note that it cannot be larger than number of observations. Default is 5.
<code>avg</code>	boolean; Averages in circles covariate - will averages in circles with different radiuses be calculated (see function <code>near.obs</code> ). Default is FALSE.
<code>increment</code>	numeric; Increment of radiuses for calculation of averages in circles with different radiuses (see function <code>near.obs</code> ). Units depends on CRS.
<code>range</code>	numeric; Maximum radius for calculation of averages in circles with different radiuses (see function <code>near.obs</code> ). Units depends on CRS.
<code>quadrant</code>	boolean; Nearest observations in quadrants covariate - will nearest observation in quadrants be calculated (see function <code>near.obs</code> ). Default is FALSE.
<code>use.idw</code>	boolean; IDW prediction as covariate - will IDW predictions from <code>n.obs</code> nearest observations be calculated (see function <code>near.obs</code> ). Default is FALSE.
<code>idw.p</code>	numeric; Exponent parameter for IDW weights (see function <code>near.obs</code> ). Default is 2.
<code>s.crs</code>	<code>st_crs</code> or <code>crs</code> ; Source CRS of data. If data contains <code>crs</code> , <code>s.crs</code> will be overwritten. Default is NA.
<code>p.crs</code>	<code>st_crs</code> or <code>crs</code> ; Projection CRS for data reprojection. If NA, <code>s.crs</code> will be used for distance calculation. Note that observations should be in projection for finding nearest observations based on Euclidean distances (see function <code>near.obs</code> ). Default is NA.
<code>cpus</code>	numeric; Number of processing units. Default is <code>detectCores()-1</code> .
<code>progress</code>	logical; If progress bar is shown. Default is TRUE.
<code>soil3d</code>	logical; If 3D soil modelling is performed and <code>near.obs.soil</code> function is used for finding <code>n</code> nearest observations and distances to them. In this case, <code>z</code> position of the <code>data.staid.x.y.z</code> points to the depth column.
<code>depth.range</code>	numeric; Depth range for location mid depth in which to search for nearest observations (see function <code>near.obs.soil</code> ). It's in the mid depth units. Default is 0.1.
<code>no.obs</code>	character; Possible values are <code>increase</code> (default) and <code>exactly</code> . If set to <code>increase</code> , in case if there is no <code>n.obs</code> observations in <code>depth.range</code> for a specific location, the <code>depth.range</code> is increased (multiplied by 2, 3, ...) until the number of observations are larger or equal to <code>n.obs</code> . If set to <code>exactly</code> , the function will raise an error when it come to the first location with no <code>n.obs</code> observations in specified <code>depth.range</code> (see function <code>near.obs.soil</code> ).
<code>...</code>	Further arguments passed to <code>ranger</code> , such as <code>quantreg</code> , <code>importance</code> , etc.

## Value

RFSI model of class `ranger`.

**Note**

Observations should be in projection for finding nearest observations based on Euclidean 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")
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)
                  n.obs = 5, # number of nearest observations
                  cpus = 2, # 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	<i>Find point pairs with equal spatial coordinates from <a href="#">STFDF-class</a> object.</i>
---------	---

---

## 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	<a href="#">STFDF-class</a> 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 longitude/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)
```



```
# 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
```

---

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	<i>Data frame containing stations' information from the OGIMET service for Serbian territory</i>
-----------------	--

---

## 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	<a href="#">Spatial-class</a> (Points, Grid or Pixels), <a href="#">sf-class</a> , <a href="#">SpatVector-class</a> , <a href="#">SpatRaster-class</a> ; Locations for which gtt is calculated with associated coordinate reference systems ( <a href="#">CRS-class</a> ). 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 July 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='tiles',
      parallel.processing=FALSE,
      cpus=6,
      ...)
```

### Arguments

rast	<a href="#">SpatRaster</a> , <a href="#">SpatialPixels*</a> object, <a href="#">SpatialGrid*</a> object or file path to raster object stored on the disk (can be read via <a href="#">rast</a> ), for more details see <a href="#">SpatRaster</a> . 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; Possible values are sf, terra sp. If specified, tiles are returned in form of points as <a href="#">sf-class</a> , <a href="#">SpatVector</a> or <a href="#">SpatialPointsDataFrame</a> .
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 parallel processing is performed via <a href="#">snowfall-calculation</a> , <a href="#">sfLapply</a> function.
cpus	integer; number of processing units.
...	character; additional arguments for for writing files, see <a href="#">writeRaster</a> .

### 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=TRUE)
plot(tiles[[2]], pch='. ' , add=TRUE)

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 $z \sim 1$ , an RFSI model using only nearest observations and distances to them as covariates will be tuned.
data	<a href="#">sf-class</a> , <a href="#">sftime-class</a> , <a href="#">SpatVector-class</a> or <a href="#">data.frame</a> ; Contains target variable (observations) and covariates used for making an RFSI model. If <a href="#">data.frame</a> 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 observations and distances to them as covariates ( $formula = z \sim 1$ ) 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 <a href="#">data.frame</a> object (e.g. <code>c(1,2,3,4)</code> ). If data is <a href="#">sf-class</a> , <a href="#">sftime-class</a> , or <a href="#">SpatVector-class</a> object, <code>data.staid.x.y.z</code> is used to point staid and z position. Set z position to NA

	(e.g. <code>c(1,2,3,NA)</code> ) or omit it (e.g. <code>c(1,2,3)</code> ) for spatial interpolation. Default is <code>NULL</code> .
<code>use.idw</code>	boolean; IDW prediction as covariate - will IDW predictions from <code>n.obs</code> nearest observations be calculated and tuned (see function <code>near.obs</code> ). Default is <code>FALSE</code> .
<code>s.crs</code>	<code>st_crs</code> or <code>crs</code> ; Source CRS of data. If data contains <code>crs</code> , <code>s.crs</code> will be overwritten. Default is <code>NA</code> .
<code>p.crs</code>	<code>st_crs</code> or <code>crs</code> ; Projection CRS for data reprojection. If <code>NA</code> , <code>s.crs</code> will be used for distance calculation. Note that observations should be in projection for finding nearest observations based on Euclidean distances (see function <code>near.obs</code> ). Default is <code>NA</code> .
<code>tgrid</code>	<code>data.frame</code> ; Possible tuning parameters. The column names are same as the tuning parameters. Possible tuning parameters are: <code>n.obs</code> , <code>num.trees</code> , <code>mtry</code> , <code>min.node.size</code> , <code>sample.fraction</code> , <code>splirule</code> , <code>idw.p</code> , and <code>depth.range</code> .
<code>tgrid.n</code>	numeric; Number of randomly chosen <code>tgrid</code> combinations for tuning of RFSI. If larger than <code>tgrid</code> , will be set to <code>length(tgrid)</code>
<code>tune.type</code>	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".
<code>k</code>	numeric; Number of random folds that will be created with <code>CreateSpacetimeFolds</code> function if <code>fold</code> s is column. Default is 5.
<code>seed</code>	numeric; Random seed that will be used to generate folds with <code>CreateSpacetimeFolds</code> function.
<code>fold</code> s	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 <code>CreateSpacetimeFolds</code> function.
<code>acc.metric</code>	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".
<code>fit.final.model</code>	boolean; Fit the final RFSI model. Default is <code>TRUE</code> .
<code>cpus</code>	numeric; Number of processing units. Default is <code>detectCores()-1</code> .
<code>progress</code>	logical; If progress bar is shown. Default is <code>TRUE</code> .
<code>soil3d</code>	logical; If 3D soil modelling is performed and <code>near.obs.soil</code> function is used for finding <code>n</code> nearest observations and distances to them. In this case, <code>z</code> position of the <code>data.staid.x.y.z</code> points to the depth column.
<code>no.obs</code>	character; Possible values are <code>increase</code> (default) and <code>exactly</code> . If set to <code>increase</code> , in case if there is no <code>n.obs</code> observations in <code>depth.range</code> for a specific location, the <code>depth.range</code> is increased (multiplied by 2, 3, ...) until the number of observations are larger or equal to <code>n.obs</code> . If set to <code>exactly</code> , the function will raise an error when it come to the first location with no <code>n.obs</code> observations in specified <code>depth.range</code> (see function <code>near.obs.soil</code> ).
<code>...</code>	Further arguments passed to <code>ranger</code> .

**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](mailto: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")
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)

# 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 = 2, # 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

```

---

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)
tvgs[[1]] # model for mean daily temp.
```

# Index

## \* datasets

- dem\_twi\_srb, 14
  - dprec, 15
  - dslp, 16
  - dsndp, 17
  - dtemp\_maxc, 20
  - dtemp\_minc, 21
  - dtempc, 18
  - dtempc\_ogimet, 19
  - dwdsp, 22
  - nlmodis20110704, 31
  - nlmodis20110712, 31
  - NLpol, 32
  - regdata, 40
  - stations, 49
  - stations\_ogimet, 50
  - tregcoef, 55
  - tvgms, 60
- acc.metric.fun, 3, 3, 10
- CreateSpacetimeFolds, 6, 10, 58
- crs, 5, 9, 13, 34, 37, 46, 58
- CRS-class, 26, 52
- cv.rfsi, 3, 4, 14, 28, 30, 34, 47, 59
- cv.strk, 3, 8
- data.frame, 5–7, 9, 10, 13, 24, 27–30, 33, 34, 36–38, 45, 57
- data.prepare, 13
- dem\_twi\_srb, 14
- difftime, 43
- dprec, 15
- dslp, 16
- dsndp, 17
- dtemp\_maxc, 20
- dtemp\_minc, 21
- dtempc, 18
- dtempc\_ogimet, 19
- dwdsp, 22
- fit.StVariogram, 10
- get\_coordinates, 23, 24
- get\_meteo, 23, 23
- knn, 27–30
- krige, 41
- krigeST, 10, 36, 38
- meteo2STFDF, 10, 25, 38
- near.obs, 5–7, 14, 27, 30, 34, 46, 47, 58, 59
- near.obs.soil, 6, 29, 34, 46, 58
- nlmodis20110704, 31
- nlmodis20110712, 31
- NLpol, 32
- pred.rfsi, 3, 4, 7, 28, 30, 32, 47, 59
- pred.strk, 3, 8, 10, 26, 36, 42, 44, 48, 55
- predict.ranger, 34
- ranger, 6, 46, 58, 59
- rast, 54
- raster, 54
- RasterLayer-class, 41
- regdata, 10, 38, 40
- rfillspgaps, 41, 44
- rfilltimegaps, 42, 43
- rfsi, 3, 4, 7, 14, 28, 30, 33, 34, 45, 59
- rm.dupl, 9, 37, 48
- sf-class, 5–7, 9, 10, 13, 27, 29, 33, 34, 36–38, 41, 45, 52, 54, 57
- sftime-class, 5–7, 9, 10, 13, 33, 34, 36–38, 45, 57
- snowfall-calculation, 54
- Spatial-class, 52, 54
- SpatialGrid-class, 41
- SpatialGridDataFrame, 31, 32
- SpatialPixels-class, 41
- SpatialPointsDataFrame, 54

SpatialPolygons, [41](#)  
SpatialPolygonsDataFrame, [41](#)  
SpatRaster, [14](#), [54](#)  
SpatRaster-class, [13](#), [27](#), [29](#), [33](#), [34](#), [37](#), [38](#),  
[41](#), [52](#), [54](#)  
SpatVector, [54](#)  
SpatVector-class, [5–7](#), [9](#), [10](#), [13](#), [27](#), [29](#), [33](#),  
[34](#), [36–38](#), [41](#), [45](#), [52](#), [57](#)  
splinefun, [43](#)  
st\_crs, [5](#), [9](#), [13](#), [34](#), [37](#), [46](#), [58](#)  
stations, [15–18](#), [20–22](#), [49](#)  
stations\_ogimet, [50](#)  
STFDF-class, [9](#), [10](#), [25](#), [26](#), [36–38](#), [40](#), [43](#), [48](#),  
[53](#)  
STIDF-class, [9](#), [10](#), [36–38](#)  
STSDF-class, [9](#), [10](#), [36–38](#)  
  
temp\_geom, [51](#)  
tgeom2STFDF, [10](#), [26](#), [38](#), [40](#), [48](#), [52](#)  
tiling, [54](#)  
tregcoef, [9](#), [10](#), [37](#), [38](#), [55](#)  
tune.rfsi, [3](#), [4](#), [7](#), [14](#), [28](#), [30](#), [34](#), [47](#), [57](#)  
tvgms, [9](#), [10](#), [37](#), [38](#), [60](#)  
  
vgmST, [9](#), [37](#)  
  
writeRaster, [54](#)