# Package 'landmap'

November 19, 2020

**Title** Automated Spatial Prediction using Ensemble Machine Learning

**Version** 0.0.6

**Description** Functions and tools for spatial interpolation and/or prediction of environmental variables (points to grids)
based on using Ensemble Machine Learning with geographical distances. Package also provides access to Global
Environmental Layers (<https://www.OpenLandMap.org>) produced by the OpenGeoHub.org foundation and collaborators. Some functions have been migrated and adopted from the Global Soil Information Facilities package.

**Depends** R (>= 3.5.0)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**URL** https://github.com/envirometrix/landmap/

**BugReports** https://github.com/envirometrix/landmap/issues/

**Imports** methods, utils, parallel, matrixStats, ranger, glmnet, mlr, parallelMap, sp, rgdal, gdalUtils, raster

**Suggests** geoR, ParamHelpers, mda, psych, spdep, fossil, xgboost, plyr, kernlab, nnet, rjson, spatstat, maptools, maxlike, RCurl, aqp, deepnet, RSAGA, soiltexture, snowfall, plotKML, boot

**RoxygenNote** 7.1.1

**SystemRequirements** C++11, GDAL (>= 2.0.1), GEOS (>= 3.4.0), PROJ (>= 4.8.0)

**NeedsCompilation** no

**Author** Tomislav Hengl [aut, cre]

**Maintainer** Tomislav Hengl <tom.hengl@opengeohub.org>

**Repository** CRAN

**Date/Publication** 2020-11-18 23:20:14 UTC

# R topics documented:



---

`buffer.dist,SpatialPointsDataFrame,SpatialPixelsDataFrame-method`
*Derive buffer distances for a list of points*

---

## Description

Derive buffer distances using the `raster::distance` function, so that these can be used as predictors for spatial prediction i.e. to account for spatial proximity to low, medium and high values.

### Usage

```
## S4 method for signature 'SpatialPointsDataFrame,SpatialPixelsDataFrame'
buffer.dist(
  observations,
  predictionDomain,
  classes,
  width,
  parallel = TRUE,
  ...
)
```

### Arguments

| | |
|---|---|
| observations | SpatialPointsDataFrame. |
| predictionDomain | SpatialPixelsDataFrame. |
| classes | vector of selected points as factors. |
| width | maximum width for buffer distance. |
| parallel | optional parallelization setting. |
| ... | optional arguments to pass to raster::distance function. |

### Value

object of class SpatialPixelsDataFrame with distances to points

### Author(s)

Tom Hengl

### Examples

```
library(raster)
library(rgdal)
demo(meuse, echo=FALSE)
b <- buffer.dist(meuse["zinc"], meuse.grid[1],
        classes=as.factor(1:nrow(meuse)), parallel=FALSE)
```

---

## download.landgis *Access and download layers from OpenLandMap.org (LandGIS data service)*

---

### Description

Access and download layers from OpenLandMap.org (LandGIS data service)

## Usage

```
download.landgis(
  coverageId,
  filename,
  scalefactor = NULL,
  subset = NULL,
  service = paste0(c("https://geoserver.opengeohub.org/landgisgeoserver/ows",
    "?service=WCS&version=2.0.1")),
  silent = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| coverageId | Coverage ID. |
| filename | Download filename. |
| scalefactor | Scale factor for WCS request. |
| subset | Subset string for WCS request. |
| service | URL of the WCS service. |
| silent | Silent output. |
| ... | optional `utils::download.file` settings. |

## Value

Locally downloaded GeoTIFF.

## Author(s)

Tom Hengl

## Examples

```
search.landgis(pattern=c("clay", "10..10cm"))
```

---

# edgeroi *The Edgeroi Data Set*

---

## Description

Soil samples and covariate layers for the Edgeroi area in NSW, Australia (ca 1500 square-km).

## Usage

```
data(edgeroi)
```

## Format

The edgeroi data set contains two data frames — sites and horizons. Sites table contains the following columns:

SOURCEID factor; unique label to help a user identify a particular site (ID in the NatSoil)

LONGDA94 numeric; longitude in decimal degrees on the GDA94 datum

LATGDA94 numeric; latitude in decimal degrees on the GDA94 datum

TAXGAUC factor; Australian Great Soil Groups (GSG; see details)

NOTEOBS character; free-form observation notes

Horizons table contains the following columns:

SOURCEID factor; unique identifier used in the NatSoil DB

LSQINT integer; a layer sequence number 1 to N

HZDUSD factor; horizon designation (primary letter)

UHDICM numeric; upper horizon depth from the surface in cm

LHDICM numeric; lower horizon depth from the surface in cm

CLYPPT numeric; weight percentage of the clay particles (<0.0002 mm)

SNDPPT numeric; weight percentage of the silt particles (0.0002–0.05 mm)

SLTPPT numeric; weight percentage of the sand particles (0.05–2 mm)

PHIHO5 numeric; pH index measured in water solution(ph_h2o in the NSCD)

ORCDRC numeric; soil organic carbon content in permille

The edgeroi.grids data frame contains a list of covariates at 250 m resolution:

DEMSRT5 numeric; SRTM DEM

TWISRT5 numeric; SAGA Topographic Wetness Index based on the SRTM DEM

PMTGEO5 factor; parent material class based on the National Geological map at scale 1:250,000 — sand with minor silty sand ("Qd"), alluvium gravel, sand, silt, clay ("Qrs"), quartz sandstone obscured by quartenary sands ("Qrt/Jp"), quartz sandstone obscured by talus material ("Qrt/Rn"), basalt obscured by talus material ("Qrt/Tv"), mottled clay, silt, sandstone and gravel ("Ts"), and basalt, dolerite, trachyte, techenite ("Tv")

EV1MOD5 numeric; first principal component of the MODIS EVI (MOD13Q1) time series data (year 2011)

EV2MOD5 numeric; second principal component of the MODIS EVI (MOD13Q1) time series data (year 2011)

EV3MOD5 numeric; third principal component of the MODIS EVI (MOD13Q1) time series data (year 2011)

x numeric; x-coordinate in the GDA94 / MGA zone 55

y numeric; y-coordinate in the GDA94 / MGA zone 55

The edgeroi.grids100 data frame contains a list of covariates at 100 m resolution prepared for the study area:

LNUABS6 factor; Australian National scale land use data

MVBSRT6 numeric; SAGA GIS Multi-resolution Index of Valley Bottom Flatness based on the SRTM DEM

TI1LAN6 numeric; principal component 1 for the Landsat band 7 (thermal) based on three periods of the Global Land Survey Landsat images (GLS1990, GLS2000, GLS2005)

TI2LAN6 numeric; principal component 2 for the Landsat band 7 (thermal) based on three periods of the Global Land Survey Landsat images (GLS1990, GLS2000, GLS2005)

PCKGAD6 numeric; percentage of Potassium estimated based on the gamma radiometrics radmap09 (GADDS)

RUTGAD6 numeric; ratio Uranium over Thorium estimated based on the gamma radiometrics radmap09 (GADDS)

PCTGAD6 numeric; parts per million of Thorium estimated based on the gamma radiometrics radmap09 (GADDS)

x numeric; x-coordinate in the GDA94 / MGA zone 55

y numeric; y-coordinate in the GDA94 / MGA zone 55

## Details

The Edgeroi is one of the standard soil data sets used to test soil mapping methods in Australia. Out of 359 profiles, 210 sites were sampled on a systematic, equilateral triangular grid with a spacing of 2.8 km between sites, the other sites are distributed more irregularly or on transects. The data set is described in detail in Malone et al. (2010) and McGarry et al. (1989). The `edgeroi` contains only a subset of the original NatSoil records. Observed soil classes for `TAXGAUC` are (alphabetically): Alluvial soil (`"A"`), Brown clay (`"BC"`), Black earth (`"BE"`), Earthy sand (`"ES"`), Grey clay (`"GC"`), Grey earth (`"GE"`), No suitable group (`"NSG"`), Prairie soil (`"PS"`), Rendzina (`"R"`), Red-brown earth (`"RBE"`), Red clay (`"RC"`), Red earth (`"RE"`), Red podzolic soil (`"RP"`), Solodic soil (`"SC"`), Soloth (`"SH"`), Solonchak (`"SK"`), Siliceous sand (`"SS"`), and Solonetz (`"SZ"`).

## Note

The Landsat images and SRTM DEM have been obtained from the Global Land Cover Facility. Scanned geology map (paper sheets) has been obtained from the Geoscience Australia, then georeferenced and rasterized to 250 m resolution. The land use map has been obtained from the Australian Collaborative Land Use and Management program. The Radiometric Map of Australia grids has been downloaded using the Geophysical Archive Data Delivery System (GADDS) on the Australian Government's Geoscience Portal (Mitny et al, 2009).
Listed gridded layers follow a standard naming convention used by WorlGrids.org (the standard 8.3 filename convention with at most eight characters): first three letter are used for the variable type e.g. `DEM` (digital elevation model); the next three letters represent the data source or collection method e.g. `SRT` (SRTM mission); the 6th character is the effective scale e.g. `5` indicates the 5th standard scale i.e. 1/600 decimal degrees (in this case 250 m).

## Author(s)

The original detailed profile description and laboratory analysis was funded by a Cotton Research and Development Corporation project in the mid-late 1980's by the CSIRO Division of Soils and available via the NatSoil DB. The gamma radiometrics images are property of the NSW Department of Primary Industries — Mineral Resources.

## References


- Malone, B.P., McBratney, A.B., Minasny, B. (2010) Mapping continuous depth functions of soil carbon storage and available water capacity. Geoderma 154, 138-152.
- McGarry, D., Ward, W.T., McBratney, A.B. (1989) Soil Studies in the Lower Namoi Valley: Methods and Data. The Edgeroi Data Set. (2 vols) (CSIRO Division of Soils: Adelaide).
- Minty, B., Franklin, R., Milligan, P., Richardson, L.M., and Wilford, J., (2009) The Radiometric Map of Australia. Exploration Geophysics, 40(4), 325-333.


## Examples

```
library(rgdal)
library(aqp)
library(sp)

data(edgeroi)
edgeroi$sites[edgeroi$sites$SOURCEID=="399_EDGEROI_ed095_1",]
edgeroi$horizons[edgeroi$horizons$SOURCEID=="399_EDGEROI_ed095_1",]
## spPoints:
sites <- edgeroi$sites
coordinates(sites) <- ~ LONGDA94 + LATGDA94
proj4string(sites) <- CRS("+proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs")
sites <- spTransform(sites, CRS("+init=epsg:28355"))


## plot points and grids:
pnts <- list("sp.points", sites, pch="+", col="black")
## load the 250 m grids:
library(RCurl)
rep = "https://raw.github.com/Envirometrix/PredictiveSoilMapping/master/extdata/"
x = tempfile(fileext = ".rda")
con <- download.file(paste0(rep, "edgeroi.grids.rda"), x, method="wget")
load(x)
str(edgeroi.grids)
gridded(edgeroi.grids) <- ~x+y
proj4string(edgeroi.grids) <- CRS("+init=epsg:28355")
spplot(edgeroi.grids[1], sp.layout=pnts)
## load the 100 m grids:
x2 = tempfile(fileext = ".rda")
con2 <- download.file(paste0(rep, "edgeroi.grids100.rda"), x2, method="wget")
load(x2)
str(edgeroi.grids100)
gridded(edgeroi.grids100) <- ~x+y
proj4string(edgeroi.grids100) <- CRS("+init=epsg:28355")
spplot(edgeroi.grids100["TI1LAN6"], sp.layout=pnts)
```

---

`fit.vgmModel,formula,data.frame,SpatialPixelsDataFrame-method`

*Fit variogram using point data*

---

## Description

Fit variogram using point data

## Usage

```
## S4 method for signature 'formula,data.frame,SpatialPixelsDataFrame'
fit.vgmModel(
  formulaString.vgm,
  rmatrix,
  predictionDomain,
  cov.model = "exponential",
  dimensions = list("2D", "3D", "2D+T", "3D+T"),
  lambda = 0.5,
  psiR = NULL,
  subsample = nrow(rmatrix),
  ini.var,
  ini.range,
  fix.psiA = FALSE,
  fix.psiR = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| formulaString.vgm | formula. |
| rmatrix | data.frame with coordinates and values of covariates. |
| predictionDomain | SpatialPixelsDataFrame. |
| cov.model | covariance model type used by the geoR package. |
| dimensions | optional 2D or 3D dimensions. |
| lambda | transformation value used by the geoR package. |
| psiR | range parameter used by the geoR package. |
| subsample | number of subset of original samples. |
| ini.var | initial variance (sill) used by the geoR package. |
| ini.range | initial range parameter used by the geoR package. |
| fix.psiA | setting used by the geoR package. |
| fix.psiR | setting used by the geoR package. |
| ... | optional arguments to pass to the geoR package. |

## Value

Fitted variogram model

### Note

Extends variogram fitting functionality from the geoR package. Can be used for 2D or 3D point data sets, with and without trend variables. Models need to be in the form `zinc ~ dist` and only numeric variables are allowed. Often reports `Singular matrix. Covariates may have different orders of magnitude.` if the covariates are perfectly aligned.

### Author(s)

Tom Hengl

### Examples

```
library(raster)
library(rgdal)
library(geoR)
demo(meuse, echo=FALSE)
vgm = fit.vgmModel(zinc~dist, as.data.frame(meuse), meuse.grid["dist"], lambda=1)
plot(variog(vgm$geodata))
lines(vgm$vgm)
```

---

## getSpatialTiles,ANY-method

*Estimate a tiling system*

---

### Description

Estimate a tiling system

### Usage

```
## S4 method for signature 'ANY'
getSpatialTiles(
  obj,
  block.x,
  block.y = block.x,
  overlap.percent = 0,
  limit.bbox = TRUE,
  return.SpatialPolygons = FALSE
)
```

### Arguments

| | |
|---|---|
| `obj` | ANY. |
| `block.x` | size of the block in x dimension. |
| `block.y` | size of the block in y dimension. |
| `overlap.percent` | optional overlap percent between tiles. |

limit.bbox optional bounding box.

return.SpatialPolygons
logical specificies whether to return a data frame or Spatial Polygons.

## Value

Tiling system for a spatial object

---

# getSpatialTiles,Spatial-method

*Split a Spatial object into tiles*

---

## Description

Split a Spatial object into tiles

## Usage

```
## S4 method for signature 'Spatial'
getSpatialTiles(
  obj,
  block.x,
  block.y = block.x,
  overlap.percent = 0,
  limit.bbox = TRUE,
  return.SpatialPolygons = TRUE
)
```

## Arguments

obj output of the GDALinfo.

block.x size of the block in x dimension.

block.y size of the block in y dimension.

overlap.percent
optional overlap percent between tiles.

limit.bbox optional bounding box.

return.SpatialPolygons
logical specificies whether to return a data frame or Spatial Polygons.

## Value

List of object result of clipping

## Author(s)

Tom Hengl

## Examples

```
library(sp)
data(meuse.grid)
gridded(meuse.grid) <- ~x+y
tl <- getSpatialTiles(meuse.grid, block.x=1000)
image(meuse.grid)
lines(as(tl, "SpatialLines"))
## all at once:
pix.lst <- tile(meuse.grid, block.x=1000)

  library(plotKML)
  ## raster files via rgdal:
  library(rgdal)
  fn = system.file("pictures/SP27GTIF.TIF",
                   package = "rgdal")
  obj <- GDALinfo(fn)
  ras.lst <- getSpatialTiles(obj, block.x=1000)
  offset <- c(ras.lst$offset.y[1], ras.lst$offset.x[1])
  region.dim <- c(ras.lst$region.dim.y[1],
                  ras.lst$region.dim.x[1])
  ## read the first tile:
  SP27GTIF_T1 <- readGDAL(fn, offset=offset,
                          region.dim=region.dim)
  str(SP27GTIF_T1)
```

---

`landgis.tables` *OpenLandMap list of GeoTIFFs global mosaics*

---

## Description

Computer names of the layers in the www.OpenLandMap.org.

## Usage

```
data(landgis.tables)
```

## Format

The `landgis.tables` data set contains four objects:

`tables` data.frame; layer metadata

`layers` list; list of layers at different resolutions

`classes` list; list of classes

`zenodo.files` list; list of URI addressed to download all OpenLandMap.org layers

## References

- https://openlandmap.org

## Examples

```
data(landgis.tables)
str(landgis.tables$tables[1,])
```

---

`landmap` *landmap: Automated Spatial Prediction using Ensemble Machine Learning*

---

## Description

Geographical distances can be used with remote sensing covariates and process-based derivatives to improve spatial prediction and/or interpolation from point data. This package shows how to fully automate process so that predictions and model errors can be generated using unbiased estimation (train.spLearner package). Additional functions are used to access global layers (from www.openlandmap.org), to process large rasters (spatial tiling) including running own customized functions in parallel.

---

`makeTiles` *Make a tiling system from a bounding box*

---

## Description

Make a tiling system from a bounding box

## Usage

```
makeTiles(
  bb,
  block.x,
  block.y,
  overlap.percent,
  limit.bbox,
  columns = NULL,
  rows = NULL
)
```

## Arguments

| | |
|---|---|
| `bb` | Bounding Box |
| `block.x` | Size of the block in X |
| `block.y` | Size of the block in Y |
| `overlap.percent` | Percent of overlap; default 0 |
| `limit.bbox` | Optional limiting bounding box |
| `columns` | Optional number of columns |
| `rows` | Optional number of rows |

### Value

A regular tiling system

### Author(s)

Tom Hengl

---

`model.data` *Overlay points and grids and prepare regression matrix*

---

### Description

Overlay points and grids and prepare regression matrix

### Usage

```
model.data(
  observations,
  formulaString,
  covariates,
  dimensions = c("2D", "3D", "2D+T", "3D+T")
)
```

### Arguments

| | |
|---|---|
| `observations` | SpatialPointsDataFrame |
| `formulaString` | Model definition |
| `covariates` | List of covariates column names |
| `dimensions` | 2D, 3D models |

### Value

Regression matrix data.frame

predict.spLearner *Predict using spLearner at new locations*

## Description

Predict using spLearner at new locations

## Usage

```
## S3 method for class 'spLearner'
predict(
  object,
  predictionLocations,
  model.error = TRUE,
  error.type = c("quantreg", "weighted.sd", "interval")[1],
  t.prob = 1/3,
  w,
  quantiles = c((1 - 0.682)/2, 1 - (1 - 0.682)/2),
  ...
)
```

## Arguments

| | |
|---|---|
| object | of type spLearner. |
| predictionLocations | SpatialPixelsDataFrame with values of all features. |
| model.error | Logical specify if prediction errors should be derived. |
| error.type | Specify how should be the prediction error be derived. |
| t.prob | Threshold probability for significant learners; only applyies for meta-learners based on lm model. |
| w | optional weights vector. |
| quantiles | list of quantiles for quantreg forest (maximum lower and upper quantile). |
| ... | optional parameters. |

## Value

Object of class SpatialPixelsDataFrame with predictions and model error.

## print.spLearner *Print object of type 'spLearner'*

### Description

Print object of type 'spLearner'

### Usage

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

### Arguments

| | |
|---|---|
| x | of type spLearner |
| ... | optional parameters |

### Value

Model summary

## search.landgis *Search for available landgis layers*

### Description

Search for available landgis layers

### Usage

```
search.landgis(
  pattern,
  layersURL = "https://landgisapi.opengeohub.org/query/layers",
  update = FALSE
)
```

### Arguments

| | |
|---|---|
| pattern | String pattern |
| layersURL | Default URL with the list of layers |
| update | Logical specify to update the layer list |

### Value

List of available landgis layers

sic1997 *The SIC 1997 Data Set*

## Description

Daily rainfall dataset from Switzerland used in the Spatial Interpolation Comparison 1997.

## Usage

```
data(sic1997)
```

## Format

The sic1997 data set contains two objects — stations (points) and covariates (1 km grids). The daily.rainfall contains the following columns:

rainfall numeric; daily rainfall measurements at 467 meteo stations on the 8th of May 1986

X numeric; easting in the local coordinate system

Y numeric; northing in the local coordinate system

Object swiss1km contains the following columns:

CHELSA_rainfall numeric; monthly rainfall based on the CHELSA climate grids

DEM numeric; digital elevation based on the SRTM DEM

border character; country border mask

## Details

The gathering of the rainfall data, provided by Giovanni Graziani from the Environment Institute of the Joint Research Centre (Ispra, Italy), has been undertaken, under JRC-Ispra funding, by the Air pollution Group at Imperial College, London. The Digital Elevation Model has been provided by EROS Data Centre from the U.S. Geological Survey (USGS). http://edcwww.cr.usgs.gov/.

## References


- Dubois, G. (1998). Spatial interpolation comparison 97: foreword and introduction. Journal of Geographic Information and Decision Analysis, 2(2), 1-10.
- https://wiki.52north.org/AI_GEOSTATS/WebHome


## Examples

```
data(sic1997)
```

`soil.classes` *Soil classification tables*

## Description

Standard soil classification tables for the United States Department of Agriculture (USDA) and IUSS / FAO World Reference Base (WRB) classification systems, including the tables used to correlate various soil classification systems.

## Usage

```
data(soil.classes)
```

## Format

Contains a list of tables:

`Canadian` data frame; Canadian soil classification system (Soil Classification Working Group, 1998)

`FAO1990.WRB` data frame; FAO 1990 (FAO-Unesco Soil Classification System) system to WRB (2006)

`USDA_GreatGroups` data frame; list of USDA Great Groups (USDA, 2010)

`WRB_versions` data frame; correlation between various FAO/WRB versions (Krasilnikov et al. 2009)

`FAO1974.WRB` data frame; correlation FAO 1974 system to WRB (IUSS Working Group WRB, 2006)

`USDA.WRB` data frame; correlation USDA system to WRB system

`Soils_World` data frame; referent soil profiles of the world (van Baren and Lof, 1987)

## Note

Some of the original tables from the literature have been adjusted / updated by the author. Correlation between various national and international systems often leads to multiple soil classes being possible equivalents. These are separated in tables using `"/"` symbol e.g. `Dark Gray Chernozem = Boralfic Boroll / Albolls`. Some national soil classification systems contain classes which are completely unique and hence most likely can not be correlated to any class in the target system.

## Author(s)

Tomislav Hengl

## References


- van Baren, H. and Lof, P. (1987) Soils of the World, published by Elsevier in Association with ISRIC, FAO, and UNESCO ISBN 0444425756
- Krasilnikov, P., Marti, J., Arnold, R., and Shoba, S., eds. (2009) A Handbook of Soil Terminology, Correlation and Classification, Earthscan LLC, pp. 448 ISBN 9781136546631
- Soil Classification Working Group, (1998) The Canadian System of Soil Classification. 3rd Ed. Agriculture and Agri-Food Canada Publication 1646, 187 pp. ISBN 0660174049
- USDA / Soil Survey Staff, (2010) Keys to Soil Taxonomy (Eleventh Edition) U.S. Department Of Agriculture, Natural Resources Conservation Service, 346 pp. ISBN 9781782662112
- IUSS Working Group WRB (2006) World Reference Base for Soil Resources 2006: A Framework for International Classification, Correlation and Communication, Food and Agriculture Organization of the United Nations 128 pp.


## Examples

```
data(soil.classes)
soil.classes$USDA_GreatGroups[1,]
DGC <- which(soil.classes$Canadian$CSSC_Great_Groups=="Dark Gray Chernozem")
soil.classes$Canadian[DGC,]
```

---

`soil.legends` *Standard color palettes for soil properties and classes*

---

## Description

Standard color palettes for soil properties and classes that can be used to display global soil data.

## Usage

```
data(soil.legends)
```

## Format

Contains a list of color palettes (data frames with class names / break points, and cumulative probabilities) for:

`ORCDRC` numeric; soil organic carbon content in permille

`PHIHOX` numeric; pH index measured in water solution

`PHIKCL` numeric; pH index measured in KCl solution

`BLDFIE` numeric; bulk density (fine earth) in kg per cubic meter

`CECSOL` numeric; Cation Exchange Capacity of soil

`SNDPPT` numeric; weight percentage of the sand particles (0.05–2 mm)

`SLTPPT` numeric; weight percentage of the silt particles (0.0002–0.05 mm)

`CLYPPT` numeric; weight percentage of the clay particles (<0.0002 mm)

CRFVOL numeric; volumetric percentage of coarse fragments (>2 mm)

TAXOUSDA factor; Keys to Soil Taxonomy suborders

TAXGWRB factor; World Reference Base groups

TAXNWRB factor; World Reference Base legend for SoilGrids250m

## Note

Breaks for continuous soil properties were determined using the quantiles function and by visually inspecting the histograms to maximize the contrast in output maps. Based on a compilation of global soil profile data (see ISRIC's World Soil Information Service WoSIS).

## Author(s)

Tomislav Hengl

## References


- Batjes, N. H., Ribeiro, E., van Oostrum, A., Leenaars, J., Hengl, T., & de Jesus, J. M. (2017). WoSIS: providing standardised soil profile data for the world. Earth System Science Data, 9(1), 1. https://doi.org/10.5194/essd-9-1-2017


## Examples

```
data(soil.legends)
pal <- soil.legends$ORCDRC$COLOR
names(pal) <- signif((soil.legends$ORCDRC$MAX +
    soil.legends$ORCDRC$MIN)/2, 3)
pal
## Munsell color codes:
data(munsell)
str(munsell)
```

---

# SpatialComponents-class

*A class for gridded components derived using the* spc *method*

---

## Description

A class containing a list of gridded components and results of principal component analysis.

## Slots

**predicted:** object of class "SpatialPixelsDataFrame"; predicted values for components

**pca:** object of class "list"; output objects from the stats::prcomp process — contains objects: 'stdev', 'rotation', 'center' and 'scale'

## Author(s)

Tomislav Hengl

## SpatialMemberships-class

*A class for membership maps derived using the* fkmeans *classification*

### Description

A class containing a list of gridded maps and results of model fitting.

### Slots

predicted: object of class "SpatialPixelsDataFrame"; predicted values (factor)

model: object of class "multinom"; output object from the nnet::multinom method

mu: object of class "SpatialPixelsDataFrame"; a list of predicted memberships

class.c: object of class "matrix"; class centres

class.sd: object of class "matrix"; class deviations

confusion: object of class "matrix"; confusion matrix

### Author(s)

Tomislav Hengl

### See Also

SpatialComponents-class

## spc,SpatialPixelsDataFrame-method

*Generate Principal Components using SpatialPixelsDataFrame object*

### Description

Combines the stats::prcomp method and predicts a list principal components for an object of type "SpatialPixelsDataFrame".

### Usage

```
## S4 method for signature 'SpatialPixelsDataFrame'
spc(obj, formulaString, scale. = TRUE, silent = FALSE)
```

### Arguments

| | |
|---|---|
| `obj` | SpatialPixelsDataFrame. |
| `formulaString` | optional model definition. |
| `scale.` | scale all numbers. |
| `silent` | silent output. |

### Value

Object of class `SpatialComponents`. List of grids with generic names `PC1`,...,`PCp`, where `p` is the total number of input grids.

### Note

This method assumes that the input covariates are cross-correlated and hence their overlap can be reduced. The input variables are scaled by default and the missing values will be replaced with 0 values to reduce loss of data due to missing pixels.

### Author(s)

Tom Hengl

### Examples

```
library(plotKML)
library(sp)
pal = rev(rainbow(65)[1:48])
data(eberg_grid)
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
formulaString <- ~ PRMGEO6+DEMSRT6+TWISRT6+TIRAST6
eberg_spc <- spc(eberg_grid, formulaString)
names(eberg_spc@predicted) # 11 components on the end;

## plot maps:
rd = range(eberg_spc@predicted@data[,1], na.rm=TRUE)
sq = seq(rd[1], rd[2], length.out=48)
spplot(eberg_spc@predicted[1:4], at=sq, col.regions=pal)
```

---

## `spfkm,formula,SpatialPointsDataFrame,SpatialPixelsDataFrame-method`

*Fit a supervised fuzzy kmeans model and predict memberships*

---

### Description

Runs supervised fuzzy *k*-means (Hengl et al., 2004) using a list of covariates layers provided as `"SpatialPixelsDataFrame-class"` object. If class centres and variances are not provided, it first fits a multinomial logistic regression model (`spmultinom`), then predicts the class centres and variances based on the output from the `nnet::multinom`.

## Usage

```
## S4 method for signature
## 'formula,SpatialPointsDataFrame,SpatialPixelsDataFrame'
spfkm(
  formulaString,
  observations,
  covariates,
  class.c = NULL,
  class.sd = NULL,
  fuzzy.e = 1.2
)
```

## Arguments

| | |
|---|---|
| formulaString | formula. |
| observations | SpatialPointsDataFrame. |
| covariates | SpatialPixelsDataFrame. |
| class.c | class centers (per variable). |
| class.sd | class standard deviation (per variable). |
| fuzzy.e | fuzzy coefficient. |

## Value

A fuzzy kmeans model

## Author(s)

Tom Hengl

## Examples

```
library(plotKML)
library(sp)

data(eberg)
# subset to 20%:
eberg <- eberg[runif(nrow(eberg))<.2,]
data(eberg_grid)
coordinates(eberg) <- ~X+Y
proj4string(eberg) <- CRS("+init=epsg:31467")
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
# derive soil predictive components:
eberg_spc <- spc(eberg_grid, ~PRMGEO6+DEMSRT6+TWISRT6+TIRAST6)
# predict memberships:
formulaString = soiltype ~ PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+PC10
eberg_sm <- spfkm(formulaString, eberg, eberg_spc@predicted)

# plot memberships:
```

```
pal = seq(0, 1, 1/50)
spplot(eberg_sm@mu, col.regions=grey(rev(pal)))
```

---

spmultinom,formula,SpatialPointsDataFrame,SpatialPixelsDataFrame-method
*Fits a multinomial logistic regression to spatial data*

---

## Description

Fits a multinomial logistic regression to spatial data

## Usage

```
## S4 method for signature
## 'formula,SpatialPointsDataFrame,SpatialPixelsDataFrame'
spmultinom(
  formulaString,
  observations,
  covariates,
  class.stats = TRUE,
  predict.probs = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| formulaString | formula. |
| observations | SpatialPointsDataFrame. |
| covariates | SpatialPixelsDataFrame. |
| class.stats | class statistics. |
| predict.probs | specify whether to derive probabilities. |
| ... | optional arguments. |

## Value

A multinomial logistic regression model

spsample.prob,SpatialPoints,SpatialPixelsDataFrame-method
*Estimate occurrence probabilities of a sampling plan (points)*

## Description

Estimates occurrence probabilities as an average between the kernel density estimation (spreading of points in geographical space) and MaxLike analysis (spreading of points in feature space). The output 'iprob' indicates whether the sampling plan has systematically missed some important locations / features, and can be used as an input for modelling (e.g. as weights for regression modeling).

## Usage

```
## S4 method for signature 'SpatialPoints,SpatialPixelsDataFrame'
spsample.prob(observations, covariates, quant.nndist = 0.95, n.sigma, ...)
```

## Arguments

| | |
|---|---|
| observations | SpatialPoints. |
| covariates | SpatialPixelsDataFrame. |
| quant.nndist | quantile used for the threshold distance. |
| n.sigma | sigma parameter for density estimation. |
| ... | optional arguments. |

## Value

Returns a list of objects where 'iprob' ("SpatialPixelsDataFrame") is the map showing the estimated occurrence probabilities.

## Note

Occurrence probabilities for geographical space are derived using kernel density estimator. The sampling intensities are converted to probabilities by deviding the sampling intensity by the maximum sampling intensity for the study area (Baddeley, 2008). The occurrence probabilities for feature space are determined using MaxLike algorithm (Royle et al., 2012). The lower the average occurrence probability for the whole study area, the lower the representation efficiency of a sampling plan.
MaxLike function might fail to produce predictions (e.g. if not at least one continuous covariate is provided and if the optim function is not able to find the global optima) in which case an error message is generated. Running Principal Component analysis i.e. standardizing the covariates prior to running spsample.prob is, thus, highly recommended.
This function can be time consuming for large grids.

## Author(s)

Tom Hengl

## References


- Baddeley, A. (2008) Analysing spatial point patterns in R. Technical report, CSIRO Australia. Version 4.
- Royle, J.A., Chandler, R.B., Yackulic, C. and J. D. Nichols. (2012) Likelihood analysis of species occurrence probability from presence-only data for modelling species distributions. Methods in Ecology and Evolution.


## Examples

```
library(plotKML)
library(maxlike)
library(spatstat)
library(maptools)

data(eberg)
data(eberg_grid)
## existing sampling plan:
sel <- runif(nrow(eberg)) < .2
eberg.xy <- eberg[sel,c("X","Y")]
coordinates(eberg.xy) <- ~X+Y
proj4string(eberg.xy) <- CRS("+init=epsg:31467")
## covariates:
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
## convert to continuous independent covariates:
formulaString <- ~ PRMGEO6+DEMSRT6+TWISRT6+TIRAST6
eberg_spc <- spc(eberg_grid, formulaString)

## derive occurrence probability:
covs <- eberg_spc@predicted[1:8]
iprob <- spsample.prob(eberg.xy, covs)
## Note: obvious omission areas:
hist(iprob[[1]]@data[,1], col="gray")

## compare with random sampling:
rnd <- spsample(eberg_grid, type="random",
               n=length(iprob[["observations"]]))
iprob2 <- spsample.prob(rnd, covs)

## compare the two next to each other:
op <- par(mfrow=c(1,2))
plot(raster(iprob[[1]]), zlim=c(0,1), col=SAGA_pal[[1]])
points(iprob[["observations"]])
plot(raster(iprob2[[1]]), zlim=c(0,1), col=SAGA_pal[[1]])
points(iprob2[["observations"]])
par(op)
dev.off()

## fit a weighted lm:
eberg.xy <- eberg[sel,c("SNDMHT_A","X","Y")]
```

```
coordinates(eberg.xy) <- ~X+Y
proj4string(eberg.xy) <- CRS("+init=epsg:31467")
eberg.xy$iprob <- over(eberg.xy, iprob[[1]])$iprob
eberg.xy@data <- cbind(eberg.xy@data, over(eberg.xy, covs))
fs <- as.formula(paste("SNDMHT_A ~ ",
                       paste(names(covs), collapse="+")))
## the lower the occurrence probability, the higher the weight:
w <- 1/eberg.xy$iprob
m <- lm(fs, eberg.xy, weights=w)
summary(m)
## compare to standard lm:
m0 <- lm(fs, eberg.xy)
summary(m)$adj.r.squared
summary(m0)$adj.r.squared
```

---

`tile,RasterLayer-method` *Tile spatial layers*

---

## Description

Tile spatial layers

## Usage

```
## S4 method for signature 'SpatialPointsDataFrame'
tile(x, y, block.x, ...)
## S4 method for signature 'SpatialPixelsDataFrame'
tile(x, y, block.x, ...)
## S4 method for signature 'SpatialPolygonsDataFrame'
tile(x, y, block.x, tmp.file = TRUE,
                                    program, show.output.on.console = FALSE, ...)
## S4 method for signature 'SpatialLinesDataFrame'
tile(x, y, block.x, tmp.file = TRUE,
                                    program, show.output.on.console = FALSE, ...)
## S4 method for signature 'RasterLayer'
tile(x, y, block.x, tmp.file = TRUE,
                                program, show.output.on.console = FALSE, ...)
```

## Arguments

| | |
|---|---|
| x | RasterLayer. |
| y | either points, pixels, polygons or lines. |
| block.x | size of the block in x direction. |
| tmp.file | temporary file name. |
| program | optional location of the gdalwarp. |

`show.output.on.console`
: shows progress.

`...`
: optional argument.

## Value

Regular tiling system

## Author(s)

Tom Hengl

---

`train.spLearner,SpatialPointsDataFrame,ANY,SpatialPixelsDataFrame-method`
*Train a spatial prediction and/or interpolation model using Ensemble Machine Learning*

---

## Description

Automated spatial predictions and/or interpolation using Ensemble Machine Learning. Extends functionality of the mlr package. Suitable for predicting numeric, binomial and factor-type variables.

## Usage

```
## S4 method for signature 'SpatialPointsDataFrame,ANY,SpatialPixelsDataFrame'
train.spLearner(
  observations,
  formulaString,
  covariates,
  SL.library,
  family = stats::gaussian(),
  method = "stack.cv",
  predict.type,
  super.learner = "regr.lm",
  subsets = 5,
  lambda = 0.5,
  cov.model = "exponential",
  subsample = 10000,
  parallel = "multicore",
  buffer.dist = FALSE,
  oblique.coords = TRUE,
  theta.list = seq(0, 180, length.out = 14) * pi/180,
  spc = TRUE,
  id = NULL,
  weights = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| observations | SpatialPointsDataFrame. |
| formulaString | ANY. |
| covariates | SpatialPixelsDataFrame. |
| SL.library | List of learners, |
| family | Family e.g. gaussian(), |
| method | Ensemble stacking method (see makeStackedLearner) usually stack.cv, |
| predict.type | Prediction type 'prob' or 'response', |
| super.learner | Ensemble stacking model usually regr.lm, |
| subsets | Number of subsets for repeated CV, |
| lambda | Target variable transformation (0.5 or 1), |
| cov.model | Covariance model for variogram fitting, |
| subsample | For large datasets consider random subsetting training data, |
| parallel | logical, Initiate parallel processing, |
| buffer.dist | Specify whether to use buffer distances to points as covariates, |
| oblique.coords | Specify whether to use oblique coordinates as covariates, |
| theta.list | List of angles (in radians) used to derive oblique coordinates, |
| spc | specifies whether to apply principal components transformation. |
| id | Id column name to control clusters of data, |
| weights | Optional weights (per row) that learners will use to account for variable data quality, |
| ... | other arguments that can be passed on to mlr::makeStackedLearner, |

## Value

object of class spLearner, which contains fitted model, variogram model and spatial grid used for Cross-validation.

## Note

By default uses oblique coordinates (rotated coordinates) as described in Moller et al. (2020) to account for geographical distribution of values. Buffer geographical distances can be added by setting buffer.dist=TRUE. Using either oblique coordinates and/or buffer distances is not recommended for point data set with distinct spatial clustering. Effects of adding geographical distances into modeling are explained in detail in Hengl et al. (2018). Default learners used for regression are: c("regr.ranger","regr.ksvm","regr.nnet","regr.cvglmnet"). Default learners used for classification / binomial variables are: c("classif.ranger","classif.svm","classif.multinom"), with predict.type="prob". When using method = "stack.cv" each training and prediction round could produce somewhat different results due to randomization of CV. Prediction errors are derived by default using quantreg (Quantile Regression) option in the ranger package (Meinshausen, 2006).

## Author(s)

Tom Hengl

## Examples

```
library(mlr)
library(rgdal)
library(geoR)
library(plotKML)
library(xgboost)
library(kernlab)
library(ranger)
library(glmnet)
library(boot)
library(raster)
demo(meuse, echo=FALSE)
## Regression:
sl = c("regr.ranger", "regr.nnet", "regr.ksvm")
m <- train.spLearner(meuse["lead"], covariates=meuse.grid[,c("dist","ffreq")],
      lambda=0, parallel=FALSE, SL.library=sl)
summary(m@spModel$learner.model$super.model$learner.model)

## regression-matrix:
str(m@vgmModel$observations@data)
meuse.y <- predict(m)
plot(raster(meuse.y$pred["response"]), col=R_pal[["rainbow_75"]][4:20],
   main="Predictions spLearner", axes=FALSE, box=FALSE)

library(parallelMap)
library(deepnet)
## Regression with default settings:
m <- train.spLearner(meuse["zinc"], covariates=meuse.grid[,c("dist","ffreq")],
        parallel=FALSE, lambda = 0)
## Ensemble model (meta-learner):
summary(m@spModel$learner.model$super.model$learner.model)
meuse.y <- predict(m)
op <- par(mfrow=c(1,2), oma=c(0,0,0,1), mar=c(0,0,4,3))
plot(raster(meuse.y$pred["response"]), col=R_pal[["rainbow_75"]][4:20],
   main="Predictions spLearner", axes=FALSE, box=FALSE)
points(meuse, pch="+")
plot(raster(meuse.y$pred["model.error"]), col=rev(bpy.colors()),
   main="Prediction errors", axes=FALSE, box=FALSE)
points(meuse, pch="+")
par(op)
dev.off()

## Classification:
SL.library <- c("classif.ranger", "classif.xgboost", "classif.nnTrain")
mC <- train.spLearner(meuse["soil"], covariates=meuse.grid[,c("dist","ffreq")],
   SL.library = SL.library, super.learner = "classif.glmnet", parallel=FALSE)
meuse.soil <- predict(mC)
spplot(meuse.soil$pred[grep("prob.", names(meuse.soil$pred))],
        col.regions=SAGA_pal[["SG_COLORS_YELLOW_RED"]], zlim=c(0,1))
spplot(meuse.soil$pred[grep("error.", names(meuse.soil$pred))],
         col.regions=rev(bpy.colors()))
```

```
## SIC1997
data("sic1997")
X <- sic1997$swiss1km[c("CHELSA_rainfall","DEM")]
mR <- train.spLearner(sic1997$daily.rainfall, covariates=X, lambda=1, parallel=FALSE)
summary(mR@spModel$learner.model$super.model$learner.model)
rainfall1km <- predict(mR)
op <- par(mfrow=c(1,2), oma=c(0,0,0,1), mar=c(0,0,4,3))
plot(raster(rainfall1km$pred["response"]), col=R_pal[["rainbow_75"]][4:20],
    main="Predictions spLearner", axes=FALSE, box=FALSE)
points(sic1997$daily.rainfall, pch="+")
plot(raster(rainfall1km$pred["model.error"]), col=rev(bpy.colors()),
    main="Prediction errors", axes=FALSE, box=FALSE)
points(sic1997$daily.rainfall, pch="+")
par(op)

## Ebergotzen data set
data(eberg_grid)
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
data(eberg)
eb.s <- sample.int(nrow(eberg), 1400)
eberg <- eberg[eb.s,]
coordinates(eberg) <- ~X+Y
proj4string(eberg) <- CRS("+init=epsg:31467")
## Binomial variable
summary(eberg$TAXGRSC)
eberg$Parabraunerde <- ifelse(eberg$TAXGRSC=="Parabraunerde", 1, 0)
X <- eberg_grid[c("PRMGEO6","DEMSRT6","TWISRT6","TIRAST6")]
mB <- train.spLearner(eberg["Parabraunerde"], covariates=X,
   family=binomial(), cov.model = "nugget", parallel=FALSE)
eberg.Parabraunerde <- predict(mB)
plot(raster(eberg.Parabraunerde$pred["prob.1"]),
   col=SAGA_pal[["SG_COLORS_YELLOW_RED"]], zlim=c(0,1))
points(eberg["Parabraunerde"], pch="+")

## Factor variable:
data(eberg)
coordinates(eberg) <- ~X+Y
proj4string(eberg) <- CRS("+init=epsg:31467")
X <- eberg_grid[c("PRMGEO6","DEMSRT6","TWISRT6","TIRAST6")]
mF <- train.spLearner(eberg["TAXGRSC"], covariates=X, parallel=FALSE)
TAXGRSC <- predict(mF)
plot(stack(TAXGRSC$pred[grep("prob.", names(TAXGRSC$pred))]),
    col=SAGA_pal[["SG_COLORS_YELLOW_RED"]], zlim=c(0,1))
plot(stack(TAXGRSC$pred[grep("error.", names(TAXGRSC$pred))]),
    col=SAGA_pal[["SG_COLORS_YELLOW_BLUE"]], zlim=c(0,0.45))
dev.off()
```

`train.spLearner.matrix` *Train a spatial prediction and/or interpolation model using Ensemble Machine Learning from a regression/classification matrix*

## Description

Train a spatial prediction and/or interpolation model using Ensemble Machine Learning from a regression/classification matrix

## Usage

```
train.spLearner.matrix(
  observations,
  formulaString,
  covariates,
  SL.library,
  family = stats::gaussian(),
  method = "stack.cv",
  predict.type,
  super.learner,
  subsets = 5,
  lambda = 0.5,
  cov.model = "exponential",
  subsample = 10000,
  parallel = "multicore",
  cell.size,
  id = NULL,
  weights = NULL,
  quantreg = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `observations` | Data frame regression matrix, |
| `formulaString` | Model formula, |
| `covariates` | SpatialPixelsDataFrame object, |
| `SL.library` | List of learners, |
| `family` | Family e.g. gaussian(), |
| `method` | Ensemble stacking method (see makeStackedLearner), |
| `predict.type` | Prediction type 'prob' or 'response', |
| `super.learner` | Ensemble stacking model usually `regr.lm`, |
| `subsets` | Number of subsets for repeated CV, |
| `lambda` | Target variable transformation for geoR (0.5 or 1), |

| | |
|---|---|
| cov.model | Covariance model for variogram fitting, |
| subsample | For large datasets consider random subsetting training data, |
| parallel | Initiate parellel processing, |
| cell.size | Block size for spatial Cross-validation, |
| id | Id column name to control clusters of data, |
| weights | Optional weights (per row) that learners will use to account for variable data quality, |
| quantreg | Fit additional ranger model as meta-learner to allow for derivation of prediction intervals, |
| ... | other arguments that can be passed on to mlr::makeStackedLearner, |

## Value

Object of class spLearner

## Author(s)

Tom Hengl

---

# TT2tri *Soil texture class to texture fractions conversion*

---

## Description

Soil texture class to texture fractions conversion

## Usage

```
TT2tri(
  TT.class,
  se.fit = TRUE,
  TT.im = NULL,
  soil.var = "TEXMHT",
 levs = c("S", "LS", "SL", "SCL", "SiL", "SiCL", "CL", "L", "Si", "SC", "SiC", "C",
    "HC")
)
```

## Arguments

| | |
|---|---|
| TT.class | based on the soiltexture package |
| se.fit | derive errors |
| TT.im | soil texture triangle image |
| soil.var | column name in the TT.im file |
| levs | texture class legends (USDA system) |

## Value

Data frame with estimated sand, silt and clay values

## Examples

```
library(soiltexture)
## convert textures by hand to sand, silt and clay:
TEXMHT <- c("CL","C","SiL","SiL","missing")
x <- TT2tri(TEXMHT)
x
```

---

# tune.spLearner,spLearner-method

*Optimize spLearner by fine-tuning parameters and running feature selection*

---

## Description

Optimize spLearner by fine-tuning parameters and running feature selection

## Usage

```
## S4 method for signature 'spLearner'
tune.spLearner(
  object,
  num.trees = 85,
  blocking,
  discrete_ps,
  rdesc = mlr::makeResampleDesc("CV", iters = 2L),
  inner = mlr::makeResampleDesc("Holdout"),
  maxit = 20,
  xg.model_Params,
  xg.skip = FALSE,
  parallel = "multicore",
  hzn_depth = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| object | spLearner object (unoptimized), |
| num.trees | number of random forest trees, |
| blocking | blocking columns, |
| discrete_ps | settings for random forest, |
| rdesc | resampling method for fine-tuning, |
| inner | resampling method for feature selection, |

| | |
|---|---|
| maxit | maximum number of iterations for feature selection, |
| xg.model_Params | xgboost parameter set, |
| xg.skip | logical, should the tuning of the XGboost should be skipped? |
| parallel | Initiate parallel processing, |
| hzn_depth | specify whether horizon depth available in the training dataframe, |
| ... | other arguments that can be passed on to mlr::makeStackedLearner, |

## Value

optimized object of type spLearner

## Note

Currently requires that two base learners are regr.ranger and regr.xgboost, and that there are at least 3 base learners in total. Fine-tuning and feature selection can be quite computational and it is highly recommended to start with smaller subsets of data and then measure processing time. The function mlr::makeFeatSelWrapper can result in errors if the covariates have a low variance or follow a zero-inflated distribution. Reducing the number of features via feature selection and fine-tuning of the Random Forest mtry and XGboost parameters, however, can result in significantly higher prediction speed and accuracy.

## Author(s)

Tom Hengl

## Examples

```
library(mlr)
library(ParamHelpers)
library(geoR)
library(xgboost)
library(kernlab)
library(ranger)
library(glmnet)
library(boot)
library(raster)
demo(meuse, echo=FALSE)
## Regression:
sl = c("regr.ranger", "regr.xgboost", "regr.ksvm", "regr.cvglmnet")
m <- train.spLearner(meuse["lead"], covariates=meuse.grid[,c("dist","ffreq")],
      lambda=0, parallel=FALSE, SL.library=sl)
summary(m@spModel$learner.model$super.model$learner.model)
## Optimize model:
m0 <- tune.spLearner(m, xg.skip = TRUE, parallel=FALSE)
summary(m0@spModel$learner.model$super.model$learner.model)
```

USDA.TT.im *Probability density for texture triangle*

## Description

Probability density for texture triangle (USDA system) based on global soil profile data (see ISRIC WoSIS).

## Usage

```
data(USDA.TT.im)
```

## Format

The USDA.TT.im data frame contains the following columns:

v numeric; probability density derived using the soiltexture::TT.kde2d function and global soil profile data

TEXMHT factor; USDA soil texture class estimated by hand (one of the following: "C", "SiC", "SC", "CL", "SiCL", "SCL", "L", "SiL", "SL", "Si", "LS", "S")

s1 numeric; horizontal coordinate (sand content 0–1) in the texture triangle system

s2 numeric; vertical coordinate (0–0.85) in the texture triangle system

## Note

Texture by hand class can be converted to sand, silt, clay content fractions by using the TT2tri function. This function uses the v column in the USDA.TT.im (i.e. prior probability densities) to adjust for texture fraction combinations that are more probable.

## Author(s)

Tomislav Hengl

## References


- Skaggs, T. H., Arya, L. M., Shouse, P. J., Mohanty, B. P., (2001) Estimating Particle-Size Distribution from Limited Soil Texture Data. Soil Science Society of America Journal 65 (4): 1038-1044.


## Examples

```
## plot prior probabilities:
library(sp)
data(USDA.TT.im)
gridded(USDA.TT.im) <- ~s1+s2
spplot(USDA.TT.im["v"])
```

# Index