Package 'GSIF'

July 20, 2015

Type Package

Title Global Soil Information Facilities

Version 0.4-7

Date 2015-07-14

Maintainer Tomislav Hengl <tom.hengl@isric.org>

Depends R (>= 2.15.0)

Imports methods, sp (>= 1.0-8), RSAGA, dismo, rgdal, raster, aqp, plotKML, gstat, stats, plyr, grDevices, graphics

Suggests rjson, RCurl, soiltexture, spatstat, stringr, XML, boot, splines, mda, psych, nortest, rpart, quantregForest, randomForest, nlme, reshape, fossil, AICcmodavg, maptools, nnet, SDMTools, rJava (>= 0.5-0), spacetime, gdalUtils, tools, maxlike, Hmisc

Description Global Soil Information Facilities - tools (standards and functions) and sample datasets for global soil mapping.

License GPL

URL http://gsif.r-forge.r-project.org/

LazyLoad yes

NeedsCompilation no

Author Tomislav Hengl [cre, aut], Bas Kempen [ctb], Gerard Heuvelink [ctb], Brendan Malone [ctb]

Repository CRAN

Date/Publication 2015-07-20 18:08:46

R topics documented:

afsp *Africa Soil Profiles Database*

Description

A merge of the Africa Soil Profiles Database (AFSP) with 17,000+ geo-referenced legacy soil profile records, and AfSIS Sentinel Site database with 9000+ sampling locations.

Usage

```
data(afsp)
```

Format

The `afsp` 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 (`ProfileID` in the AFSP)

`SOURCEDB` factor; source data base

`LONWGS84` numeric; longitude in decimal degrees on the WGS84 datum (`X_LonDD` in the AFSP)

`LATWGS84` numeric; latitude in decimal degrees on the WGS84 datum (`Y_LatDD` in the AFSP)

`TIMESTRR` character; the date on which this particular soil was described or sampled (`T_Year` in the AFSP)

`TAXGWRB` factor; abbreviated soil group based on the WRB classification system (`WRB06rg` in the AFSP)

`TAXNUSDA` factor; Keys to Soil Taxonomy taxon name e.g. `"Plinthic Udoxic Dystropept"` (`USDA` in the AFSP)

`BDRICM` numeric; depth to bedrock in cm

`DRAINFAO` factor; drainage class based on the FAO guidelines for soil description: `E` (excessively drained), `S` (somewhat excessively drained), `W` (well drained), `M` (moderately well drained), `I` (somewhat poorly drained) and `V` (very poorly drained)

Horizons table contains the following columns:

`SOURCEID` factor; a short label to help a user identify a particular site

`UHDICM` numeric; upper horizon depth from the surface in cm

`LHDICM` numeric; lower horizon depth from the surface in cm

`MCOMNS` factor; Munsell color moist

`ORCDRC` numeric; soil organic carbon content in permilles

`PHIHOX` numeric; pH index measured in water solution

`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; volume percentage of coarse fragments (> 2 mm)

`BLD` numeric;bulk density in tonnes per cubic-meter

`CEC` numeric; Cation exchange capacity (fine earth fraction) in cmolc/kg

`NTO` numeric; total N content in permille or g/kg

`EMGX` numeric; exchangable Mg in cmolc/kg

Author(s)

The Africa Soil Profiles Database have been prepared by Johan Leenaars <johan.leenaars@wur.nl>. This is a subset of the original database that can be downloaded via www.isric.org. The AfSIS Sentinel Site database is one of the main deliverables of the Africa Soil Information Service project.

References

- Leenaars, J.G.B. (2014) Africa Soil Profiles Database, Version 1.2. A compilation of geo-referenced and standardized legacy soil profile data for Sub Saharan Africa (with dataset). ISRIC report 2012/03. Africa Soil Information Service (AfSIS) project and ISRIC — World Soil Information, Wageningen, the Netherlands.
- Africa Soil Information Service (http://africasoils.net)

Examples

```
## Not run:
library(rgdal)
library(aqp)
library(sp)

data(afsp)
sites <- afsp$sites
coordinates(sites) <- ~ LONWGS84 + LATWGS84
proj4string(sites) <- "+proj=longlat +datum=WGS84"
## obtain country borders:
library(maps)
country.m = map('world', plot=FALSE, fill=TRUE)
IDs <- sapply(strsplit(country.m$names, ":"), function(x) x[1])
require(maptools)
country <- as(map2SpatialPolygons(country.m, IDs=IDs), "SpatialLines")
proj4string(country) = "+proj=longlat +datum=WGS84"
## overlay and plot points and maps:
plot(country, col="darkgrey", xlim=c(-25.3,57.8), ylim=c(-34.8, 37.4))
points(sites, pch=21, bg="white", cex=.6, col="black")

## End(Not run)
```

`as.data.frame` *Converts an object of class* `"SoilProfileCollection"` *to a data frame*

Description

Converts an object of class `"SoilProfileCollection"` to an object of class `"data.frame"` with both site and horizon data sorted in one row. Each original column name in the horizons table receives a sufix `*_A, B, ..., Z` where alphabetic letters represent horizon sequence.

Usage

```
## S4 method for signature 'SoilProfileCollection'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	object of class `"SoilProfileCollection"`
`row.names`	character; giving the row names for the data frame (missing values are not allowed)
`optional`	logical; if 'TRUE', setting row names and converting column names (to syntactic names: see 'make.names') is optional
`...`	optional arguments

Details

The advantage of converting the `"SoilProfileCollection"` data to a single table is that, once both tables have been merged to a single data frame, it can be more easily exported and visualized in a GIS and/or imported into a data base.

Note

Few profiles with a large number of horizons can make the whole data frame become large. Consider removing such locations or aggregating measured values per horizon to a lower number of horizons.

Author(s)

Tomislav Hengl and Brendan Malone

See Also

`as.geosamples`, `mpspline`

Examples

```
library(aqp)
library(plyr)
library(rgdal)
library(sp)
## sample profile from Nigeria:
lon = 3.90; lat = 7.50; id = "ISRIC:NG0017"; FAO1988 = "LXp"
top = c(0, 18, 36, 65, 87, 127)
bottom = c(18, 36, 65, 87, 127, 181)
ORCDRC = c(18.4, 4.4, 3.6, 3.6, 3.2, 1.2)
## prepare a SoilProfileCollection:
prof1 <- join(data.frame(id, top, bottom, ORCDRC),
    data.frame(id, lon, lat, FAO1988),
    type='inner')
depths(prof1) <- id ~ top + bottom
site(prof1) <- ~ lon + lat + FAO1988
coordinates(prof1) <- ~ lon + lat
proj4string(prof1) <- CRS("+proj=longlat +datum=WGS84")
## convert to a simple table:
x <- as.data.frame(prof1)
str(x)
## horizons only
horizons <- getHorizons(x, idcol="id", sel=c("top", "bottom", "ORCDRC"))
horizons
```

---

as.geosamples *Converts an object to geosamples class*

Description

Converts an object of class "SoilProfileCollection" or "SpatialPointsDataFrame" to an object of class "geosamples" with all measurements broken into individual records. Geosamples are standardized spatially and temporally referenced samples from the Earth's surface.

Usage

```
## S4 method for signature 'SoilProfileCollection'
as.geosamples(obj,
   registry = as.character(NA), sample.area = 1, mxd = 2, TimeSpan.begin, TimeSpan.end)
## S4 method for signature 'SpatialPointsDataFrame'
as.geosamples(obj,
   registry = as.character(NA), sample.area = 1, mxd = 2, TimeSpan.begin, TimeSpan.end)
```

Arguments

obj	object of class "SoilProfileCollection"
...	optional arguments

registry	URI specifying the metadata registry (web-service that carries all metadata connected to the certain method ID and/or sample ID)
sample.area	standard sample area in square meters (assumed to be 1 by 1 m)
mxd	maximum depth of interest in meters
TimeSpan.begin	vector of class "POSIXct"; begin of the measurement period
TimeSpan.end	vector of class "POSIXct"; end of the measurement period

Value

Returns an object of type "geosamples". Many columns required by the "geosamples" class might be not available and will result in NA values. To ensure compatibility, when building an object of type "SoilProfilesCollection", use some standard naming convention to attach attributes to each measurement (horizons and sites slots in the "SoilProfileCollection-class"):

"locationError" can be used to attach location errors in meters to each spatial location

"sampleArea" can be used to attach spatial support to each measurement (usually 1 by 1 meter)

"measurementError" can be used to attach specific measurement errors to each measurement in both site and horizons table

"IGSN" can be used to attach the unique identifier (International Geo Sample Number) to each specific observation (corresponds to the "observationid" column)

Author(s)

Tomislav Hengl and Hannes I. Reuter

See Also

geosamples-class, as.data.frame, aqp::SoilProfileCollection

Examples

```
library(aqp)
library(plyr)
library(rgdal)
library(sp)
# sample profile from Nigeria:
lon = 3.90; lat = 7.50; time = as.POSIXct("1978", format="%Y")
id = "ISRIC:NG0017"; TAXNFAO8 = "LXp"
top = c(0, 18, 36, 65, 87, 127)
bottom = c(18, 36, 65, 87, 127, 181)
ORCDRC = c(18.4, 4.4, 3.6, 3.6, 3.2, 1.2)
methodid = c("TAXNFAO8", "ORCDRC")
description = c("FAO 1988 classification system group",
    "Method of Walkley-Black (Org. matter = Org. C x 1.72)")
units = c("FAO 1988 classes", "permille")
detectionLimit = c(as.character(NA), "0.1")
# prepare a SoilProfileCollection:
prof1 <- join(data.frame(id, top, bottom, ORCDRC),
    data.frame(id, lon, lat, time, TAXNFAO8), type='inner')
```

```
depths(prof1) <- id ~ top + bottom
site(prof1) <- ~ lon + lat + time + TAXNFAO8
coordinates(prof1) <- ~ lon + lat + time
proj4string(prof1) <- CRS("+proj=longlat +datum=WGS84")
# add measurement errors:
attr(prof1@horizons$ORCDRC, "measurementError") <- c(1.5, 0.5, 0.5, 0.5, 0.5, 0.5)
attr(prof1@sp@coords, "locationError") <- 1500
# add the metadata:
prof1@metadata <- data.frame(methodid, description, units, detectionLimit)
# convert to geosamples:
x <- as.geosamples(prof1)
x
# print only the sampled values of ORCDRC:
ORCDRC <- subset(x, "ORCDRC")
ORCDRC[,c("sampleid", "altitude", "observedValue")]

# convert object of type SpatialPointsDataFrame:
data(meuse)
# prepare columns:
names(meuse)[which(names(meuse)=="x")] = "longitude"
names(meuse)[which(names(meuse)=="y")] = "latitude"
meuse$altitude = -.15
meuse$time = unclass(as.POSIXct("1992-01-01"))
coordinates(meuse) <- ~ longitude + latitude + altitude + time
proj4string(meuse) <- CRS("+init=epsg:28992")
library(plotKML)
hm <- reproject(meuse[,c("zinc", "copper")])
hm.geo <- as.geosamples(hm)
hm.geo
```

`autopredict-methods` *Auto predict numeric or factor type variables*

Description

Fits either geostatistical model via the `fit.gstatModel` function (in the case of numeric variable) or a multinomial logistic regression model via the `spmultinom` function (factor-type variable) and generates predictions.

Usage

```
## S4 method for signature 'SpatialPointsDataFrame,SpatialPixelsDataFrame'
autopredict(target,covariates,
     auto.plot=TRUE, ...)
```

Arguments

`target`	object of class `"SpatialPointsDataFrame"` containing observations of the target variable

covariates	object of class "SpatialPixelsDataFrame"; spatial covariates
auto.plot	logical; specifies whether to immediately plot the data via the plotKML function
...	other optional arguments that can be passed to fit.gstatModel or spmultinom

Author(s)

Tomislav Hengl

See Also

fit.gstatModel, spmultinom

Examples

```
## Ebergotzen data:
library(sp)
library(gstat)
library(randomForest)
library(plotKML)

## load input data:
data(eberg)
eberg <- eberg[runif(nrow(eberg))<.1,]
coordinates(eberg) <- ~X+Y
proj4string(eberg) <- CRS("+init=epsg:31467")
data(eberg_grid)
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")

## predict sand content:
SNDMHT_A <- autopredict(eberg["SNDMHT_A"], eberg_grid,
   auto.plot=FALSE)
plot(SNDMHT_A)

## predict soil types:
soiltype <- autopredict(eberg["soiltype"], eberg_grid,
  auto.plot=FALSE)
spplot(soiltype@predicted)
```

AWCPTF *Available soil water capacity*

Description

Derive available soil water capacity (in cubic-meter per cubic-meter) based on a Pedo-Transfer Function developed using the Africa Soil Profile Database (Hodnett and Tomasella, 2002; Wösten et al. 2013).

Usage

```
AWCPTF(SNDPPT, SLTPPT, CLYPPT, ORCDRC,
    BLD=1400, CEC, PHIHOX, h1=-10, h2=-20, h3=-31.6,
    pwp=-1585, PTF.coef, fix.values=TRUE, print.coef=TRUE)
```

Arguments

SNDPPT	numeric; sand content in percent
SLTPPT	numeric; silt content in percent
CLYPPT	numeric; clay content in percent
ORCDRC	numeric; soil organic carbon concentration in permille or g / kg
BLD	numeric; bulk density in kg / cubic-meter for the horizon/solum
CEC	numeric; Cation Exchange Capacity in cmol per kilogram
PHIHOX	numeric; soil pH in water suspension
h1	numeric; moisture potential in kPa e.g. -10 (pF 2.0)
h2	numeric; moisture potential in kPa e.g. -20 (pF 2.3)
h3	numeric; moisture potential in kPa e.g. -31.6 (pF 2.5)
pwp	numeric; moisture potential at wilting point in kPa e.g. -1585 (pF 4.2)
PTF.coef	data.frame; optional conversion coefficients (Pedo-Transfer Function) with rows "ai1", "sand", "silt", "clay", "oc", "bd", "cec", "ph", "silt^2", "clay^2", "sand*silt", "sand*clay" and colums "lnAlfa", "lnN", "tetaS" and "tetaR" (see Wösten et al. 2013 for more details)
fix.values	logical; specifies whether to correct values of textures and bulk density to avoid creating nonsensical values
print.coef	logical; specifies whether to attach the PTF coefficients to the output object

Value

Returns a data frame with the following columns:

- AWCh1: available soil water capacity (volumetric fraction) for h1;
- AWCh2: available soil water capacity (volumetric fraction) for h2;
- AWCh3: available soil water capacity (volumetric fraction) for h3;
- WWP: available soil water capacity (volumetric fraction) until wilting point;
- tetaS: saturated water content;

Note

Pedotransfer coefficients (PTF.coef) developed by Hodnett and Tomasella (2002). fix.values will correct sand, silt and clay fractions so they sum up to 100, and will replace bulk density values using global minimum maximum values.

Author(s)

Johan Leenaars, Maria Ruiperez Gonzalez and Tomislav Hengl

References

- Hodnett, M. G., & Tomasella, J. (2002). Marked differences between van Genuchten soil water-retention parameters for temperate and tropical soils: a new water-retention pedo-transfer functions developed for tropical soils. Geoderma, 108(3), 155-180.
- Wösten, J. H. M., Verzandvoort, S. J. E., Leenaars, J. G. B., Hoogland, T., & Wesseling, J. G. (2013). Soil hydraulic information for river basin studies in semi-arid regions. Geoderma, 195, 79-86.

Examples

```
SNDPPT = 30
SLTPPT = 25
CLYPPT = 48
ORCDRC = 23
BLD = 1200
CEC = 12
PHIHOX = 6.4
x <- AWCPTF(SNDPPT, SLTPPT, CLYPPT, ORCDRC, BLD, CEC, PHIHOX)
str(x)
attr(x, "coef")

## predict AWC for AfSP DB profile:
data(afsp)
names(afsp$horizons)
## profile of interest:
sel <- afsp$horizons$SOURCEID=="NG 28440_Z5"
hor <- afsp$horizons[sel,]
## replace missing values:
BLDf <- ifelse(is.na(hor$BLD),
   mean(hor$BLD, na.rm=TRUE), hor$BLD)
hor <- cbind(hor, AWCPTF(hor$SNDPPT, hor$SLTPPT,
  hor$CLYPPT, hor$ORCDRC, BLD=BLDf*1000, hor$CEC,
  hor$PHIHOX))
str(hor)
```

---

cookfarm *The Cook Agronomy Farm data set*

Description

The R.J. Cook Agronomy Farm (`cookfarm`) is a Long-Term Agroecosystem Research Site operated by Washington State University, located near Pullman, Washington, USA. Contains spatio-temporal (3D+T) measurements of three soil properties and a number of spatial and temporal regression covariates.

Usage

```
data(cookfarm)
```

Format

The cookfarm data set contains four data frames. The readings data frame contains measurements of volumetric water content (cubic-m/cubic-m), temperature (degree C) and bulk electrical conductivity (dS/m), measured at 42 locations using 5TE sensors at five standard depths (0.3, 0.6, 0.9, 1.2, 1.5 m) for the period "2011-01-01" to "2012-12-31":

SOURCEID factor; unique station ID

Date date; observation day

Port*VW numeric; volumetric water content measurements at five depths

Port*C numeric; soil temperature measurements at five depths

Port*EC numeric; bulk electrical conductivity measurements at five depths

The profiles data frame contains soil profile descriptions from 142 sites:

SOURCEID factor; unique station ID

Easting numeric; x coordinate in the local projection system

Northing numeric; y coordinate in the local projection system

TAXNUSDA factor; Keys to Soil Taxonomy taxon name e.g. "Caldwell"

HZDUSD factor; horizon designation

UHDICM numeric; upper horizon depth from the surface in cm

LHDICM numeric; lower horizon depth from the surface in cm

BLD bulk density in tonnes per cubic-meter

PHIHOX numeric; pH index measured in water solution

The grids data frame contains values of regression covariates at 10 m resolution:

DEM numeric; Digital Elevation Model

TWI numeric; SAGA GIS Topographic Wetness Index

MUSYM factor; soil mapping units e.g. "Thatuna silt loam"

NDRE.M numeric; mean value of the Normalized Difference Red Edge Index (time series of 11 RapidEye images)

NDRE.sd numeric; standard deviation of the Normalized Difference Red Edge Index (time series of 11 RapidEye images)

Cook_fall_ECa numeric; apparent electrical conductivity image from fall

Cook_spr_ECa numeric; apparent electrical conductivity image from spring

X2011 factor; cropping system in 2011

X2012 factor; cropping system in 2012

The weather data frame contains daily temperatures and rainfall from the nearest meteorological station:

Date date; observation day

Precip_wrcc numeric; observed precipitation in mm

MaxT_wrcc numeric; observed maximum daily temperature in degree C

MinT_wrccc numeric; observed minimum daily temperature in degree C

Note

The farm is 37 ha, stationed in the hilly Palouse region, which receives an annual average of 550 mm of precipitation, primarily as rain and snow in November through May. Soils are deep silt loams formed on loess hills; clay silt loam horizons commonly occur at variable depths. Farming practices at Cook Farm are representative of regional dryland annual cropping systems (direct-seeded cereal grains and legume crops).

Author(s)

Caley Gasch, Tomislav Hengl and David J. Brown

References

- Gasch, C., Hengl, T., Gräler, B., Meyer, H., Magney, T., Brown, D.J., 2015. Spatio-temporal interpolation of soil water, temperature, and electrical conductivity in 3D+T: the Cook Agronomy Farm data set. Spatial Statistics Journal, accepted.

Examples

```
## An example for 3D+T modelling applied to the cookfarm data set can be assesed via
## demo(cookfarm_3DT_kriging)
## demo(cookfarm_3DT_RF)
## Please note that the demo's might take 10-15 minutes to complete.
library(rgdal)
library(sp)
library(spacetime)
library(aqp)
library(splines)
library(randomForest)
library(plyr)
library(plotKML)
data(cookfarm)

## gridded data:
grid10m <- cookfarm$grids
gridded(grid10m) <- ~x+y
proj4string(grid10m) <- CRS(cookfarm$proj4string)
spplot(grid10m["DEM"], col.regions=SAGA_pal[[1]])

## soil profiles:
profs <- cookfarm$profiles
levels(cookfarm$profiles$HZDUSD)
## Bt horizon:
sel.Bt <- grep("Bt", profs$HZDUSD, ignore.case=FALSE, fixed=FALSE)
profs$Bt <- 0
profs$Bt[sel.Bt] <- 1
depths(profs) <- SOURCEID ~ UHDICM + LHDICM
site(profs) <- ~ TAXSUSDA + Easting + Northing
coordinates(profs) <- ~Easting + Northing
proj4string(profs) <- CRS(cookfarm$proj4string)
profs.geo <- as.geosamples(profs)
```

```
## fit model for Bt horizon:
m.Bt <- GSIF::fit.gstatModel(profs.geo, Bt~DEM+TWI+MUSYM+Cook_fall_ECa
   +Cook_spr_ECa+ns(altitude, df = 4), grid10m, fit.family = binomial(logit))
plot(m.Bt)

## fit model for soil pH:
m.PHI <- fit.gstatModel(profs.geo, PHIHOX~DEM+TWI+MUSYM+Cook_fall_ECa
    +Cook_spr_ECa+ns(altitude, df = 4), grid10m)
plot(m.PHI)
```

`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; lower horizon depth from the surface in cm

`LHDICM` numeric; upper 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"))

## Not run:
```

```
## plot points and grids:
pnts <- list("sp.points", sites, pch="+", col="black")
## load the 250 m grids:
con <- url("http://gsif.isric.org/lib/exe/fetch.php?media=edgeroi.grids.rda")
load(con)
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:
con2 <- url("http://gsif.isric.org/lib/exe/fetch.php?media=edgeroi.grids100.rda")
load(con2)
str(edgeroi.grids100)
gridded(edgeroi.grids100) <- ~x+y
proj4string(edgeroi.grids100) <- CRS("+init=epsg:28355")
spplot(edgeroi.grids100["TI1LAN6"], sp.layout=pnts)

## End(Not run)
```

ERDICM *Effective Rooting Zone depth*

Description

Derive Effective Rooting Zone depth i.e. an effective depth suitable for plant growth. Usually minimum depth of soil out of three standard rooting depths: limiting soil properties, depth to water-stagnating layer and depth to bedrock.

Usage

```
ERDICM(UHDICM, LHDICM, minimum.LRI, DRAINFAO, BDRICM,
    threshold.LRI=20, srd=150, drain.depths, smooth.LRI=TRUE)
```

Arguments

UHDICM	numeric; upper horizon depth in cm
LHDICM	numeric; lower horizon depth in cm
minimum.LRI	numeric; minimum Limiting Rootability index
DRAINFAO	factor; FAO drainage class e.g. "V", "P", "I", "M", "W", "S", "E"
BDRICM	numeric; depth to bedrock in cm
threshold.LRI	numeric; treshold index for LRI
srd	numeric; maximum depth of interest
drain.depths	data.frame; estimate effective rooting depth per drainage class (DRAINFAO)
smooth.LRI	logical; specify whether to smooth LRI values using splines

Value

Returns a vector of effective rooting depth in cm.

Author(s)

Johan Leenaars, Maria Ruiperez Gonzalez and Tomislav Hengl

See Also

LRI

`extract` *Extracts values at points from a list of files*

Description

Overlays and extracts values at points from a list of raster layers defined as file names (e.g. GeoTiffs). Extends the `extract` function from the raster package. Especially suitable for extracting values of a large list of rasters that have not been organized into a mosaick (a virtual stack), for example a list of Landsat scenes.

Usage

```
## S4 method for signature 'SpatialPoints,character'
extract(x, y,
   path=".", ID = "SOURCEID",
   method = "simple", is.pattern = FALSE, force.projection = TRUE,
   NAflag = "", show.progress=TRUE, isFactor=FALSE, ...)
## S4 method for signature 'SpatialPointsDataFrame,character'
extract(x, y,
   path = ".", ID = "SOURCEID",
   method = "simple", is.pattern = FALSE, force.projection = TRUE,
   NAflag = "", show.progress=TRUE, isFactor=FALSE, ...)
```

Arguments

`x`	object of class `"SpatialPoints*"`
`y`	character; list of files that can be read using the `raster` function
`path`	optional working directory where the files are stored
`ID`	character; column name for the unique identifier (if object is of class `"SpatialPoints"` `"SOURCEID"` column is automatically generated)
`method`	character; resampling method (see `raster::extract`)
`is.pattern`	logical; specifies whether the list is a pattern
`force.projection`	logical; specifies whether the reprojection should be ignored

NAflag	character; missing value flag (all missing values are removed by default)
show.progress	logical; specifies whether to display the progress bar
isFactor	logical; turns aggregation on off for factor type variable
...	additional arguments that can be passed to the raster::extract function

Note

The method will try to reproject the values to the native coordinate system, hence it is highly advisible to embed the proj4 string into the GeoTiffs. If both x and y are in the same coordinate system, then reprojection can be turned off by setting force.projection = FALSE. In the case is.pattern = TRUE (search by pattern), missing values are removed by default and if multiple rasters covering the same area are found, values are aggregated to the mean value.

Author(s)

Tomislav Hengl

See Also

raster::extract, warp

FAO.SoilProfileCollection-class

A class for FAO SoilProfileCollection

Description

A class for harmonized (FAO) soil profile records. Extends the "SoilProfileCollection" class from the aqp package.

Slots

idcol: object of class "character"; column name containing IDs

depthcols: object of class "character"; two element vector with column names for horizon top, bottom depths

metadata: object of class "data.frame"; metadata table

horizons: object of class "data.frame"; table containing observations at different depths

site: object of class "data.frame"; table containing observations at site locations

sp: object of class "SpatialPoints"; locations of profiles

diagnostic: object of class "data.frame"; table containing diagnostic properties

Data of class "FAO.SoilProfileCollection" must satisfy all of the following requirements (class validity):

- All variable names must be registered in the Global Soil Data Registry;

- All variable domains must correspond to the FAO Guidelines (2006 or later) for soil description or similar;
- All values must pass the validity checks i.e. numeric values must be within physical limits defined in the SoilGrids Global Soil Data Registry;

Author(s)

Tomislav Hengl

References

- Beaudette, D. E., Roudier, P., & O'Geen, A. T. (2013) Algorithms for quantitative pedology: A toolkit for soil scientists. Computers & Geosciences, 52, 258-268.
- FAO (2006) Guidelines for Soil Description. Food and Agriculture Organization of the United Nations, 4th Ed.

See Also

SoilGrids-class, SpatialComponents-class, geosamples-class

Examples

```
library(aqp)
library(sp)

LONWGS84 = 3.90
LATWGS84 = 7.50
UHDICM = 0
LHDICM = 30
SOURCEID = "ISRIC:NG0017"
SOURCEDB = "AfSP DB"
SPDFAO = "3"
TEXMHT = "SCL"
DCOMNS = "7.5YR_3_2"

sp1 <- new("FAO.SoilProfileCollection",
  depthcols=c('UHDICM','LHDICM'),
  metadata=soil.vars,
  horizons=data.frame(SOURCEID, UHDICM, LHDICM, TEXMHT, DCOMNS),
  site=data.frame(SOURCEID, SPDFAO, SOURCEDB),
  sp=SpatialPoints(data.frame(LONWGS84, LATWGS84),
     proj4string=CRS("+proj=longlat +datum=WGS84"))
)
str(sp1)
```

`fit.gstatModel-methods`

Methods to fit a regression-kriging model

Description

Tries to automatically fit a 2D or 3D regression-kriging model for a given set of points (object of type "SpatialPointsDataFrame" or "geosamples") and covariates (object of type "SpatialPixelsDataFrame"). It first fits a regression model (e.g. Generalized Linear Model, regression tree, random forest model or similar) following the formulaString, then fits variogram for residuals usign the fit.variogram method from the gstat package. Creates an output object of class gstatModel-class.

Usage

```
## S4 method for signature
## 'SpatialPointsDataFrame,formula,SpatialPixelsDataFrame'
fit.gstatModel(observations, formulaString, covariates,
     method = list("GLM", "rpart", "randomForest", "quantregForest"),
     dimensions = list("2D", "3D", "2D+T", "3D+T"),
     fit.family = gaussian(), stepwise = TRUE, vgmFun = "Exp",
     subsample = 5000, subsample.reg = 10000, ...)
## S4 method for signature 'geosamples,formula,SpatialPixelsDataFrame'
fit.gstatModel(observations, formulaString, covariates,
     method = list("GLM", "rpart", "randomForest", "quantregForest"),
     dimensions = list("2D", "3D", "2D+T", "3D+T"),
     fit.family = gaussian(), stepwise = TRUE,
     vgmFun = "Exp", subsample = 5000, subsample.reg = 10000, ...)
## S4 method for signature 'geosamples,formula,list'
fit.gstatModel(observations, formulaString, covariates,
     method = list("GLM", "rpart", "randomForest", "quantregForest"),
     dimensions = list("2D", "3D", "2D+T", "3D+T"),
     fit.family = gaussian(), stepwise = TRUE,
     vgmFun = "Exp", subsample = 5000, subsample.reg = 10000, ...)
## S4 method for signature 'geosamples,list,list'
fit.gstatModel(observations, formulaString, covariates,
     method = list("GLM", "rpart", "randomForest", "quantregForest"),
     dimensions = list("2D", "3D", "2D+T", "3D+T"),
     fit.family = gaussian(), stepwise = TRUE,
     vgmFun = "Exp", subsample = 5000, subsample.reg = 10000, ...)
```

Arguments

observations	object of type "SpatialPointsDataFrame" or "geosamples-class"
formulaString	object of type "formula" or a list of formulas
covariates	object of type "SpatialPixelsDataFrame", or list of grids

method	character; family of methods considered e.g. "GLM"
dimensions	character; "3D", "2D", "2D+T", "3D+T" models
fit.family	character string defyning the GLM family (for more info see stats::glm)
stepwise	specifies whether to run step-wise regression on top of GLM to get an optimal subset of predictors
vgmFun	variogram function ("Exp" by default)
subsample	integer; maximum number of observations to be taken for variogram model fitting (to speed up variogram fitting)
subsample.reg	integer; maximum number of observations to be taken for regression model fitting (currently only used for randomForest)
...	other optional arguments that can be passed to glm and/or fit.variogram

Details

The GLM method by default assumes that the target variable follows a normal distribution `fit.family = gaussian()`. Other possible families are:

normal distribution `fit.family = gaussian()` (default setting)

log-normal distribution `fit.family = gaussian(log)`

binomial variable `fit.family = binomial(logit)`

variable following a poisson distribution `fit.family = poisson(log)`

Note

Residuals (response residuals from the model) will be checked for normality and problems reported by default. The warning messages should be taken with care, as when the sample size is small, even big departures from normality will not be reported; when the sample size is large, even the smallest deviation from normality might lead to a warning. Likewise, if the variogram fitting fails, consider fitting a variogram manually or using the fit.vgmModel method.

Author(s)

Tomislav Hengl, Gerard B.M. Heuvelink and Bas Kempen

References

- chapter 8 "Interpolation and Geostatistics" in Bivand, R., Pebesma, E., Rubio, V., (2008) Applied Spatial Data Analysis with R. Use R Series, Springer, Heidelberg, pp. 378.
- Hengl, T. (2009) A Practical Guide to Geostatistical Mapping, 2nd Edt. University of Amsterdam, www.lulu.com, 291 p.

See Also

gstatModel-class, fit.regModel, test.gstatModel, geosamples-class, stats::glm, gstat::fit.variogram

Examples

```
# 2D model:
library(sp)
library(boot)
library(aqp)
library(plyr)
library(rpart)
library(splines)
library(gstat)
library(randomForest)
library(quantregForest)
library(plotKML)

## load the Meuse data set:
demo(meuse, echo=FALSE)

## simple model:
omm <- fit.gstatModel(meuse, om~dist+ffreq, meuse.grid,
   family = gaussian(log))
om.rk <- predict(omm, meuse.grid)
plot(om.rk)
## it was succesful!

## fit a GLM with a gaussian log-link:
omm <- fit.gstatModel(meuse, om~dist+ffreq, meuse.grid,
   fit.family = gaussian(log))
summary(omm@regModel)
om.rk <- predict(omm, meuse.grid)
plot(om.rk)

## fit a regression-tree:
omm <- fit.gstatModel(meuse, log1p(om)~dist+ffreq, meuse.grid,
   method="rpart")
summary(omm@regModel)
## plot a regression-tree:
plot(omm@regModel, uniform=TRUE)
text(omm@regModel, use.n=TRUE, all=TRUE, cex=.8)
omm@vgmModel

## fit a randomForest model:
omm <- fit.gstatModel(meuse, om~dist+ffreq, meuse.grid,
   method="randomForest")
## plot to see how good is the fit:
plot(omm)
## plot the estimated error for number of bootstrapped trees:
plot(omm@regModel)
omm@vgmModel
om.rk <- predict(omm, meuse.grid)
plot(om.rk)
## Compare with "quantregForest" package:
omm <- fit.gstatModel(meuse, om~dist+ffreq, meuse.grid,
   method="quantregForest")
```

```
## Not run:
om.rk <- predict(omm, meuse.grid, nfold=0)
plot(om.rk)
## plot the results in Google Earth:
plotKML(om.rk)

## End(Not run)

## binary variable (0/1):
meuse$soil.1 <- as.numeric(I(meuse$soil==1))
som <- fit.gstatModel(meuse, soil.1~dist+ffreq, meuse.grid,
   fit.family = binomial(logit))
summary(som@regModel)
som.rk <- predict(som, meuse.grid)
plot(som.rk)
## Not run: # plot the results in Google Earth:
plotKML(som.rk)

## End(Not run)

## 3D model:
library(plotKML)
data(eberg)
## list columns of interest:
s.lst <- c("ID", "soiltype", "TAXGRSC", "X", "Y")
h.lst <- c("UHDICM","LHDICM","SNDMHT","SLTMHT","CLYMHT")
sel <- runif(nrow(eberg))<.05
## get sites table:
sites <- eberg[sel,s.lst]
## get horizons table:
horizons <- getHorizons(eberg[sel,], idcol="ID", sel=h.lst)
## create object of type "SoilProfileCollection"
eberg.spc <- join(horizons, sites, type='inner')
depths(eberg.spc) <- ID ~ UHDICM + LHDICM
site(eberg.spc) <- as.formula(paste("~", paste(s.lst[-1], collapse="+"), sep=""))
coordinates(eberg.spc) <- ~X+Y
proj4string(eberg.spc) <- CRS("+init=epsg:31467")
## convert to geosamples:
eberg.geo <- as.geosamples(eberg.spc)
## covariates:
data(eberg_grid)
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
glm.formulaString = as.formula(paste("SNDMHT ~ ",
  paste(names(eberg_grid), collapse="+"), "+ ns(altitude, df=4)"))
SNDMHT.m <- fit.gstatModel(observations=eberg.geo, glm.formulaString,
  covariates=eberg_grid)
plot(SNDMHT.m)
## problems with the variogram?
## Not run: ## remove classes from the PRMGEO6 that are not represented in the model:
sel = !(levels(eberg_grid$PRMGEO6) %in% levels(SNDMHT.m@regModel$model$PRMGEO6))
fix.c = levels(eberg_grid$PRMGEO6)[sel]
summary(eberg_grid$PRMGEO6)
```

```
for(j in fix.c){
  eberg_grid$PRMGEO6[eberg_grid$PRMGEO6 == j] <- levels(eberg_grid$PRMGEO6)[7]
}
## prepare new locations:
new3D <- sp3D(eberg_grid)
## regression only:
SNDMHT.rk.sd1 <- predict(SNDMHT.m, new3D[[1]], vgmmodel=NULL)
## regression-kriging:
SNDMHT.rk.sd1 <- predict(SNDMHT.m, new3D[[1]])
## plot the results in Google Earth:
plotKML(SNDMHT.rk.sd1, z.lim=c(5,85))

## End(Not run)
```

`fit.regModel-methods` *Fits a regression model to spatial data*

Description

Fits a regression or a trend model (e.g. a GLM) and, if not available, a variogram for the response residuals using the default settings.

Usage

```
## S4 method for signature
## 'formula,data.frame,SpatialPixelsDataFrame,character'
fit.regModel(formulaString, rmatrix, predictionDomain,
      method = list("GLM", "rpart", "randomForest", "quantregForest", "lme"),
      dimensions = NULL, fit.family = gaussian(), stepwise = TRUE, rvgm,
      GLS = FALSE, steps, subsample, subsample.reg, ...)
```

Arguments

`formulaString`	object of class `"formula"` — regression model
`rmatrix`	object of class `"data.frame"`; regression matrix produced as a result of spatial overlay
`predictionDomain`	object of class `"SpatialPixelsDataFrame"`; spatial domain of interest
`method`	character; family of methods considered e.g. `"GLM"`, `"rpart"` (regression trees), `"randomForest"` (random forest)
`dimensions`	character; `"2D"`, `"3D"`, `"2D+T"`, or `"3D+T"`
`fit.family`	family to be passed to the `glm` (see examples below)
`stepwise`	specifies whether to run step-wise regression on top of GLM to get an optimal subset of predictors
`rvgm`	residual variogram (to avoid fitting the variogram set as NULL)

GLS	fit trend model using Generalized Least Squares implemented in the nlme package
steps	integer; the maximum number of steps to be considered for step-wise regression; see stats::step for more details
subsample	integer; maximum number of observations to be taken for variogram model fitting (to speed up variogram fitting)
subsample.reg	integer; maximum number of observations to be taken for regression model fitting (especially important for randomForest modelling)
...	other optional arguments that can be passed to gstat::fit.variogram

Details

Produces an object of class "gstatModel" that contains: (1) fitted regression model (e.g. a GLM, cubist model, or randomForest model), (2) fitted variogram, and (c) object of class "SpatialPoints" with observation locations. To combine overlay and model fitting operations, consider using fit.gstatModel.

Author(s)

Tomislav Hengl, Mario Antonio Guevara Santamaria and Bas Kempen

See Also

fit.gstatModel, stats::glm, gstat::fit.variogram, randomForest::randomForest

Examples

```
## Meuse data:
library(sp)
library(rpart)
library(nlme)
library(gstat)
library(randomForest)
library(quantregForest)

## load the Meuse data set:
demo(meuse, echo=FALSE)

## prepare the regression matrix:
ov <- over(meuse, meuse.grid)
ov <- cbind(data.frame(meuse["om"]), ov)
## skip variogram fitting:
m <- fit.regModel(om~dist+ffreq, rmatrix=ov, meuse.grid,
    fit.family=gaussian(log), method="GLM", rvgm=NULL)
m@regModel
m@vgmModel
plot(m)
## fit a GLM with variogram:
m1 <- fit.regModel(om~dist+ffreq, rmatrix=ov, meuse.grid,
    fit.family=gaussian(log), method="GLM")
m1@vgmModel
```

```
plot(m1)
## fit a regression tree with variogram:
m2 <- fit.regModel(log1p(om)~dist+ffreq, rmatrix=ov, meuse.grid,
     method="rpart")
plot(m2)
## fit a lme model with variogram:
m3 <- fit.regModel(log1p(om)~dist, rmatrix=ov, meuse.grid,
     method="lme", random=~1|ffreq)
plot(m3)
## fit a randomForest model with variogram
## NOTE: no transformation required
m4 <- fit.regModel(om~dist+ffreq, rmatrix=ov, meuse.grid,
     method="randomForest")
plot(m4)
## RF is very sensitive to the 'mtry' argument:
m4b <- fit.regModel(om~dist+ffreq, rmatrix=ov, meuse.grid,
     method="randomForest", mtry=2)
plot(m4b)
## RF with uncertainty (quantregForest package)
m5 <- fit.regModel(om~dist+ffreq, rmatrix=ov, meuse.grid,
     method="quantregForest")
plot(m5)
```

`fit.vgmModel-methods` *Fits a 2D or 3D variogram model to spatial data*

Description

Fits a 2D or 3D variogram model based on a regression matrix and spatial domain of interest.

Usage

```
## S4 method for signature 'formula,data.frame,SpatialPixelsDataFrame'
fit.vgmModel(formulaString,
      rmatrix, predictionDomain, vgmFun = "Exp",
      dimensions = list("2D", "3D", "2D+T", "3D+T"),
      anis = NULL, subsample = nrow(rmatrix), ivgm, cutoff = NULL,
      width, cressie = FALSE, ...)
```

Arguments

`formulaString`	object of class `"formula"` — regression model
`rmatrix`	object of class `"data.frame"`; regression matrix produced as a result of spatial overlay
`predictionDomain`	object of class `"SpatialPixelsDataFrame"`; spatial domain of interest
`vgmFun`	character; variogram function (`"Exp"` by default)
`dimensions`	character; `"3D"`, `"2D"`, `"2D+T"`, `"3D+T"` models

anis	vector containing 2, 5 or more anisotropy parameters; see gstat::vgm for more info
subsample	integer; size of the subset
ivgm	vgm; initial variogram model
cutoff	numeric; distance up to which point pairs are included in semivariance estimates
width	numeric; sample variogram bin width
cressie	logical; specifies whether to use cressie robust estimator
...	other optional arguments that can be passed to gstat::fit.variogram

Details

It will try to fit a variogram to multidimensional data. If the data set is large, this process can be time-consuming, hence one way to speed up fitting is to subset the regression matrix using the subsample argument (i.e. randomly subset observations).

Author(s)

Tomislav Hengl

See Also

fit.regModel, fit.gstatModel, gstat::fit.variogram

Examples

```
library(sp)
library(gstat)

## fit variogram to the Meuse data:
demo(meuse, echo=FALSE)
# produce a regression matrix:
ov <- over(meuse, meuse.grid)
ov <- cbind(data.frame(meuse["om"]), ov)
# fit a model:
v <- fit.vgmModel(om~1, rmatrix=ov, meuse.grid, dimensions="2D")
plot(variogram(om ~ 1, meuse[!is.na(meuse$om),]), v$vgm)
```

---

geochm *NGS database samples for Indiana State*

Description

A subset of the National Geochemical Survey (NGS) samples covering the Indiana and Illinois State. Contains a total of 2681 point samples.

Usage

```
data(geochm)
```

Format

Data frame; contains the following columns:

`REC_NO` factor; unique record identifier

`DATASET` factor; abbreviated dataset group e.g. "AK+MI"

`TYPEDESC` factor; abbreviated description of sample type: stream, pond, spring, soil etc

`COLL_DATE` integer; sampling date

`LONGITUDE` numeric; longitude in decimal degrees (NAD27 datum)

`LATITIUDE` numeric; latitude in decimal degrees (NAD27 datum)

`DATUM` factor; geodetic datum if different from NAD83

`RELIEF` factor; relief in drainage basin from which sample was collected

`FORMATION` factor; code or name of geologic formation in which sample area was located

`ROCK_TYPE` factor; rock type in area of sample collection e.g. "carbonate"

`SOIL_HORIZ` factor; soil horizon from which the sample was collected

`COLOR` factor; observed color of powdered sample during splitting

`MEDIUM` factor; sample medium — rock, sediment, standard, or unknown

`SOURCE` factor; geological source of the sample medium that was collected e.g. "Beach"

`AS_ICP40` numeric; As (ppm) by Inductively Coupled Plasma Spectrometry (ICP) after acid dissolution

`CD_ICP40` numeric; Cd (ppm)

`CR_ICP40` numeric; Cr (ppm)

`CU_ICP40` numeric; Cu (ppm)

`NI_ICP40` numeric; Ni (ppm)

`ZN_ICP40` numeric; Zn (ppm)

`AS_AA` numeric; As (ppm) by Hydride Atomic Absorption

`HG_AA` numeric; Hg (ppm) by Hydride Atomic Absorption

`PB_ICP40` numeric; Pb (ppm)

`C_TOT` numeric; total carbon (weight percentage) by combustion

`C_ORG` numeric; organic carbon (weight percentage) as a difference between `C_TOT` and `C_CO3`

`C_CO3` numeric; carbonate carbon (weight percentage) by Coulometric Titration

`S_TOT` numeric; total sulfur (weight percentage) by combustion

Note

Negative values of the heavy metal concetrations indicate a determination that is below the limit of detection for the analytic method used. The magnitude of the negative number indicates the detection limit. For example, -10 ppm means the result should be regarded as < 10 ppm.

Author(s)

National Geochemical Survey database is maintaned by the USGS National Geochemical Survey Team (contact: Peter Schweitzer). This subset has been prepared for the purpose of testing various geostatistical mapping algoriths by Tomislav Hengl (tom.hengl@wur.nl).

References

- The National Geochemical Survey Team, (2008) The National Geochemical Survey: database and documentation. U.S. Geological Survey Open-File Report 2004-1001, U.S. Geological Survey, Reston VA.
- National Geochemical Survey database (http://tin.er.usgs.gov/geochem/)

Examples

```
library(sp)

# Load the NGS data:
data(geochm)
coordinates(geochm) <- ~LONGITUDE+LATITUDE
proj4string(geochm) <- CRS("+proj=longlat +ellps=clrk66 +datum=NAD27 +no_defs")
## Not run:
require(plotKML)
data(SAGA_pal)
# replace the missing values with half the detection limit:
geochm$PB_ICP40 <- ifelse(geochm$PB_ICP40 < 0, 2, geochm$PB_ICP40)
shape = "http://maps.google.com/mapfiles/kml/pal2/icon18.png"
kml(geochm, shape = shape, colour = log1p(PB_ICP40), labels = "",
    colour_scale = SAGA_pal[[1]], kmz = TRUE)

## End(Not run)
```

---

geosamples-class *A class for spatially and temporally referenced samples*

Description

A class for spatially and temporally referenced samples with fixed column names (standardized geosamples). Corresponds to the point "Placemark" in the KML schema.

Slots

`registry`: object of class "character"; URI of the online registry i.e. the URL where the "producerid" column can be linked to all other connected metadata

`methods`: object of class "data.frame"; a table with method names ("methodid"), a one sentence description of each method ("description"), measurement units or levels ("units"), and associated detection limits ("detectionLimit")

`data`: object of class `"data.frame"`; a standardized table with fixed column names: `"observationid"` (unique observation ID; as specified in the data registry service), `"sampleid"` (producer's ID; usually site ID and horizon ID or sequence number), `"longitude"` (longitude on the WGS84 ellipsoid), `"latitude"` (latitude on the WGS84 ellipsoid), `"locationError"` (error radius in meters), `"TimeSpan.begin"` (begin of the measurement period), `"TimeSpan.end"` (end of the measurement period), `"altitude"` (height above ground or above the sea level in meters), `"altitudeMode"` (one of the KML schema altitude modes), `"sampleArea"` (spatial support in square meters), `"sampleThickness"` (thickness of horizons in meters or vertical support), `"observedValue"` (measured value), `"methodid"` (method name; see `methods` table), `"measurementError"` (estimated measurement error for that specific observation)

The column names in the data slot largely reflect the KML schema elements. Geosamples are interoperable with the OGC Observations and measurements specifications, but do not necessarily contain all required fields (i.e. there is no validity check for the OGC specifications). Geosamples-class can be used to store and manipulate geological, hydrological, geochemical, biodiversity, soil science and similar field samples near or below land surface. Geological and soil samples can also be registered via the geosamples.org, in which case the `"observationid"` will correspond to the unique sample identifier. `"sampleid"` column allows linking geosamples to the original ID's.

Methods

show `signature(obj = "geosamples")`: summarize object by listing methods, total number of observations, total area covered etc.

subset `signature(obj = "geosamples")`: subset to a single variable type; returns a data frame

over `signature(x = "SpatialPixelsDataFrame" or "RasterStack", y = "geosamples")`: overlay geosamples and spatial pixels

stack `signature(x = "geosamples")`: stacks all observed values into a single table using reshape function

write.data `signature(obj = "geosamples")`: write geosamples to an external format e.g. GeoEAS

Author(s)

Tomislav Hengl

References

- Dyson, E., (2003) Online Registries: The DNS and Beyond.... Edventure, Vol 21(8).
- International Geo Sample Number (http://en.wikipedia.org/wiki/International_Geo_Sample_Number)
- KML Reference (https://developers.google.com/kml/documentation/kmlreference)
- OGC Observations and Measurements standard (http://www.opengeospatial.org/standards/om/)
- SESAR, the System for Earth Sample Registration (http://www.geosamples.org)

See Also

`as.geosamples`

getID *Derive 1 degree cell IDs*

Description

Derives ID's of the 1 degree cells in the default land mask for a given polygon defining the spatial domain of interest.

Usage

```
## S4 method for signature 'SpatialPolygons'
getID(obj, pixsize = 3/3600, empty.tif = FALSE,
     compress = FALSE, zipname = set.file.extension(tempfile(tmpdir = getwd()), "zip"))
```

Arguments

obj	object of class "SpatialPolygons"; must be in geographical coordinates (WGS84)
pixsize	grid cell size in decimal degrees (set at 0.0008333333 or 100 m around equator)
empty.tif	logical; specify whether a GeoTiff mask file should be created
compress	logical; specify whether to compress GeoTiffs
zipname	(optional); zip archive file name

Value

The output is a vector of grid cell ID names e.g. W79_N83. These can be further used to automate digital soil mapping for large areas.

Note

This operation can be time consuming for large areas (e.g. continents).

Author(s)

Tomislav Hengl

See Also

landmask

Examples

```
library(sp)
## Bounding box for Malawi:
bbox = expand.grid(lon=c(32.67152, 35.915046), lat=c(-17.12721, -9.363796))
bbox[5,] <- bbox[1,]
crs = CRS("+proj=longlat +datum=WGS84")
x <- SpatialPolygons(list(Polygons(list(Polygon(bbox)), ID="1")), proj4string=crs)
ID.lst <- getID(x)
```

getSpatialTiles *Get a list of tiles (regular blocks)*

Description

Creates a list of tiles ("SpatialPolygons") for a given spatial domain i.e. extent. Input can be any object of class "Spatial" or "GDALobj".

Usage

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

Arguments

obj	object of class "Spatial*"
block.x	numeric; size of block in x-direction (meters or corresponding mapping units)
block.y	numeric; size of block in y-direction (meters or corresponding mapping units)
overlap.percent	numeric; percentage overlap (must be a positive number)
limit.bbox	logical; specifies whether to limit the extent of tiles to the bounding box only
return.SpatialPolygons	logical; specifies whether to return a list of tiles as "SpatialPolygons" or a data frame with bounding box coordinates

Details

The first output tile starts by default at the lower left corner. getSpatialTiles-method can only be used to generate regular tiles.

Value

Returns a list of tiles either as a list of "SpatialPolygons" or a data frame with with bounding box coordinates.

Author(s)

Tomislav Hengl

See Also

tile, sp::spsample

`GlobalSoilMap-class` *A class for GlobalSoilMap soil property maps*

Description

A class containing predictions of target soil property at six standard depths following the GlobalSoilMap.net specifications: `sd1` = 2.5 cm (0–5), `sd2` = 10 cm (5–15), `sd3` = 22.5 cm (15–30), `sd4` = 45 cm (30–60), `sd5` = 80 cm (60–100), `sd6` = 150 cm (100–200).

Slots

`varname`**:** object of class `"character"`; abbreviated variable name registered in the Global Soil Data registry

`TimeSpan`**:** object of class `"list"`; contains begin and end of the sampling period of class `"POSIXct"`

`sd1`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 2.5 cm (0–5)

`sd2`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 10 cm (5–15)

`sd3`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 22.5 cm (15–30)

`sd4`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 45 cm (30–60)

`sd5`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 80 cm (60–100)

`sd6`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 150 cm (100–200)

References

- Hartemink, A. E., Hempel, J., Lagacherie, P., McBratney, A., McKenzie, N., MacMillan, R. A., ... & Zhang, G. L. (2010). GlobalSoilMap.net — A New Digital Soil Map of the World. In Digital Soil Mapping (pp. 423–428). Springer Netherlands.
- Sanchez, P. A., S. Ahamed, F. Carre, A. E. Hartemink, J. Hempel, J. Huising, P. Lagacherie, A. B. McBratney, N. J. McKenzie, M L. de Mendonça-Santos, et al., (2009) Digital Soil Map of the World. Science, 325(5941): 680–681.

See Also

`SoilGrids-class`, `SpatialComponents-class`, `geosamples-class`

GSIF.env *GSIF specific environmental variables / paths*

Description

Sets the environmental, package specific parameters and settings (URLs, names, default cell size and similar) that can be later on passed to other functions.

Usage

```
GSIF.env(wps.server = "http://wps.worldgrids.org",
    ref_CRS = "+proj=longlat +datum=WGS84",
    NAflag = -99999,
    license_url = "http://creativecommons.org/licenses/by/3.0/",
    project_url = "http://gsif.r-forge.r-project.org/",
    stdepths = c(-2.5, -10, -22.5, -45, -80, -150)/100,
    stsize = c(5, 10, 15, 30, 40, 100)/100,
    cellsize = rev(c(6/120, 3/120, 1/120, 1/240, 1/600, 1/1200, 1/3600)),
    REST.server = 'http://rest.soilgrids.org/',
    attributes = c("ORCDRC","PHIHOX","SNDPPT","SLTPPT","CLYPPT",
                   "CFRVOL","CEC","BLD","TAXGWRB","TAXOUSDA"),
   TimeSpan = list(begin=as.POSIXct("1950-01-01"), end=as.POSIXct("2005-12-30")),
    show.env = TRUE)
```

Arguments

wps.server	character; location of the WPS server
ref_CRS	the referent CRS proj4string ("+proj=longlat +datum=WGS84")
NAflag	the default missing value flag (usually "-99999")
license_url	the default license URL
project_url	the default location of the package documentation
stdepths	numeric; standard depths
stsize	numeric; standard horizon thicknesses
cellsize	numeric; standard grid cell sizes on WGS84 geographical coordinates
REST.server	character; location of the SoilGrids REST service
attributes	character; default soil variables of interest
TimeSpan	list; default begin end times (temporal coverage of SoilGrids)
show.env	logical; specify whether to print all environmental parameters

Note

To further customize the GSIF options, consider putting:

```
library(GSIF); GSIF.env(..., show.env = FALSE)
```

in your "/etc/Rprofile.site".

Author(s)

Tomislav Hengl

Examples

```
# environmental variables:
GSIF.env()
get("cellsize", envir = GSIF.opts)
```

---

gstatModel-class *A class for a geostatistical model*

Description

A class containing fitted parameters of a geostatistical model to be used to run predictions by regression-kriging. It comprises regression model (e.g. a GLM), variogram model, and observation locations of sampled values used to fit the model.

Details

Any model passed to the regModel slot must come with generic functions such as residuals, fitted.values, summary, formula and predict.

Slots

regModel: object of class "ANY"; output of fitting a generalized linear model (GLM) or any similar regression model

svgmModel: object of class "data.frame"; sample variogram with semivariances and distances

vgmModel: object of class "data.frame"; the fitted gstat variogram model parameters containing variogram model, nugget, sill, range and the five anisotropy parameters

sp: object of class "SpatialPointsDataFrame"; observation locations

Methods

predict `signature(obj = "gstatModel")`: makes predictions for a set of given predictionLocations (gridded maps) at block support corresponding to the `cellsize` slot in the object of class `"SpatialPixelsDataFrame"`; to produce predictions at point support, submit the `predictionLocations` as `"SpatialPointsDataFrame"`

validate `signature(obj = "gstatModel")`: runs *n*-fold cross-validation of the existing gstatModel (it re-fits the model using existing formula string and model data, then estimates the mapping error at validation locations)

plot `signature(obj = "gstatModel", ...)`: plots goodness of fit and variogram model

Note

`"SpatialPredictions"` saves results of predictions for a single target variable, which can be of type numeric or factor. Multiple variables can be combined into a list. When using `nsim` argument with the `predict` method, the output result will be of type:

`plotKML::RasterBrickSimulations-class`

i.e. *N* number of equiprobable realizations. To generate an object of type:

`plotKML::SpatialPredictions-class`

set `nsim = 0`.

Author(s)

Tomislav Hengl and Gerard B.M. Heuvelink

See Also

predict.gstatModel, test.gstatModel, plotKML::SpatialPredictions-class, plotKML::RasterBrickSimulations
gstat::gstat, stats::glm

Examples

```
## load observations:
library(plotKML)
library(sp)
demo(meuse, echo=FALSE)
data(meuse)
coordinates(meuse) <- ~x+y
proj4string(meuse) <- CRS("+init=epsg:28992")
## load grids:
data(meuse.grid)
coordinates(meuse.grid) <- ~x+y
gridded(meuse.grid) <- TRUE
proj4string(meuse.grid) <- CRS("+init=epsg:28992")
```

```
## fit a model:
omm <- fit.gstatModel(meuse, om~dist+ffreq,
  fit.family=gaussian(link="log"), meuse.grid)
plot(omm)
## produce SpatialPredictions:
om.rk <- predict(omm, predictionLocations = meuse.grid)
plot(om.rk)
## run a proper cross-validation:
rk.cv <- validate(omm)
## RMSE:
sqrt(mean((rk.cv$validation$var1.pred-rk.cv$validation$observed)^2))
```

`isis` *ISRIC Soil Information System*

Description

ISRIC's collection of global soil monoliths that represent the main soil reference groups of the World Reference Base for Soil Resources (WRB). Includes some 950 monoliths (785 with coordinates) from over 70 countries with detailed soil profile and environmental data.

Usage

```
data(isis)
```

Format

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

`SOURCEID` factor; unique ISIS code

`LONWGS84` numeric; longitude in decimal degrees on the WGS84 datum

`LATWGS84` numeric; latitude in decimal degrees on the WGS84 datum

`TIMESTRR` Date; the date on which this particular soil was described or sampled

`TAXGWRB` factor; soil group based on the WRB classification system

`TAXNUSDA` factor; Keys to Soil Taxonomy taxon name e.g. `"Natraqualf"`

`BDRICM` numeric; depth to bedrock (R horizon) if observed

`SOURCEDB` factor; source data base

Horizons table contains the following columns:

`SOURCEID` factor; unique ISIS code

`UHDICM` numeric; upper horizon depth from the surface in cm

`LHDICM` numeric; lower horizon depth from the surface in cm

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

`PHIHOX` numeric; pH index measured in water solution

PHIKCL numeric; pH index measured in KCl solution

ORCDRC numeric; soil organic carbon content in permilles

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)

CEC numeric; Cation Exchange Capacity in cmol+/kg

BLD bulk density in tonnes per cubic-meter

Author(s)

ISRIC — World Soil Information

Examples

```
library(rgdal)
library(sp)

data(isis)
sites <- isis$sites
coordinates(sites) <- ~ LONWGS84 + LATWGS84
proj4string(sites) <- "+proj=longlat +datum=WGS84"
## Not run:
## obtain country borders:
library(maps)
country.m = map('world', plot=FALSE, fill=TRUE)
IDs <- sapply(strsplit(country.m$names, ":"), function(x) x[1])
require(maptools)
country <- as(map2SpatialPolygons(country.m, IDs=IDs), "SpatialLines")
proj4string(country) = "+proj=longlat +datum=WGS84"
## overlay and plot points and maps:
plot(country, col="darkgrey")
points(sites, pch=21, bg="red", cex=.6, col="black")

## End(Not run)
```

---

landmask *Global coarse resolution land / soil mask maps*

Description

Land mask showing the 1-degree cells (about 19 thousand in total) in the geographical coordinates, and the productive soils mask (areas with a positive Leaf Area Index at least once in the period 2002–2011). The land mask is based on the Global Self-consistent, Hierarchical, High-resolution Shoreline Database data (GSHHS 2.1), the productive soils mask on the MODIS Leaf Area Index monthtly product (MOD15A2), and the water mask is based on the MOD44W product. The map of the Keys to Soil Taxonomy soil suborders of the world at 20 km is based on the USDA-NRCS map of the global soil regions.

Usage

```
data(landmask)
```

Format

`landmask` data set is a data frame with the following columns:

`mask` percent; land mask value

`soilmask` boolean; soil mask value

`watermask` percent; water mask value

`Lon_it` indication of the longitude quadrant (W or E)

`Lat_it` indication of the latitude quadrant (S or N)

`cell_id` cell id code e.g. `W79_N83`

`x` longitudes of the center of the grid nodes

`y` latitudes of the center of the grid nodes

`landmask20km` data set is an object of class `SpatialGridDataFrame` with the following columns:

`mask` percent; land mask value

`suborder` factor; Keys to Soil Taxonomy suborder class e.g. Histels, Udolls, Calcids, ...

`soilmask` factor; global soil mask map based on the land cover classes (see: SMKISR3)

Note

The land mask has been generated from the layer `GSHHS_shp/h/GSHHS_h_L1.shp` (level-1 boundaries).

References

- Carroll, M., Townshend, J., DiMiceli, C., Noojipady, P., Sohlberg, R. (2009) A New Global Raster Water Mask at 250 Meter Resolution. International Journal of Digital Earth, 2(4).
- Global Self-consistent, Hierarchical, High-resolution Shoreline Database (http://en.wikipedia.org/wiki/GSHHS)
- USDA-NRCS Global Soil Regions Map (http://www.nrcs.usda.gov/)
- Savtchenko, A., D. Ouzounov, S. Ahmad, J. Acker, G. Leptoukh, J. Koziana, and D. Nickless, (2004) Terra and Aqua MODIS products available from NASA GES DAAC. Advances in Space Research 34(4), 710-714.
- Wessel, P., Smith, W.H.F., (1996) A Global Self-consistent, Hierarchical, High-resolution Shoreline Database. Journal of Geophysical Research, 101, 8741-8743.

See Also

`rworldmap::rworldmapExamples`, `maps::map`

Examples

```
library(rgdal)
library(sp)

data(landmask)
gridded(landmask) <- ~x+y
proj4string(landmask) <- "+proj=longlat +datum=WGS84"
## Not run:  ## plot maps:
library(maps)
country.m = map('world', plot=FALSE, fill=TRUE)
IDs <- sapply(strsplit(country.m$names, ":"), function(x) x[1])
library(maptools)
country <- as(map2SpatialPolygons(country.m, IDs=IDs), "SpatialLines")
spplot(landmask["mask"], col.regions="grey", sp.layout=list("sp.lines", country))
spplot(landmask["soilmask"], col.regions="grey", sp.layout=list("sp.lines", country))

## End(Not run)
## also available in the Robinson projection at 20 km grid:
data(landmask20km)
image(landmask20km[1])
summary(landmask20km$suborder)
summary(landmask20km$soilmask)
```

LRI *Limiting Rootability*

Description

Derive Limiting Rootability using observed soil properties at at least three depths.

Usage

```
LRI(UHDICM, LHDICM, SNDPPT, SLTPPT, CLYPPT, CRFVOL, BLD,
   ORCDRC, ECN, CEC, ENA, EACKCL, EXB, PHIHOX, CRB, GYP, tetaS,
   fix.values=TRUE, thresholds, print.thresholds=FALSE)
```

Arguments

UHDICM	numeric; upper horizon depth in cm
LHDICM	numeric; lower horizon depth in cm
SNDPPT	numeric; sand content in percent
SLTPPT	numeric; silt content in percent
CLYPPT	numeric; clay content in percent
CRFVOL	numeric; volume percentage of coarse fragments (> 2 mm)
BLD	numeric; bulk density in kg per cubic-meter for the horizon/solum
ORCDRC	numeric; soil organic carbon concentration in permille or g per kg

ECN	numeric; electrical conductivity in dS per m
CEC	numeric; Cation Exchange Capacity in cmol per kilogram
ENA	numeric; exchangable Na in cmol per kilogram
EACKCL	numeric; exchangable acidity in cmol per kilogram
EXB	numeric; exchangable bases in cmol per kilogram
PHIHOX	numeric; soil pH in water suspension
CRB	numeric; CaCO3 (carbonates) in g per kg
GYP	numeric; CaSO4 (gypsum) in g per kg
tetaS	numeric; volumetric percentage (optional; if not provided it will be derived using the AWCPTF Pedo-Transfer Function)
fix.values	logical; specifies whether to correct values of textures and bulk density to avoid creating nonsensical values
thresholds	data.frame; optional table containing threshold values for "CRFVOL", "tetaS" (volumetric percentage), "BLD.f" (clay-adjusted BLD), "SNDPPT", "CLY.d" (difference in clay between horizons), "SND.d" (difference in sand between horizons), "PHIHOX.L" (lower limits for pH), "PHIHOX.H" (upper limits for pH), "ECN", "ENA.f" (exchangable saturated Na), "ENA", "EACKCL.f" (exchangable saturated acidity), "CRB" (carbonates), and "GYP" (gypsum)
print.thresholds	logical; specifies whether to attach the threshold values to the output object

Value

Returns a vector with TRUE / FALSE values where FALSE indicates rooting not possible. Threshold values used to derive Limiting Rootability scores are set based on common soil agricultural productivity tresholds (e.g. in this case for maize), and can be adjusted via the thresholds argument. This functions also accounts for textural changes (sudden changes in sand and clay content) and saturated water content.

Note

Horizons need to be sorted by depth e.g. 0-5, 5-15, 15-30...For each soil property at least three depths are needed otherwise the function reports an error. Missing values are automatically replaced using smoothing splines.

Author(s)

Johan Leenaars and Maria Ruiperez Gonzalez

References

- Driessen, P. M., & Konijn, N. T. (1992) Land-use systems analysis. Wageningen Agricultural University.
- Rijsberman, F. R., & Wolman, M. G. (1985) Effect of erosion on soil productivity: an international comparison. Journal of soil and water conservation, 40(4), 349-354.

See Also

AWCPTF, ERDICM

Examples

```
## sample profile from Nigeria (ISRIC:NG0017):
UHDICM = c(0, 18, 36, 65, 87, 127)
LHDICM = c(18, 36, 65, 87, 127, 181)
SNDPPT = c(66, 70, 54, 43, 35, 47)
SLTPPT = c(13, 11, 14, 14, 18, 23)
CLYPPT = c(21, 19, 32, 43, 47, 30)
CRFVOL = c(17, 72, 73, 54, 19, 17)
BLD = c(1.57, 1.60, 1.52, 1.50, 1.40, 1.42)*1000
PHIHOX = c(6.5, 6.9, 6.5, 6.2, 6.2, 6.0)
CEC = c(9.3, 4.5, 6.0, 8.0, 9.4, 10.9)
ENA = c(0.1, 0.1, 0.1, 0.1, 0.1, 0.2)
EACKCL = c(0.1, 0.1, 0.1, NA, NA, 0.5)
EXB = c(8.9, 4.0, 5.7, 7.4, 8.9, 10.4)
ORCDRC = c(18.4, 4.4, 3.6, 3.6, 3.2, 1.2)
x <- LRI(UHDICM=UHDICM, LHDICM=LHDICM, SNDPPT=SNDPPT,
   SLTPPT=SLTPPT, CLYPPT=CLYPPT, CRFVOL=CRFVOL,
   BLD=BLD, ORCDRC=ORCDRC, CEC=CEC, ENA=ENA, EACKCL=EACKCL,
   EXB=EXB, PHIHOX=PHIHOX, print.thresholds=TRUE)
x
## Most limiting: BLD.f and CRFVOL, but nothing < 20

## Effective Rootable Depth:
sel <- x==FALSE
if(!all(sel==FALSE)){
  UHDICM[which(sel==TRUE)[1]]
} else {
  max(LHDICM)
}

xI <- attr(x, "minimum.LRI")
## derive Effective rooting depth:
ERDICM(UHDICM=UHDICM, LHDICM=LHDICM, minimum.LRI=xI, DRAINFAO="M")
```

---

make.3Dgrid *Methods to prepare 3D prediction locations*

Description

Generates a list of objects of type "SpatialPixelsDataFrame" with longitude, latitude and altitude coordinates (these names are used by default for compatibility with the geosamples-class).

Usage

```
## S4 method for signature 'SpatialPixelsDataFrame'
make.3Dgrid(obj,
      proj4s = get("ref_CRS", envir = GSIF.opts),
      pixsize = get("cellsize", envir = GSIF.opts)[2],
      resampling_method = "bilinear",
      NAflag = get("NAflag", envir = GSIF.opts),
      stdepths = get("stdepths", envir = GSIF.opts),
      tmp.file = TRUE, show.output.on.console = TRUE, ...)
## S4 method for signature 'RasterBrick'
make.3Dgrid(obj,
      proj4s = get("ref_CRS", envir = GSIF.opts),
      pixsize = get("cellsize", envir = GSIF.opts)[2],
      resampling_method = "bilinear",
      NAflag = get("NAflag", envir = GSIF.opts),
      stdepths = get("stdepths", envir = GSIF.opts),
      tmp.file = TRUE, show.output.on.console = TRUE, ...)
```

Arguments

`obj`	object of class `"SpatialPixelsDataFrame"` or `"RasterBrick"`
`proj4s`	character; proj4string describing the target coordinate system
`pixsize`	grid cell size in decimal degrees (set by default at 1/1200 (0.0008333333 or 100 m around equator)
`resampling_method`	character; resampling method to be passed the reprojection algorithm
`NAflag`	character; missing value flag
`stdepths`	numeric; list of standard depths
`tmp.file`	logical; specifies whether a temporary file name should be generated
`show.output.on.console`	logical; specifies whether to print out the progress
`...`	optional arguments that can be passed to the reprojetion algorithm

Value

The output is list of objects of class `"SpatialPixelsDataFrame"` where the number of elements in the list corresponds to the number of standard depths.

Note

If the input object is of class `"SpatialPixelsDataFrame"`, the method by default uses `FWTools` (`warp` command) to resample grids, otherwise the `raster::projectRaster` command is passed. FWTools must be installed separately.
Note: this operation can be time consuming for large areas (e.g. » 1e6 pixels).

Author(s)

Tomislav Hengl

References

- Bivand, R.S., Pebesma, E.J., and Gómez-Rubio, V., (2008) Applied Spatial Data Analysis with R. Springer, 378 p.
- FWTools (http://fwtools.maptools.org)
- gdalUtils package (http://CRAN.R-project.org/package=gdalUtils)
- Raster package (http://CRAN.R-project.org/package=raster)

See Also

spc, geosamples-class, plotKML::reproject

Examples

```
## grids Ebergotzen:
library(plotKML)
library(rgdal)
library(raster)

data(eberg_grid)
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
##  convert to spatial components:
formulaString <- ~ PRMGEO6+DEMSRT6+TWISRT6+TIRAST6
eberg_spc <- spc(eberg_grid, formulaString)
## create 3D locations in the original coordinate system:
eberg_3Dxy <- sp3D(eberg_spc@predicted)
## Not run: ## wrapper function to create 3D locations in the default WGS84 system:
eberg_3D <- make.3Dgrid(eberg_spc@predicted)
image(eberg_3D[[1]]["PC1"])
## downscale 100 m resolution imagery to 25 m:
data(eberg_grid25)
gridded(eberg_grid25) <- ~x+y
proj4string(eberg_grid25) <- CRS("+init=epsg:31467")
eberg_grid25@data <- cbind(eberg_grid25@data,
    warp(eberg_grid, pixsize=eberg_grid25@grid@cellsize[1],
    GridTopology=eberg_grid25@grid, resampling_method="cubicspline")@data)
## this function requires FWTools!

## End(Not run)
```

makeGstatCmd *Make a gstat command script*

Description

Generates a command script based on the regression model and variogram. This can then be used to run predictions/simulations by using the pre-compiled binary `gstat.exe`.

Usage

```
makeGstatCmd(formString, vgmModel, outfile, easfile,
        nsim = 0, nmin = 20, nmax = 40, radius, zmap = 0,
        predictions = "var1.pred.hdr", variances = "var1.svar.hdr",
        xcol = 1, ycol = 2, zcol = 3, vcol = 4, Xcols)
```

Arguments

`formString`	object of class `"formula"` — regression model
`vgmModel`	object of class `"vgmmodel"` or `"data.frame"`
`outfile`	character; output file for the command script
`easfile`	character; file name for the GeoEAS file with observed values
`nsim`	integer; number of simulations
`nmin`	integer; smallest number of points in the search radius (see gstat user's manual)
`nmax`	integer; largest number of points in the search radius (see gstat user's manual)
`radius`	numeric; search radius (see gstat user's manual)
`zmap`	numeric; fixed value for the 3D dimension in the case of 3D kriging
`predictions`	character; output file name for predictions
`variances`	character; output file name for kriging variances
`xcol`	integer; position of the x column in the GeoEAS file
`ycol`	integer; position of the y column in the GeoEAS file
`zcol`	integer; position of the z column in the GeoEAS file
`vcol`	integer; position of the target variable column in the GeoEAS file
`Xcols`	integer; column numbers for the list of covariates

Details

To run the script under Windows OS you need to obtain the pre-compiled `gstat.exe` program from the www.gstat.org website, and put it in some directory e.g. `c:/gstat/`. Then add the program to your path (see environmental variable under Windows > Control panel > System > Advanced > Environmental variables), or copy the exe program directly to some windows system directory.

Note

The advantage of using `gstat.exe` is that it loads large grids much faster to memory than if you use gstat in R, hence it is potentially more suited for computing with large grids. The draw back is that you can only pass simple linear regression models to `gstat.exe`. The stand-alone gstat is not maintained by the author of gstat any more.

Author(s)

Tomislav Hengl

References

- Bivand, R.S., Pebesma, E.J., and Gómez-Rubio, V., (2008) Applied Spatial Data Analysis with R. Springer, 378 p.
- Pebesma, E., (2003) Gstat user's manual. Dept. of Physical Geography, Utrecht University, p. 100, www.gstat.org

See Also

`write.data`, `fit.gstatModel`, `gstat::krige`

Examples

```
## Not run:
library(sp)
library(gstat)

# Meuse data:
demo(meuse, echo=FALSE)
# fit a model:
omm <- fit.gstatModel(observations = meuse, formulaString = om~dist,
  family = gaussian(log), covariates = meuse.grid)
str(omm@vgmModel)
# write the regression matrix to GeoEAS:
meuse$log_om <- log1p(meuse$om)
write.data(obj=meuse, covariates=meuse.grid["dist"],
    outfile="meuse.eas", methodid="log_om")
writeGDAL(meuse.grid["dist"], "dist.rst", drivername="RST", mvFlag="-99999")
makeGstatCmd(log_om~dist, vgmModel=omm@vgmModel,
    outfile="meuse_om_sims.cmd", easfile="meuse.eas",
    nsim=50, nmin=20, nmax=40, radius=1500)
# compare the processing times:
system.time(system("gstat meuse_om_sims.cmd"))
vgmModel = omm@vgmModel
class(vgmModel) <- c("variogramModel", "data.frame")
system.time(om.rk <- krige(log_om~dist, meuse[!is.na(meuse$log_om),],
    meuse.grid, nmin=20, nmax=40, model=vgmModel, nsim=50))

## End(Not run)
```

MaxEnt *Prediction and cross-validation using the Maximum Entropy*

Description

Runs MaxEnt algorithm on a set of observations ("ppp" class from the spatstat package) and environmental covariates (of "SpatialPixelsDataFrame" class) and returns predicted probability of occurrence and cross-validation of models with presence/absence data.

Usage

```
## S4 method for signature 'ppp,SpatialPixelsDataFrame'
MaxEnt(occurrences, covariates,
          nfold = 5, Npoints = 1000, sciname = as.character(NA),
          period = c(Sys.Date()-1, Sys.Date()), ...)
```

Arguments

occurrences	object of type "ppp"; occurrences
covariates	object of type "SpatialPixelsData"; list of covariate layers
nfold	object of type "integer"; number of folds used for cross-validation
Npoints	object of type "integer"; number of points used for cross-validation
sciname	object of type "character"; usually species latin name (it can also be a surveyor's team name or a sampling design)
period	object of type "Date"; sampling period
...	for more additional arguments see dismo::predict

Value

Returns an object of type "SpatialMaxEntOutput" with the following slots: sciname (usually latin "genus" and "species" name), occurrences (occurrence-only records), TimeSpan.begin (begin of sampling), TimeSpan.end (end of sampling), maxent (object of class "MaxEnt" produced as an output of the dismo::maxent function), sp.domain (assumed spatial domain), and predicted (results of prediction produced using the MaxEnt software).

Note

MaxEnt is one of the standard tools used in ecology for Niche analysis and species distribution modelling. What makes it especially robust is the fact that it can take both continuous and factor data as inputs, and has no requirements considering the distribution of covariates (Phillips et al., 2006). In the example below, I use MaxEnt to analyze representation of feature space by a given soil sampling pattern (i.e. mis-representation or the sampling preference by the surveyors). For more information on how to install MaxEnt and use it in R, see dismo package documentation.

Author(s)

Tomislav Hengl

References

- Phillips, S.J., Anderson, R.P., Schapire, R.E., (2006) Maximum entropy modeling of species geographic distributions. Ecological Modelling, 190:231-259.
- MaxEnt software (http://www.cs.princeton.edu/~schapire/maxent/)
- Dismo package (http://CRAN.R-project.org/package=dismo)

See Also

dismo::maxent, plotKML::SpatialMaxEntOutput-class

Examples

```
# load data:
library(plotKML)
library(rJava)
library(spatstat)
library(maptools)
library(dismo)
library(rgdal)

data(eberg)
data(eberg_grid)
## prepare data for spatial analysis:
eberg.xy <- eberg[runif(nrow(eberg))<0.3,]
coordinates(eberg.xy) <- ~X+Y
proj4string(eberg.xy) <- CRS("+init=epsg:31467")
## format gridded data:
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
## convert to a "ppp" object:
eberg.ppp <- as.ppp(eberg.xy)
## run MaxEnt analysis (evaluates sampling bias or mis-representation):
jar <- paste(system.file(package="dismo"), "/java/maxent.jar", sep='')
if(file.exists(jar)){
  me.eberg <- MaxEnt(occurrences=eberg.ppp, covariates=eberg_grid)
  ## NOTE: MaxEnt can be time-consuming!
  # plot the results:
  par(mfrow=c(1,2), mar=c(0.5,0.5,0.5,0.5), oma=c(0,0,0,0))
  image(as(me.eberg@predicted, "SpatialPixelsDataFrame"),
    col=rev(heat.colors(25)), xlab="", ylab="")
  points(me.eberg@occurrences, pch="+", cex=.7)
  image(me.eberg@sp.domain, col="grey", xlab="", ylab="")
}
```

merge *Merge multiple predictions*

Description

Merges objects of class "SpatialPredictions" or "RasterBrickSimulations" and produces average predictions where the two objects overlap spatially. If the predictions are available at different resolutions, then it downscales all other grids to the smallest grid cell size using bicubic splines (for predictions) i.e. nearest neighbor algorithm (for simulations). Weigths can be passed via the RMSE.l argument, otherwise they will be estimated from validation slot (if objects are of the class "SpatialPredictions").

Usage

```
## S4 method for signature 'SpatialPredictions,SpatialPredictions'
merge(x, y, ..., RMSE.l = NULL, silent = TRUE)
```

Arguments

x	object of class "SpatialPredictions" or "RasterBrickSimulations"
y	object of class "SpatialPredictions" or "RasterBrickSimulations"
...	additional objects of class "SpatialPredictions" or "RasterBrickSimulations"
RMSE.l	numeric; list of mean prediction errors for each object (these are used as weights during the averaging)
silent	logical; specifies whether to print out the progress and used RMSE's

Value

Returns an object of type "SpatialPixelsDataFrame" or "RasterBrickSimulations" that contains only the merged values.

Note

Merging of multiple spatial predictions using weighted averaging is a heuristic approach to mapping. This method assumes that the predictions are completely independent (independent covariates, independent models), but this not might be the case and hence the merged predictions will be sub-optimal. Merging multiple predictions is however attractive for situations where the predictions do not have the same extent, so that spatial predictions with larger coverage can be used to fill in the gaps in locally produced predictions.

Author(s)

Tomislav Hengl and Gerard B.M. Heuvelink

References

- Heuvelink, G.B.M., Bierkens, M.F.P. (1992) Combining soil maps with interpolations from point observations to predict quantitative soil properties. Geoderma 55(1-2): 1-15.

`mpspline` *Fits a mass preserving spline*

Description

Fits a mass preserving spline to a soil profile data.

Usage

```
## S4 method for signature 'SoilProfileCollection'
mpspline(obj, var.name,
        lam = 0.1, d = t(c(0,5,15,30,60,100,200)), vlow = 0,
        vhigh = 1000, show.progress=TRUE)
```

Arguments

`obj`	object of class "`SoilProfileCollection`"
`var.name`	character; target variable name (must be a numeric variable)
`lam`	numeric; lambda the smoothing parameter
`d`	numeric; standard depths
`vlow`	numeric; smallest value of the target variable (smaller values will be replaced)
`vhigh`	numeric; highest value of the target variable (larger values will be replaced)
`show.progress`	logical; specifies whether to display the progress bar

Value

Returns a list with four elements:

`idcol` site ID column

`var.fitted` matrix; are are spline-estimated values of the target variable at observed depths (upper and lower depths are indicated as attributes)

`var.std` matrix; are spline-estimated values of the target variable at standard depths

`var.1cm` matrix; are spline-estimated values of the target variable using the 1 cm increments

Note

Target variable needs to be a numeric vector measured at least 2 horizons for the spline to be fitted. Profiles with 1 horizon are accepted and processed as per output requirements, but no spline is fitted as such. Only positive numbers for upper and lower depths can be accepted. It is assumed that soil variables collected per horizon refer to block support i.e. they represent averaged values for the whole horizon. This operation can be time-consuming for large data sets.

Author(s)

Brendan Malone and Tomislav Hengl

References

- Bishop, T.F.A., McBratney, A.B., Laslett, G.M., (1999) Modelling soil attribute depth functions with equal-area quadratic smoothing splines. Geoderma, 91(1-2): 27-45.
- Malone, B.P., McBratney, A.B., Minasny, B., Laslett, G.M. (2009) Mapping continuous depth functions of soil carbon storage and available water capacity. Geoderma, 154(1-2): 138-152.

See Also

`stats::spline`

Examples

```
library(aqp)
library(plyr)
library(sp)
## sample profile from Nigeria:
lon = 3.90; lat = 7.50; id = "ISRIC:NG0017"; FAO1988 = "LXp"
top = c(0, 18, 36, 65, 87, 127)
bottom = c(18, 36, 65, 87, 127, 181)
ORCDRC = c(18.4, 4.4, 3.6, 3.6, 3.2, 1.2)
munsell = c("7.5YR3/2", "7.5YR4/4", "2.5YR5/6", "5YR5/8", "5YR5/4", "10YR7/3")
## prepare a SoilProfileCollection:
prof1 <- join(data.frame(id, top, bottom, ORCDRC, munsell),
         data.frame(id, lon, lat, FAO1988), type='inner')
depths(prof1) <- id ~ top + bottom
site(prof1) <- ~ lon + lat + FAO1988
coordinates(prof1) <- ~ lon + lat
proj4string(prof1) <- CRS("+proj=longlat +datum=WGS84")
## fit a spline:
ORCDRC.s <- mpspline(prof1, var.name="ORCDRC")
str(ORCDRC.s)

## Example with multiple soil profiles
## Make some fake, but reasonable profiles:
rand.prof <- ldply(1:20, random_profile, n=c(6, 7, 8), n_prop=1, method='LPP')
## promote to SPC and plot
depths(rand.prof ) <- id ~ top + bottom
plot(rand.prof, color='p1')
## fit MP spline by profile
try( m <- mpspline(rand.prof, 'p1') )
```

OCSKGM *Soil organic carbon stock*

Description

Derive soil organic carbon stock / storage (in kilograms per square-meter) and propagated uncertainty for a given horizon/solum depth and based on soil organic carbon concentration, horizon/solum thickness, bulk density and percentage of coarse fragments.

Usage

```
OCSKGM(ORCDRC, BLD=1400, CRFVOL=0, HSIZE,
     ORCDRC.sd=10, BLD.sd=100, CRFVOL.sd=5, se.prop=TRUE)
```

Arguments

ORCDRC	numeric; soil organic carbon concentration in permille or g / kg
BLD	numeric; bulk density in kg / cubic-meter for the horizon/solum
CRFVOL	numeric; percentage of coarse fragments (above 2 mm in diameter) in the sample
HSIZE	numeric; thickness of the horizon/solum in cm
ORCDRC.sd	numeric; standard error of estimating ORCDRC (must be positive number)
BLD.sd	numeric; standard error of estimating BLD (must be positive number
CRFVOL.sd	numeric; standard error of estimating CRFVOL (must be positive number)
se.prop	logical; specifies whether to derive propagated error

Value

Soil organic carbon stock in kilograms per square-meter. To convert to tonnes per hectar multiply by 10.

Note

Propagated error (attached as an attribute) is estimated using the Taylor Series Method and shows only an approximate estimate. A more robust way to estimate the propagated uncertainty would be to use (geo)statistical simulations. See Heuvelink (1998) for more info.

Author(s)

Tomislav Hengl, Niels Batjes and Gerard Heuvelink

References

- Heuvelink, G. B. (1998) Error propagation in environmental modelling with GIS. CRC Press, 150 p.
- Nelson, D.W., and L.E. Sommers (1982) Total carbon, organic carbon, and organic matter. p. 539-580. In A.L. Page et al. (ed.) Methods of soil Analysis. Part 2. 2nd ed. Agron. Monogr. 9. ASA and SSSA, Madison, WI.

Examples

```
Area <- 1E4  ## 1 ha
HSIZE <- 30 ## 0--30 cm
ORCDRC <- 50  ## 5%
ORCDRC.sd <- 10  ## +/-1%
BLD <- 1500  ## 1.5 tonnes per cubic meter
BLD.sd <- 100  ## +/-0.1 tonnes per cubic meter
CRFVOL <- 10  ## 10%
CRFVOL.sd <- 5  ## +/-5%
x <- OCSKGM(ORCDRC, BLD, CRFVOL, HSIZE, ORCDRC.sd, BLD.sd, CRFVOL.sd)
x  ## 20.25 +/-4.41 kg/m^2
## in tonnes per ha:
x[[1]] * Area / 1000
```

---

predict.gstatModel-method

Predict from an object of class "gstatModel"

Description

Predicts from an object of class gstatModel-class using new prediction locations. The function combines predictions by regression (e.g. GLM) and interpolation of residuals (kriging) via the Regression-Kriging (RK) or Kriging with External Drift (KED, also known as Universal Kriging) framework.

Usage

```
## S4 method for signature 'gstatModel'
predict(object,
     predictionLocations, nmin = 10, nmax = 30, debug.level = -1,
     predict.method = c("RK", "KED"), nfold = 5, verbose = FALSE,
     nsim = 0, mask.extra = TRUE, block,
     zmin = -Inf, zmax = Inf, subsample = length(object@sp),
     coarsening.factor = 1, vgmmodel = object@vgmModel,
    subset.observations = !is.na(object@sp@coords[,1]), betas = c(0,1), extend = .5, ...)
## S4 method for signature 'list'
predict(object,
     predictionLocations, nmin = 10, nmax = 30, debug.level = -1,
     predict.method = c("RK", "KED"), nfold = 5, verbose = FALSE,
     nsim = 0, mask.extra = TRUE, block,
     zmin = -Inf, zmax = Inf, subsample = length(object@sp), ...)
```

Arguments

object	object of type "gstatModel"

predictionLocations
: object of type "SpatialPixelsDataFrame" prediction locations (must contain all covariates from the model)

nmin
: integer; minimum number of nearest observations sent to gstat::krige

nmax
: integer; maximum number of nearest observations sent to gstat::krige

debug.level
: integer; default debug level mode sent to gstat::krige

predict.method
: character; mathematical implementation of the gstat::krige interpolation method with covariates: Regression-Kriging (RK) or Kriging with External Drift (KED)

nfold
: integer; n-fold cross validation sent to gstat::krige.cv

verbose
: logical; specifies whether to supress the progress bar of the gstat::krige.cv

nsim
: integer; triggers the geostatistical simulations

mask.extra
: logical; specifies whether to mask out the extrapolation pixels (prediction variance exceeding the global variance)

block
: numeric; support size (block support for objects of type "SpatialPixelsDataFrame" is chosen by default)

zmin
: numeric; lower physical limit for the target variable

zmax
: numeric; upper physical limit for the target variable

subsample
: integer; sub-sample point observations to speed up the processing

coarsening.factor
: integer; coarsening factor (1:5) to speed up the processing

vgmmodel
: object of class data.frame corresponding to the gstat::vgm variogram

subset.observations
: logical; vector specifying the subset of observations used for interpolation

extend
: numeric; fraction of the range for which the spatial domain should be extended when searching for observations for kriging

betas
: numeric; vector of the beta coefficients to be passed to the gstat::krige

...
: other optional arguments that can be passed to gstat::krige and/or predict.glm

Details

Selecting predict.method = "KED" invokes simple kriging with external drift with betas set at 0 (intercept) and 1 (regression predictions used as the only covariate). This assumes that the regression model already results in an unbiased estimator of the trend model.
If not speficied otherwise, subset.observations by default selects only obserations within the spatial domain (bounding box) of the predictionLocations plus 50% of the one third of the extent of the area (extend). In the case of spatial duplicates in 2D or 3D, subset.observations will automatically remove all duplicates before running kriging. All points in 3D that stand exactly above each other will be removed by default.
Predictions can be speed up by using a larger coarsening.factor e.g. 2 to 5, in which case the ordinary kriging on residuals will run at a coarser resolution, and the output would be then downscaled to the original resolution using splines (via the warp method). In the case of predict.method = RK, the kriging variance is derived as a sum of the GLM variance and the OK variance, which is statistically sub-optimal.

Note

Predictions using `predict.method = "KED"` (the default gstat setting) can be time consuming for large data set and can result in instabilities (singular matrix problems) if the search radius is small and/or if all covariates contain exactly the same values. Predictions using `predict.method = "RK"` on the other hand can be speed up, but will typically underestimate the prediction variance (taken as a simple sum of the regression and ordinary kriging variances). Compare to the "KED" variance that includes also a cross-term (see Hengl et al. 2007 for more details).

Author(s)

Tomislav Hengl, Gerard B.M. Heuvelink and Bas Kempen

References

- Hengl T., Heuvelink G.B.M., Rossiter D.G., 2007. About regression-kriging: from equations to case studies. Computers and Geosciences, 33(10): 1301-1315.

See Also

`gstatModel-class`, `fit.gstatModel`

`REST.SoilGrids-class` *A class for SoilGrids REST API*

Description

A class for SoilGrids REST API Service. Can be used to overlay points or fetch grid values from SoilGrids Soil Information System.

Slots

`server`**:** object of class `"character"`; contains the location of the server that executes REST.SoilGrids calls

`query`**:** object of class `"list"`; contains parameters or REST.SoilGrids query

`stream`**:** object of class `"character"`; contains parameters or REST.SoilGrids stream operation

Methods

over `signature(x = "REST.SoilGrids", y = "SpatialPoints")`: overlays spatial points and the target grids defined via the REST.SoilGrids-class (point-by-point) and returns list of objects of `"SpatialPixelsDataFrame"`-class

Note

More examples of overlay and download functions are available via `http://rest.soilgrids.org/`. `over` method is not recommended for large point data sets.

Author(s)

Tomislav Hengl & Jorge S. Mendes de Jesus

References

- SoilGrids — a system for automated soil mapping (http://www.soilgrids.org)

See Also

SoilGrids-class, WPS-class

Examples

```
## Not run:
library(rjson)
library(sp)
## 2 points:
pnts <- data.frame(lon=c(10.65,5.36), lat=c(51.81,51.48), id=c("p1","p2"))
coordinates(pnts) <- ~lon+lat
proj4string(pnts) <- CRS("+proj=longlat +datum=WGS84")
pnts
## REST example:
soilgrids.r <- REST.SoilGrids(c("ORCDRC","PHIHOX"))
ov <- over(soilgrids.r, pnts)
ORCDRC.pnt1 <- data.frame(
  top=unlist(ov[1,grep("depthCodesMeters", names(ov))])*-100,
  M=unlist(ov[1,grep("ORCDRC.M", names(ov))]),
  L=unlist(ov[1,grep("ORCDRC.L", names(ov))]),
  U=unlist(ov[1,grep("ORCDRC.U", names(ov))]))
ORCDRC.pnt1$variable <- "ORCDRC"
## plot the result:
library(lattice)
library(aqp)
data(soil.legends)
## Soil organic carbon:
ORCDRC.range = range(soil.legends[["ORCDRC"]]$MIN, soil.legends[["ORCDRC"]]$MAX)
dev.new(width=5, height=6)
xyplot(top ~ M | variable, data=ORCDRC.pnt1, ylab='Depth in cm',
  xlab='5th and 95th percentiles', xlim=ORCDRC.range,
  lower=ORCDRC.pnt1$L, upper=ORCDRC.pnt1$U, ylim=c(150,0),
  panel=panel.depth_function,
  alpha=0.25, sync.colors=TRUE,
  par.settings=list(superpose.line=list(col='RoyalBlue', lwd=3)),
  strip=strip.custom(bg=grey(0.8))
)

## Soil pH:
PHIHOX.range = range(soil.legends[["PHIHOX"]]$MIN, soil.legends[["PHIHOX"]]$MAX)
for(i in 1:nrow(ov)){
  PHIHOX.pnt <- data.frame(
    top=unlist(ov[i,grep("depthCodesMeters", names(ov))])*-100,
    M=unlist(ov[i,grep("PHIHOX.M", names(ov))]),
```

```
    L=unlist(ov[i,grep("PHIHOX.L", names(ov))]),
    U=unlist(ov[i,grep("PHIHOX.U", names(ov))]))
  PHIHOX.pnt$variable <- "PHIHOX"
  png(paste("PHIHOX_depth_", i, ".png", sep=""), width=300, height=6/5*300)
  p <- xyplot(top ~ M/10 | variable, data=PHIHOX.pnt, ylab='Depth in cm',
    xlab='5th and 95th percentiles', xlim=PHIHOX.range/10,
    lower=PHIHOX.pnt$L/10, upper=PHIHOX.pnt$U/10, ylim=c(150,0),
    panel=panel.depth_function,
    alpha=0.25, sync.colors=TRUE,
    par.settings=list(superpose.line=list(col='Red', lwd=3)),
    strip=strip.custom(bg=grey(0.8))
  )
  print(p)
  graphics.off()
}
## plot in Google Earth:
library(plotKML)
kml(pnts, colour=id, file="PHIHOX_depth.kml",
  shape=paste("PHIHOX_depth_", 1:nrow(ov), ".png", sep=""),
  size=6, points_names=pnts$id,
  colour_scale=rep("#FFFFFF", 2))

## End(Not run)
```

`sample.grid` *sample spatial points by grids*

Description

Get a subset of a object of class "SpatialPoints" or "SpatialPointsDataFrame" avoiding spatial clustering.

Usage

```
## S4 method for signature 'SpatialPoints'
sample.grid(obj, cell.size, n, bbox, ...)
## S4 method for signature 'SpatialPointsDataFrame'
sample.grid(obj, cell.size, n, bbox, ...)
```

Arguments

obj	"SpatialPoints*" object
cell.size	numeric; the cell size of the overlayed "SpatialGridDataFrame" in the form of c(x,y)
n	integer; specifies maximum number points in each grid
bbox	matrix; the bounding box of output "SpatialPoints" or "SpatialPointsDataFrame"; it is set the same as the obj if missing
...	other optional arguments that can be passed to over

Value

Returns a list of two objects: (1) an object of type "SpatialPoints" or "SpatialPointsDataFrame" that contains a subset of the obj, and (2) resulting grid.

Note

Spatial points are overlayed with spatial grids with a specified cell size and then get a subset from each grid with a specified number at most. If one grid has less points than the specified number, all the points are taken. If one grid has more points than the specified number, only this number of points are taken by sample. This function can be used when there are too much point observations to be handled, especially for spatially clustered observations. The total number of sampled points are determined by cell.size and n together. You will get fewer the sampled points when cell.size is larger, or/and when n is smaller. Similar sample sizes can be achieved by differen combination of cell.size and n.

Author(s)

Wei Shangguan

Examples

```
library(sp)
data(isis)
profs <- isis[["sites"]]
coordinates(profs) <- ~  LONWGS84 + LATWGS84
proj4string(profs) <- CRS("+proj=longlat +datum=WGS84")
## sample SpatialPointsDataFrame:
#bbox <- matrix(c(-180, -90, 180, 90), nrow=2)
prof1 <- sample.grid(profs, cell.size = c(5,5), n = 1)
l0 <- list("sp.points", profs, pch=1, col="red")
l1 <- list("sp.points", prof1$subset, pch="+", col="black", cex=1.2)
spplot(prof1$grid, scales=list(draw=TRUE),
   col.regions="grey", sp.layout=list(l0, l1))
## Subsampling ratio:
round(length(prof1$subset)/length(profs)*100, 1)
```

---

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

`BLD` numeric; bulk density in kg per cubic meter

`CEC` numeric; Cation Exchange Capacity

`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)

`TAXGWRB` factor; World Reference base groups

`TAXOUSDA` factor; Keys to Soil Taxonomy suborders

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 (http://soilprofiles.org).

Author(s)

Tomislav Hengl

References

- Global Soil Information Facilities (http://gsif.isric.org)

Examples

```
data(soil.legends)
pal <- soil.legends$ORCDRC$COLOR
names(pal) <- signif((soil.legends$ORCDRC$MAX +
    soil.legends$ORCDRC$MIN)/2, 3)
pal
data(soil.vars)
soil.vars[soil.vars$varname=="ORCDRC",]
## make SAGA GIS palette:
makeSAGAlegend(x=as.factor(names(pal)), col_pal=pal,
    filename="ORCDRC.txt")
```

`SoilGrid.validator` *Validate SoilGrid (spatial predictions)*

Description

Validate SoilGrid (spatial predictions) i.e. soil property maps following the GSIF validation protocol.

Usage

```
SoilGrid.validator(obj, domain, ground.truth, N.sample=2000,
   xml.file, z.lim, md.type="INSPIRE", test.URL=FALSE)
```

Arguments

`obj`	`"GDALobj"` object i.e. a pointer to a spatial layer of interest (single slice)
`domain`	`"GDALobj"` object i.e. a pointer to a spatial layer contain soil mask
`ground.truth`	`"SpatialPointsDataFrame"`; contains values of the target variable at exactly the same depth / same support size sampled either using Simple Random Sampling or regular sampling on a grid
`N.sample`	integer; random sampling size
`xml.file`	character; metadata file (should have the same name as `obj` file)
`z.lim`	numeric; upper and lower physical limits
`md.type`	character; metadata standard (currently INSPIRE)
`test.URL`	logical; specifies whether to validate XML schema / test download times and proj4 string

Value

Returns a list with validation results. Explanation of codes is available in the SoilGrids.org data validation protocol.

Note

One SoilGrid layer (2D slice) basically contains predictions on a regular grid for a specific soil depth (at either point or block support). The `ground.truth` data must refer to the exactly the same depth and the same support size and should ideally be collected using some probability spatial sampling (see e.g. `sp::spsample`). To estimate values of soil properties at standard depths, consider using `mpspline` function.
Numeric resolution is derived as estimated RMSE/2. Numeric resolution can be best specified as `Attribute_Measurement_Resolution` (the smallest unit increment to which an attribute value is measured).
Increasing `N.sample` can lead to more precise results at the cost of higher computing time.

Author(s)

Tomislav Hengl

See Also

`plotKML::spMetadata`

`SoilGrids-class` *A class for SoilGrids — soil property and/or class maps*

Description

A class containing predictions and prediction error (or multiple realizations) of some of the target global soil property at six standard depths. Standard depths used are based on the GlobalSoilMap.net specifications: `sd1` = 2.5 cm (0–5), `sd2` = 10 cm (5–15), `sd3` = 22.5 cm (15–30), `sd4` = 45 cm (30–60), `sd5` = 80 cm (60–100), `sd6` = 150 cm (100–200).

Slots

`varname`**:** object of class `"character"`; abbreviated variable name registered in the Global Soil Data registry

`TimeSpan`**:** object of class `"list"`; contains begin and end of the sampling period of class `"POSIXct"`

`sd1`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 2.5 cm (0–5)

`sd2`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 10 cm (5–15)

`sd3`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 22.5 cm (15–30)

`sd4`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 45 cm (30–60)

`sd5`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 80 cm (60–100)

`sd6`**:** object of class `"SpatialPixelsDataFrame"`; predictions and variances, or number of realizations of the target variable at depth 150 cm (100–200)

Gridded data submitted to `sd*` slots of the `"SoilGrids"` class must satisfy all of the following requirements (class validity):

- All grids submitted must have the same grid topology (identical `grid` slot in the object of class `"SpatialPixelsDataFrame"`);
- All grids must be projected in the referent coordinate system WGS84 (geographical coordinates), with 3D dimension (altitude) expressed as distance from the land surface in meters (e.g. altitude of `-.025` corresponds to the 2.5 cm depth);
- The grid cell size must correspond to some standard resolution e.g. 0.0008333333 (1/1200 or about 100 m), 0.0016666667 (1/600 or about 250 m) or similar;
- Only standard abbreviated names registered in the Global Soil Data registry can be used in the `varname` slot;

Methods

summary `signature(x = "SoilGrids")`: generates summary statistics for the object

Author(s)

Tomislav Hengl and Robert A. MacMillan

References

- SoilGrids — a system for automated soil mapping (http://www.soilgrids.org)

See Also

GlobalSoilMap-class, SpatialComponents-class, geosamples-class

Examples

```
# load soil samples from the plotKML package:
library(plotKML)
library(aqp)
library(plyr)
library(splines)
library(rgdal)
library(raster)

data(eberg)
## subset data to 10%:
eberg <- eberg[runif(nrow(eberg)) < .1,]
## sites table:
s.lst <- c("ID", "soiltype", "TAXGRSC", "X", "Y")
h.lst <- c("UHDICM","LHDICM","SNDMHT","SLTMHT","CLYMHT")
sites <- eberg[,s.lst]
## get horizons table:
horizons <- getHorizons(eberg, idcol="ID", sel=h.lst)
## create object of type "SoilProfileCollection"
eberg.spc <- join(horizons, sites, type='inner')
depths(eberg.spc) <- ID ~ UHDICM + LHDICM
site(eberg.spc) <- as.formula(paste("~", paste(s.lst[-1], collapse="+"), sep=""))
coordinates(eberg.spc) <- ~X+Y
proj4string(eberg.spc) <- CRS("+init=epsg:31467")
## convert to logits:
eberg.spc@horizons$SNDMHT.t <- log((eberg.spc@horizons$SNDMHT/100)/
    (1-eberg.spc@horizons$SNDMHT/100))
## convert to geosamples:
eberg.geo <- as.geosamples(eberg.spc)
## load gridded data:
data(eberg_grid)
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
## derive spc's:
formulaString <- ~ PRMGEO6+DEMSRT6+TWISRT6+TIRAST6
eberg_spc <- spc(eberg_grid, formulaString)
```

```
## build a 3D "gstatModel":
glm.formulaString = as.formula(paste("SNDMHT.t ~ ",
     paste(names(eberg_spc@predicted), collapse="+"), "+ ns(altitude, df=4)"))
## Not run:
SNDMHT.m <- fit.gstatModel(observations=eberg.geo, glm.formulaString,
     covariates=eberg_spc@predicted)
summary(SNDMHT.m@regModel)
SNDMHT.m@vgmModel
## prepare new locations (6 standard depths):
new3D <- sp3D(eberg_spc@predicted)
## Make predictions at six depths:
sd.l <- lapply(new3D, FUN=function(x){predict(SNDMHT.m, predictionLocations=x, nfold=0)})
## back-transformation function:
invlogit = function(x){exp(x)/(1+exp(x))*100}
## for the back-transformation for the mean value see Diggle and Ribeiro, 2007, p. 148:
invlogit.m = function(x, v){((1+exp(-x))^(-1)-.5*v*exp(-x)*(1-exp(-x))*(1+exp(-x))^(-3) )*100}
## back-transform values from logits:
for(j in 1:length(sd.l)){
    sd.l[[j]]@predicted$M <- round(invlogit.m(sd.l[[j]]@predicted$SNDMHT.t,
       sd.l[[j]]@predicted$var1.var))
    sd.l[[j]]@predicted$L <- round(invlogit(sd.l[[j]]@predicted$SNDMHT.t
     - 1.645*sqrt(sd.l[[j]]@predicted$var1.var)))
    sd.l[[j]]@predicted$U <- round(invlogit(sd.l[[j]]@predicted$SNDMHT.t
     + 1.645*sqrt(sd.l[[j]]@predicted$var1.var)))
}
str(sd.l[[1]]@predicted@data)

## reproject to WGS84 system (100 m resolution):
p = get("cellsize", envir = GSIF.opts)[1]
s = get("stdepths", envir = GSIF.opts)
sd.ll <- sapply(1:length(sd.l), FUN=function(x){
     make.3Dgrid(sd.l[[x]]@predicted[c("L","M","U")],
     pixsize=p, stdepths=s[x])})
## save to a "SoilGrids" object:
SNDMHT.gsm <- SoilGrids(obj=sd.ll, varname="SNDPPT",
             TimeSpan=list(begin="1999-02-01", end="2001-07-01"))
str(SNDMHT.gsm, max.level=2)
## visualize all maps in Google Earth:
data(R_pal)
z0 = mean(eberg_grid$DEMSRT6, na.rm=TRUE)
## export grids:
for(j in 1:length(sd.ll)){
  kml(slot(SNDMHT.gsm, paste("sd", j, sep="")),
     folder.name = paste("eberg_sd", j, sep=""),
     file = paste("SNDMHT_sd", j, ".kml", sep=""),
     colour = M, z.lim=c(10,85),
     raster_name = paste("SNDMHT_sd", j, ".png", sep=""),
     altitude = z0+5000+(s[j]*2500))
}

## End(Not run)
```

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

See Also

spc

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

spfkm, SpatialComponents-class

spc *Derive Spatial Predictive Components*

Description

Derives Spatial Predictive Components for a given set of covariates. It wraps the `stats::prcomp` method and predicts a list principal components for an object of type `"SpatialPixelsDataFrame"`.

Usage

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

Arguments

`obj`	object of class `"SpatialPixelsDataFrame"` (must contain at least two grids) or a list of objects of type `"SpatialPixelsDataFrame"`
`formulaString`	object of class `"formula"` or a list of formulas
`scale.`	object of class `"logical"`; specifies whether covariates need to be scaled
`silent`	object of class `"logical"`; specifies whether to print the progress
`...`	additional arguments that can be passed to `stats::prcomp`

Value

`spc` returns an object of type `"SpatialComponents"`. This is a 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. This operation can be time consuming for large grids.

Author(s)

Tomislav Hengl

See Also

`stats::prcomp`, `SpatialComponents-class`

Examples

```
# load data:
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;
## Not run: # 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)

## End(Not run)
```

---

spfkm *Supervised fuzzy* k*-means on spatial pixels*

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 string
observations	object of type "SpatialPointsData"; occurrences of factors
covariates	object of type "SpatialPixelsData" or "RasterBrick"; list of covariate layers
class.c	object of type "matrix"; class centres (see examples below)
class.sd	object of type "matrix"; class deviations (see examples below)
fuzzy.e	object of type "numeric"; fuzzy exponent

Value

Returns an object of type "SpatialMemberships" with following slots: predicted (classes predicted either by the multinomial logistic regression or fuzzy *k*-means), model (the multinomial logistic regression model; if available), mu (memberships derived using the fuzzy *k*-means), class.c (submitted or derived class centres), class.sd (submitted or derived class deviations), confusion (confusion matrix).

Note

Although nnet::multinom is consider to be robust and suited for large data sets, function might not converge in some cases or result in artifacts. If this happens try setting up the class centres and variances manually.

Author(s)

Tomislav Hengl and Bas Kempen

References

- Burrough, P. A., Gaans, P.F.M., and Van Hootsmans, R., (1997) Continuous classification in soil survey: spatial correlation, confusion and boundaries. Geoderma, 77(2-4), 115–135.
- Hengl T., Walvoort D.J.J., Brown, A., (2004) A double continuous approach to visualisation and analysis of categorical maps. Int. Jou. of Geographical Information Science, 18(2): 183-202.

See Also

spmultinom, SpatialMemberships-class, nnet::multinom

Examples

```
# load data:
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)
## Not run: # plot memberships:
pal = seq(0, 1, 1/50)
```

```
spplot(eberg_sm@mu, col.regions=grey(rev(pal)))
# predict soil properties using memberships:
glm.formulaString = as.formula(paste("SNDMHT_A ~ ",
    paste(names(eberg_sm@mu), collapse="+"), "-1"))
SNDMHT.m2 <- fit.gstatModel(observations=eberg, glm.formulaString,
    covariates=eberg_sm@mu)
summary(SNDMHT.m2@regModel)
# Coefficients correspond to the class centres;

## End(Not run)
```

`spline.krige` *Kriging combined with splines*

Description

Combines kriging and spline interpolation to speed up the kriging with minimal loss in precision, whilst reducing generation of artifacts. Spline interpolation is implemented via the SAGA GIS function `"Multilevel B-Spline Interpolation"` (SAGA GIS needs to be installed separately).

Usage

```
spline.krige(formula, locations, newdata, newlocs = NULL, model,
    te = as.vector(newdata@bbox), file.name, silent = FALSE,
    t_cellsize = newdata@grid@cellsize[1], optN = 20, quant.nndist = .5,
    nmax = 30, predictOnly = FALSE, resample = TRUE, saga.env,
    saga.lib=c("grid_spline","grid_tools"), saga.module=c(4,0), ...)
```

Arguments

`formula`	formula that defines the dependent variable as a linear model of independent variables; usually in the form `z~1`
`locations`	object of class `SpatialPoints`; sampling locations
`newdata`	object of class `SpatialPixels*`; spatial domain of interest
`newlocs`	object of class `SpatialPoints*`; prediction locations produced using the `resample.grid` function (if missing it will be generated using the `resample.grid` function)
`model`	variogram model of dependent variable (or its residuals); see `gstat::krige`
`te`	numeric; a vector in the form `c(xmin,ymin,xmax,ymax)`; sets bounding box of the kriging predictions
`file.name`	character; optional output file name pattern (without any file extension)
`silent`	logical; specifies whether to print out the progress
`t_cellsize`	numeric; target cell size (output grid)
`optN`	integer; optimal number of prediction locations per sampling location e.g. 1 sampling location is used to predict values for 20 new pixels
`quant.nndist`	numeric; threshold probability to determine the search radius (sigma)

nmax	integer; the number of nearest observations that should be used for kriging
predictOnly	logical; specifies whether to generate only predictions (var1.pred column)
resample	logical; specifies whether to down or upscale SAGA GIS grids to match the grid system of newdata
saga.env	list; path to location of the SAGA binaries (extracted using rsaga.env())
saga.lib	character; names of the SAGA libraries used
saga.module	integer; corresponding module numbers
...	other optional arguments that can be passed to function gstat::krige

Value

Returns an object of class "SpatialGridDataFrame", or an output file name.

Note

This function adjusts grid density (prediction locations) in reference to the actual local sampling intensity. High resolution grids are created where sampling density is higher and vice versa (Hengl, 2006). Low resolution grids (due to sparse data) are then downscaled to the target resolution using spline interpolation. This allows for speeding up the kriging with minimal loss in precision, whilst reducing generation of artifacts. Spline interpolation is implemented via the SAGA GIS v2.1 function "Multilevel B-Spline Interpolation" using the default settings. This function is especially suitable for producing predictions for large grids where the sampling locations show high spatial clustering. It is NOT intended for predicting using point samples collected using sampling designs with constant spatial sampling intensity e.g. point samples collected using simple random sampling or grid sampling.

Author(s)

Tomislav Hengl

References

- Hengl T., (2006) Finding the right pixel size. Computers and Geosciences, 32(9): 1283-1298.
- SAGA GIS (http://sourceforge.net/projects/saga-gis/)
- SpatStat package (http://www.spatstat.org)

Examples

```
## Not run:
library(plotKML)
library(spatstat)
library(RSAGA)
library(gstat)
library(raster)
data(eberg)
data(eberg_grid)
data(eberg_grid25)
library(sp)
```

```
coordinates(eberg) <- ~X+Y
proj4string(eberg) <- CRS("+init=epsg:31467")
m <- vgm(psill=320, model="Exp", range=1200, nugget=160)
plot(variogram(SNDMHT_A~1, eberg[!is.na(eberg$SNDMHT_A),]), m)
## prediction locations:
gridded(eberg_grid) <- ~x+y
proj4string(eberg_grid) <- CRS("+init=epsg:31467")
gridded(eberg_grid25) <- ~x+y
proj4string(eberg_grid25) <- CRS("+init=epsg:31467")
## prepare prediction locations for spline.krige:
grd <- resample.grid(locations=eberg["SNDMHT_A"], t_cellsize=25,
   newdata=eberg_grid25, optN=5, quant.nndist=.9)
## plot resampled grid:
plot(raster(grd$density))
plot(grd$newlocs)
points(eberg, pch=19, col="red", cex=.7)
env <- rsaga.env()
if(exists("env") & env$version=="2.1.0"){
 ## compare processing time:
 system.time( SND.sok <- spline.krige(locations=eberg["SNDMHT_A"],
      t_cellsize=25, newdata=eberg_grid25,
      newlocs=grd$newlocs, model=m, nmax=30) )
 system.time( SND.ok <- krige(SNDMHT_A~1,
      eberg[!is.na(eberg$SNDMHT_A),],
      newdata=eberg_grid, m,
      debug.level = -1, nmax=30) )
 system.time( SND.ok25 <- krige(SNDMHT_A~1,
      eberg[!is.na(eberg$SNDMHT_A),],
      newdata=eberg_grid25, m,
      debug.level = -1, nmax=30) )
 ## compare outputs visually:
 par(mfrow=c(1,3))
 plot(raster(SND.sok[1]), main="spline.krige (25 m)")
 plot(raster(SND.ok25[1]), main="krige (25 m)")
 plot(raster(SND.ok[1]), main="krige (100 m)")
}

## End(Not run)
## conclusion: spline.krige produces less artifacts,
## and is at order of magnitude faster than simple 'krige'
```

`spmultinom` *Multinomial logistic regression on spatial objects*

Description

Runs the multinomial logistic regression via `nnet::multinom` to produce spatial predictions of the target factor-type variable. It requires point locations of observed classes and a list of covariate layers provided as `"SpatialPixelsDataFrame-class"` object. The resulting predicted classes are then used to estimate class centres and variances per class.

Usage

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

Arguments

formulaString	formula string
observations	object of type "SpatialPointsData"; occurrences of factors
covariates	object of type "SpatialPixelsData"; list of covariate layers
class.stats	logical; species wether to estimate class centres
predict.probs	logical; species wether to predict probabilities per class
...	optional arguments

Value

Returns an object of type "SpatialMemberships" with following slots: predicted (classes predicted by the multinomial logistic regression, model (the multinomial logistic regression model), mu (probabilities derived using the mutinom model), class.c (derived class centres), class.sd (derived class deviations), confusion (confusion matrix).

Author(s)

Bas Kempen and Tomislav Hengl

References

- Multinomial logistic regression (http://en.wikipedia.org/wiki/Multinomial_logit)
- Nnet package (http://CRAN.R-project.org/package=nnet)

See Also

spfkm, SpatialMemberships-class

Examples

```
# load data:
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 <- spmultinom(formulaString, eberg, eberg_spc@predicted)
## Not run: # plot memberships:
pal = seq(0, 1, 1/50)
spplot(eberg_sm@mu, col.regions=pal)
image(eberg_sm@mu[1], col=pal)
text(eberg@coords, paste(eberg$soiltype), cex=.6, col="black")
# classes predicted:
Ls = length(levels(eberg_sm@predicted$soiltype))
pnts = list("sp.points", eberg, pch="+", cex=.6, col="black")
spplot(eberg_sm@predicted, col.regions=rainbow(Ls)[rank(runif(Ls))], sp.layout=pnts)

## End(Not run)
```

`spsample.prob` *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 geostatistical modelling (e.g. as weights for regression modeling).

Usage

```
## S4 method for signature 'SpatialPoints,SpatialPixelsDataFrame'
spsample.prob(observations, covariates,
  quant.nndist=.95, n.sigma, ...)
```

Arguments

`observations`	object of class `SpatialPoints`; sampling locations
`covariates`	object of class `SpatialPixelsDataFrame`; list of covariates of interest
`quant.nndist`	numeric; threshold probability to determine the search radius (sigma)
`n.sigma`	numeric; size of sigma used for kernel density estimation (optional)
`...`	other optional arguments that can be passed to function `spatstat::density`

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)

Tomislav 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.

See Also

`maxlike-package`, `spatstat-package`

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

## 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

## all at once:
gm <- fit.gstatModel(eberg.xy, fs, covs, weights=w)
plot(gm)
```

---

summary-methods *Summarize an object of class* "SpatialPredictions"

Description

Derives a statistical summary for an object of class "SpatialPredictions".

Usage

```
## S4 method for signature 'SpatialPredictions'
summary(object)
```

Arguments

`object`	object of class `"SpatialPredictions"`

Details

The function creates a summary table with standard column names. These tell us what is the summary accuracy of the spatial predictions and what are the effective bytes of information produced.

Value

The summary returns a data.frame with the following columns:

`"variable"` variable name

`"minium"` lowest value observed

`"maximum"` largest value observed

`"npoints"` number of observations

`"area"` lowest value observed

`"area.units"` area units either square-m or square-arcdegrees

`"covariates"` list of covariates used

`"family"` GLM family (if applicable)

`"RMSE"` RMSE derived using cross-validation

`"tvar"` variance percent explained by the model using the cross-validation

`"npixels"` total number of produced pixels

`"breaks"` breaks based on the half RMSE

`"bonds"` lower and upper boundaries for effective classes

`"Bytes"` effective bytes produced (see Hengl et al (2012) for more details)

`"compress"` compression algorithm used

Author(s)

Tomislav Hengl

References

- Hengl, T., Nikolic, M., MacMillan, R.A., (2013) Mapping efficiency and information content. International Journal of Applied Earth Observation and Geoinformation, special issue Spatial Statistics Conference, 22: 127–138.

See Also

`plotKML::SpatialPredictions-class`

Examples

```
## load observations:
library(sp)
library(rgdal)
library(gstat)
demo(meuse, echo=FALSE)
## fit a model:
omm <- fit.gstatModel(meuse, om~dist,
  fit.family=gaussian(link="log"), meuse.grid)
show(omm@regModel)
## produce SpatialPredictions:
om.rk <- predict(omm, predictionLocations = meuse.grid)
x = summary(om.rk)
str(x)
```

---

test.gstatModel-methods

Methods to test predictability of a regression-kriging model

Description

Tests predictability of a regression-kriging model on a sample data set. Automates model fitting, cross-validation and prediction and prints out: (1) RMSE at validation points under different sampling intensities, (2) number of predictions per second and (3) number of prediction failures (failure = predictions where cross-validation z-scores exceed value of +/- 1.5 or cross-validation residuals exceed three standard deviations of the observed values).

Usage

```
## S4 method for signature
## 'SpatialPointsDataFrame,formula,SpatialPixelsDataFrame'
test.gstatModel(observations, formulaString, covariates, Ns,
   predictionLocations, save.predictions = TRUE, debug.level = 0, nfold = 5, ...)
## S4 method for signature 'geosamples,formula,SpatialPixelsDataFrame'
test.gstatModel(observations, formulaString, covariates, Ns,
   predictionLocations, save.predictions = TRUE, debug.level = 0, nfold = 5, ...)
```

Arguments

observations	object of type "SpatialPointsDataFrame" or "geosamples-class"
formulaString	object of type "formula" or a list of formulas
covariates	object of type "SpatialPixelsDataFrame", or list of grids
Ns	vector; list of sampling intensities (maximum should not exceed the total number of samples)

predictionLocations
: object of class "SpatialPixelsDataFrame"; if not specified then passes the object covariates

save.predictions
: logical; indicates whether the prediction results should also be saved

debug.level
: integer; gstat's setting to hide the progress output

nfold
: integer; number of folds for cross-validation

...
: other optional arguments that can be passed to fit.gstatModel

Note

Vector of sampling intensities, if not provided, will be estimated as: sequence of 10 numbers on square root scale (where N minimum is determined as 20 + number of covariates times 10 and N maximum is the total number of observations). Where no model can be fitted, function returns an empty set. This function can be time consuming for large data sets and is hence recommended only for testing a mapping algorithm using sample data.

Author(s)

Tomislav Hengl, Gerard B.M. Heuvelink

See Also

fit.gstatModel, gstatModel-class

Examples

```
# 2D model:
library(sp)
## load the Meuse data set:
demo(meuse, echo=FALSE)
## model diagnostics:
t1 <- test.gstatModel(meuse, om~dist+ffreq, meuse.grid,
   fit.family = gaussian(log), Ns=c(80, 155))
t1[[1]]
```

---

tile *Tiles (subsets or clips) a spatial object to regular blocks*

Description

Tiles objects of class "Spatial*" or "RasterLayer*" into regular blocks.

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	object of class "Spatial*" "RasterLayer"
y	list of "SpatialPolygons"; if missing will be derived based on block.x
block.x	numeric; size of block in meters or corresponding mapping units
tmp.file	logical; specifies whether to generate a temporary file
program	character; location of the auxiliary program in the system
show.output.on.console	logical; specifies whether to print the progress of a function
...	optional arguments that can be passed to the getSpatialTiles

Details

When working with objects of type "SpatialLinesDataFrame", "SpatialPolygonsDataFrame" and or "RasterLayer", the function looks for FWTools binary files ogr2ogr and warp. FWTools is a separate program and must be installed separately.

Value

Returns a list of objects of the same class as the input object.

Author(s)

Tomislav Hengl

See Also

getSpatialTiles

Examples

```
## spatial pixels:
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)
## Not run: ## lines:
library(plotKML)
data(eberg_contours)
line.lst <- tile(eberg_contours, block.x=5000)
spplot(line.lst[[1]][2])
## polygons:
data(eberg_zones)
## this one requires ogr2ogr function:
pol.lst <- tile(eberg_zones, block.x=5000)
spplot(pol.lst[[1]][1])
## 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)

## End(Not run)
```

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

Description

Probability density for texture triangle (USDA system) based on global soil profile data (http://soilprofiles.org).

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.

See Also

FAO.SoilProfileCollection, soil.dom

Examples

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

## Not run: library(soiltexture)
## convert textures by hand to sand, silt and clay:
TEXMHT <- c("CL","C","SiL","SiL","missing")
x <- TT2tri(TEXMHT)
x

## End(Not run)
```

warp *(GDAL) warp function from FWTools*

Description

Reproject and resample using (GDAL) warp program.

Usage

```
## S4 method for signature 'SpatialPixelsDataFrame'
warp(obj, proj4s = proj4string(obj),
      GridTopology = NULL, pixsize,
      resampling_method = "bilinear",
      NAflag = get("NAflag", envir = GSIF.opts),
      tmp.file = FALSE, show.output.on.console = FALSE, program)
## S4 method for signature 'RasterLayer'
warp(obj, proj4s = proj4string(obj),
      GridTopology = NULL, pixsize,
      resampling_method = "bilinear",
      NAflag = get("NAflag", envir = GSIF.opts),
      tmp.file = FALSE, show.output.on.console = FALSE, program)
```

Arguments

obj	object of class "SpatialPixelsDataFrame" or class "RasterLayer"
proj4s	character; proj4string describing the target coordinate system
GridTopology	optional grid topology from sp package
pixsize	grid cell size in decimal degrees
resampling_method	character; resampling method (see gdalwarp options)
NAflag	character; missing value flag
tmp.file	logical; specifies whether a temporary file name should be generated
show.output.on.console	logical; specifies whether to print out the progress
program	full path to the (GDAL) warp program

Note

FWTools must be installed separately. See also gdalUtils package.

Author(s)

Tomislav Hengl

See Also

make.3Dgrid, plotKML::reproject

WPS-class *A class for a Web Processing Service*

Description

A class for a Web Processing Service. Can be used to overlay points or fetch grid values for rasters located remotely on a server and specified via the inRastername slot.

Slots

server: object of class "list"; contains the location of the CGI script that executes WPS ("URI"); service name ("service.name"), version ("version"), request type ("request"), identifier ("identifier")

inRastername: object of class "character"; name of the objects on the server

Methods

show signature(object = "WPS"): gets the complete server capabilities

getProcess signature(x = "WPS"): gets a list of processes available from a server

describe signature(x = "WPS"): lists parameters specific to some service identifier

over signature(x = "WPS", y = "SpatialPoints"): overlays spatial points and the target grids defined via the WPS-class (point-by-point)

subset signature(x = "WPS"): subsets a grid (from server) and loads it to R; use bbox argument to specify the bounding box

Note

More examples of overlay, subset and aggregation functions are available via WorldGrids.org. WPS WorldGrids.org uses the PyWPS module on a Debian system with Webserver, GDAL, Python and Scipy. The standard format for the gridded data on the WorldGrids.org repository is "GeoTiff". Use of the "bbox" object to obtain grids that cover more than 30 percent of the global coverage is not recommended. Consider instead downloading the compressed images directly from WorldGrids.org.

Author(s)

Tomislav Hengl & Hannes I. Reuter

References

- PyWPS module (http://pywps.wald.intevation.org)
- WorldGrids.org (http://worldgrids.org)

See Also

landmask

Examples

```
## Not run:
library(XML)
library(sp)
URI = "http://wps.worldgrids.org/pywps.cgi"
server <- list(URI=URI, request="execute",
    version="version=1.0.0", service.name="service=wps",
    identifier="identifier=sampler_local1pt_nogml")
glcesa3.wps <- new("WPS", server=server, inRastername="glcesa3a")
# show(biocl15.wps)
prl <- getProcess(glcesa3.wps)
prl[7]
describe(glcesa3.wps, identifier="overlay")
p1 <- data.frame(lon=15, lat=15)
coordinates(p1) <- ~lon+lat
proj4string(p1) <- CRS("+proj=longlat +datum=WGS84")
p1
over(glcesa3.wps, p1)
# fetch grids and load the to R:
glcesa3 <- subset(glcesa3.wps, bbox=matrix(c(20,40,22,42), nrow=2))
image(glcesa3)

## End(Not run)
```

Index