Package ‘sptemExp’

August 27, 2017

Type Package

Title Ensemble Spatiotemporal Mixed Models for Exposure Estimation

Version 0.1.0

Author Lianfa Li

Maintainer Lianfa Li `<lspatial@gmail.com>`

Description The approach of ensemble spatiotemporal mixed models is to make reliable estimation of air pollutant concentrations at high resolutions.
This package is an ensemble spatiotemporal modeling tool with constrained optimization and also provides the functionality of grid mapping of air pollutants.
Specifically, it includes the following functionality:
(1) Extraction of covariates from the satellite images such as GeoTiff and NC4 raster (e.g NDVI, AOD, and meteorological parameters);
(2) Generation of temporal basis functions to simulate the seasonal trends in the study regions;
(3) Generation of the regional monthly or yearly means of air pollutant concentration;
(4) Generation of Thiessen polygons and spatial effect modeling;
(5) Ensemble modeling for spatiotemporal mixed models, supporting multicore parallel computing;
(6) Integrated predictions with or without weights of the model's performance, supporting multicore parallel computing;
(7) Constrained optimization to interpolate the missing values;
(8) Generation of the grid surfaces of air pollutant concentrations at high resolution;
(9) Block kriging for regional mean estimation at multiple scales.

Depends R (>= 2.14)

Imports Rcpp (>= 0.12.12), methods, rgdal, raster,deldir, SpatioTemporal,plyr,sp,limSolve, R2BayesX,BayesX,BayesXsrc,ncdf4, bcv, rgeos, splines, parallel, foreach,doParallel,automap

LinkingTo Rcpp, RcppEigen

SystemRequirements C++11

License GPL

Encoding UTF-8

LazyData true

NeedsCompilation yes

RoxygenNote 6.0.1

R topics documented:

`abatchModel` *A Batch Modeing Training Inner Functions*

Description

This function is for a batch training models. The users can call parSpModel rather than this for training of multiple models.

Usage

```
abatchModel(td,bnd,fS,iF,iT,tidF,tids,mPath,idF="siteid",dateF="date",obsF="obs"
```

Arguments

`td`	Training dataset
`bnd`	Map object used in spatial effect model. For specific format, refer to BayesX
`fS`	Formular string
`iF`	Staring time id
`iT`	Ending time id
`tidF`	Time field name
`tids`	Time vector
`mPath`	The path for the models trained to be saved
`idF`	location id name
`dateF`	Date or time field name
`obsF`	observed value field name
`nM`	number of models to be trained

Details

This is an inner function to be called by parSpModel.

Value

The trained models will be saved on the appointed path. No direct output for this function.

Examples

```
#An example of PM2.5 data from Shandong


dPath=tempdir()
modelPath=paste(dPath,"/models",sep="")
unlink(modelPath,recursive = TRUE)
dir.create(modelPath)

data("trainsample","bnd")
aform=paste0('logpm25 ~sx(rid,bs ="mrf",map =bnd)+sx(monthAv,bs="rw2")')
aform=paste0(aform,'+sx(ndvi,bs="rw2")+sx(aod,bs="rw2")+sx(wnd_avg,bs="rw2")')

formulaStrs=c(aform)

trainsample$tid=as.numeric(strftime(trainsample$date,format= "%j"))
trainsample$logpm25=log(trainsample$pm25)
tids=c(91)
abatchModel(trainsample,bnd,formulaStrs,1,1,"tid",tids,modelPath,"siteid","date","pm25",3
```

`allPre500` *Dataset of the prediction result for some days for 2014 Shandong, interpolated by constrained optimization.*

Description

The dataset of the prediction result for some days for 2014 Shandong, interpolated by constrained optimization.

Usage

```
allPre500
```

Format

DataFrame

dk predicted (by ensemble mixed model) or interpolated (by constrained optimization) estimate for rowname id and the kth year of day

row name location id, corresponding to the raster id #'

Source

Collected

Examples

```
 allPre500
```

`bkriging` *Regional Mean Estimation by Block Kriging*

Description

Block kriging can use the measured or prediceted values to estimate the regional mean with minimum variance.

Usage

```
bkriging(samples, rtargets,tarStr,paras,model)
```

Arguments

`samples`	the sample data used to estimate the regional mean, must include the x and y coordinates. Format:`DataFrame`.
`rtargets`	the points within the target region used to represent the region to be predicted for the regional means. The points determines the density, shape and size of the region. Format:`dataframe`
`tarStr`	The target variable name (field name)
`paras`	variogram parameters: format: `vector`, (range,sill,nugget)
`model`	variogram model: default: "exponential"

Value

vector format: (kriged mean, kriged standard deviation, regular average, regular standard deviation)

Author(s)

Lianfa Li lspatial@gmail.com

Examples

```
#Test for simulated data

dataDt=data.frame(x=sample(c(1:3000),500),y=sample(c(1:2500),500))
dataDt$z=(2*dataDt$x+5*dataDt$y)%%10+rnorm(500)
dataDtSp=dataDt
sp::coordinates(dataDtSp) <- ~x+y
cl=colorGrinf(dataDt$z)
raster::plot(dataDtSp,col=cl$cols[cl$index])
tarDt=data.frame()
for(i in c(1:20)){
  for(j in c(1:20)){
    index=(i-1)*20+j
    tarDt[index,"x"]=i*10
    tarDt[index,"y"]=j*10
  }
}

varg=automap::autofitVariogram(z~1,input_data =dataDtSp,model="Exp")
paras=c(varg$var_model[2,3],varg$var_model[2,2],varg$var_model[1,2])
krigeMean=bkriging(dataDt, tarDt,"z",paras,model="Exp")
krigeMean


#Test using PM2.5 data of the 2014 PM2.5 of Shandong province

data("spointspre")
spointspresub=spointspre[!is.na(spointspre$pre_m),]
spointspresub$log_pre=log(spointspresub$pre_m)
sz=as.integer(nrow(spointspresub)/1)
index=sample(c(1:sz),size=as.integer(sz/2))
samples=spointspresub[index,]
rtargets=(spointspresub[c(1:sz),])[-index,]
paras=c(50000,0.0278,0.2)
samples@data$x=sp::coordinates(samples)[1]
samples@data$y=sp::coordinates(samples)[2]
rtargets@data$x=sp::coordinates(rtargets)[1]
rtargets@data$y=sp::coordinates(rtargets)[2]
sampledata=samples@data
rtargetsdata=rtargets@data
krigeMean=bkriging(sampledata, rtargetsdata,"log_pre",paras,model="Exp")
exp(krigeMean)
```

`bnd` *BND spatial topology data for use in spatial effect modeling.*

Description

BND spatial topology data for use in spatial effect modeling.

Usage

```
bnd
```

Format

BND format for use in BayesX package
#'List data object

Source

Collected

Examples

```
bnd
```

`colorCusGrinf` *Customed Color Generation by the Number of the Levels*

Description

A function for generation of colors by the numebr of levels for use in the map making.

Usage

```
colorCusGrinf(brkpts, cols)
```

Arguments

`brkpts`	a vector to contain the breakpoints
`cols`	Selection of colors for different timelines.

Value

A colors of gradient levels

Examples

```
data("spointspre","countylayer")
praster=sptemExp::points2Raster(spointspre,"d91")
dtStr=as.character(as.Date(91,origin=as.Date("2014-01-1")))
title=expression("PM"[2.5]*" Concentration Estimated")
par(mar=c(4,4,1,1))
breakpoints = c(0,50,100,200,350,600)
colors=colorCusGrinf(breakpoints,c("darkgreen","yellow","darkred"))
raster::plot(praster,breaks=breakpoints,col=colors,
    main=title,xlab=paste("Shandong Province, China (",dtStr,")",sep=""))
```

`colorGrinf` *Generation of Customed Gradient Colors*

Description

This function is to generate the color gradient with the customed levels

Usage

```
colorGrinf(x, levels=NA, colors=c("green","yellow","red"), colsteps=10)
```

Arguments

`x`	A vector value
`levels`	levels of gradient colors
`colors`	Color ranges
`colsteps`	Levels of color gradient.

Value

`levels`	Level of values for legend use
`cols`	Color ranges for legeng use
`index`	Color values for map.

Examples

```
#Example

x=sample(c(1:1000),size=100)
x=x[order(x)]
ret=colorGrinf(x)

# A block kriging example :

data("spointspre","countylayer")
tarF="d91" # target variable to be kriged
regionName="NAME_3"
```

```
bkRes=sptemExp::getTidBKMean(spointspre,countylayer,regionName,tarF,2)

bkRes=bkRes[!is.na(bkRes$bkm_fill),]
levels=c(30,60,100,150,250)
cr=sptemExp::colorGrinf(bkRes$bkm_fill,levels,colors=c("darkgreen","yellow","darkred"))
par(mar=c(1,1,1,1))
title=expression("Regional Block Kriged PM"[2.5]*" Concentration Estimated")
raster::plot(bkRes,col =cr$cols[cr$index],main=title)
legend("bottomright", fill =cr$cols, legend = cr$levels,col =cr$cols, cex=1,bty="n",bg="r
```

---

conOpt *Function of Constrained Optimization*

Description

Constrained optimization to construct the long-term series of air pollutants .

Usage

```
conOpt(ptrends,tSet,preF="con",paras=c(2.5,-5.5,-0.6,-0.1,-0.25,0.25),maxC)
```

Arguments

ptrends	seasonal trends such as temporal basis functions.
tSet	Train dataset (observed or estimated values) to get the solution.
preF	Predicted field name.
paras	A vector, constraints for the coefficients of temporal basis functions, respectively correponding to b0, b1 and b2. Different pollutants have different constraint parameters.
maxC	Maximum values for conentration of air pollutants.

Value

a vector of the coefficients for temporal basis functions.

Author(s)

Lianfa Li <lspatial@gmail.com>

References

Lianfa Li et al, 2017, Constrained Mixed-Effect Models with Ensemble Learning for Prediction of Nitrogen Oxides Concentrations at High Spatiotemporal Resolution, ES & T, DOI: 10.1021/acs.est.7b01864

Examples

```
#PM2.5 exmaple:

data("allPre500","shdSeries2014")
#Get the temporal basis functions
season_trends=getTBasisFun(shdSeries2014,idStr="siteid",dateStr="date",
                           valStr="obs",df=10,n.basis=2,tbPath=NA)

asiteMe=allPre500[1,]
ndays=ncol(allPre500)
trainSet=NA
days=as.integer(gsub("d","",colnames(allPre500)))
for(k in c(1:ndays)){
  aday=paste("d",days[k],sep="")
  if(!is.na(asiteMe[,aday])){
    atrainPnt=data.frame(b0=1,b1=season_trends$pv1[days[k]],
          b2=season_trends$pv2[days[k]],con=log(asiteMe[,aday]))
    if(inherits(trainSet,"logical")){
      trainSet=atrainPnt
    }else{
      trainSet=rbind(trainSet,atrainPnt)
    }
  }
}
#Set the PM2.5 constriants:
paras=c(2.5,-5.5,-0.6,-0.1,-0.25,0.25)
maxCon=750
res=conOpt(season_trends,trainSet,preF="con",paras,maxCon)
```

`countylayer` *County layer map for illustration of block kriging.*

Description

County layer map for illustration of block kriging.

Usage

```
countylayer
```

Format

SpatialPolygonDataFrame

ID_0 Privence name

NAME_3 county name

... ... #'

Source

Collected

Examples

```
countylayer
```

`extractVNC4` *Extract Values for Point from NC4 Image*

Description

A program to extarct the values from the NC4 image by overlaying the subject locations.

Usage

```
extractVNC4(tarshp, ncin, bandVar, prj)
```

Arguments

`tarshp`	The point objects, format: `SpatialPointDataFrame`
`ncin`	the nc4 object by nc_open.
`bandVar`	The band name to be used for extration from the NC4 file
`prj`	The project information, default: NA

Details

This function can be used to extract values from the NC4 images (such as satellite images)

Value

The values extracted in the same sequence with the point object. Format: `vector`

Author(s)

Lianfa Li, `<lspatial@gmail.com>`

References

http://disc.sci.gsfc.nasa.gov/daac-bin/FTPSubset2.pl)

Examples

```
data("samplepnt")
nc4File=file.path(system.file(package = "sptemExp"), "extdata", "ancdata.nc4")
ncin0=ncdf4::nc_open(nc4File)
extRes=extractVNC4(samplepnt,ncin0,"TLML")
extRes
```

`extractVTIF` *Extract GeoTiff Data*

Description

Extract geotiff image.

Usage

```
extractVTIF(tarshp, tifRaster)
```

Arguments

`tarshp`	spatial point object, data format: `PointDataFrame`
`tifRaster`	tifRaster object

Value

values for extracted. Format: `vector`

`fillNASVD` *Function to Use SVD to Impute the Missing Values for Training Dataset*

Description

Singular value decomposition (SVG) is used to impute the missing values for the training dataset. For each monitoring location, the time series of multivariate data is leveraged to impute the missing values using SVD.

Usage

```
fillNASVD(dset, cols, idF, dateF)
```

Arguments

`dset`	The dataframe having many missing values. Data format: `dataframe`
`cols`	A character vector to contain the column names (including the columns with missing values) used to impute the missing valeus
`idF`	Unique location identification
`dateF`	Date column name if any

Value

A dataframe base on the input dset, but with filled values.

Examples

```
# Use the covariates for PM2.5 data as a example:

data("trainsample")
cols=c("ndvi","aod","wnd_avg","monthAv")
n=nrow(trainsample)
p=0.05
pn=as.integer(p*n)
trainsample2missed=trainsample
for(col in cols){
  index=sample(n,pn)
  trainsample2missed[index,col]=NA
}
trainsample2filled=fillNASVD(trainsample2missed,cols,"siteid","date")

#Examine the accuracy:
for(col in cols){
  index=which(is.na(trainsample2missed[,col]))
  obs=trainsample[index,col]
  missed=trainsample2missed[index,]
  sindex=match(interaction(missed$siteid,missed$date),
               interaction(trainsample2filled$siteid,trainsample2filled$date))
  pre=trainsample2filled[sindex,col]
  print(paste(col," missing value correlation: ",round(cor(obs,pre),2)))
  print(paste(col," missing value cv rmse: ",round(rmse(obs,pre),2)))
}
```

`fillNASVDSer` *SVD to Interpolate the Missing Values in the Time Series Data*

Description

Function to Use SVD to Interpolate the Missing Values in the Time Series Data

Usage

```
fillNASVDSer(dset, idF, dateF, valF, k)
```

Arguments

`dset`	The data frame for time series. Data format: siteid, date, obs `dataframe`.
`idF`	The unique location id like siteid.
`dateF`	The time column name.
`valF`	The target variable column name.
`k`	the priciple component, default 1

Details

This function can be used to fill the missing values in time series for many locations.

Value

The data frame similar to the input dset's structure but with filled values.

Examples

```
#Using the 2014 PM2.5 time series as an example
data("shdSeries2014")
n=nrow(shdSeries2014)
p=0.1 # Set the proportion of missing values
np=as.integer(n*p)
index=sample(n,np)
shdSeries2014missed=shdSeries2014
shdSeries2014missed[index,"obs"]=NA
shdSeries2014filled=fillNASVDSer(shdSeries2014missed,"siteid","date","obs",k=1)

#Exmine the accuracy:
cor(shdSeries2014filled[index,"obs"],shdSeries2014[index,"obs"])
rmse(shdSeries2014filled[index,"obs"],shdSeries2014[index,"obs"])
```

`genRaster` *Generation of Raster Covering the Side Map*

Description

Generaye the raster to cover the study region with the preset resolution.

Usage

```
genRaster(sideSdf, dx, dy, idStr)
```

Arguments

`sideSdf`	The SpatialPolygonDataFrame obejct used to constrain the grid border.
`dx`	x resolution
`dy`	y resolution
`idStr`	id name

Value

`PntObj`	The SpatialPointDataFrame extracted from the generated raster.
`Rst`	The raster object covering the study region.

Examples

```
## Use the Shandong province as an example:
data("prnside")
ret=genRaster(prnside,dx=2000,dy=2000,idStr="gid")
raster::plot(ret$Rst)
raster::plot(ret$PntObj)
```

`GetARegionBK` *Get a Regional Kriging*

Description

Estimate a regional mean for the regions.

Usage

```
GetARegionBK(rNames,rF,rT,rlayer,paras,spnts,regF,obsF="pre_mf_log")
```

Arguments

`rNames`	region data
`rF`	id of the start region
`rT`	id of the end
`rlayer`	regional layer
`paras`	parameters of variogram
`spnts`	spatial points for preiction
`regF`	region field name
`obsF`	observed field name

Value

Regiona means by block kriging.

`getPolyMMean` *Generation of Regional Monthly Mean Based on the Input Polygons*

Description

Generate the regional monthly mean of air pollutant concentrations based on the input polygons

Usage

```
getPolyMMean(polys,samp,tse,idF="siteid",ridF="rid",obsF="obs",dateF="date")
```

Arguments

`polys`	The input region polygon map object (SpatialPolygonsDataFrame) to be used the regions for generation of the regiona monthly means
`samp`	The sample spatial location map. Data format: `SpatialPointDataFrame`
`tse`	Time series for the siteid and date used for generation of monthly mean.
`idF`	location id name
`ridF`	region id name
`obsF`	observed value field name.
`dateF`	Date name

Value

A data frame of data format: rid, year, month, mean

Author(s)

Lianfa Li `<lspatial@gmail.com>`

Examples

```
#Use the PM2.5 concentration as an example.

data("samplepnt","prnside","shdSeries2014")
tpolys=tpolygonsByBorder(samplepnt,prnside)$tpolys
regionmmean=getPolyMMean(tpolys, samplepnt, shdSeries2014,"siteid", "rid", "obs","date")
```

`getRidbytpoly` *getRidbytpoly for Assignment of Thiessen polygon id to point object*

Description

Assign the polygon id to the data points.

Usage

```
getRidbytpoly(tpolys,pntlayer,isnearest)
```

Arguments

`tpolys`	Thiessen polygons, data format: SpatialPolygonsDataFrame.
`pntlayer`	Points for assignment for polygons. SpatialPointsDataFrame.
`isnearest`	whether to use nearest method to assign polygon id for no overlay with polygons, default: TRUE

Value

polygon id

Author(s)

Lianfa Li `<lspatial@gmail.com>`

Examples

```
  data("samplepnt","prnside")
  # Point
  x=samplepnt
  # Border
  sidepoly=prnside
  # Get the Thiessen polygons
```

```
res=tpolygonsByBorder(x,sidepoly)
# Assign the regional id
rids=getRidbytpoly(res$tpolys,x)
```

---

getTBasisFun *Generation of Temporal Basis Function*

Description

Generation of temporal basis function

Usage

```
getTBasisFun(serDf, idStr, dateStr, valStr, df = 25, n.basis = 2, tbPath = NA)
```

Arguments

`serDf`	Time series dataframe, format: (siteid,date,observed value)
`idStr`	Location id name
`dateStr`	Date id name
`valStr`	The target variable's name
`df`	Degree of freedom
`n.basis`	Number of temporal basis function
`tbPath`	The path to save the plots of each temporal basis component. Default: NA, no plots generated

Value

A dataframe of temporal basis function: (date, pvi (the ith temporal basis function output for a date))

References

Finkenstadt, B., Held, L., Isham, V., 2007. Statistical Methods for Spatio-Temporal Systems. Chapman & Hall/CRC, New York.

Examples

```
#Use PM2.5 as example:

data("shdSeries2014")
result=getTBasisFun(shdSeries2014,"siteid","date","obs",df=10,n.basis=2)
```

`getTidBKMean` *Batch Block Kriging for Estimate of Regional Means*

Description

A batch program to implement block kriging for estimate of regional mean for air pollutant. Support the multi-core parallel computation.

Usage

```
getTidBKMean(spt,rlayer,regF="NAME_3",tarF="pre_mf",n=1)
```

Arguments

`spt`	Spatial point layer (shape file) corresponding to the grid `spointspre`
`rlayer`	Regional layer to crop the points for estimate of regional means `regionlayer`
`regF`	Regiona field name `regionName`
`tarF`	the target variable to be estimated `tarVar`
`n`	Core number of CPU for parallel support `ncore`

Value

The spatial polygon dataframe including the field of kriged means.

Author(s)

Lianfa Li <`lspatial@gmail.com`>

Examples

```
# PM2.5 example

data("spointspre","countylayer")
regionName="NAME_3"
tarF="d91" # field target name to be estimated (2014-04-01 for 91 day of 2014)
bkRes=getTidBKMean(spointspre,countylayer,regionName,tarF="d91",n=2)
```

`gtifRst` *The 2014 time series of PM2.5 concentrations of Shandong province, with many missing values.*

Description

The 2014 time series of PM2.5 concentrations of Shandong province, with many missing values.

Usage

```
gtifRst
```

Format

Raster
GeoTiff format image of AOD to demonstrate use of extractVTIF

Source

NASA MAIMIC data

Examples

```
gtifRst
```

`inter2conOpt` *Batch Interpolation of the Missing Values for Time Series Using Constrained Optimization.*

Description

This function provides batch implementation for interpolation of the missing values for multiple locations for a raster, supporting multi-core parallel computing.

Usage

```
inter2conOpt(tarPDf, pol_season_trends, ncore)
```

Arguments

`tarPDf`	The target data frame with missing values. Each row corresponds to a location (rowname as location id) and each column corresonds to a time point. The sequence of the location and time should be in sequence in spatial and temporal dimension. This dataset comes from the raster dataset and the sequence is kept for convenience of making raster with the interpolated value.
`pol_season_trends`	The temporal basis function using `getTBasisFun`
`ncore`	number of cores for parallel computing.

Details

This function aims to implement the batch computing to use constrained optimization to get the concentrations for the missing values of a time series, such as PM2.5 concentration.

Value

A data frame similar to the input data frame, tarPDf but with the missing values interpolated by constrained optimizaiotn.

Author(s)

Lianfa Li `<lspatial@gmail.com>`

Examples

```
# Here is the sample for the first 500 locations.
# In practice, you may need more point locations and more cores.
data("allPre500","shdSeries2014")
# Get the temporal basis functions to be used in constrained optimization
season_trends=getTBasisFun(shdSeries2014,idStr="siteid",dateStr="date",
                           valStr="obs",df=10,n.basis=2,tbPath=NA)

#Constrained optimization
season_trends$tid=as.numeric(strftime(season_trends$date, format = "%j"))
allPre_part_filled=inter2conOpt(tarPDf=allPre500[c(1:6),],pol_season_trends=season_trends
```

---

noweiAvg *Averages over the Ensemble Predictions of Mixed Models (No weighted)*

Description

Average and standard deviation of multiple models by ensemble learning.

Usage

```
noweiAvg(path, preStr = "preno2", idStr = "id", dateStr = "s_date")
```

Arguments

path	Path for the prediction files from multiple models with the unified format and field names. File format: CSV with head:(gid,rid,pre)
preStr	prediction field names
idStr	unique identifier string
dateStr	date string. You can set it as the same as idStr

Value

id and corresponding mean and standard deviation. Format: `dataframe`

Author(s)

Lianfa Li: <lspatial@gmail.com>

Examples

```
# Generate the prediction dataset, but you can use parATimePredict function
# to make the prediction in application

dPath=tempdir()
pPath=paste(dPath,"/preds",sep="")
unlink(pPath, recursive=TRUE, force=TRUE)
dir.create(pPath)
```

```
nr=2000
for(i in c(1:80)){ # i =1
  dset=data.frame(gid=c(1:nr),rid=sample(c(1:30),size=nr,replace=TRUE))
  dset$pre=dset$gid%%80+rnorm(nr,mean=5,sd=9)+runif(nr,0,1)
  afile=paste(pPath,"/m_",i,".csv",sep="")
  write.csv(dset,file=afile,row.names = FALSE)
}
result=noweiAvg(pPath, preStr="pre",idStr="gid",dateStr="gid")
```

---

parATimePredict *Batch Prediction for Time Series Using the Ensemble Models*

Description

Batch predictions for the time series using ensemble models generated by the function, parSpModel

Usage

```
parATimePredict(mdPath,newPnts,cols=NA,bnd,c=1,outPath="/tmp",idF="siteid",ridF=
```

Arguments

`mdPath`	The path where multiple ensemble models are saved by parSpModel
`newPnts`	New data locations corresponding to the predcitions.
`cols`	Columns where there are NAs. NAs must be removed before prediction. Default: NA
`bnd`	The same BND object as that used in parSpModel, for spatial effect models.
`c`	CPU cores to support parallel computing.
`outPath`	The output file path, file named after the model id.
`idF`	Unique identifier
`ridF`	Region id used in spatial effect modeling

Details

This function aims to use the muiltiple models with their performance metrics to make the predictions for the new dataset with their spatial location.

Value

The prediction result will be saved in the assigned path

Author(s)

Lianfa Li <lspatial@gmail.com>

References

Breiman, L., 1996. Bagging Predictors. Machine Learning 24, 123-140. Lianfa Li et al, 2017, Constrained Mixed-Effect Models with Ensemble Learning for Prediction of Nitrogen Oxides Concentrations at High Spatiotemporal Resolution, ES & T, DOI: 10.1021/acs.est.7b01864

Examples

```
#Use the PM2.5 examples

dPath=tempdir()
modelPath=paste(dPath,"/models",sep="")
unlink(modelPath,recursive = TRUE)
dir.create(modelPath)

prePath=paste(dPath,"/preds",sep="")
unlink(prePath,recursive = TRUE)
dir.create(prePath)

data("trainsample","bnd")
aform=paste0('logpm25 ~sx(rid,bs ="mrf",map =bnd)+sx(monthAv,bs="rw2")')
aform=paste0(aform,'+sx(ndvi,bs="rw2")+sx(aod,bs="rw2")+sx(wnd_avg,bs="rw2")')

formulaStrs=c(aform)

trainsample$tid=as.numeric(strftime(trainsample$date, format = "%j"))

trainsample$logpm25=log(trainsample$pm25)
tids=c(91)

parSpModel(trainsample,bnd,formulaStrs,tidF="tid",tids,c=2,
       nM=3,modelPath,idF="siteid",dateF="date",obsF="pm25")

amodelPath=paste(dPath,"/models/t_",tids[1],"_models",sep="")
data("shd140401pcovs","bnd")
shd140401pcovs_part=shd140401pcovs[c(1:1000),]

cols=c("aod","ndvi","wnd_avg","monthAv")
parATimePredict(amodelPath,newPnts=shd140401pcovs_part,
       cols,bnd=bnd,c=2,prePath,idF="gid",ridF="rid")
```

`parSpModel` *Generation of Spatiotemporal Models by Bootstrap Aggregating*

Description

Generation of multiple models using bootstrap aggregating, supporting multi-cores based parallel computing.

Usage

```
parSpModel(tSet,bnd,fS,tidF="tid",tids, c=1,
     nM=30,mPath,idF="siteid",dateF="date",obsF="pm25")
```

Arguments

tSet	Dataframe of the training dataset, including the measurements of the target variable and covariates.
bnd	BND object used in saptial effect modeling (BayesX)
fS	Formula string, like that used in BayesX
tidF	time id (ensemble models for each time point)
tids	all the time ids for which multiple models will be trained.
c	CPU core number
nM	Number of ensemble models for each time point.
mPath	Path where the models will be saved.
idF	Unique location name
dateF	Time id
obsF	Target variable name

Details

Batch training of the models using the multi-cores based parallel computing

Value

The model will be saved into the assigned path.

Author(s)

Lianfa Li <lspatial@gmail.com>

References

Breiman, L., 1996. Bagging Predictors. Machine Learning 24, 123-140. Lianfa Li et al, 2017, Constrained Mixed-Effect Models with Ensemble Learning for Prediction of Nitrogen Oxides Concentrations at High Spatiotemporal Resolution, ES & T, DOI: 10.1021/acs.est.7b01864

Examples

```
# Example the PM2.5 data for Shandong

dPath=tempdir()
mPath=paste(dPath,"/models",sep="")
unlink(mPath,recursive = TRUE)
dir.create(mPath)

data("trainsample","bnd")

aform=paste0('logpm25 ~sx(rid,bs ="mrf",map =bnd)+sx(monthAv,bs="rw2")')
aform=paste0(aform,'+sx(ndvi,bs="rw2")+sx(aod,bs="rw2")+sx(wnd_avg,bs="rw2")')

formulaStrs=c(aform)

trainsample$tid=as.numeric(strftime(trainsample$date, format = "%j"))
trainsample$logpm25=log(trainsample$pm25)
tids=c(91)
```

```
parSpModel(trainsample,bnd,formulaStrs,tidF="tid",
   tids,c=2,nM=3,mPath,idF="siteid",dateF="date",obsF="pm25")
```

`parTemporalBImp` *Function to Fill Missing Values by Constraint Optimization*

Description

A function to use constraint pptimization to predict the missing values.

Usage

```
parTemporalBImp(allPre_, siteids_, isite_, pol_season_trends_)
```

Arguments

`allPre_`	Prediction dataset including many mssing values
`siteids_`	Ponitoring station id
`isite_`	Target site field name
`pol_season_trends_`	Temporal basis function.

Value

Parameters for temporal basis functions.

`perMdPrediction` *Batch Prediction Using the Trained Models*

Description

Batch Prediction Using the Trained Models

Usage

```
    perMdPrediction(mPath,mFiles,mids,mF,mT,bnd,dset,outPath,idF,ridF)
```

Arguments

`mPath`	Model path
`mFiles`	Model file path
`mids`	Set of mids
`mF`	From field name
`mT`	To field name
`bnd`	BND object used in spatial effect modeling
`dset`	newDataset to be predicted
`outPath`	Ourput path to save the predictions.
`idF`	id field name
`ridF`	regional id field name

Value

No straightforward output. All output saved in the appointed path.

`points2Raster` *Generation of Grid Surface Using the predicted/Interpolated Values*

Description

This combines the predicted values's output with the corresponding spatial point data frame to generate the grid surface. Please use this function with the output spatial point data frame generated by genRaster

Usage

```
points2Raster(spoints, tarVar, dx = 2000, dy = 2000)
```

Arguments

`spoints`	Spatial point data frame. This data frame is based on the output by the function, genRaster with its predicted or interpolated value field.
`tarVar`	Field name such as pollutant concentration used to make the grid.
`dx`	Size of resolution along x coordinate
`dy`	Size of resolution along y coordinate

Value

Convert the points into Raster

Examples

```
data("spointspre")
praster=points2Raster(spointspre,"pre_m",dx=2000,dy=2000)
raster::plot(praster)
```

`pol_season_trends` *pol_season_trends* .

Description

pol_season_trends. Temoral basis function output of log PM2.5 for Shan dong 2014 data

Usage

```
pol_season_trends
```

Format

DataFrame

date date

pv1 1st temporal basis function

pv2 2st temporal basis function

tid unique temporal identifier #'

Source

simulated

Examples

```
pol_season_trends
```

`prnside` *Side to limit the Thiessen's polygons.*

Description

Side shape file

Usage

```
prnside
```

Format

SpatialPolygonsDataFrame

AREA AREA

PERIMETER PERIMETER

BOU2_4M_ BOU2_4M_

BOU2_4M_ID BOU2_4M_ID

ADCODE93 ADCODE93

ADCODE99 ADCODE99

Source

Collected

Examples

```
prnside
```

---

rmse *RMSE function*

Description

A function to calculate rmse.

Usage

```
rmse(obs, pre)
```

Arguments

`obs`	Observed values
`pre`	Predicted values

Value

A scalar value, RMSE

Examples

```
obs=runif(400,1,100)
pre=obs+rnorm(400,5,10)
rmse(obs,pre)
```

---

rSquared *Coefficient of Determination*

Description

A function to calculate the rSquared.

Usage

```
rSquared(obs, res)
```

Arguments

`obs`	A vector of the observed values.
`res`	A vector of residuals

Value

rsquared value

Examples

```
obs=runif(400,1,100)
pre=obs+rnorm(400,5,10)
res=obs-pre
rSquared(obs,res)
```

`samplepnt` *Sample data for generation of Thiessen polygons.*

Description

Sample data for generation of Thiessen polygons.

Usage

```
samplepnt
```

Format

SpatialPointDataFrame

provence Privence name

city city name

geocode geocode

name name

code code

x x

y y

rid region id #'

Source

Collected

Examples

```
samplepnt
```

`shd140401pcovs` *The dataset of 04/01/2014 prediction dataset for the raster spoint_pre covering the Shandong with 2km x 2km grid .*

Description

The dataset of 04/01/2014 prediction dataset for the raster spoint_pre covering the Shandong with 2km x 2km grid .

Usage

```
shd140401pcovs
```

Format

SpatialPointDataFrame

layer

gid unique location identifier

rid region id

ndvi ndvi

aod aod

wnd_avg wind speed

monthAv regional monthy average

Source

Collected

Examples

```
shd140401pcovs
```

`shdSeries2014` *The 2014 time series of PM2.5 concentrations of Shandong province, with missing approach.*

Description

The 2014 time series of PM2.5 concentrations of Shandong province, with missing approach.

Usage

```
shdSeries2014
```

Format

DataFrame

siteid site id for monitoring station

date monitoring date

obs average over the hourly observed values or imputed value by SVD #'

Source

Collected

Examples

```
shdSeries2014
```

`spointspre` *SpatialPointDataFrame as container of raster to geo-link with the specific date prediction of PM2.5.*

Description

Container of raster to geo-link with the specific date prediction of PM2.5, and will be used to generate the surface of PM2.5 concentration at high resolution for Shandong Province.

Usage

```
spointspre
```

Format

`SpatialPointsDataFrame`

ogc_fid inner id

layer layers value

pre_m predicted value

pre_sd estimate of standard variance of the predicted value

Source

Collected

Examples

```
spointspre
```

`tpolygonsByBorder` *tpolygonsByBorder for Generation of Thiessen polygons*

Description

Generate Thiessen polygons according to the point spatialframes and border.

Usage

```
tpolygonsByBorder(x,sidepoly)
```

Arguments

`x`	SpatialPointsDataFrame "`SpatialPointsDataFrame`".
`sidepoly`	SpatialPolygonsDataFrame, e.g. `SpatialPolygonsDataFrame`.

Value

A list object:

`tpolys`	Thiessen polygons, data format: SpatialPolygonsDataFrame
`bnd`	BND object used in the model in the BayesX.

Author(s)

Lianfa Li `<lspatial@gmail.com>`

Examples

```
data("samplepnt","prnside")
x=samplepnt
sidepoly=prnside
tpoly=tpolygonsByBorder(x,sidepoly)$tpolys
raster::plot(tpoly)
```

`trainsample` *The dataset of 2014 training sample for the Shandong with missing values imputed using SVD.*

Description

The dataset of 2014 training sample for the Shandong with missing values imputed using SVD.

Usage

```
trainsample
```

Format

`DataFrame`

provence provence

city city

geocode geocode

name name

code code

x x

y y

siteid siteid

rid rid

ndvi ndvi

aod aod

wnd_avg wind speed

monthAv regional monthy average

date date

month month

pm25 pm2.5

logpm25 log PM2.5

tid tid

Source

Collected

Examples

```
trainsample
```

`voronoipolygons2` *Generation of Thiesseon Polygons By Points*

Description

Generation of Thiessen polygons by spatial points

Usage

```
voronoipolygons2(x, poly)
```

Arguments

`x`	Spatial point object, data format:`SpatialPointsDataFrame`
`poly`	The border polygons object to limit the Thiessen polygons. Data format:`SpatialPolygonsDataI`

Value

The spatial polygongs objects. Data format: `SpatialPolygonsDataFrame`

Examples

```
data("samplepnt","prnside")
x=samplepnt
sidepoly=prnside
prjinf=sp::proj4string(x)
sidepoly_p=sp::spTransform(sidepoly,prjinf)
extBnd=as(raster::extent(sidepoly_p), 'SpatialPolygons')
sp::proj4string(extBnd)=prjinf
pzn.coords=voronoipolygons2(x,extBnd)
sp::proj4string(pzn.coords)=prjinf
```

`weiA2Ens` *Ensemble Weighted Prediction of Mixed Models*

Description

Weighted average and standard deviation of multiple models. The weights can be the model's performance metrics such as R2 or RMSE.

Usage

```
weiA2Ens(pPath,mFile,metrF="rmse",preF="pre",idF="gid",dateF=NA)
```

Arguments

`pPath`	Path for the prediction files from multiple models with the unified format and field names. File format: CSV with head.
`mFile`	File path for the corresponding multiple models's performance. CSV format:mid, r2, rmse.
`metrF`	target metric such as rmse or r2 to weigh the model's output
`preF`	prediction field name
`idF`	unique identifier string
`dateF`	date string if any

Value

id and corresponding mean and standard deviation. Format: `dataframe`

Author(s)

Lianfa Li: `<lspatial@gmail.com>`

Examples

```
#First generate the prediction dataset and metrics.
# In application, you can use parSpModel to train models and
# get the models's performance metrics, and use the parATimePredict function to make the

# Simulared data

dPath=tempdir()
pPath=paste(dPath,"/preds",sep="")
unlink(pPath, recursive=TRUE, force=TRUE)
dir.create(pPath)

nr=2000;nmod=80
for(i in c(1:nmod)){ # i =1
  dset=data.frame(gid=c(1:nr),rid=sample(c(1:30),size=nr,replace=TRUE),stringsAsFactors =
  dset$pre=dset$gid%%nmod+rnorm(nr,mean=5,sd=9)+runif(nr,0,1)
  dset$gid=paste("c",dset$gid,sep="")
  afile=paste(pPath,"/m",i,".csv",sep="")
  write.csv(dset,file=afile,row.names = FALSE)
}

modelsMetrics=data.frame(mid=c(1:nmod),r2=runif(nmod,0.6,0.9),rmse=runif(nmod,20,60))
mfile=paste(dPath,"/model_metrics.csv",sep="")
write.csv(modelsMetrics,file=mfile,row.names = FALSE)
result=weiA2Ens(pPath,mfile,metrF="rmse","pre","gid","gid")
```

`weightedstat` *Weighted Average for Multiple Models*

Description

Returns an R `dataframe` containing the character vector `c("foo", "bar") c(0, 1).`

Usage

```
weightedstat(path,modelpath,metric,preStr, idStr,dateStr)
```

Arguments

`path`	path of the prediction dataset files.
`modelpath`	model metric file
`metric`	performance weight
`preStr`	prediction value's name
`idStr`	id name
`dateStr`	date name

Author(s)

Lianfa Li `<lspatial@gmail.com>`

Examples

```
# Simulared data


dPath=tempdir()
pPath=paste(dPath,"/preds",sep="")
unlink(pPath, recursive=TRUE, force=TRUE)
dir.create(pPath)

nr=2000;nmod=80
for(i in c(1:nmod)){ # i =1
  dset=data.frame(gid=c(1:nr),rid=sample(c(1:30),size=nr,replace=TRUE),stringsAsFactors =
  dset$pre=dset$gid%%nmod+rnorm(nr,mean=5,sd=9)+runif(nr,0,1)
  dset$gid=paste("c",dset$gid,sep="")
  afile=paste(pPath,"/m",i,".csv",sep="")
  write.csv(dset,file=afile,row.names = FALSE)
}

modelsMetrics=data.frame(mid=c(1:nmod),r2=runif(nmod,0.6,0.9),rmse=runif(nmod,20,60))
modelsMetrics$rmse2=1/modelsMetrics$rmse
mfile=paste(dPath,"/model_metrics.csv",sep="")
write.csv(modelsMetrics,file=mfile,row.names = FALSE)
res=weightedstat(pPath,modelpath=mfile,metric="rmse2",preStr="pre",idStr="gid",dateStr="g
```

Index