Package ‘ParallelDSM’

November 16, 2022

Type Package

Title Parallel Digital Soil Mapping using Machine Learning

Version 0.3.7

Description Parallel computing, multi-core CPU is used to efficiently compute and process multi-dimensional soil data.This package includes the parallelized 'Quantile Regression Forests' algorithm for Digital Soil Mapping and is mainly dependent on the package 'quantregForest' and 'snowfall'. Detailed references to the R package and the web site are described in the methods, as detailed in the method documentation.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0), snowfall, raster, sp

Imports methods, pryr, utils, caret, gstat, quantregForest, randomForest, stringr, rgdal, stats,

RoxygenNote 7.1.1

NeedsCompilation no

Author Xiaodong Song [aut],
Peicong Tang [aut, cre],
Wentao Zhu [aut],
Gaoqiang Ge [aut],
Jun Zhu [aut],
Ganlin Zhang [aut]

Maintainer Peicong Tang <peicongtang0409@163.com>

Repository CRAN

Date/Publication 2022-11-16 08:20:02 UTC

R topics documented:

`CVfunction` *For the gap between the predicted value and expected value of the model, the model validates the function*

Description

For the gap between the predicted value and expected value of the model, the model validates the function

Usage

```
CVfunction(pred, actual)
```

Arguments

`pred`	: Value predicted by the model
`actual`	: The real value

Value

Vector-value after model accuracy verification

Examples

```
test.pred <- c(2,4,5,7,2,4)
test.obs <- c(1,2,3,4,5,6)
myres <- CVfunction(test.pred,test.obs)
print(myres)
```

DataProcess *Parallel computing initialization preparation(This function is not open to users)*

Description

Parallel computing initialization preparation(This function is not open to users)

Usage

```
DataProcess(mymodel)
```

Arguments

`mymodel` : The models were selected, including QRF,RF and MLR.

Value

Represents whether the loading of the required variables and dependent packages is complete

Examples

```
#This function only serves the ParallelComputing function.
DataProcess(mymodel = "QRF")
```

df.dem *Sampling test data of the dem*

Description

A dataset containing the df.dem and other attributes of almost 212000 df.dem The variables are as follows:

Usage

```
df.dem
```

Format

A data frame with 211415 rows and 3 variables:

dem data variable DEM

x The coordinate variable x

y The coordinate variable y

`df.input` *Sampling test data*

Description

A dataset containing the testdata and other attributes of almost 110 socd030 The variables are as follows:

Usage

```
df.input
```

Format

A data frame with 109 rows and 6 variables:

socd030 data variable socd030

dem data variable DEM

plancur data variable plancur

procur data variable procur

mrrtf data variable mrrtf

twi data variable twi

`df.mrrtf` *Sampling test data of the mrrtf*

Description

A dataset containing the df.mrrtf and other attributes of almost 212000 df.mrrtf The variables are as follows:

Usage

```
df.mrrtf
```

Format

A data frame with 211415 rows and 3 variables:

mrrtf data variable MRRTF

x The coordinate variable x

y The coordinate variable y

`df.plancur` *Sampling test data of the plancur*

Description

A dataset containing the df.plancur and other attributes of almost 212000 df.plancur The variables are as follows:

Usage

```
df.plancur
```

Format

A data frame with 211415 rows and 3 variables:

plancur data variable PLANCUR

x The coordinate variable x

y The coordinate variable y

`df.procur` *Sampling test data of the procur*

Description

A dataset containing the df.procur and other attributes of almost 212000 df.procur The variables are as follows:

Usage

```
df.procur
```

Format

A data frame with 211415 rows and 3 variables:

procur data variable PROCUR

x The coordinate variable x

y The coordinate variable y

df.twi *Sampling test data of the twi*

Description

A dataset containing the df.twi and other attributes of almost 212000 df.twi The variables are as follows:

Usage

```
df.twi
```

Format

A data frame with 211415 rows and 3 variables:

twi data variable TWI

x The coordinate variable x

y The coordinate variable y

GetPredictorSubset *calculation function for cutting spatial data (tool function,Not as an open function, only for function calls)*

Description

calculation function for cutting spatial data (tool function,Not as an open function, only for function calls)

Usage

```
GetPredictorSubset(
  predictor.name,
  iblock,
  nblock,
  fn,
  nr,
  nc,
  resolutions,
  pro,
  from,
  to
)
```

Arguments

`predictor.name` : the name of the predictor variable

`iblock` : sequence code of parallel computing

`nblock` : number of target blocks (integer)

`fn` : The passed value of a global variable

`nr` : The passed value of a global variable

`nc` : The passed value of a global variable

`resolutions` : The passed value of a global variable

`pro` : The passed value of a global variable

`from` : Which row to start cutting the matrix

`to` : Where does the last row of the cut matrix go

Value

Parallel calculation of the cut part of the data box data

References

Breiman, L. (2001). Random forests. Mach. Learn. 45, 5–32. Meinshausen, N. (2006) "Quantile Regression Forests", Journal of Machine Learning Research 7, 983-999 http://jmlr.csail.mit.edu/papers/v7/

Examples

```
GetPredictorSubset("dem",4,10,"covariate",486,777,NULL,NULL,1,10)
```

`InsepectionVariable` *A function that checks the parallel computation for missing data*

Description

A function that checks the parallel computation for missing data

Usage

```
InsepectionVariable(model = "MLR", block, outputDirectory)
```

Arguments

`model` : The models were selected, including QRF,RF and MLR.

`block` : The number of blocks for data cutting.

`outputDirectory` : The directory of output files.

Examples

```
InsepectionVariable(model = "MLR", block = 30, outputDirectory = "MlrOutput")
```

Insepect_MLR *A function that checks the parallel computation for missing data of MLR model.*

Description

A function that checks the parallel computation for missing data of MLR model.

Usage

```
Insepect_MLR(block, outputDirectory)
```

Arguments

block	: The number of blocks for data cutting.
outputDirectory	: The directory of output files.

Examples

```
Insepect_MLR(30, "./MlrOutput")
```

Insepect_QRF *A function that checks the parallel computation for missing data of QRF model.*

Description

A function that checks the parallel computation for missing data of QRF model.

Usage

```
Insepect_QRF(block, outputDirectory)
```

Arguments

`block`	: The number of blocks for data cutting.
`outputDirectory`	: The directory of output files.

Examples

```
Insepect_QRF(30, "./QrfOutput")
```

`Insepect_RF` *A function that checks the parallel computation for missing data of RF model.*

Description

A function that checks the parallel computation for missing data of RF model.

Usage

```
Insepect_RF(block, outputDirectory)
```

Arguments

`block`	: The number of blocks for data cutting.
`outputDirectory`	: The directory of output files.

Examples

```
Insepect_RF(30, "./RfOutput")
```

MergingTiles *A function that combines the results of parallel cutting into a single file*

Description

A function that combines the results of parallel cutting into a single file

Usage

```
MergingTiles(df_dem, f.i.d, f.iblock, n.block, f.o.d, f.suffix)
```

Arguments

`df_dem`	: The predicted source file before merging
`f.i.d`	: Enter the absolute path to the file
`f.iblock`	: The filename prefix of the resulting result
`n.block`	: The number of blocks cut is calculated in parallel
`f.o.d`	: The absolute output path of the file
`f.suffix`	: The suffix for the output of the file

Value

1

Examples

```
# you must have a file, which is name "myres"
# Merging files, for example:
# f.input.directory <- c("e:/test/")
# f.input.iblock <- c("sics030_")
# n.block <- 100
# f.output.directory <- c("e:/test/myoutput")
# f.output.suffix <- c("sics030_together.tif")
# Naming rules: file.name.directory + file.name.iblock + ".tif"

rmap_dem <- raster("E:/test/dem.tif")
spdf_dem <- as(rmap_dem,"SpatialPointsDataFrame")
df_dem <- as.data.frame(spdf_dem)

# mergeing results together
n.block <- 100
f.i.d <- c("E:/test/mapping/")
f.o.d <- c("E:/test/mapping_merge/")
f.iblock <- c("mlr.ak05.")
f.suffix <- c("mlr.ak05.tif")
```

```
MergingTiles(df_dem, f.i.d, f.iblock, n.block, f.o.d, f.suffix)
```

`NormalizeData` *Standardize and normalize data elements*

Description

Standardize and normalize data elements

Usage

```
NormalizeData()
```

Examples

```
# This function is optional to the user, depending on the data situation.
NormalizeData()
```

`ParallelComputing` *ParallelComputing Functions*

Description

ParallelComputing Functions

Usage

```
ParallelComputing(outpath, mymodels)
```

Arguments

`outpath`	: Output path of the result of the prediction file. The default is "output".
`mymodels`	: The models were selected, including QRF,RF and MLR.

Details

This function is the main function that performs parallel computations The outpath field refers to the filename of the data output The mymodels field has three modes to choose from: QRF,RF and MLR ‘QRF??? stands for Quantile Regression Forest Model Prediction Method ‘RF??? stands for Random Forest Model Prediction Method ‘MLR??? stands for Multiple Linear Regression Prediction Model

References

Breiman, L. (2001). Random forests. Mach. Learn. 45, 5???32. Meinshausen, N. (2006) "Quantile Regression Forests", Journal of Machine Learning Research 7, 983-999 http://jmlr.csail.mit.edu/papers/v7/

Examples

```
## This function performs parallel computing, of which the parameters are as follows:
## outpath: the filename of the data output
## mymodels: which model user want to use. Three modes are available:
## Quantile Regression Forest (QRF),Random Forest (RF) and Multiple Linear Regression (MLR)

#####################################################################################
# Example 1: Using random forest to produce soil map based on data in this package
# Loads related data sets
data("df.input" , package = "ParallelDSM")
data("df.mrrtf" , package = "ParallelDSM")
data("df.dem" , package = "ParallelDSM")

# Sets the path to the folder where the dataset will be stored
sampledata <- system.file("extdata" , "covariate", package = "ParallelDSM")

# Initializing the parameters for parallel computing
# ParallelInit_Test is same as ParallelInit
ParallelInit_Test(sampledata,df.input,dsmformul="socd030 ~ dem + mrrtf")
NormalizeData()
ParallelComputing(outpath = "mlrOutput" , mymodels = "MLR")
####################################################################################


####################################################################################
##  Example 2: Performing soil mapping based on my data with 3 CPUs ##

myinput <- "./all.input.csv"
# The sample data represents the file name where the data file is stored

# 'covariate' is the path name of a file
sampledata <- "./covariate" # the directory and filename
# The third parameter represents the name of the TIF file.
# nblock is used to partition the tif data into several blocks in the terms of row
# An appropriate nblock may optimize the speedup of parallel computing
ParallelInit(myinput,sampledata,"socd030 ~ twi + dem", nblock = 30 , ncore = 3)

ParallelComputing(outpath = "qrfOutput" , mymodels = "QRF")
####################################################################################
```

`ParallelInit` *As a data ParallelIniting function, sets some global variables that are not visible to the user*

Description

As a data ParallelIniting function, sets some global variables that are not visible to the user

Usage

```
ParallelInit(
  Fpath = "",
  fn = "",
  dsmformula = "",
  nblock = 6,
  ncore = 2,
  Fc = 1
)
```

Arguments

`Fpath` : The file path to the CSV file

`fn` : Name of the folder in which the soil data is stored

`dsmformula` : Symbolic description of a soil fitting model

`nblock` : the number of blocks for data cutting

`ncore` : Computes the CPU's kernel in parallel(fill in according to the computer configuration)

`Fc` : the encoding of file

References

Breiman, L. (2001). Random forests. Mach. Learn. 45, 5???32. Meinshausen, N. (2006) "Quantile Regression Forests", Journal of Machine Learning Research 7, 983-999 http://jmlr.csail.mit.edu/papers/v7/

Examples

```
############################################################################
## Example code                                                           ##
## Select your own reading method, as shown below                        ##
############################################################################
mydatas <- system.file("extdata", "all.input.csv", package = "ParallelDSM")
sampledatas <- system.file("extdata", "covariate", package = "ParallelDSM")
ParallelInit(mydatas,sampledatas,"socd030 ~ twi + dem + pa")

############################################################################
## If you want to use test cases, load the relevant data sets            ##
############################################################################
```

```
# Select the data set that comes with this package
# data(”df.input”, package = ”ParallelDSM”)
# data(”df.dem”, package = ”ParallelDSM”)

############################################################################
##  Use the data file references that come with this package              ##
############################################################################
# sampledatas <- system.file(”extdata”, ”covariate”, package = ”ParallelDSM”)

############################################################################
## Select your own data file references, as shown below                   ##
############################################################################
# sampledatas <- ”C:/mySampleDatas/”

############################################################################
## Use ParallelInit functions to process the data that is loaded in       ##
############################################################################
# ParallelInit(myinput,sampledata,”socd030 ~ twi + procur + dem”)

############################################################################
## This function is the main function that performs parallel computations ##
## The outpath field refers to the filename of the data output            ##
## The mymodels field has three modes to choose from: QRF,RF and MLR      ##
## 'QRF' stands for Quantile Regression Forest Model Prediction Method    ##
## 'RF' stands for Random Forest Model Prediction Method                  ##
## 'MLR' stands for Multiple Linear Regression Prediction Model           ##
## 'from' and 'to' are reserved fields that can be left unused by the user##
############################################################################
# ParallelComputing(outpath = ”myoutputs”, mymodels = ”MLR”)
```

ParallelInit_Test *Data initialization function is the first step to complete parallel training*

Description

Data initialization function is the first step to complete parallel training

Usage

```
ParallelInit_Test(
  fn = "",
  icsv = NULL,
  dsmformula = NULL,
  nblock = 6,
  ncore = 2
)
```

Arguments

`fn` : Name of the folder in which the soil data is stored

`icsv` : Use df.input from the built-in dataset

`dsmformula` :Symbolic description of a soil fitting model

`nblock` : the number of blocks for data cutting

`ncore` : Computes the CPU's kernel in parallel(fill in according to the computer configuration)

References

Breiman, L. (2001). Random forests. Mach. Learn. 45, 5???32. Meinshausen, N. (2006) "Quantile Regression Forests", Journal of Machine Learning Research 7, 983-999 http://jmlr.csail.mit.edu/papers/v7/

Examples

```
############################################################################
## Example code                                                           ##
## If you want to use test cases, load the relevant data sets             ##
## Select the data set that comes with this package                       ##
############################################################################
library(ParallelDSM)
data("df.input",package = "ParallelDSM")
data("df.dem",package = "ParallelDSM")
data("df.twi",package = "ParallelDSM")
sampledata <- system.file("extdata", "covariate", package = "ParallelDSM")
ParallelInit_Test(sampledata,df.input,dsmformula = "socd030 ~ twi + dem")
#ParallelComputing(outpath = "qrfOutput",mymodels = "QRF")

############################################################################
##  Use the data file references that come with this package              ##
############################################################################
# sampledatas <- system.file("extdata", "covariate", package = "ParallelDSM")

############################################################################
## Use ParallelInit_Test functions to process the data that is loaded in  ##
############################################################################
# ParallelInit_Test(sampledata,df.input,dsmformula = "socd030 ~ dem + twi")

############################################################################
## This function is the main function that performs parallel computations ##
## The outpath field refers to the filename of the data output            ##
## The mymodels field has three modes to choose from: QRF,RF and MLR      ##
## 'QRF' stands for Random Forest Model Prediction Method                 ##
## 'RF' stands for Machine Learning Model Prediction Method               ##
## 'MLR' stands for Multiple Linear Regression Prediction Model           ##
## 'from' and 'to' are reserved fields that can be left unused by the user##
############################################################################
# ParallelComputing(outpath = "myoutputs",mymodels = "MLR",from=1,to=200)
```

`smalltesttoy` *Black box test function to test whether R package was installed successfully*

Description

Black box test function to test whether R package was installed successfully

Usage

```
smalltesttoy(myflag)
```

Arguments

`myflag`	: Mark the successful installation of the R package

Examples

```
flag <- ""
smalltesttoy(flag)
```

Index