Package 'AgroBayes'

January 19, 2023

Type Package

Title Generates static and dynamic Bayesian networks for forecasting crop results

Version 0.1.0

Author Guilherme Afonso Halal [aut, cre], Ana Paula Ludtke Ferreira [ctb]

Maintainer Guilherme Afonso Halal <guilhermehalal@gmail.com>

Description This package permits, from variables related to agricultural production, to generate static and dynamic Bayesian networks that allow forecasting crop results. The package allows comparing the effectiveness of models generated from some network structure learning methods. The networks are generated based on functions available in the bnlearn (static networks) and dbnR (dynamic networks) packages. The forecasting of the crop production can be done from sets of data geographically separated (specific areas of the plantation) and also from chronological cuts (phenological phases).
All functions identified with the term TEST FUNCTION in the documentation and with a name starting with 'test', were added to the package to run the demonstration of the package's functionalities. The /AgroBayes/vignettes/demo.Rmd provides an interactive example of how the package works.
The final user must organize the dataset in dataframes representing the interval referring to a phenological phase and group them in lists according to the area of the plantation site.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Imports bnlearn, dbnR, Hmisc, dplyr, caTools, ggplot2, data.table, caret, stats, graphics, grDevices

R topics documented:

createDbn 2

createNetworks 3
runNetworks 4
testBuildSimulationData 5
testCreateDataFrames 5
testDefVars 6
testRunDataGen 7
testSetSimVarValues 8
validateNetwork 8

Index **10**

`createDbn` *Creates dynamic Bayesian networks for performance evaluation*

Description

Using harvest data from all phenological phases of the cultivar, from a specific area of the plantation, dynamic Bayesian networks are generated (using the `dbnR::learn_dbn_struc` functions with `natPsoho` and `dmmhc` methods), trained (using the `dbnR::fit_dbn_params` function) and evaluated for performance (using `dbnR::forecast_ts` and `caret::defaultSummary`). Two networks are created, one using `natPsoho` learning method and other using `dmmhc` method. The DBN are tested using `dbnR::forecast_ts` with `exact` and `approx` methods then, from the return of these functions, the RMSE, R-squared and MAE metrics are calculated and returned

Usage

```
createDbn(area)
```

Arguments

`area`	A dataframe of continuous data, from a specific area of the plantation to be used

Value

A dataframe with the dynamics networks metrics (RMSE, R-squared, MAE)

Examples

```
metricsDbn = createDbn(area_1)
```

createNetworks *Creates Bayesian networks for performance evaluation*

Description

Using harvest data from a phenological phase of the cultivar, from a specific area of the plantation, Bayesian network are generated (using the bnlearn::hc and bnlearn::mmhc functions), trained (using the bnlearn::bn.fit function) and evaluated for performance (using validateNetwork). Four networks are created, two from the pre-established topology and two learned only from the presented data.

Usage

```
createNetworks(areaphase, blacklist, whitelist)
```

Arguments

areaphase	the dataframe of discretized data to be used
blacklist	a dataframe of character string with two columns, it is passed as a parameter to bnlearn learn functions in order to avoid these arcs composing the final network
whitelist	a dataframe of character string with two columns, it is passed as a parameter to bnlearn learn functions in order to guarantee these arcs composing the final network

Value

Network evaluation metrics, as calculated in the validateNetwork function

Examples

```
metricsArea1Phase1 = list()
blacklist = data.frame(
from = c("X_1","X_1",
         "X_2", "X_2",
         "X_3", "X_3",
         "harvest",  "harvest",  "harvest" ),
to = c("X_2", "X_3", #from v1
       "X_1", "X_3", #from v2
       "X_1", "X_2", #from v3
       "X_1", "X_2", "X_3")) #from col
whitelist = data.frame(
from = c("X_1", "X_2", "X_3"),
to = c("harvest", "harvest", "harvest"))
areaphase = data.frame(area1_phase_1)
metricsArea1Phase1 = createNetworks (areaphase, blacklist, whitelist)
```

runNetworks *Executes* createNetworks *function to an area*

Description

Executes the createNetworks function for all phenological phases of an area. It also organizes the generated metrics in a dataframe.

Usage

```
runNetworks(arealist, blacklist, whitelist)
```

Arguments

arealist	list os dataframes of all phenological phase on a specific area of the plantation
blacklist	a dataframe of character string with two columns, it is passed as a parameter to bnlearn learn functions in order to avoid these arcs composing the final network
whitelist	a dataframe of character string with two columns, it is passed as a parameter to bnlearn learn functions in order to guarantee these arcs composing the final network

Value

A dataframe organizing the network metrics generated in the createNetworks function. The lines represent the network performance learned in each phenological phase of the area (arealist)

Examples

```
blacklist = data.frame(
from = c("X_1","X_1",
         "X_2", "X_2",
         "X_3", "X_3",
         "harvest",  "harvest",  "harvest" ),
to = c("X_2", "X_3", #from v1
       "X_1", "X_3", #from v2
       "X_1", "X_2", #from v3
       "X_1", "X_2", "X_3")) #from harvest
whitelist = data.frame(
from = c("X_1", "X_2", "X_3"),
to = c("harvest", "harvest", "harvest"))
arealist <- list(area1)
metricsArea1 <- createNetworks (arealist, blacklist, whitelist)
```

`testBuildSimulationData` *Initialize the values of all variables*

Description

Initialized by the `testRunDataGen` function, it generates all the values of the independent variables and the dependent variable, returning a list of areas with harvest data.

Usage

```
testBuildSimulationData(nHarvests, nPhases, nAreas = NULL, ...)
```

Arguments

`nHarvests`	number of harvests.
`nPhases`	number of phenological phases.
`nAreas`	number of areas of the plantation site if not informed the default is 6 areas.
`...`	parameters passed to `testDefVars` function

Value

list of areas with harvest data.

Examples

```
nHarvests = 1000
nphases = 5
nAreas = 10
nVars = 6
areas_list  = testBuildSimulationData(nHarvests, nphases)
areas_list_2  = testBuildSimulationData(nHarvests, nPhases, nAreas, nVars)
```

`testCreateDataFrames` *Organizes data in dataframes so it can be used in Bayesian learning functions*

Description

Called by the `testRunDataGen` function. Function receives the list of variables referring to an area and organizes it into dataframes, one dataframe for each phenological phase.

Usage

```
testCreateDataFrames(data)
```

Arguments

data	list of variables referring to an area

Value

list of dataframes, one for each phenological stage.

Examples

```
area_1_df = testCreateDataFrames(area_1_list)
```

---

testDefVars *Defines the type and quantity of independent variables*

Description

Called by the testBuildSimulationData function. Function allows defining the type and quantity of initialized independent variables for data generation. It is possible to define a number of variables greater than 3. If the type of variable is defined, it is mandatory to inform a list with the name of the variables as well. If only the number of variables are defined, the default nomenclature is X_1, X_2, ..., X_n-1, X_n is applied, with randomly defined behavior. If the number of variables is not defined, 3 variables will be initialized, one that always grows over time, a constant and one that oscillates over time.

Usage

```
testDefVars(n_var = NULL, type_var = NULL, name_var = NULL)
```

Arguments

n_var	number of variables (not counting the dependent variable) if not informed the default is 3 variables.
type_var	a list of types of variables. If informed the name_var parameter must be informed too.
name_var	a list of names of variables. If informed the type_var parameter must be informed too.

Value

list of variables, that is a list containing (name of the variable, type of variable, value min and value max)

Examples

```
defProdVariables = testDefVars(...)
defProdVariables = testDefVars(
4,
c("precipitation", "Mn_rate", "Zn_rate", "avg_temp"),
 c(1, 3, 3, 2))
```

`testRunDataGen` *Starts data generation*

Description

It receives the parameters for the `testBuildSimulationData` function to generate simulated data for carrying out the package tests. It also distributes the data generated in lists of dataframes representing the set of phenological phases of each productive area. The data are still treated so that there is a copy of the data set discretized and another not, for the purpose of comparing performance between dynamic and static networks.

Usage

```
testRunDataGen(nHarvests, nphases, nAreas, nVars, nClass, ...)
```

Arguments

`nHarvests`	number of harvests.
`nphases`	number of phenological phases.
`nAreas`	number of areas of the plantation site.
`nVars`	number of variables (not counting the dependent variable).
`nClass`	Number of classes for dataframe discretization. Must be 5 or 3.
`...`	parameters passed to `testBuildSimulationData` function

Value

list with two dataframes one with continuous data and the other with discrete data

Examples

```
areas <- testRunDataGen(nHarvests, nphases, nAreas, nVars, nClass)
```

testSetSimVarValues *Generates the values of dependent and independent variables*

Description

Called by the testBuildSimulationData function. Function starts the values of dependent and independent variables The 'harvest' variables is is calculated from rules that depend on the number of independent variables and the type of area where this harvest was generated. There are 6 types of areas for standard situations, 3 independent variables, and 6 types of areas for situations where there are more than 3 variables.

Usage

```
testSetSimVarValues(nHarvests, areatype, prodvars, nPhases)
```

Arguments

nHarvests	number of harvests.
areatype	integer that defines the type of area. It determines the kind of relationship between the independent variables and the 'harvest' variable
prodvars	list of independent variables, that is a list containing (name of the variable, type of variable, value min and value max)
nPhases	number of phenological phases.

Value

list of values of all variables

Examples

```
variables =
testSetSimVarValues(nHarvests, areatype, defProdVariables, nPhases)
```

validateNetwork *Generates Bayesian networks performance evaluation*

Description

Using the functions available in the repository https://github.com/KaikeWesleyReis/bnlearn-multivar-prediction- calculates the metrics of the four Bayesian networks generated in the createNetworks function when executed in context of runNetworks function.

Usage

```
validateNetwork(
  test,
  train,
  dag_fitted1,
  dag_fitted2,
  dag_fitted3,
  dag_fitted4
)
```

Arguments

test	dataframe to be used to test the Bayesian networks. It is composed of a 25 the createNetworks function.
train	dataframe to be used to train the Bayesian networks. It is composed of a 75 the createNetworks function.
dag_fitted1	Fitted Bayesian network to be tested
dag_fitted2	Fitted Bayesian network to be tested
dag_fitted3	Fitted Bayesian network to be tested
dag_fitted4	Fitted Bayesian network to be tested

Value

List of values returned from bnMetricsMultiVarPrediction.

Index

createDbn, 2
createNetworks, 3, *4*, *8, 9*

runNetworks, 4, *8*

testBuildSimulationData, 5, *6–8*
testCreateDataFrames, 5
testDefVars, *5*, 6
testRunDataGen, *5*, 7
testSetSimVarValues, 8

validateNetwork, *3*, 8