Package ‘vdra’

September 9, 2021

Title Vertical Distributed Regression Analysis

Maintainer Thomas Kent <kentedegrees@gmail.com>

Type Package

Version 1.0.0

Author Thomas Kent [aut, cre],
Yuji Samizo [aut]

Description Implements linear, logistic, and Cox regression on vertically partitioned data across several data partners. Data is not shared between data partners or the analysis center and the computations can be considered secure. Three different protocols are implemented. 2-Party: two data partners which communicate directly without an intermediate analysis center; 2T-Party: two data partners communicate indirectly via an analysis center, and KT-Party: two or more data partners plus an analysis center are all allowed to communicate directly. 2-Party and 2^T-Party use a form of secure multiplication as found in Karr, et. al. (2009) ``Privacy-Preserving Analysis of Vertically Partitioned Data Using Secure Matrix Products'' and Slavkovic et. al. (2007) ``Secure Logistic Regression of Horizontally and Vertically Partitioned Distributed Databases'' <doi:10.1109/ICDMW.2007.114>. Full details can be found in Samizo (In preparation).

BugReports https://github.com/kentedegrees/vdra/issues

License GPL-3

NeedsCompilation yes

URL

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

Suggests survival (>= 3.2-7), knitr (>= 1.28), rmarkdown (>= 2.2)

VignetteBuilder knitr

RoxygenNote 7.1.1

Repository CRAN

Date/Publication 2021-09-09 06:20:02 UTC

R topics documented:

differentModel *Fitting Different Linear Models*

Description

Models are specified symbolically. A typical model is of the form `response ~ term_1 + term_2 + ... + term_k` where `response` and `term_i` are variables names used in the orgional linear model which created the object `x`. The `response` can be the orginal respose or any of the other covariates. Interactions are not allowed. Not all variables in the original model have to be used.

Usage

```
differentModel(formula, x)
```

Arguments

`formula`	an object of class "`formula`": a symbolic description of the model to be fitted. The model must be additive with no interactions.
`x`	an object of class `vdralinear`.

Value

Returns an object of class `vdralinear`.

See Also

`AnalysisCenter.2Party`, `AnalysisCenter.3Party`, `AnalysisCenter.KParty`

Examples

```
fit = differentModel(Change_BMI ~ Exposure + Age + NumRx, vdra_fit_linear_A)
summary(fit)

fit = differentModel(Age ~ Change_BMI + Exposure + NumRx, vdra_fit_linear_A)
summary(fit)
```

---

distributed2party *Two Party Vertical Distributed Regression Analysis*

Description

AnalysisCenter.2Party and DataPartner.2Party are used in conjuction with PopMedNet to perform linear, logistic, or cox regression on data that has been partitioned vertically between two data partners. The data partner which holds the response variable(s) uses AnalysisCener.2Party and the other data partner uses DataPartner.2Party. While both data partners share information with each other in order to perform the regression, data is kept secure and not shared, nor is any information shared that would allow one data partner to reconstruct part of the other data partners data. Final coefficients and other regression statistics are computed by the analysis center and shared with the other data partner.

Usage

```
AnalysisCenter.2Party(regression = "linear", data = NULL, response = NULL,
                      strata = NULL, mask = TRUE, monitorFolder = NULL,
                      msreqid = "v_default_00_000", blocksize = 500,
                      tol = 1e-8, maxIterations = 25, sleepTime = 10,
                      maxWaitingTime = 86400, popmednet = TRUE,
                      trace = FALSE, verbose = TRUE)

DataPartner.2Party(regression = "linear", data = NULL, strata = NULL,
                   mask = TRUE, monitorFolder = NULL, sleepTime = 10,
                   maxWaitingTime = 86400, popmednet = TRUE,
                   trace = FALSE, verbose = TRUE)
```

Arguments

regression	the model to be used to fit the data. The default regression "linear" fits a least squares linear model to the data. Alternatively, "logistic" returns a fitted logistic model, and "cox" returns a fitted Cox proportional hazards model.
data	a data.frame or matrix which contains the data to be used in the model. For DataPartner.2Party(), all columns will be used as covariates in the regression. For AnalysisCenter.2Party(), all columns, with the expection of the column specified by response, will be used as covariates in the regression.

`response` for "linear" and "logistic" regression, the name of the column in `data` which holds the response variable. If `reponse = NULL`, then the first column of `data` will be used as the response variable. For "cox" regression response hold the name of the column which is time to event and the name of the column which is the event type (0 = censored, 1 = event). If `response = NULL`, then the first column of `data` is assumed to be the time to even and the second column is assumed to be the event type.

`strata` for "cox" regression only. A `vector` of character strings identifying the names of the covariates from either party which will be used as strata. Both `AnalysisCenter.2party` and `DataPartner.2Party` must specify the same vector of strata.

`mask` logical value: If `FALSE`, strata levels for the strata which belong to the party which specified `FALSE` will be identified by name. If `TRUE`, levels for the strata which belong to the party which specified `TRUE` will be put in a random order and level names will be changed to `NA`.

`monitorFolder` the folder where the directories `dplocal`, `inputfiles`, `macros`, `msoc`, and `rprograms` are located.

`msreqid` a character string specifying the name of the *Request ID* as specified when creating the Distributed Regresion request on PopMedNet. Used for logging purposes only.

`blocksize` the minimium size used to horizontally partition the data for data transfer between the two parties.

`tol` the tolerance used to determine convergence in "logistic" and "cox" regression.

`maxIterations` the maximum number of iterations to perform "logistic" or "cox" regression before non-convergence is declared.

`sleepTime` the number of seconds to wait after writing the last file to disk before signalling the PMN Datamart Client that files are ready to be transferred.

`maxWaitingTime` the number of seconds to wait to receive files before a transfer error is declared and the program halts execution.

`popmednet` logical value: if `TRUE`, assumes that PopMednet is being used to transfer the files and implements PopMedNet specific routines. In particular, a 15 second offset terminiation of routines that execute in parallel is implemented.

`trace` logical value: if `TRUE` and `verbose == TRUE`, prints every function called during execution. Used for debugging.

`verbose` logical value. If `TRUE`, prints out information to document the progression of the computation.

Value

Returns an object of `class` `vdralinear` for linear regression, `vdralogistic` for logistic regression, or `vdracox` for cox regression.

See Also

`AnalysisCenter.3Party`, `AnalysisCenter.KParty`

Examples

```
## Not run:
## 2 party linear regression

# Analysis Center -- To be run in one instance of R.
# The working directory should be the same as specified in the PopMedNet
# requset for the analysis center.

fit = AnalysisCenter.2Party(regression = "linear", data = vdra_data[, c(1, 5:7)],
        response = "Change_BMI", monitorFolder = tempdir())

# Data Partner -- To be run in second instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.

fit = DataPartner.2Party(regression = "linear", data = vdra_data[, 8:11],
        monitorFolder = tempdir())

## 2 party logistic regression

# Analysis Center -- To be run in one instance of R.
# The working directory should be the same as specified in the PopMedNet
# requset for the analysis center.

fit = AnalysisCenter.2Party(regression = "logistic", data = vdra_data[, c(2, 5:7)],
        response = "WtLost", monitorFolder = tempdir())

# Data Partner -- To be run in second instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.

fit = DataPartner.2Party(regression = "logistic", data = vdra_data[, 8:11],
        monitorFolder = tempdir())

## 2 party cox regression

# Analysis Center -- To be run in one instance of R.
# The working directory should be the same as specified in the PopMedNet
# requset for the analysis center.

fit = AnalysisCenter.2Party(regression = "cox", data = vdra_data[, c(3:4, 5:7)],
        response = c("Time", "Status"), strata = c("Exposure", "Sex"),
        monitorFolder = tempdir())

# Data Partner -- To be run in second instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.

fit = DataPartner.2Party(regression = "cox", data = vdra_data[, 8:11],
        strata = c("Exposure", "Sex"), monitorFolder = tempdir())

## End(Not run)
```

distributed3party *Three Party Vertical Distributed Regression Analysis*

Description

AnalysisCenter.3Party, DataPartner1.3Party and DataPartner2.3Party are used in conjuction with PopMedNet to perform linear, logistic, or cox regression on data that has been partitioned vertically between two data partners. The data partner which holds the response variable(s) uses Datapartner1.3Party and the other data partner uses DataPartner2.3Party. Data partners are not allowed to communicate with each other, but share inforamtion via a trusted third party analysis center. While any infomration that is shared with the analysis center by a data partner, with the exception of some summary statistics, is encrypted by the sending data parter, if the infomration needs to be sent on to the other data parter for futher analysis, the analysis center further encrypts the data. That way, any information that deals directly with the raw data that moves between two data partners is doubly encrypted to keep both the analysis center and the other data partner from learning it. Thus, no information is shared between the data partners or analysis center that would allow one data partner to reconstrut part of the other data partners data. Final coefficients and other regression statistics are computed by the analysis center and shared with the data partners.

Usage

```
AnalysisCenter.3Party(regression = "linear", monitorFolder = NULL,
                      msreqid = "v_default_00_000", blocksize = 500,
                      tol = 1e-8, maxIterations = 25, sleepTime = 10,
                      maxWaitingTime = 86400, popmednet = TRUE,
                      trace = FALSE, verbose = TRUE)

DataPartner1.3Party(regression = "linear", data = NULL, response = NULL,
                    strata = NULL, mask = TRUE, monitorFolder = NULL,
                    sleepTime = 10, maxWaitingTime = 86400, popmednet = TRUE,
                    trace = FALSE, verbose = TRUE)

DataPartner2.3Party(regression = "linear", data = NULL, strata = NULL,
                    mask = TRUE, monitorFolder = NULL, sleepTime = 10,
                    maxWaitingTime = 86400, popmednet = TRUE,
                    trace = FALSE, verbose = TRUE)
```

Arguments

regression — the model to be used to fit the data. The default regression "linear" fits a least squares linear model to the data. Alternatively, "logistic" returns a fitted logistic model, and "cox" returns a fitted Cox proportional hazards model.

data — a data.frame or matrix which contains the data to be used in the model. For DataPartner2.3Party(), all columns will be used as covariates in the regression. For DataPartner1.3Party(), all columns, with the expection of the column specified by response, will be used as covariates in the regression.

`response` for `"linear"` and `"logistic"` regression, the name of the column in `data` which holds the response variable. If `reponse = NULL`, then the first column of `data` will be used as the response variable. For `"cox"` regression response hold the name of the column which is time to event and the name of the column which is the event type (0 = censored, 1 = event). If `response = NULL`, then the first column of `data` is assumed to be the time to even and the second column is assumed to be the event type.

`strata` for `"cox"` regression only. A `vector` of character strings identifying the names of the covariates from either party which will be used as strata. Both `DataPartner1.3party` and `DataPartner2.3Party` must specify the same vector of strata.

`mask` logical value: If `FALSE`, strata levels for the strata which belong to the party which specified `FALSE` will be identified by name. If `TRUE`, levels for the strata which belong to the party which specified `TRUE` will be put in a random order and level names will be changed to `NA`.

`monitorFolder` the folder where the directories `dplocal`, `inputfiles`, `macros`, `msoc`, and `rprograms` are located.

`msreqid` a character string specifying the name of the *Request ID* as specified when creating the Distributed Regresion request on PopMedNet. Used for logging purposes only.

`blocksize` the minimium size used to horizontally partition the data for data transfer between the two parties.

`tol` the tolerance used to determine convergence in `"logistic"` and `"cox"` regression.

`maxIterations` the maximum number of iterations to perform `"logistic"` or `"cox"` regression before non-convergence is declared.

`sleepTime` the number of seconds to wait after writing the last file to disk before signalling the PMN Datamart Client that files are ready to be transferred.

`maxWaitingTime` the number of seconds to wait to receive files before a transfer error is declared and the program halts execution. Should be the same for both parties when `delayOffset = TRUE`.

`popmednet` logical value: if `TRUE`, assumes that PopMednet is being used to transfer the files and implements PopMedNet specific routines. In particular, a 15 second offset between terminiation of routines that execute in parallel is implemented.

`trace` logical value: if `TRUE` and `verbose == TRUE`, prints every function call. Used for debugging.

`verbose` logical value. If `TRUE`, prints out information to document the progression of the computation.

Value

Returns an object of `class` `vdralinear` for linear regression, `vdralogistic` for logistic regression, or `vdracox` for cox regression.

See Also

`AnalysisCenter.2Party`, `AnalysisCenter.KParty`

Examples

```
## Not run:
## 3 party linear regression

# Analysis Center -- To be run in one instance of R.
# The working directory should be the same as specified in the PopMedNet
# requset for the analysis center.
fit = AnalysisCenter.3Party(regression = "linear", monitorFolder = tempdir())

# Data Partner 1 -- To be run in second instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner1.3Party(regression = "linear", data = vdra_data[, c(1, 5:7)],
          response = "Change_BMI", monitorFolder = tempdir())

# Data Partner 2 -- To be run in third instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner2.3Party(regression = "linear", data = vdra_data[, 8:11],
          monitorFolder = tempdir())

## 3 party logistic regression

# Analysis Center -- To be run in one instance of R.
# The working directory should be the same as specified in the PopMedNet
# requset for the analysis center.
fit = AnalysisCenter.3Party(regression = "logistic", monitorFolder = tempdir())

# Data Partner 1 -- To be run in second instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner1.3Party(regression = "logistic", data = vdra_data[, c(2, 5:7)],
          response = "WtLost", monitorFolder = tempdir())

# Data Partner 2 -- To be run in third instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner2.3Party(regression = "logistic", data = vdra_data[, 8:11],
          monitorFolder = tempdir())

## 3 party cox regression

# Analysis Center -- To be run in one instance of R.
# The working directory should be the same as specified in the PopMedNet
# requset for the analysis center.
fit = AnalysisCenter.3Party(regression = "cox", monitorFolder = tempdir())

# Data Partner 1 -- To be run in second instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner1.3Party(regression = "cox", data = vdra_data[, c(3:4, 5:7)],
        response = c("Time", "Status"), strata = c("Exposure", "Sex"),
        monitorFolder = tempdir())

# Data Partner 2 -- To be run in third instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner2.3Party(regression = "cox", data = vdra_data[, 8:11],
         strata = c("Exposure", "Sex"), monitorFolder = tempdir())

## End(Not run)
```

distributedKparty *K-Party Vertical Distributed Regression Analysis*

Description

AnalysisCenter.KParty and DataPartner.KParty are used in conjuction with PopMedNet to perform linear, logistic, or cox regression on data that has been partitioned vertically between two or more data partners. The data partners which holds the data use DataPartner.KParty while a trusted "third" party uses AnalysisCenter.KParty. Data partners are allowed to communicate with each other and the analysis center, no information is shared between the data partners or analysis center that would allow one data partner or the analysis center to reconstrut part of the other data partners data. Final coefficients and other regression statistics are computed by the analysis center and shared with the data partners.

Usage

```
AnalysisCenter.KParty(regression = "linear", numDataPartners = NULL,
                      monitorFolder = NULL, msreqid = "v_default_00_000",
                      tol = 1e-8, maxIterations = 25, sleepTime = 10,
                      maxWaitingTime = 86400, popmednet = TRUE,
                      trace = FALSE, verbose = TRUE)

DataPartner.KParty(regression = "linear", data = NULL, response = NULL,
                   strata = NULL, mask = TRUE, numDataPartners = NULL,
                   dataPartnerID = NULL, monitorFolder = NULL,
                   sleepTime = 10, maxWaitingTime = 86400, popmednet = TRUE,
                   trace = FALSE, verbose = TRUE)
```

Arguments

regression — the model to be used to fit the data. The default regression "linear" fits a least squares linear model to the data. Alternatively, "logistic" returns a fitted logistic model, and "cox" returns a fitted Cox proportional hazards model.

data — a data.frame or matrix which contains the data to be used in the model. All columns will be used as covariates in the regression with the exception of the data partner which has dataPartnerID = 1. For this data partner, all columns, with the expection of the column specified by response, will be used as covariates in the regression.

`response` only used for data parther with `dataPartnerID = 1`. For `"linear"` and `"logistic"` regression, the name of the column in `data` which holds the response variable. If `reponse = NULL`, then the first column of `data` will be used as the response variable. For `"cox"` regression response hold the name of the column which is time to event and the name of the column which is the event type (0 = censored, 1 = event). If `response = NULL`, then the first column of `data` is assumed to be the time to even and the second column is assumed to be the event type.

`strata` for `"cox"` regression only. A `vector` of character strings identifying the names of the covariates from either party which will be used as strata. All data partners must specify the same vector of strata.

`mask` logical value: If `FALSE`, strata levels for the strata which belong to the party which specified `FALSE` will be identified by name. If `TRUE`, levels for the strata which belong to the party which specified `TRUE` will be put in a random order and level names will be changed to `NA`.

`numDataPartners`
the number of data partners which are supplying data for the regression.

`dataPartnerID` a unique identifier for each data partner. The data partner with the response variable(s) must have `dataPartnerID = 1`. All other data partners must have an integer value from 2 to `numDataPartners`.

`monitorFolder` the folder where the directories `dplocal`, `inputfiles`, `macros`, `msoc`, and `rprograms` are located.

`msreqid` a character string specifying the name of the *Request ID* as specified when creating the Distributed Regresion request on PopMedNet. Used for logging purposes only.

`tol` the tolerance used to determine convergence in `"logistic"` and `"cox"` regression.

`maxIterations` the maximum number of iterations to perform `"logistic"` or `"cox"` regression before non-convergence is declared.

`sleepTime` the number of seconds to wait after writing the last file to disk before signalling the PMN Datamart Client that files are ready to be transferred.

`maxWaitingTime` the number of seconds to wait to receive files before a transfer error is declared and the program halts execution. Should be the same for all parties when `delayOffset = TRUE`.

`popmednet` logical value: if `TRUE`, assumes that PopMednet is being used to transfer the files and implements PopMedNet specific routines. In particular, a 15 second offset between terminiation of routines that execute in parallel is implemented.

`trace` logical value: if `TRUE` and `verbose == TRUE`, prints every function call. Used for debugging.

`verbose` logical value. If `TRUE`, prints out information to document the progression of the computation.

Value

Returns an object of `class` `vdralinear` for linear regression, `vdralogistic` for logistic regression, or `vdracox` for cox regression.

See Also

AnalysisCenter.2Party, AnalysisCenter.KParty

Examples

```
## Not run:
## 3 party linear regression

# Analysis Center -- To be run in one instance of R.
# The working directory should be the same as specified in the PopMedNet
# requset for the analysis center.
fit = AnalysisCenter.KParty(regression = "linear", numDataPartners = 2,
              monitorFolder = tempdir())

# Data Partner 1 -- To be run in second instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner.KParty(regression = "linear", data = vdra_data[, c(1, 5:7)],
          response = "Change_BMI", numDataPartners = 2, dataPartnerID = 1,
          monitorFolder = tempdir())

# Data Partner 2 -- To be run in third instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner.KParty(regression = "linear", data = vdra_data[, 8:11],
          numDataPartners = 2, dataPartnerID = 2, monitorFolder = tempdir())

## 3 party logistic regression

# Analysis Center -- To be run in one instance of R.
# The working directory should be the same as specified in the PopMedNet
# requset for the analysis center.
fit = AnalysisCenter.KParty(regression = "logistic", numDataPartners = 2,
              monitorFolder = tempdir())

# Data Partner 1 -- To be run in second instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner.KParty(regression = "logistic", data = vdra_data[, c(2, 5:7)],
          response = "WtLost", numDataPartners = 2, dataPartnerID = 1,
          monitorFolder = tempdir())

# Data Partner 2 -- To be run in third instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner.KParty(regression = "logistic", data = vdra_data[, 8:11],
          numDataPartners = 2, dataPartnerID = 2, monitorFolder = tempdir())

## 3 party cox regression

# Analysis Center -- To be run in one instance of R.
# The working directory should be the same as specified in the PopMedNet
# requset for the analysis center.
fit = AnalysisCenter.KParty(regression = ”cox”, numDataPartners = 2,
              monitorFolder = tempdir())

# Data Partner 1 -- To be run in second instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner.KParty(regression = ”cox”, data = vdra_data[, c(3:4, 5:7)],
        response = c(”Time”, ”Status”), strata = c(”Exposure”, ”Sex”),
        numDataPartners = 2, dataPartnerID = 1, monitorFolder = tempdir())

# Data Partner 2 -- To be run in third instand of R, on perhaps a different machine.
# The working directory should be the same as specified in the PopMedNet
# request for the data partner.
fit = DataPartner.KParty(regression = ”cox”, data = vdra_data[, 8:11],
         strata = c(”Exposure”, ”Sex”), numDataPartners = 2, dataPartnerID = 2,
         monitorFolder = tempdir())

## End(Not run)
```

HoslemTest — *Hosmer-Lemeshow Test for Vertical Distributed Logistic Regression*

Description

Run the Hosmer-Lemeshow test for an object created by 2-party, 3-party, or K-party vdra logistic regression. Only the party that holds the response may invoke this function.

Usage

```
HoslemTest(x, nGroups = 10)
```

Arguments

`x` an object of type `vdralogistic`.

`nGroups` the number of groups that the data will be sperated into.

Value

Returns an object of class `hoslemdistributed`. Objects of this class have a method for the function `print`. The following component must be included in a `hoslemdistributed` object.

`hoslem` a vector containing three numeric quantities: the chi-square value of the test, the degrees of freedom of the test, and p-value of the test, in that order.

Examples

```
HoslemTest(vdra_fit_logistic_A)

HoslemTest(vdra_fit_logistic_A, 20)
```

plot.survfitDistributed *Plotting Survival Curves for Vertical Distributed Cox Regression*

Description

Plots a survivial curve as specified by `survfitDistribuited` object.

Usage

```
## S3 method for class 'survfitDistributed'
plot(x, merge = FALSE, ...)
```

Arguments

`x`	a `survfitDistributed` object.
`merge`	logical. It `TRUE`, plots all strata of the survival curve on one plot. If `FALSE`, plots all strata in different plots.
`...`	common graphical parameters (not fully implemented).

Value

No return value.

See Also

`survfitDistributed`

Examples

```
sfit = survfitDistributed(vdra_fit_cox_A)
plot(sfit)

# From Data Partner 1
sfit = survfitDistributed(vdra_fit_cox_A, ~Exposure, data = vdra_data[, c(3:4, 5:7)])
plot(sfit)
plot(sfit, merge = FALSE)

# From Data Partner 2
sfit = survfitDistributed(vdra_fit_cox_B, ~Race + Sex, data = vdra_data[, 8:11])
plot(sfit, merge = FALSE)
```

pmn *PopMedNet Simulator*

Description

This function is intended to act as a proxy for PopMedNet when developing code to run on PopMedNet, or when testing out the distributed regression programs provided with this package. When used, it is expected that this function, the analysis center, and the data partner(s) will each be run in their own instances of R. The analysis center and the data partner(s) will share a common directory where the subdirectories dp0, dp1, ... will be stored. The directory dp0 is the monitor folder for the analysis center. With the exception of 2-party regression, it is assumed that the data partner using dp1 is also the data partner which holds the response. In the case of 2-party regression, the analysis center holds the response.

Usage

```
## Not run:
pmn(numParty, directory = NULL, verbose = TRUE)
## End(Not run)
```

Arguments

`numParty` the number of parties (analysis center + data partners) involved in the multiple regression. If a data partner is also acting as the analysis center, then that data partner is only counted once.

`directory` the directory where the directories dp0, dp1, ... are located, which are used by the analysis center and data partner(s) to save data and receive data from each other.

`verbose` logical value. If `TRUE`, prints out information to document the progression of the computation.

Value

Returns `NULL`.

See Also

`AnalysisCenter.2Party`, `AnalysisCenter.3Party`, `AnalysisCenter.KParty`

RocTest *Create the ROC for Vertical Distributed Logistic Regression*

Description

Generate the receiver operator curve on an object created by 2-party, 3-party, or K-party vdra logistic regression. Only the party that holds the response may invoke this function.

Usage

```
RocTest(x, bins = 10)
```

Arguments

x	an object of type vdralogistic.
bins	the number of bins the data will be separated into.

Value

Returns an object of class rocdistributed. Objects of this class have a method for the function print. The following components must be included in a rocdistributed object.

roc	a two column matrix containing the cordinates of 1 - specifity and sensitivity.
auc	numeric value which is area under the curve.

Examples

```
RocTest(vdra_fit_logistic_A)

RocTest(vdra_fit_logistic_A, 40)
```

summary.vdracox *Summary Method for Vertical Distributed COX Models*

Description

Produces a summary of a fitted vdra cox model.

Usage

```
  ## S3 method for class 'vdracox'
  ## S3 method for class 'vdracox'
summary(object, ...)
```

Arguments

`object` a `vdracox` object.

`...` futher argumetns passed to or from other methods.

Value

Returns an object of class `summary.vdracox`. Objects of this class have a method for the function `print`. The following components must be included in `summary.vdracox` object.

`failed` logical value. If `FALSE`, then there was an error processing the data. if `TRUE`, there were no errors.

`converged` logical value. If `TRUE`, the regression converged. If `FALSE`, it did not.

`party` a vector which indicates the party from which each covariate came.

`coefficients` the vector of coefficients. If the model is over-determined, there will be missing values in the vector corresponding to the redudant columns model matrix.

`expcoef` a vector which represents exp(coefficients).

`secoef` the vector of the standard error of the coefficients.

`zvals` the z-values of the coefficients.

`pvals` the p-values of the coefficients.

`expncoef` a vector which represents exp(-coefficients).

`lower95` a vector of the lower bounds of the 95% confidence interval for exp(coefficients).

`upper95` a vector of the upper bounds of the 95% confidence interval for exp(coefficients).

`n` the number of observations in the data.

`nevent` the number of events used in the fit.

`concordance` a vector containing the number of events which are concordant, discordant, tied.risk, tied.time. Also contains the concordance statistic and its standard error. Calculated using the `survival` package, if installed. If not installed, all values are `NA`.

`rsquare` a vector containing an r-square value for the fit and its p-value.

`lrt` a vector contaiing the likelihood ratio test statistic and its p-value.

`df` the degrees of freedom.

`wald.test` a vector containg the Wald test statistic and its p-value.

`score` a vector contining the score test statistic and its p-value.

`iter` the number of iterations of the cox algorithm before convergence.

See Also

`vdracox`

Examples

```
summary(vdra_fit_cox_A)
```

summary.vdralinear *Summary Method for Vertical Distributed Linear Regression Models*

Description

Produces a summary of a fitted vdra linear regression model.

Usage

```
  ## S3 method for class 'vdralinear'
  ## S3 method for class 'vdralinear'
summary(object, ...)
```

Arguments

`object`	a `vdralinear` object.
`...`	futher argumetns passed to or from other methods.

Value

Returns an object of class `summary.vdralinear`. Objects of this class have a method for the function `print`. The following components must be included in `summary.vdralinear` object.

`failed`	logical value. If `FALSE`, then there was an error processing the data. if `TRUE`, there were no errors.
`party`	a vector which indicates the party from which each covariate came.
`coefficients`	the vector of coefficients. If the model is over-determined, there will be missing values in the vector corresponding to the redudant columns model matrix.
`secoef`	the vector of the standard error of the coefficients.
`tvals`	the t-values of the coefficietns.
`pvals`	the p-values of the coefficients.
`rstderr`	residual standard error.
`rsquare`	r squared.
`adjrsquare`	adjusted r squared.
`Fstat`	the F-statistic for the linear regression.
`df1`	the numerator degrees of freedom for the F-statistic.
`df2`	the denominator degrees of freedom for the F-statistic.
`Fpval`	the p-value of the F-statistic for the linear regression.

See Also

`vdralinear`

Examples

```
summary(vdra_fit_linear_A)
```

summary.vdralogistic *Summary Method for Vertical Distributed Logistic Regression Models*

Description

Produces a summary of a fitted vdra logistic regression model.

Usage

```
  ## S3 method for class 'vdralogistic'
  ## S3 method for class 'vdralogistic'
summary(object, ...)
```

Arguments

`object` a `vdralogistic` object.

`...` futher argumetns passed to or from other methods.

Value

Returns an object of class `summary.vdralogistic`. Objects of this class have a method for the function `print`. The following components must be included in `summary.vdralogistic` object.

`failed` logical value. If `FALSE`, then there was an error processing the data. if `TRUE`, there were no errors.

`converged` logical value. If `TRUE`, the regression converged. If `FALSE`, it did not.

`party` a vector which indicates the party from which each covariate came.

`coefficients` the vector of coefficients. If the model is over-determined, there will be missing values in the vector corresponding to the redudant columns model matrix.

`secoef` the vector of the standard error of the coefficients.

`tvals` the t-values of the coefficietns.

`pvals` the p-values of the coefficients.

`nulldev` the null deviance of the fit.

`nulldev_df` the degrees of freedom for the null deviance.

`resdev` the residual deviance of the fit.

`resdev_df` the degrees of freedome for the residual deviance.

`aic` the AIC of the fit.

`bic` the BIC of the fit.

`iter` the number of iterations of the cox algorithm before convergence.

See Also

`vdralogistic`

Examples

```
summary(vdra_fit_logistic_A)
```

---

survfitDistributed *Create Survival Curves for Vertical Distributed Cox Regression*

Description

This function creates survival curves for a previously defined vdracox object. The function also accepts a formula and the original data supplied by the calling party allowing exploration of other potential strata. Both formula and data must be NULL or both must be specified.

Usage

```
survfitDistributed(x, formula = NULL, data = NULL)
```

Arguments

x	an object of type vdracox.
formula	a formula which defines alternative strata for the survival curve.
data	if formula is specified, this should be the data that was supplied by the calling party.

Value

Returns an object of class survfitDistributed. Objects of this class have methods for the functions print and plot. The following components must be included in a legitimate survfitDistributed object.

n	the total number of subjects in each curve.
time	the time points at which the curve has a step.
n.risk	the number of subjects at risk at each time point.
n.event	the number of events that occour at each time point.
n.censor	the number of subjects who are censored at each time point.
strata	the number of points in each strata.
surv	the estimate of the survival time at each time step.
type	the type of censoring. Currently, always "right".

See Also

plot.survfitDistributed

Examples

```
sfit = survfitDistributed(vdra_fit_cox_A)
print(sfit)
plot(sfit)

# From Data Partner 1
sfit = survfitDistributed(vdra_fit_cox_A, ~Exposure, data = vdra_data[, c(3:4, 5:7)])
print(sfit)
plot(sfit)

# From Data Partner 2
sfit = survfitDistributed(vdra_fit_cox_B, ~Race + Sex, data = vdra_data[, 8:11])
print(sfit)
plot(sfit, merge = TRUE)
```

`vdracox` *Vertical Distributed Cox Regression Results Object*

Description

This class of object is returned by the two party, three party, and K-party distributed regression analysis programs when "cox" regression is specified. Objects of this class have methods for the functions `print` and `summary`.

Arguments

The following components must be included in a legitimate `vdracox` object.

logical value. If `FALSE`, then there was an error processing the data. if `TRUE`, there were no errors.

`failed` `converged` logical value. If `TRUE`, the regression converged. If `FALSE`, it did not.

`party` a vector which indicates the party from which each covariate came.

`coefficients` the vector of coefficients. If the model is over-determined, there will be `NA` values in the vector corresponding to the redudant columns model matrix.

`expcoef` a vector which represents exp(coefficients).

`expncoef` a vector which represents exp(-coefficients).

`var` the variance matrix of the coefficients. Rows and columns corresponding to any missing coefficients are set to zero.

`secoef` the vector of the standard error of the coefficients.

`zvals` the z-values of the coefficients.

`pvals` the p-values of the coefficients.

`lower95` a vector of the lower bounds of the 95% confidence interval for exp(coefficients).

`upper95` a vector of the upper bounds of the 95% confidence interval for exp(coefficients).

`loglik` a vector holding the loglikelihood and null loglikelihood.

n the number of observations in the data.

nevent the number of events used in the fit.

iter the number of iterations of the cox algorithm before convergence.

df the degrees of freedom.

score a vector contining the score test statistic and its p-value.

method "efron". The method used for the cox algorithm.

lrt a vector contaiing the likelihood ratio test statistic and its p-value.

rsquare a vector containing an r-square value for the fit and its p-value.

wald.test a vector containg the Wald test statistic and its p-value.

concordance a vector containing the number of events which are concordant, discordant, tied.risk, tied.time. Also contains the concordance statistic and its standard error. Calculated using the survival package, if installed. If not installed, all values are NA.

survival a matrix of values used to compute the surivival curve.

strata a data.frame of the strata used in the computation.

See Also

survfitDistributed, AnalysisCenter.2Party, AnalysisCenter.3Party, AnalysisCenter.KParty

---

vdralinear *Vertical Distributed Linear Regression Results Object*

Description

This class of object is returned by the two party, three party, and K-party distributed regression analysis programs when "linear" regression is specified. Objects of this class have methods for the functions print and summary.

Arguments

The following components must be included in a legitimate vdralinear object.

logical value. If FALSE, then there was an error processing the data. if TRUE, there were no errors.

failed converged logical value. If TRUE, the regression converged. If FALSE, it did not.

party a vector which indicates the party from which each covariate came.

coefficients the vector of coefficients. If the model is over-determined, there will be NA values in the vector corresponding to the redudant columns model matrix.

tvals the t-values of the coefficietns.

secoef the vector of the standard error of the coefficients.

pvals the p-values of the coefficients.

`sse` sum of squared errors.

`rstderr` residual standard error.

`rsquare` r squared.

`adjrsquare` adjusted r squared.

`Fstat` the F-statistic for the linear regression.

`Fpval` the p-value of the F-statistic for the linear regression.

`df1` The numerator degrees of freedom for the F-statistic.

`df2` The denominator degrees of freedom for the F-statistic.

`n` the number of observations in the data.

`xtx` a matrix of the transpose of the covariates times the covarites. Used by `differentModel`.

`xty` a matrix of the transpose of the covarites times the response. Used by `differentModel`.

`yty` sum of squares of the reponse. Used by `differentModel`.

`meansy` the mean of the response. Used by `differentModel`.

`means` the mean of each covaraite. Used by `differentModel`.

See Also

`differentModel`, `AnalysisCenter.2Party`, `AnalysisCenter.3Party`, `AnalysisCenter.KParty`

`vdralogistic` *Vertical Distributed Logistic Regression Results Object*

Description

This class of object is returned by the two party, three party, and K-party distributed regression analysis programs when "logistic" regression is specified. Objects of this class have methods for the functions `print` and `summary`.

Arguments

The following components must be included in a legitimate `vdralogistic` object.

logical value. If `FALSE`, then there was an error processing the data. if `TRUE`, there were no errors.

`failed` `converged` logical value. If `TRUE`, the regression converged. If `FALSE`, it did not.

`party` a vector which indicates the party from which each covariate came.

`coefficients` the vector of coefficients. If the model is over-determined, there will be `NA` values in the vector corresponding to the redudant columns model matrix.

`secoef` the vector of the standard error of the coefficients.

`tvals` the t-values of the coefficietns.

`pvals` the p-values of the coefficients.

n the number of observations in the data.

nulldev the null deviance of the fit.

resdev the residual deviance of the fit.

aic the AIC of the fit.

bic the BIC of the fit.

nulldev_df the degrees of freedom for the null deviance.

resdev_df the degrees of freedome for the residual deviance.

hoslem the Hosmer Lemshow Test statistics.

ROC a list containing the coordinates for an ROC curve.

iter the number of iterations of the cox algorithm before convergence.

Y a matrix of the response. Only returned to the party which holds the response.

FinalFitted a matrix of final fitted values of the regression. Only returned to the party which holds the response.

See Also

HoslemTest, RocTest, AnalysisCenter.2Party, AnalysisCenter.3Party, AnalysisCenter.KParty

vdra_data *Simulated data from a weight loss study.*

Description

Simulated data based on a weight loss study performed by Harvard School of Medicine. The original study was looking for comorbidities to predict weight loss / weight gain in obese subjects. The various outcomes in this simulated data set don't really correlate to each other. The purpose of this simulated data set is to illustrate the funcationality of the package, not to draw valid statistical inferences.

Usage

```
vdra_data
```

Format

Change_BMI: Continuous response used for linear regression.

WtLost: Binary response used for logistic regression.

Time: Used for time to event in Cox regression. Continuous. Ranges from 1 to 459.

Status: Used for censoring in Cox regression. A binary categroical variable.

Exposure: A binary categorical variable.

Age: A continuous variable ranging from 3 to 80.

ComorbidScore: A ordinal variable with 11 variables: 0 to 10.

NumRx: A ordinal varible with 15 levels: -2 to 12.

BMI_pre: A continuous variable ranging from 35.01 to 92.79.

Race: A factor with 6 levels: "Race 0" to "Race 5".

Sex: A binary factor with 2 levels: M and F.

vdra_fit *Return values from the various distributed regression functions in the* vdra *package.*

Description

The objects vdra_fit_cox_A, vdra_fit_linear_A, and vdra_logistic_A are example fitted models that are obtained by the data partner which holds the response variable(s). The objects vdra_fit_cox_B, vdra_fit_linear_B, and vdra_logistic_B are example fitted models that are obtained by the data partner which does not hold the response variable(s). These are provided so the user may see what the summary and print outputs look like before trying to run the full vertical distributed regression. They also allow the user to experiement with the functions differentModel (linear regression); RocTest and HoslemTest (Logistic Regression); and survfitDistributed and plot.survfitDistributed (Cox Regression).

Usage

```
vdra_fit_cox_A
vdra_fit_cox_B
vdra_fit_linear_A
vdra_fit_linear_B
vdra_fit_logistic_A
vdra_fit_logistic_B
```

Index