Package 'predcomps'

June 10, 2014

Title Average Predictive Comparisons

Description An implementation of Gelman/Pardoe's average predictive comparisons

Version 0.1

Author David Chudzicki

Maintainer David Chudzicki <dchudz@gmail.com>

Depends R (>= 3.0.2), ggplot2, gridExtra, dplyr, reshape2, assertthat

License MIT

R topics documented:

ComputeApcFromPairs 1
GetComparisonDFFromPairs 2
GetPairs 2
GetPredCompsDF 3
GetSingleInputApcs 4
GetSingleInputPredComps 4
PlotPairCumulativeWeights 5
PlotPredCompsDF 6

Index 7

`ComputeApcFromPairs` *ComputeApcFromPairs*

Description

ComputeApcFromPairs

Usage

```
ComputeApcFromPairs(predictionFunction, pairs, u, v, absolute = FALSE,
  impact = FALSE)
```

GetComparisonDFFromPairs *GetComparisonDFFromPairs*

Description

(abstracted this into a separate function from GetApc so we can more easily do things like GetApcWithAbsolute)

Usage

```
GetComparisonDFFromPairs(predictionFunction, pairs, u, v)
```

GetPairs *GetPairs*

Description

Form all pairs of rows in X and compute Mahalanobis distances based on v.

Usage

```
GetPairs(X, u, v, numForTransitionStart = NULL, numForTransitionEnd = NULL,
  onlyIncludeNearestN = NULL, mahalanobisConstantTerm = 1)
```

Arguments

X	data frame
u	input of interest
v	other inputs
mahalanobisConstantTerm	Weights are (1 / (mahalanobisConstantTerm + Mahalanobis distance))
numForTransitionStart	number of rows to use as the start points of transitions (defaulting to 'NULL', we use all rows)
numForTransitionEnd	number of rows to use as potential end points of transitions (defaulting to 'NULL', we use all rows)
onlyIncludeNearestN	for each transition start, we only include as transition end points the nearest 'onlyIncludeNearestN' rows (defaulting to 'NULL', we use all rows)

Details

To help with computational constraints, you have the option to not form pairs between all rows of X but instead of specify a certain number (numForTransitionStart) to randomly be selected as rows from which transitions start, and another number (numForTransitionEnd) to be randomly selected as where transitions end. We then form all pairs between transition-start rows and transition-end rows.

In order to get a smaller data frame for later manipulations (and maybe just because it's a good idea), you can also specify onlyIncludeNearestN, in which case we return only the nearest onlyIncludeNearestN transition ends for each transition start (instead of all pairs).

Value

a data frame with the inputs v from the first of each pair, u from each half (with ".B" appended to the second), and the Mahalanobis distances between the pairs.

Examples

```
v <- rnorm(100)
u <- v + 0.3*rnorm(100)
qplot(v,u)
X = data.frame(v=v,u=u)
pairsDF <- GetPairs(X, "v", "u")
pairsDFRow1 <- subset(pairsDF, OriginalRowNumber==1)
# When we subset to one "original row number", all of the vs are the same:
print(pairsDFRow1$v)
# ... and us corresponding to closer v.B (the v in the second element of the pair) have higher weight:
qplot(u.B, Weight, data=pairsDFRow1)
```

`GetPredCompsDF` *GetApcDF*

Description

makes average predictive comparison for all specified inputs

Usage

```
GetPredCompsDF(model, df, inputVars = NULL, ...)
```

Arguments

`model`	Either a function (from a data frame to vector of predictions) or a model we know how to deal with (lm, glm)
`df`	data frame with data
`inputVars`	inputs to the model
`...`	extra parguments passed to GetPairs used to control Weight function

Examples

```
n <- 200
x1 <- runif(n = n, min = 0, max = 1)
x2 <- runif(n = n, min = 0, max = 1)
x3 <- runif(n = n, min = 0, max = 10)
y <- 2 * x1 + (-2) * x2 + 1 * x3 + rnorm(n, sd = 0.1)
df <- data.frame(x1, x2, x3, y)
fittedLm <- lm(y ~ ., data = df)
apcDF <- GetPredCompsDF(fittedLm, df = df)
apcDF
```

GetSingleInputApcs *GetSingleInputApcs*

Description

makes predictive comparison summaries (both per unit input and impact, both absolute and signed) by forming an data frame of pairs with appropriate weights and then calling 'ComputeApcFromPairs'. Only works fore continuous inputs right now

Usage

```
GetSingleInputApcs(predictionFunction, X, u, v, ...)
```

Arguments

predictionFunction	this could be a function (which takes data frame and makes returns a vector of predictions) or an object of class 'lm', 'glm', or 'randomForest'
X	a data frame with all inputs
u	a string naming the input of interest
v	a string naming the other inputs
...	other arguments to be passed to 'GetPairs'

Value

a list with: signed (the usual Apc) and absolute (Apc applied to the absolute value of the differences)

Examples

```
n <- 200
x1 <- runif(n = n, min = 0, max = 1)
x2 <- runif(n = n, min = 0, max = 1)
x3 <- runif(n = n, min = 0, max = 10)
y <- 2 * x1 + (-2) * x2 + 1 * x3 + rnorm(n, sd = 0.1)
df <- data.frame(x1, x2, x3, y)
fittedLm <- lm(y ~ ., data = df)
fittedLm
GetSingleInputApcs(fittedLm, df, "x2", c("x1", "x3"))
```

GetSingleInputPredComps *GetSingleInputPredComps*

Description

makes predictive comparison summaries (APC and impact, absolute and signed) by forming an data frame of pairs with appropriate weights and then calling 'ComputeApcFromPairs'. Only works fore continuous inputs right now

Usage

```
GetSingleInputPredComps(predictionFunction, X, u, v, ...)
```

Arguments

predictionFunction	this could be a function (which takes data frame and makes returns a vector of predictions) or an object of class 'lm', 'glm', or 'randomForest'
X	a data frame with all inputs
u	a string naming the input of interest
v	a string naming the other inputs
...	other arguments to be passed to 'GetPairs'

Value

a list with: signed (the usual Apc) and absolute (Apc applied to the absolute value of the differences)

PlotPairCumulativeWeights

PlotPairCumulativeWeights

Description

For a sample of transition start rows, we plot rank of transition end (by increasing weight) vs. cumulative weight. This gives a sense of how much weight is going into the nearest points vs. further ones.

Usage

```
PlotPairCumulativeWeights(pairs, numOriginalRowNumbersToPlot = 20)
```

Examples

```
v <- rnorm(100)
u <- v + 0.3*rnorm(100)
X = data.frame(v=v,u=u)
pairsDF <- GetPairs(X, "v", "u")
pairsDFRow1 <- subset(pairsDF, OriginalRowNumber==1)
# For most original rows, we get 75% of the weight in 50% of the pairs:
PlotPairCumulativeWeights(pairsDF)
```

PlotPredCompsDF *PlotApcDF*

Description

plots the output of GetApcDF – this is my preferred display for now

Usage

```
PlotPredCompsDF(apcDF, variant = "Impact")
```

Arguments

apcDF	the output of GetApcDF

Examples

```
n <- 200
x1 <- runif(n = n, min = 0, max = 1)
x2 <- runif(n = n, min = 0, max = 1)
x3 <- runif(n = n, min = 0, max = 10)
y <- 2 * x1 + (-2) * x2 + 1 * x3 + rnorm(n, sd = 0.1)
df <- data.frame(x1, x2, x3, y)
fittedLm <- lm(y ~ ., data = df)
apcDF <- GetPredCompsDF(fittedLm, df = df)
PlotPredCompsDF(apcDF, variant = "PerUnitInput") + theme_gray(base_size = 18)
```

Index

ComputeApcFromPairs, 1

GetComparisonDFFromPairs, 2
GetPairs, 2
GetPredCompsDF, 3
GetSingleInputApcs, 4
GetSingleInputPredComps, 4

PlotPairCumulativeWeights, 5
PlotPredCompsDF, 6