Package 'predictMe'

May 9, 2026

Type Package

Title Visualize Individual Prediction Performance

Version 0.1

Date/Publication 2022-05-24 09:40:02 UTC

Author Marcel Miché

Maintainer Marcel Miché <marcel.miche.predictme@gmail.com>

Description Enables researchers to visualize the prediction performance of any algorithm on the individual level (or close to it), given that the predicted outcome is either binary or continuous. Visual results are instantly comprehensible.

License MIT + file LICENSE

Repository CRAN

URL https://github.com/mmiche/predictMe

Encoding UTF-8

Imports Rdpack, ggplot2, reshape2

RdMacros Rdpack

RoxygenNote 7.1.2

Depends R (>= 3.3.0)

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Contents

`binBinary` *Categorization of predicted probabilities and the corresponding mean number of events for each category.*

Description

Predicted probabilities are categorized as bins, depending on the selected 'binWidth', and corresponding mean outcome per bin is computed.

Usage

```
binBinary(x = NULL, measColumn = NULL, binWidth = 20)
```

Arguments

`x`	A data.frame with exactly two columns, one of the columns must be the measured outcome, the other column must be the predicted outcome values, as returned by some algorithm.
`measColumn`	A single integer number that denotes which of the two columns of function argument 'x' contains the measured outcome.
`binWidth`	A single integer value greater than 0 and less than 100, which separates 100 into equal bins, e.g., 20 (100/20 = 5 equal bins).

Details

Predicted values (probability in percent) less than 0 or greater than 100 are replaced by 0 and 100, respectively.

Beware: Since binning continuous values always introduces noise, some of the differences in column 7 (bin differences) require explicit attention. When the outcome is binary, the binning of the predicted probabilities (fitted values) will also automatically introduce noise in column 5, since the mean number of measured events depends on the width and on the exact borders of the bins (see package vignette, headline **Bin noise**).

Value

a list with two data.frames and one vector. Each data.frame has 7 columns:

1. xTrans Data set, with columns 1 and 2 being categorized, according to the user's selected bin width. Each in percent, column 3 displays the observed frequencies per bin, whereas column 4 display the predicted probabilities (fitted values) per bin. Column 5 shows the difference between values in column 3 and column 4. Column 6 shows the unique individual identifiers. Column 7 shows the differences in terms of bins. See **Details**.
2. xTrans2 Same as xTrans, only that original or transformed values less than 0 or greater than 100 have not been replaced with 0 or 100, respectively.
3. idxExceed logical vector. TRUE shows the row of xTrans or xTrans2 where values were either less than 0 or greater than 100.

Author(s)

Marcel Miché

Examples

```
# Simulate data set with binary outcome
dfBinary <- quickSim(type="binary")
# Logistic regression, used as algorithm to predict the response variable
# (estimated probability of outcome being present).
glmRes <- glm(y~x1+x2,data=dfBinary,family="binomial")
# Extract measured outcome and the predicted probability (fitted values)
# from the logistic regression output, put both in a data.frame.
glmDf <- data.frame(measOutcome=dfBinary$y,
                    fitted=glmRes$fitted.values)
# Apply function binBinary, generate 5 equal bins (probabilities in
# percent, bin width 20, yields 5 bins).
x100b <- binBinary(x=glmDf, measColumn = 1, binWidth = 20)
```

`binContinuous` *Categorization of measured and predicted outcome values.*

Description

Measured and predicted continuous outcome values (transformed to range between 0 and 100) are categorized as bins, depending on the selected 'binWidth'.

Usage

```
binContinuous(
  x = NULL,
  measColumn = NULL,
  binWidth = 20,
  computeRange = TRUE,
  range_x = c(0, 0)
)
```

Arguments

`x`	A data.frame with exactly two columns, one of the columns must be the measured outcome, the other column must be the predicted outcome values, as returned by some algorithm.
`measColumn`	A single integer number that denotes which of the two columns of function argument 'x' contains the measured outcome.
`binWidth`	A single integer value greater than 0 and less than 100, which separates 100 into equal bins, e.g., 20 (100/20 = 5 equal bins).
`computeRange`	Logical value, defaults to TRUE, meaning that the range of the column with the measured outcome values will be computed. Else set this argument to FALSE (see **Details**).

range_x A vector with the minimum and maximum possible value of the continuous outcome scale (see **Details**).

Details

Regarding function arguments 'computeRange' and 'range_x': If either the minimum or maximum possible value of the outcome scale has not occurred, e.g., none of the participants selected the maximum possible answer option, then the user must pass the possible range of outcome values to this function, using the function argument 'range_x', e.g., range_x = c(1, 5), if the original outcome scale ranged from 1 to 5.

Regarding function output 'xTrans' (see **Value**): Predicted values less than 0 or greater than 100 are replaced by 0 and 100, respectively.

Beware: The differences in column 5 are as accurate (no information loss) as if the original measured and predicted outcome values were subtracted from one another. However, since binning continuous values always introduces noise, some of the differences in column 7 (bin differences) require explicit attention (see package vignette, headline **Bin noise**).

Value

a list with two data.frames and one vector. Each data.frame has 7 columns:

1. xTrans Data set, with columns 1 and 2 being categorized, according to the user's selected bin width. Column 3 displays the observed outcome values, whereas column 4 displays the predicted outocme values (fitted values), both transformed to range between 0 and 100. Column 5 shows the difference between values in column 3 and column 4. Column 6 shows the unique individual identifiers. Column 7 shows the differences in terms of bins. See **Details**.
2. xTrans2 Same as xTrans, only that original or transformed values less than 0 or greater than 100 have not been replaced with 0 or 100, respectively.
3. idxExceed logical vector. TRUE shows the row of xTrans or xTrans2 where values were either less than 0 or greater than 100.

Author(s)

Marcel Miché

Examples

```
# Simulate data set with continuous outcome (use all default values)
dfContinuous <- quickSim()
# Use multiple linear regression as algorithm to predict the outcome.
lmRes <- lm(y~x1+x2,data=dfContinuous)
# Extract measured outcome and the predicted outcome (fitted values)
# from the regression output, put both in a data.frame.
lmDf <- data.frame(measOutcome=dfContinuous$y,
                   fitted=lmRes$fitted.values)
# Apply function binContinuous, generate 5 equal bins (transformed
# outcome 0-100, bin width = 20, yields 5 bins).
x100c <- binContinuous(x=lmDf, measColumn = 1, binWidth = 20)
```

get2by2 *Return of five common results, based on the 2x2 cross-table (a.k.a. confusion matrix).*

Description

Upon receiving two binary variables (only 0 and 1 permitted) of equal length, return sensitivity, specificity, positive predictive value, negative predictive value, and the base rate of the outcome.

Usage

```
get2by2(xr, measColumn = NULL, print2by2 = FALSE)
```

Arguments

xr	A data.frame with exactly two columns, one of the columns must be the binary measured outcome, the other column must be the binary predicted outcome, based on some algorithm's predictions (see **Details**).
measColumn	A single integer number that denotes which of the two columns of function argument 'x' contains the measured outcome.
print2by2	Logical value, defaults to FALSE. If set TRUE, two 2by2 matrices will be printed with explanations of what they display.

Details

The r in the argument 'xr' stands for response, meaning that the predicted probabilities must have been transformed to a binary outcome, usually by using the default cutoff of 0.5; although it may also be any other cutoff between 0 and 1.

If you wish to additionally print the 2x2 matrix, set the argument 'print2by2' TRUE (default: FALSE).

Value

a list with five elements (seven, if argument print2by2 is set TRUE; see **Details**):

1. sens Sensitivity (a.k.a.: Recall, True Positive Rate).
2. spec Specificity (a.k.a.: True Negative Rate).
3. ppv Positive Predictive Value (a.k.a.: Precision).
4. npv Negative Predictive Value.
5. br Base rate of the outcome (mean outcome occurrence in the sample).
6. tbl1 2x2 matrix. Test-theoretic perspective: Specificity in top left cell, sensitivity in bottom right cell.
7. tbl2 2x2 matrix. Test-practical perspective (apply test in the real world): Negative predictive value (npv) in top left cell, positive predictive value (ppv) in bottom right cell.

Author(s)

Marcel Miché

Examples

```
# Simulate data set with binary outcome
dfBinary <- quickSim(type="binary")
# Logistic regression, used as algorithm to predict the response variable
# (response = estimated probability of outcome being present).
glmRes <- glm(y~x1+x2,data=dfBinary,family="binomial")
# Extract measured outcome and the predicted probability (fitted values)
# from the logistic regression output, put both in a data.frame.
glmDf <- data.frame(measOutcome=dfBinary$y,
                    fitted=glmRes$fitted.values)
# binary outcome, based on the default probability threshold of 0.5.
get2by2Df <- data.frame(
    measuredOutcome=glmDf$measOutcome,
    predictedOutcome=ifelse(glmDf$fitted<.5, 0, 1))
# Demand 2x2 matrix to be part of the resulting list.
my2x2 <- get2by2(xr=get2by2Df, measColumn=1, print2by2 = TRUE)
# Display both 2x2 matrices
# tbl1: Theoretical perspective, with specificity in top left cell,
# sensitivity in bottom right cell.
my2x2$tbl1
# tbl2: Practical perspective, with negative predictive value (npv)
# in top left cell, positive predictive value (ppv) in bottom right
# cell.
my2x2$tbl2
```

---

makeDiffPlot — *Plot individual differences between measured and predicted outcome values.*

Description

Plot the differences between measured and predicted outcome for all individuals.

Usage

```
makeDiffPlot(xd = NULL, idCol = NULL)
```

Arguments

xd	A data.frame with exactly two columns, one of the columns must be the identifier of all individuals, the other column must be the differences between the measured and the predicted outcome values.
idCol	A single integer that denotes which of the columns of the data.frame contains the identifier of the individuals.

Details

The d in 'xd' stands for differences, meaning that the column of interest contain the differences between the measured and the predicted outcome values, logically requiring the column that identifies the individuals.

Irrespective of whether the original outcome was continuous or binary, outcome values always range between 0 and 100. For instance, for a binary outcome the 'probabilities' are represented as percentage.

Use the column diff (from function `binContinuous`) or diffPerc (from function `binBinary`) and column xAxisIds, both columns being part of both data.frames that are returned by the two mentioned functions.

Value

a list with the plot that shows the differences between the measured and predicted outcome for all individuals. See **Details**.

Author(s)

Marcel Miché

References

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.

Examples

```
# Simulate data set with continuous outcome (use all default values)
dfContinuous <- quickSim()
# Use multiple linear regression as algorithm to predict the outcome.
lmRes <- lm(y~x1+x2,data=dfContinuous)
# Extract measured outcome and the predicted outcome (fitted values)
# from the regression output, put both in a data.frame.
lmDf <- data.frame(measOutcome=dfContinuous$y,
                   fitted=lmRes$fitted.values)
# Apply function binContinuous.
x100c <- binContinuous(x=lmDf, measColumn = 1, binWidth = 20)
# Apply function makeDiffPlot, using columns 5 and 6 from x100c[["xTrans"]]
# The second of columns 5 and 6 contains the identifiers of the individuals.
dp <- makeDiffPlot(x100c[["xTrans"]][,5:6], idCol = 2)
# dp is the plot that shows the individual differences.
# makeDiffPlot works the same way if binBinary had be used instead of
# binContinuous.
```

`makeDiffPlotColor` *Same as function makeDiffPlot, but add information by using colors.*

Description

Does the same as makeDiffPlot. However, additionally the difference between bins are added by using colors.

Usage

```
makeDiffPlotColor(xdc = NULL, idCol = NULL, colorCol = NULL)
```

Arguments

`xdc`	A data.frame with exactly three columns, one of the columns must be the identifier of all individuals, another column must be the differences between the measured and the predicted outcome values, and the third column must be the absolute differences between the bins of the measured and the predicted outcome.
`idCol`	A single integer that denotes which of the columns of the data.frame contains the identifier of the individuals.
`colorCol`	A single integer that denotes which of the columns of the data.frame contains the absolute differences between the bins of the measured and the predicted outcome.

Details

Recommendation: Use some of the ggplot2 options to enhance the plot, e.g., using the function facet_wrap (for an example, see vignette **predictMe Why and how to?, headline 'Function makeDiffPlotColor (to go into more detail)'**).

Value

a list with the plot that shows the differences between the measured and predicted outcome for all individuals, using colorized points that express the differences in terms of number of bins.

Author(s)

Marcel Miché

References

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.

Examples

```
# Simulate data set with continuous outcome (use all default values)
dfContinuous <- quickSim()
# Use multiple linear regression as algorithm to predict the outcome.
lmRes <- lm(y~x1+x2,data=dfContinuous)
# Extract measured outcome and the predicted outcome (fitted values)
# from the regression output, put both in a data.frame.
lmDf <- data.frame(measOutcome=dfContinuous$y,
                   fitted=lmRes$fitted.values)
# Apply function binContinuous.
x100c <- binContinuous(x=lmDf, measColumn = 1, binWidth = 20)
# Apply function makeDiffPlotColor, using columns 5 and 6 from x100c[["xTrans"]]
# The second of columns 5 and 6 contains the identifiers of the individuals.
dpc <- makeDiffPlotColor(x100c[["xTrans"]][,5:7], idCol = 2, colorCol=3)
# dpc is the plot that shows the individual differences, in colorized form.
# makeDiffPlotColor works the same way if binBinary had be used instead of
# binContinuous.
```

`makeTablePlot` *Tabularize the essential result of the predictMe package.*

Description

Provides the essential result of the predictMe package, three tables, and, optionally, two plots.

Usage

```
makeTablePlot(xc = NULL, measColumn = NULL, plot = FALSE, plotCellRes = TRUE)
```

Arguments

`xc`	A data.frame with exactly two columns, one of the columns must be the categorized measured outcome, the other column must be the categorized predicted outcome.
`measColumn`	A single integer number that denotes which of the two columns of function argument 'x' contains the measured outcome.
`plot`	Logical value, defaults to FALSE. If set TRUE, two complementary plots will be part of the list that this function returns.
`plotCellRes`	Logical value, defaults to TRUE (is ignored if function argument 'plot' is set FALSE). If set FALSE, the heatmap is returned without frequency results in the cellls.

Details

The c in 'xc' stands for categorized, meaning that the outcome values are expected to have been categorized, so that both columns contain the exact same categories, and are of the class factor.

Columns 1 and 2 of the output 'xTrans' from function binBinary and from function binContinuous provide the expected input of this makeTablePlot function (see **Examples**).

The returned list will contain 7 items, if function argument 'plot' is set TRUE, if FALSE, it will return the first 5 items (see **Values**).

Value

a list with five or seven items (see **Details**):

1. totalCountTable A table with the total counts.
2. rowSumTable A table with proportions that sum up to 1, per row (summing across columns).
3. colSumTable A table with proportions that sum up to 1, per column (summing across rows).
4. rowSumTable_melt The rowSumTable, reformated by the function melt of the reshape2 package.
5. colSumTable_melt The colSumTable, reformated by the function melt of the reshape2 package.
6. rowSumTable_plot The rowSumTable_melt data, plotted by the function ggplot of the ggplot2 package.
7. colSumTable_plot The colSumTable_melt data, plotted by the function ggplot of the ggplot2 package.

Author(s)

Marcel Miché

References

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.

Wickham H (2007). "Reshaping Data with the reshape Package." *Journal of Statistical Software*, **21**(12), 1–20. https://www.jstatsoft.org/v21/i12/.

Examples

```
# Simulate data set with continuous outcome (use all default values)
dfContinuous <- quickSim()
# Use multiple linear regression as algorithm to predict the outcome.
lmRes <- lm(y~x1+x2,data=dfContinuous)
# Extract measured outcome and the predicted outcome (fitted values)
# from the regression output, put both in a data.frame.
lmDf <- data.frame(measOutcome=dfContinuous$y,
                   fitted=lmRes$fitted.values)
# Apply function binBinary
x100c <- binContinuous(x=lmDf, measColumn = 1, binWidth = 20)
# Apply function makeDiffPlot, using columns 1 and 2 from x100c[["xTrans"]]
# The first of columns 1 and 2 contains the measured outcome values.
tp <- makeTablePlot(x100c[["xTrans"]][,1:2], measColumn = 1, plot = TRUE)
# tp is a list with 7 items, items 6 and 7 are the plots that represent
```

```
# the numeric information of items 2 and 3 (and 4 and 5, which merely have
# a different format).
# Display item 6 (plot no.1). Perfect performance if the diagonal cells all
# contain the value 1.
tp$rowSumTable_plot
```

`predictMe` *Documentation of this predictMe package.*

Description

This package enables researchers to visualize the prediction performance of an algorithm, either on the individual level or approximating this level. The visualized result is instantly comprehensible, only depending on being familiar with the concept of 'difference' (yes or no) and the related concept of 'distance' (if difference yes, how large is it). The predictMe package can be applied to the output of any algorithm, given that the measured (and therefore also the predicted) outcome is either continuous or binary.

Importantly, predictMe only takes the two relevant columns, that is, the measured outcome and the predicted outcome. The values in the two columns will be transformed, to range between 0 and 100 (see **Details** in the documentation of functions `binContinuous` or `binBinary`), finally returning the transformed values as bins. The user can decide how small the bins shall be, using the function argument `binWidth`. The smaller the bins, the more bins will be produced, which means the more will the visualized prediction performance approximate the individual level (see function `makeTablePlot`). Differences between measured and predicted outcome on the individual level can also be visualized (see function `makeDiffPlot`).

The predictMe package provides the transformed data (see functions `binContinuous` or `binBinary`) and the visualization (see functions functions `makeTablePlot` or `makeDiffPlot`). Nevertheless, the user is free to experiment with visualizing the results, which are returned in different formats (see vignette of predictMe for a few examples of how the data may be visualized).

The predictMe package depends on two packages: ggplot2 (Wickham, 2016) for providing suggested visualizations, and reshape (Wickham, 2007) for providing the results in a format that is readily compatible with ggplot2 experimentation. The conventional format may also be used, which is compatible with base R plotting functions.

Importantly, the predictMe package was developed with the aim of extreme ease of both, use and comprehension of the output. This, I hope, may make this package powerful, in terms of being actually used. The first four out of the six references (see below) contain bits of the intended usefulness of this package (see **Note** below). The actual idea for this package came while trying to achieve something specific, using the ggplot2 package (Wickham, 2016).

Note

These are the bits in the first four references below, that pertain to the intended usefulness of the predictMe package:

Altman and Royston (2000) provide this introductory quote (by Alvan Feinstein): 'Validation is one of those words ... that is constantly used and seldom defined.' This surely is strange in the vicinity of developing prognostic models, especially in the machine learning age, unless the statement was

meant as a joke (which appears not to be the case), or is no longer valid in 2022 (which might be true or false, who knows).

Bickel and Lehman (2012): If two different people, who both provided the exact same relevant input data for an algorithm, with which a risk percentage of some adverse outcome is computed, say complications due to an operation, they will receive the exact same risk estimation, e.g., 1 percent. However, both individuals may understand this number very differently, depending on their individual inclinations in general and/or at that moment. Therefore, one of the two individuals may simply say ok to the operation, while the other individual may ask for more detailed information. This more detailed information can be computed with the predictMe functions `binContinuous` or `binBinary`, and visualized with the predictMe function `makeDiffPlot`. The differences can be colorized with the function `makeDiffPlotColor`, which may help in seeing how far away an individual's prediction is from being perfect (no difference between measured and predicted outcome). Even though perfect prediction is practically utopian, it still might be relevant to the individual whether his or her predictions are closer to this utopian reference, compared to the predictions of all individuals, who have been used to develop the model that underlies this algorithm's individual predictions.

Assel et al. (2017): In line with Altman and Royston (2000), Assel et al. (2017) recommend to clarify whether a published prediction model is at an early stage of development or whether it approaches an advanced stage, maybe even suggesting implementation in the real world. In the latter case, much stricter performance criteria must be met, compared to the former case (early stage of model development), due to actual individuals of the real world being the supposed beneficiaries of the algorithmic decision support.

Offord and Kraemer (2000): In line with Altman and Royston (2000), Offord and Kraemer (2000) emphasize that a risk factor must in any case demonstrate that it can accurately split a group into individuals with low risk and individuals with high risk. In the real world, this requires much more than meeting statistical significance criteria or meeting other (similarly thin) model fit criteria. Again, if model development was at an early stage (see Assel et al., 2017), such criteria may suffice. However, at later stages, real world criteria must be met, that is, real-world relevant results must either replace or at least complement the commonly reported results of prediction performance.

Conclusion: The predictMe package provides the opportunity to provide some real-world relevant 'results', if visualized individual prediction performance may be considered as 'results'.

References

Altman DG, Royston P (2000). "What do we mean by validating a prognostic model?" *Statistics in medicine*, **19**(4), 453–473.

Assel M, Sjoberg DD, Vickers AJ (2017). "The Brier score does not evaluate the clinical utility of diagnostic tests or prediction models." *Diagnostic and prognostic research*, **1**(1), 1–7.

Bickel PJ, Lehmann EL (2012). "Frequentist interpretation of probability." In *Selected Works of EL Lehmann*, 1083–1085. Springer.

Offord DR, Kraemer HC (2000). "Risk factors and prevention." *Evidence-Based Mental Health*, **3**(3), 70–71.

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.

Wickham H (2007). "Reshaping Data with the reshape Package." *Journal of Statistical Software*, **21**(12), 1–20. https://www.jstatsoft.org/v21/i12/.

quickSim *Quick simulation of a data.frame for demonstration purposes.*

Description

Quick simulation of a data.frame, either with a continuous or with a binary outcome. This is merely to enable showcasing the main purpose of the predictMe package.

Usage

```
quickSim(
  n = 1000,
  intercept = 1,
  coefs = c(2, 3),
  errMean = 30,
  errSD = 3,
  seed = 1,
  type = "continuous"
)
```

Arguments

n	Sample size, defaults to 1000.
intercept	Intercept of a simulated model output, defaults to 1.
coefs	Regression coefficients of a simulated model output, defaults to two predictors with coefficients 2 and 3, respectively.
errMean	Mean prediction error, present in the simulated data, defaults to 30 (will be ignored, if function argument 'type' (see below) is set to 'binary').
errSD	Standard deviation of the error, present in the simulated data, defaults to 3 (will be ignored, if function argument 'type' (see below) is set to 'binary').
seed	A single integer value. Setting a seed ensures reproducibility of a once simulated data set.
type	A single character value, either 'continuous' or 'binary', depending on what scale the simulated outcome shall have.

Details

The returned simulated data set will have as many predictors, as the user entered regression coefficients to the function argument 'coefs'. For instance, coefs = c(.5, -2, -.9) will result in three predictors x1, x2, and x3 in the returned data set.

The simulated data set is merely serving the need to provide the main functions of this package with the data they require (demonstration purpose; several simulation packages exist in R).

Value

simDf A data.frame with one outcome column y, and as many predictor columns (named: x1, x2, ...) as the user selected (default: 2). See **Details**.

Author(s)

Marcel Miché

References

Simulation code inside this function was largely taken from Stéphane Laurent's answer on StackExchange.

Examples

```
# Simulate data set with continuous outcome (use all default values)
dfContinuous <- quickSim()
# Simulate data set with continuous outcome (set sample size to 149)
dfContinuous <- quickSim(n = 149)
nrow(dfContinuous) # 149
# Simulate data set with binary outcome (set sample size to 100, and
# coefficients to 3, 1, and -2.5)
```

Index