Package ‘recalibratiNN’

April 15, 2024

Title Quantile Recalibration for Regression Models

Version 0.2.0

Description Enables the diagnostics and enhancement of calibration of regression models. It offers both global and local visualization tools to calibration diagnostics and provides one recalibration method : Torres R, Nott DJ, Sisson SA, Rodrigues T, Reis JG, Rodrigues GS (2024) <doi:10.48550/arXiv.2403.05756>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

URL `https://github.com/cmusso86/recalibratiNN`, `https://cmusso86.github.io/recalibratiNN/`

BugReports `https://github.com/cmusso86/recalibratiNN/issues`

Imports stats(>= 4.3.0), dplyr(>= 1.0.0), ggplot2 (>= 3.0.0), purrr(>= 1.0.0), RANN(>= 2.0.0), tidyr(>= 1.0.0), tibble(>= 3.0.0), glue (>= 1.0.0), magrittr(>= 2.0.0), Hmisc (>= 5.0.0), Rdpack

RdMacros Rdpack

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Carolina Musso [aut, cre, cph] (<`https://orcid.org/0000-0002-8107-6458`>), Ricardo Torres [aut, cph] (<`https://orcid.org/0009-0000-9100-7125`>), João Reis [aut, cph], Guilherme Rodrigues [aut, ths, cph] (<`https://orcid.org/0000-0003-2009-4844`>)

Maintainer Carolina Musso <`cmusso86@gmail.com`>

Repository CRAN

Date/Publication 2024-04-15 15:50:02 UTC

R topics documented:

gg_CD_global — *Plots Cumulative Distributions of PIT-values for global calibration diagnose.*

Description

ggplot to visualize predicted vs empirical cumulative distributions of PIT-values.

Usage

```
gg_CD_global(pit, ycal, yhat, mse)
```

Arguments

`pit`	vector of global PIT-values
`ycal`	vector of y calibration set
`yhat`	vector of predicted y on calibration set
`mse`	Mean Squared Error from calibration set

Value

a ggplot point graph

Examples

```
n <- 10000
split <- 0.8

# generating heterocedastic data
mu <- function(x1){
10 + 5*x1^2
}

sigma_v <- function(x1){
30*x1
}
```

```
x <- runif(n, 1, 10)
y <- rnorm(n, mu(x), sigma_v(x))

x_train <- x[1:(n*split)]
y_train <- y[1:(n*split)]

x_cal <- x[(n*split+1):n]
y_cal <- y[(n*split+1):n]

model <- lm(y_train ~ x_train)

y_hat <- predict(model, newdata=data.frame(x_train=x_cal))

MSE_cal <- mean((y_hat - y_cal)^2)

pit <- PIT_global( y_cal, y_hat,  MSE_cal)

gg_CD_global(pit,y_cal, y_hat, MSE_cal)
```

`gg_CD_local` *Plots Cumulative Distributions of PIT-values for local calibration diagnose.*

Description

ggplot to visualize predicted vs empirical cumulative distributions of PIT-values locally

Usage

```
gg_CD_local(
  pit_local,
  psz = 0.01,
  abline = "black",
  pal = "Set2",
  facet = FALSE,
  ...
)
```

Arguments

`pit_local`	A data frame obtained from PIT_local_lm
`psz`	double that indicates size of the points that compose the lines. Default is 0.001
`abline`	Color of horizontal line that indicates density 1. Default is"red"
`pal`	Palette name from RColorBrewer. Default is "Set2’
`facet`	logical value in case separate visualization is preferred. Default is F
`...`	Other parameters to pass ggplot

Value

a ggplot graph

Examples

```
n <- 10000
split <- 0.8

mu <- function(x1){
10 + 5*x1^2
}

sigma_v <- function(x1){
30*x1
}


x <- runif(n, 1, 10)
y <- rnorm(n, mu(x), sigma_v(x))

x_train <- x[1:(n*split)]
y_train <- y[1:(n*split)]

x_cal <- x[(n*split+1):n]
y_cal <- y[(n*split+1):n]

model <- lm(y_train ~ x_train)

y_hat <- predict(model, newdata=data.frame(x_train=x_cal))

MSE_cal <- mean((y_hat - y_cal)^2)

pit_local <- PIT_local(xcal = x_cal, ycal=y_cal, yhat=y_hat, mse=MSE_cal)

gg_CD_local(pit_local)
gg_CD_local(pit_local, facet=TRUE)
```

`gg_PIT_global` *Plots Density Distributions of PIT-values for global calibration diagnose.*

Description

A function based on ggplot2 to observe the global the density of PIT-values. For more detailed edition of layers a posteriori, please refer to ggplot2 User Guide.

Usage

```
gg_PIT_global(
  pit,
  type = "density",
  fill = "steelblue4",
  alpha = 0.8,
  print_p = TRUE
)
```

Arguments

`pit`	vector of pit values
`type`	either "density" or "histogram" to change type of graph.
`fill`	The color to fill the density plot. The default is 'stealblue4.
`alpha`	The opacity of the density plot filling. Default is set to 0.8.
`print_p`	Logical value indicating whether or not to print the p-value of ks.test()

Value

a ggplot density graph

Examples

```
n <- 10000
split <- 0.8

# generating heterocedastic data
mu <- function(x1){
10 + 5*x1^2
}

sigma_v <- function(x1){
30*x1
}

x <- runif(n, 1, 10)
y <- rnorm(n, mu(x), sigma_v(x))

x_train <- x[1:(n*split)]
y_train <- y[1:(n*split)]

x_cal <- x[(n*split+1):n]
y_cal <- y[(n*split+1):n]

model <- lm(y_train ~ x_train)

y_hat <- predict(model, newdata=data.frame(x_train=x_cal))

MSE_cal <- mean((y_hat - y_cal)^2)
```

```
pit <- PIT_global(ycal=y_cal, yhat=y_hat, mse=MSE_cal)

gg_PIT_global(pit)
```

gg_PIT_local *Plots Density Distributions of PIT-values for global calibration diagnose.*

Description

A function based on ggplot2 to observe the local the density of PIT-values. To use this function we recommend providing the PIT-values returned by the PIT_local function from this package or an object of equivalent format. Layers can be edited like in http://cran.nexr.com/web/packages/ggpmisc/vignettes/user-guide-4.html.

Usage

```
gg_PIT_local(
  pit_local,
  alpha = 0.4,
  linewidth = 1,
  pal = "Set2",
  facet = FALSE
)
```

Arguments

`pit_local`	A tibble with five column names "part", "y_cal", "y_hat", "pit" and "n".
`alpha`	double 0-1 to indicate transparency of fill. Default is 0.4.
`linewidth`	integer linewidth of density line. Default set to 1.
`pal`	a chosen RBrewer color pallete. Default is "Set2"
`facet`	Logical iforming if the plot should use face_wrap() to separate the different localities.

Value

A ggplot

Examples

```
 n <- 10000
 mu <- function(x1){
  10 + 5*x1^2
  }

sigma_v <- function(x1){
 30*x1
}

x <- runif(n, 2, 20)
y <- rnorm(n, mu(x), sigma_v(x))

x_train <- x[1:(n*0.8)]
y_train <- y[1:(n*0.8)]

x_cal <- x[(n*0.8+1):n]
y_cal <- y[(n*0.8+1):n]

model <- lm(y_train ~ x_train)

y_hat <- predict(model, newdata=data.frame(x_train=x_cal))

MSE_cal <- mean((y_hat - y_cal)^2)

pit_local <- PIT_local(xcal = x_cal, ycal=y_cal, yhat=y_hat, mse=MSE_cal)

gg_PIT_local(pit_local)
gg_PIT_local(pit_local, facet=TRUE)
```

PIT_global *Obtain the PIT-values of a model.*

Description

A function to obtain the (possibly uncalibrated) PIT-values of any fitted model that assumes a normal distribution for the output, such as (but not limited to), a lm() or a neural network that used the Mean Squared Error as the loss function.

Usage

```
PIT_global(ycal, yhat, mse)
```

Arguments

ycal	observations of the recalibration set
yhat	predictions of the recalibration set from the uncalibrated model
mse	Mean Squared Error of validation set.

Value

Vector of PIT-values

Examples

```
n <- 10000
split <- 0.8

# generating heterocedastic data
mu <- function(x1){
10 + 5*x1^2
}

sigma_v <- function(x1){
30*x1
}


x <- runif(n, 1, 10)
y <- rnorm(n, mu(x), sigma_v(x))

x_train <- x[1:(n*split)]
y_train <- y[1:(n*split)]

x_cal <- x[(n*split+1):n]
y_cal <- y[(n*split+1):n]

model <- lm(y_train ~ x_train)

y_hat <- predict(model, newdata=data.frame(x_train=x_cal))

MSE_cal <- mean((y_hat - y_cal)^2)

PIT_global(ycal=y_cal, yhat=y_hat, mse=MSE_cal)
```

`PIT_local` *Obtain local PIT-values from a model*

Description

Return local PIT-values. Centroids for localization is obtained by k-means method from stats package. The vicinity of such centroids are selected though a aproximate k-nearst neighboors method from RANN package.

Usage

```
PIT_local(
  xcal,
  ycal,
  yhat,
  mse,
  clusters = 6,
  p_neighbours = 0.2,
  PIT = PIT_global
)
```

Arguments

`xcal`	features/covariates from calibration set
`ycal`	observations of calibration set
`yhat`	predicted outputs from the calibrations et
`mse`	Mean Squared Error of the model
`clusters`	Number of partitions to create for local calibration. Centroids calculated by k-means method. Default set to 6.
`p_neighbours`	Proportion of xcal to localize neighboors in the KNN method. Default is 0.2.
`PIT`	function to return the PIT-values. Default set to PIT_global() from this package.

Value

A tibble with five containing in the first column containing unique names for the partition, "y_cal", the second column containing the yhat the third the pit-values and the last the number of neighbors in each partition.

Examples

```
n <- 10000
split <- 0.8

mu <- function(x1){
10 + 5*x1^2
}

sigma_v <- function(x1){
 30*x1
}

x <- runif(n, 1, 10)
y <- rnorm(n, mu(x), sigma_v(x))

x_train <- x[1:(n*split)]
y_train <- y[1:(n*split)]

x_cal <- x[(n*split+1):n]
y_cal <- y[(n*split+1):n]

model <- lm(y_train ~ x_train)
```

```
y_hat <- predict(model, newdata=data.frame(x_train=x_cal))

MSE_cal <- mean((y_hat - y_cal)^2)

PIT_local(xcal = x_cal, ycal=y_cal, yhat=y_hat, mse=MSE_cal)
```

---

recalibrate   *Obtain recalibrated samples of the predictive distribution.*

Description

This function currently offers one recalibration technique, based on the method by Torres R. et al. (2024). It offers two approaches (local and global) to obtain samples and the mean of a recalibrated predictive distribution for any regression Gaussian model that used Mean Squared Error (MSE) as the loss function.

Usage

```
recalibrate(
  yhat_new,
  pit_values,
  mse,
  space_cal = NULL,
  space_new = NULL,
  type = c("local", "global"),
  p_neighbours = 0.1,
  epsilon = 0
)
```

Arguments

`yhat_new`	Predicted values of the new (test) set.
`pit_values`	Global Probability Integral Transform (PIT) values calculated on the calibration set.
`mse`	Mean Squared Error of the calibration/validation set. which extremes corresponds to the usual extremes for a 95% confidence interval and the central value corresponds to the median.
`space_cal`	Used in local recalibration. The covariates/features of the calibration/validation set or any representation of those covariates, such as an intermediate layer or an output layer of a neural network.
`space_new`	Used in local recalibration. A new set of covariates or other representation of those covariates, provided they are in the same space as the ones in space_cal.
`type`	Choose between local or global calibration.

`p_neighbours`	Double between (0,1] that represents the proportion of the x_cal is to be used as the number of neighboors for the KNN. If p_neighbours=1 calibration but weighted by distance. Default is set to 0.1.
`epsilon`	Approximation for the K-nearest neighbors (KNN) method. Default is epsilon = 0, which returns the exact distance. This parameter is available when choosing local calibration.

Details

The method implemented here is designed to generate recalibrated samples from regression models that have been fitted using the least-squares method. It's important to note that the least-squared method will only render a probabilistic interpretation if the output to be modeled follows a normal distribution, and that assumption was used to implement this method.

The current available methods, draws inspiration from Approximate Bayesian Computation and the Inverse Transform Theory. The calibration methods can be applied either locally or globally. When tipe="global", the calibration will use a uniform kernel.

Alternatively, one can choose the "local" calibration with a p_neighbours=1. This way, the calibration will use the whole calibration set (that is, globally), but instead of an uniform kernel, it will use a Epanechnikov kernel.

Value

A list containing the calibrated predicted mean/variance along with samples from the recalibrated predictive distribution with its respective weights. Weights are calculated with an Epanechnikov kernel. over the distances obtained from KNN.

References

Torres R, Nott DJ, Sisson SA, Rodrigues T, Reis JG, Rodrigues GS (2024). “Model-Free Local Recalibration of Neural Networks.” *arXiv preprint arXiv:2403.05756*. doi:10.48550/arXiv.2403.05756.

Examples

```
n <- 1000
split <- 0.8

# Auxiliary functions
mu <- function(x1){
10 + 5*x1^2
}

sigma_v <- function(x1){
30*x1
}

# Generating heteroscedastic data.
x <- runif(n, 1, 10)
y <- rnorm(n, mu(x), sigma_v(x))
```

```
# Train set
x_train <- x[1:(n*split)]
y_train <- y[1:(n*split)]

# Calibration/Validation set.
x_cal <- x[(n*split+1):n]
y_cal <- y[(n*split+1):n]

# New observations or the test set.
x_new <- runif(n/5, 1, 10)

# Fitting a simple linear regression, which will not capture the heteroscedasticity
model <- lm(y_train ~ x_train)

y_hat_cal <- predict(model, newdata=data.frame(x_train=x_cal))
MSE_cal <- mean((y_hat_cal - y_cal)^2)

y_hat_new <- predict(model, newdata=data.frame(x_train=x_new))

pit <- PIT_global(ycal=y_cal, yhat= y_hat_cal, mse=MSE_cal)

recalibrate(
  space_cal=x_cal,
  space_new=x_new,
  yhat_new=y_hat_new,
  pit_values=pit,
  mse= MSE_cal,
  type="local")
```

Index