Package: recalibratiNN (via r-universe)

September 4, 2024

Title Quantile Recalibration for Regression Models

Version 0.3.0

Description Enables the diagnostics and enhancement of regression model calibration.It offers both global and local visualization tools for 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>. The method leverages on Probabilistic Integral Transform (PIT) values to both evaluate and perform the calibration of statistical models. For a more detailed description of the package, please refer to the bachelor's thesis available bellow.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

URL `https://bdm.unb.br/handle/10483/38504`, `https://github.com/cmusso86/recalibratiNN`, `https://cmusso86.github.io/recalibratiNN/`

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

Imports stats(>= 3.0.0), 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 knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Repository https://cmusso86.r-universe.dev

RemoteUrl https://github.com/cmusso86/recalibratinn

RemoteRef HEAD

RemoteSha 9e48ed2eada1a76fad6a39edbc7bddf8dc52d23d

Contents

`gg_CD_global` *Plots the cumulative distributions of PIT-values for global calibration diagnostics.*

Description

Visualizes the predicted vs. empirical cumulative distributions of PIT-values using ggplot.

This function creates a ggplot graph that compares the cumulative distributions of predicted and empirical Probability Integral Transform (PIT) values. It shows the calibration quality of a regression model by examining how well the predicted values conform to the observed values.

Usage

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

Arguments

`pit`	Numeric vector of global PIT-values. It is recommended to calculate these using the `PIT_global()` function.
`ycal`	Numeric vector representing the true observations (y-values) of the response variable from the calibration dataset.
`yhat`	Numeric vector of predicted response (y-hat-values) on the calibration dataset.
`mse`	Mean Squared Error calculated from the calibration dataset.

Value

A `ggplot` object displaying a point graph of the empirical versus predicted cumulative distributions 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 <- PIT_global( y_cal, y_hat,  MSE_cal)

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

`gg_CD_local` *Plots the cumulative distributions of PIT-values for local calibration diagnostics.*

Description

This function generates a ggplot visual representation to compare the predicted versus empirical cumulative distributions of Probability Integral Transform (PIT) values at a local level. It is useful for diagnosing the calibration in different regions within the dataset, since miscalibration patterns may differ across the covariate space. The function allows for customization of the plot layers to suit specific needs. For advanced customization of the plot layers, refer to the ggplot2 User Guide.

Usage

```
gg_CD_local(
```

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

Arguments

pit_local	A data frame of local PIT-values, typically obtained from `PIT_local()`.
psz	Double indicating the size of the points on the plot. Default is 0.001.
abline	Color of the diagonal line. Default color is "red".
pal	Palette name from RColorBrewer for coloring the plot. Default is "Set2".
facet	Logical value indicating if a separate visualization for each subgroup is preferred. Default is FALSE.
...	Additional parameters to customize the ggplot.

Details

This funcion will work with the output of the `PIT_local()` function, which provides the PIT-values for each subgroup pf the covariate space in the appropriate format.

Value

A `ggplot` object displaying the cumulative distributions of PIT-values that that can be customized as needed.

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 Diagnostics*

Description

This function generates a ggplot visual representation of the density of Probability Integral Transform (PIT) values globally. For advanced customization of the plot layers, refer to the ggplot2 User Guide.

Usage

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

Arguments

`pit`	Vector of PIT values to be plotted.
`type`	Character string specifying the type of plot: either "density" or "histogram". This determines the representation style of the PIT values.
`fill`	Character string defining the fill color of the plot. Default is 'steelblue4'.
`alpha`	Numeric value for the opacity of the plot fill, with 0 being fully transparent and 1 being fully opaque. Default is 0.8.
`print_p`	Logical value indicating whether to print the p-value from the Kolmogorov-Smirnov test. Useful for statistical diagnostics.

Details

This function also tests the PIT-values for uniformity using the Kolmogorov-Smirnov test (`ks.test`). The p-value from the test is printed on the plot if `print_p` is set to `TRUE`.

Value

A `ggplot` object depicting a density graph of PIT-values, which can be further customized.

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 Diagnostics*

Description

A function based on ggplot2 to observe the density of PIT-values locally. It is recommended to use PIT-values obtained via the `PIT_local` function from this package or an object of equivalent format. For advanced customization of the plot layers, refer to the ggplot2 User Guide.This function also tests the PIT-values for uniformity using the Kolmogorov-Smirnov test (`ks.test`). The p-value from the test is printed on the plot if `facet` is set to `TRUE`.

Usage

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

Arguments

`pit_local`	A tibble with five columns: "part", "y_cal", "y_hat", "pit", and "n", representing the partitions, calibration data, predicted values, PIT-values, and the count of observations, respectively.
`alpha`	Numeric value between 0 and 1 indicating the transparency of the plot fill. Default is set to 0.4.
`linewidth`	Integer specifying the linewidth of the density line. Default is set to 1.
`pal`	A character string specifying the RColorBrewer palette to be used for coloring the plot. Default is "Set2".
`facet`	Logical indicating whether to use `facet_wrap()` to separate different covariate regions in the visualization. If TRUE, the p-value from the Kolmogorov-Smirnov test is printed on the plot.

Value

A `ggplot` object representing the local density distributions of PIT-values, which can be further customized through ggplot2 functions.

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 calculate the Probability Integral Transform (PIT) values for any fitted model that assumes a normal distribution of the output.

Usage

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

Arguments

ycal	Numeric vector representing the true observations (y-values) of the response variable from the calibration dataset.
yhat	Numeric vector of predicted y-values on the calibration dataset.
mse	Mean Squared Error calculated from the calibration dataset.

Details

This function is designed to work with models that is, even implicitly, assuming normal distribution of the response variable. This includes, but is not limited to, linear models created using `lm()` or neural networks utilizing Mean Squared Error as the loss function. The OLS method is used to minimized residuals in these models. This mathematical optimization will also yield a probabilistic optimization when normal distribution of the response variable is assumed, since OLS and maximum likelihood estimation are equivalent under normality. Therefore, in order to render a probabilistic interpretation of the predictions, the model is intrinsically assuming a normal distribution of the response variable.

Value

Returns a numeric 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

This function calculates local Probability Integral Transform (PIT) values using localized subregions of the covariate space from the calibration set. The output will be used for visualization of calibration quality using the `gg_CD_local()` and `gg_PIT_local()`function.

Usage

```
PIT_local(
  xcal,
  ycal,
  yhat,
  mse,
  clusters = 6,
```

```
  p_neighbours = 0.2,
  PIT = PIT_global
)
```

Arguments

- `xcal` Numeric matrix or data frame of features/covariates (x-values) from the calibration dataset.
- `ycal` Numeric vector representing the true observations (y-values) of the response variable from the calibration dataset.
- `yhat` Numeric vector of predicted response (y-hat-values) from the calibration dataset.
- `mse` Mean Squared Error calculated from the calibration dataset.
- `clusters` Integer specifying the number of partitions to create for local calibration using the k-means method. Default is set to 6.
- `p_neighbours` Proportion of xcal used to localize neighbors in the KNN method. Default is 0.2.
- `PIT` Function used to calculate the PIT-values. Default is set to `PIT_global()` from this package, that assumes a Gaussian distribution.

Details

It calculates local Probability Integral Transform (PIT) values using localized subregions of the covariate space from the calibration set. The centroids of such regions are derived from a k-means clustering method (from the `stats` package). The local areas around these centroids are defined through an approximate k-nearest neighbors method from the `RANN` package. Then, for this subregion, the PIT-values are calculated using the `PIT` function provided by the user. At the moment this function is tested to work with the `PIT_global()` function from this package, which assumes a Gaussian distribution. Eventually, it can be used with other distributions.

Value

A tibble with five columns containing unique names for each partition ("part"), "y_cal" (true observations), "y_hat" (predicted values), "pit" (PIT-values), and "n" (number of neighbors) for 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` *Generates Recalibrated Samples of the Predictive Distribution*

Description

This function offers recalibration techniques for regression models that assume Gaussian distributions by using the Mean Squared Error (MSE) as the loss function. Based on the work by Torres R. et al. (2024), it supports both local and global recalibration approaches to provide samples from a recalibrated predictive distribution. A detailed algorithm can also be found in Musso C. (2023).

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`	Numeric vector with predicted response values for the new (or test) set.
`pit_values`	Numeric vector of Global Probability Integral Transform (PIT) values calculated on the calibration set. We recommend using the PIT_global function.
`mse`	Mean Squared Error calculated from the calibration/validation set.

`space_cal`	Numeric matrix or data frame representing the covariates/features of the calibration/validation set, or any intermediate representation (like an intermediate layer of a neural network).
`space_new`	Similar to space_cal, but for a new set of covariates/features, ensuring they are in the same space as those in space_cal for effective local recalibration.
`type`	Character string to choose between 'local' or 'global' calibration.
`p_neighbours`	Proportion (0,1] of the calibration dataset to be considered for determining the number of neighbors in the KNN method. Default is set to 0.1. With p_neighbours=1, calibration is global but weighted by distance.
`epsilon`	Numeric value for approximation in the K-nearest neighbors (KNN) method. Default is 0, indicating exact distances.

Details

The calibration technique implemented here draws inspiration from Approximate Bayesian Computation and Inverse Transform Theorem, allowing for recalibration either locally or globally. The global method employs a uniform kernel, while the local method employs an Epanechnikov kernel.

It's important to note that the least squares method will only yield a probabilistic interpretation if the output to be modeled follows a normal distribution, and this assumption was used to implement this function.

The local recalibration method is expected to improve the predictive performance of the model, especially when the model is not able to capture the heteroscedasticity of the data. However, there is a trade off between refinement of localization and the Monte Carlo error, which can be controlled by the number of neighbors. That is, when more localized, the recalibration will grasp local changes better, but the Monte Carlo error will increase, because of the reduced number of neighbors.

When p_neighbours=1, recalibration is performed using the entire calibration dataset but with distance-weighted contributions.

Value

A list containing the calibrated predicted mean and variance, along with samples from the recalibrated predictive distribution and their respective weights calculated using 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.
Musso C (2023). "Recalibration of Gaussian Neural Network Regression Models: The RecalibratiNN Package." Undergraduate Thesis (Bachelor in Statistics), University of Brasília. Available at: `https://bdm.unb.br/handle/10483/38504`.

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