Package ‘AutoNLS’

December 29, 2024

Type Package

Title Automated Non-Linear Regression

Version 1.0.0

Author Adrian Antico [aut, cre, cph]

Maintainer Adrian Antico <adriantantico@gmail.com>

Description AutoNLS is a comprehensive package for automated non-linear regression modeling, evaluation, and visualization. It supports dynamic selection of non-linear models, tools for scoring and comparing models, and powerful visualizations using the `echarts4r` package. The package is designed for ease of use and extensibility, making it ideal for analysts, data scientists, and researchers.

Imports R6,
data.table,
dplyr,
echarts4r,
minpack.lm,
mgcv

Suggests testthat,
shiny,
DT,
bs4Dash,
readxl

License AGPL (>= 3) + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://github.com/AdrianAntico/AutoNLS

BugReports https://github.com/AdrianAntico/AutoNLS/issues

Language en-US

NeedsCompilation no

R topics documented:

`dummy_data` *Dummy Data for AutoNLS Examples*

Description

A synthetic dataset for testing and demonstrating AutoNLS functions.

Usage

```
dummy_data
```

Format

A data.frame with 100 rows and 3 variables:

X-Value Numeric predictor variable.

GroupVar Categorical grouping variable.

Target Numeric response variable.

Source

Generated synthetically for the AutoNLS package.

`EDA` *EDA (Exploratory Data Analysis) Class*

Description

Provides tools for automated exploratory data analysis, including summary statistics, correlation matrices, and customizable visualizations using `echarts4r`.

Methods

- `initialize(data)`: Initializes the class with a `data.table`.
- `summarize()`: Computes summary statistics.
- `correlate()`: Computes a correlation matrix for numeric columns.
- `visualize_distributions()`: Creates histogram and density visualizations for numeric columns.
- `visualize_scatterplots()`: Creates pairwise scatterplots for numeric columns.
- `generate_3d_scatter_plot()`: Creates a 3D scatterplot for numeric columns.

Public fields

`data` A `data.table` containing the dataset for analysis.

`summary_stats` A `data.table` storing the summary statistics of the dataset.

`correlation_matrix` A correlation matrix for numeric columns.

`plots` A list of `echarts4r` plots generated during the analysis. Initialize the EDA class

Methods

Public methods:

- `EDA$new()`
- `EDA$summarize()`
- `EDA$correlate()`
- `EDA$visualize_distributions()`
- `EDA$visualize_scatterplots()`
- `EDA$generate_3d_scatter_plot()`
- `EDA$clone()`

Method `new()`**:**

Usage:

```
EDA$new(data)
```

Arguments:

`data` A `data.table` containing the dataset for analysis.

Method `summarize()`**:** Calculates mean, median, sd, and the count of missing values for each column.

Usage:

```
EDA$summarize()
```

Returns: A `data.table` containing the summary statistics.

Method `correlate()`**:** Calculates both Pearson and Spearman correlations between the `target_col` and all (or listed via `input_cols`) numeric columns.

Usage:

```
EDA$correlate(target_col = NULL, input_cols = NULL)
```

Arguments:

`target_col` the target variable in the data set

`input_cols` the independent variables

Returns: A data.table with the Pearson and Spearman correlation values for each numeric predictor.

Method `visualize_distributions()`**:** Generates histograms for numeric columns and optionally overlays density lines.

Usage:

```
EDA$visualize_distributions(
  input_cols = NULL,
  title_prefix = "Distribution of",
  bins = 20,
  add_density = TRUE,
  tooltip_trigger = "axis",
  theme = "westeros",
  density_opacity = 0.4
)
```

Arguments:

`input_cols` Names of numeric variables to plot

`title_prefix` Character. Prefix for the plot title.

`bins` Integer. Number of bins for the histogram. Defaults to Sturges' formula.

`add_density` Logical. Whether to add a density line. Defaults to `TRUE`.

`tooltip_trigger` "axis"

`theme` Character. Theme for the plot

`density_opacity` numeric. default 0.4

Returns: A list of `echarts4r` histogram plots.

Method `visualize_scatterplots()`**:** Generates scatterplots for all target and input pairs of numeric columns and overlays fitted lines from Generalized Additive Models (GAM) for different `k` values.

Usage:

```
EDA$visualize_scatterplots(
  target_col = NULL,
  input_cols = NULL,
  title_prefix = "Scatterplot of",
  theme = "westeros",
  k_values = c(3, 5, 7)
)
```

Arguments:

`target_col` Name of target variable

`input_cols` Names of input variables

`title_prefix` Character. Prefix for the plot title.

`theme` Character. Theme for the plot

`k_values` Numeric vector. Values of `k` (basis dimension) for GAM fits. Defaults to `c(3, 5, 7)`.

Returns: A list of `echarts4r` scatter plots with GAM fitted lines.

Method `generate_3d_scatter_plot()`**:** Generates a 3D scatter plot for three numeric variables.

Usage:

```
EDA$generate_3d_scatter_plot(
  input_col1,
  input_col2,
  target_col,
  rank_values = TRUE,
  theme = "westeros"
)
```

Arguments:

`input_col1` The name of the first numeric column.

`input_col2` The name of the second numeric column.

`target_col` The name of the third numeric column, the target variable.

`rank_values` Logical. Whether to transform variables to their percentile ranks. Defaults to TRUE.

`theme` Name of theme for `echarts4r` plots

Returns: An `echarts4r` 3D scatter plot.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
EDA$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ModelEvaluator` *ModelEvaluator*

Description

ModelEvaluator

ModelEvaluator

Details

An R6 class to evaluate non-linear regression models. Includes tools to generate tables of statistics and visualizations to compare models against data.

This method evaluates each fitted model by calculating various metrics such as Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), residual standard error, and R-squared values. It ensures compatibility across all models and gracefully handles cases where a model fails to fit properly by excluding it from the final summary.

This method generates a comparison plot for each fitted model, allowing users to visually assess how well the models align with the observed data. It supports customization options such as theming and dynamic adjustment of the x-axis range. Models that fail to fit are gracefully excluded, ensuring clean and informative outputs.

Public fields

`fit_results` A list of fitted model objects.

`evaluation_metrics` A data.table containing model evaluation metrics.

`plots` A list of visualizations comparing models against data.

`data` The original dataset used for fitting models. Initialize the ModelEvaluator class

Methods

Public methods:

- `ModelEvaluator$new()`
- `ModelEvaluator$generate_metrics()`
- `ModelEvaluator$generate_comparison_plot()`
- `ModelEvaluator$clone()`

Method `new()`**:**

Usage:

```
ModelEvaluator$new(fit_results, data)
```

Arguments:

`fit_results` A list of fitted model objects (e.g., output from NonLinearFitter).

`data` The original dataset used for fitting models.

Returns: A new instance of the ModelEvaluator class.

Method `generate_metrics()`**:** Computes and summarizes key performance metrics for all fitted models, including goodness-of-fit statistics, residual standard errors, and information criteria. The metrics provide a comprehensive evaluation of each model's performance on the given dataset.

Usage:

```
ModelEvaluator$generate_metrics(y_col = NULL, x_col = NULL)
```

Arguments:

`y_col` target variable

`x_col` x variable

Returns: A data.table of evaluation metrics with fitted equations.

Method `generate_comparison_plot()`**:** Creates visualizations comparing the fitted models against the observed data to assess their fit and predictive behavior. The plots include the fitted curves overlaid on the original data points for easy comparison.

Usage:

```
ModelEvaluator$generate_comparison_plot(
  data,
  x_col,
  y_col,
  theme = "westeros",
  lower_bound = 0.025,
  upper_bound = 0.975,
  n_sim = 1000
)
```

Arguments:

`data` A `data.table` or `data.frame` containing the dataset used for evaluation.

`x_col` A string specifying the name of the x variable in the dataset.

`y_col` A string specifying the name of the y variable in the dataset.

`theme` Echarts theme

`lower_bound` Lower bound probability. Defaults to 0.025

`upper_bound` Upper bound probability. Defaults to 0.975

`n_sim` Number of simulations to run for prediction interval

Returns: An `echarts4r` plot showing observed vs. predicted data, with weighted comparisons if available.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ModelEvaluator$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ModelFitter` *ModelFitter*

Description

ModelFitter

ModelFitter

Details

An R6 class for automatically fitting non-linear regression models. Includes a library of pre-defined models to simplify selection.

This method returns a list containing information about the supported non-linear models, including their names, descriptions, and the formulas they use. It serves as a convenient reference for users to understand the available options and select the most appropriate models for their dataset and objectives.

This method enables the inclusion of both built-in and custom models in the fitting process. Users can select a predefined model from the library or provide the necessary components to define a custom model, including a formula, starting parameters, and a function to calculate predictions.

For custom models, the following components must be provided:

- `name`: Name of the model
- `formula`: A mathematical representation of the model.
- `start_params`: A named list of starting parameter values for optimization.
- `model_function`: A function that takes input `x` and parameter values as arguments and returns the predicted `y` values.

The choice of fitting method depends on the arguments:

- If `force_optim = TRUE`, the function will use `optim()` regardless of whether weights are supplied.
- If `weights` is supplied, `optim()` will be used even if `force_optim = FALSE`.
- If neither `force_optim` nor `weights` is supplied, the function defaults to `nls()` for unweighted fitting.

Behavior Examples:

1. **Default behavior**: `nls()` is used when `weights = NULL` and `force_optim = FALSE`.
2. **Weighted fitting**: `optim()` is used when `weights` is provided, even if `force_optim = FALSE`.
3. **Forced optimization**: `optim()` is used when `force_optim = TRUE`, regardless of whether `weights` is supplied.

This method overlays the predictions of selected fitted models onto the observed data across a specified range of the independent variable. The output is an interactive plot generated using the `echarts4r` package, allowing users to visually assess the goodness of fit, trends, and differences among models.

By default, predictions are normalized to allow fair comparison across models with different scales. Users can customize the plotting range and apply specific visual themes for enhanced interpretability.

Public fields

`data` A data.table containing the dataset for modeling.

`models` A list of non-linear models to test.

`fit_results` A list to store the results of model fits.

`evaluation_metrics` A list to store evaluation metrics for each model.

`plots` A list to store plots of model fits.

`model_library` A pre-defined library of common non-linear models. Initialize the NonLinearFitter class

Methods

Public methods:

- `ModelFitter$new()`
- `ModelFitter$list_models()`
- `ModelFitter$add_model()`
- `ModelFitter$fit_models()`
- `ModelFitter$model_comparison_plot()`
- `ModelFitter$clone()`

Method `new()`:

Usage:

```
ModelFitter$new(data)
```

Arguments:

`data` A data.table containing the dataset for modeling. Must include the predictor and response variable columns.

Returns: A new instance of the NonLinearFitter class.

Method `list_models()`: Retrieves a list of all non-linear models available for fitting in the `NonLinearFitter` class, along with their descriptions and key details. This method provides an overview of the models users can select for analysis.

Usage:

```
ModelFitter$list_models()
```

Returns: A list summarizing available models.

Method `add_model()`: Adds a specified non-linear model to the list of models to be fitted. This method allows users to include predefined models or define their own custom models for analysis.

Usage:

```
ModelFitter$add_model(
  name,
  formula = NULL,
  start_params = NULL,
  model_function = NULL
)
```

Arguments:

`name` The name of the model (e.g., "Hill").

`formula` The non-linear formula for the model (optional if using pre-defined model).

`start_params` A list of starting parameters for the model (optional if using pre-defined model).

`model_function` A function used in fitting and prediction. See model_library for examples

Returns: NULL

Method `fit_models()`**:** `fit_models()` first standardizes your data before fitting the model. The fitting method depends on whether a `weights_col` is provided: `nls()` is used for unweighted fitting, while `optim()` is used for weighted fitting. The returned parameters are based on the standardized data. However, when scoring models, the results are back-transformed to align with the original data scale.

Usage:

```
ModelFitter$fit_models(
  x_col,
  y_col,
  weights_col = NULL,
  control = list(maxiter = 1024),
  force_optim = FALSE,
  ...
)
```

Arguments:

`x_col` The name of the predictor variable.

`y_col` The name of the response variable.

`weights_col` The name of the weights variable.

`control` A list of control parameters for the optimizer, such as `maxiter`. Default is `list(maxiter = 200)`.

`force_optim` Logical; if `TRUE`, forces the use of `optim()` regardless of whether weights are supplied. Defaults to `FALSE`.

`...` Additional arguments to be passed to the underlying fitting functions (`nlsLM` for unweighted models or `optim` for weighted models). Examples include `trace`, `lower`, and `upper` for `nlsLM`, or `reltol`, `parscale`, and others for `optim`.

Returns: A list of fitted model objects.

Method `model_comparison_plot()`**:** Creates a visual comparison of multiple fitted non-linear models against the observed data. This method helps evaluate the performance and fit of different models in a single, intuitive plot.

Usage:

```
ModelFitter$model_comparison_plot(
  x_range = seq(1, 100, by = 1),
  normalize = TRUE,
  theme = "westeros"
)
```

Arguments:

`x_range` A numeric vector specifying the range of x values to evaluate.

`normalize` Logical. If TRUE, normalizes the y values for each model to fall between 0 and 1. Defaults to TRUE.

`theme` A string specifying the plot theme (e.g., "macarons").

Returns: An `echarts4r` object representing the comparison plot.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ModelFitter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ModelScorer` *ModelScorer*

Description

ModelScorer

ModelScorer

Details

An R6 class to score non-linear regression models on new data and visualize the results.

The `score_new_data` method enables users to evaluate how well the fitted models generalize to unseen data. It uses the stored parameters of the fitted models and applies them to the specified independent variable (`x_col`) in the new dataset. If the fitted models include transformations, the predictions are back-transformed to match the scale of the original data.

The output is a data frame containing the original data and the predicted values for each model, making it easy to compare model predictions side-by-side.

The `generate_score_plot` method produces an interactive plot comparing the predicted values of a fitted model against the actual observed values from the new dataset. The independent variable (`x_col`) is used on the x-axis, while the predicted and actual dependent variable values are displayed on the y-axis. This visualization can be used to assess how well the model generalizes to new data and to identify areas where predictions deviate from observations.

The plot leverages the `echarts4r` package for interactive and customizable visualizations.

Public fields

`fit_results`	A list of fitted model objects.
`scored_data`	A list of data.tables containing scored data.
`score_plots`	A list of plots visualizing scored data.

Methods

Public methods:

- `ModelScorer$new()`
- `ModelScorer$score_new_data()`
- `ModelScorer$generate_score_plot()`
- `ModelScorer$clone()`

Method `new()`:

Usage:

```
ModelScorer$new(fit_results)
```

Arguments:

`fit_results` A list of fitted model objects (e.g., output from NonLinearFitter).

Returns: A new instance of the ModelScorer class.

Method `score_new_data():` Generates predictions for new data using the fitted non-linear models. This method applies each model to the new dataset and returns the predicted values.

Usage:

```
ModelScorer$score_new_data(
  new_data,
  x_col,
  lower_bound = 0.025,
  upper_bound = 0.975
)
```

Arguments:

`new_data` A data.table containing the new data to score.

`x_col` The predictor column in `new_data`.

`lower_bound` Lower bound of prediction interval. Defaults to 0.025

`upper_bound` Upper bound of prediction interval. Defaults to 0.975

Returns: A list of data.tables with predicted values for each model.

Method `generate_score_plot():` Creates a visual representation of predictions versus actual values for a given fitted model and a new dataset. This plot helps evaluate the model's performance on unseen data.

Usage:

```
ModelScorer$generate_score_plot(model_name, x_col, theme = "westeros")
```

Arguments:

`model_name` The name of the model to plot.

`x_col` The predictor column in scored data.

`theme` Echarts theme.

Returns: A plot visualizing the scored data.

Method `clone():` The objects of this class are cloneable with this method.

Usage:

```
ModelScorer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`run_shiny_app` *Run the AutoNLS Shiny App*

Description

This function launches the interactive Shiny application for AutoNLS.

Usage

```
run_shiny_app(launch_browser = TRUE)
```

Arguments

`launch_browser` Logical. If TRUE, the app opens in the default web browser. Defaults to TRUE.

Index