Package ‘autovi’

November 18, 2024

Type Package

Title Auto Visual Inference with Computer Vision Models

Version 0.4.1

Description Provides automated visual inference of residual plots using computer vision models, facilitating diagnostic checks for classical normal linear regression models.

License MIT + file LICENSE

URL `https://tengmcing.github.io/autovi/`,
`https://github.com/TengMCing/autovi/`

BugReports `https://github.com/TengMCing/autovi/issues`

Encoding UTF-8

RoxygenNote 7.2.1

Imports bandicoot, ggplot2, stats, tibble, tools, reticulate, utils, cassowaryr, cli

Suggests rmarkdown, knitr, testthat (>= 3.0.0), covr

Config/testthat/edition 3

Depends R (>= 2.10)

NeedsCompilation no

Author Weihao Li [aut, cre, cph] (<`https://orcid.org/0000-0003-4959-106X`>)

Maintainer Weihao Li <`llreczx@gmail.com`>

Repository CRAN

Date/Publication 2024-11-18 04:40:12 UTC

Contents

`AUTO_VI` *AUTO_VI class environment*

Description

This is the class of auto visual inference, inherited from bandicoot::BASE. It is an environment with S3 class `bandicoot_oop`.

Usage

```
auto_vi(
  fitted_model,
  keras_model = NULL,
  data = NULL,
  node_index = 1L,
  env = new.env(parent = parent.frame()),
  init_call = sys.call()
)

residual_checker(
  fitted_model,
  keras_model_name = "vss_phn_32",
  data = NULL,
  node_index = 1L,
  env = new.env(parent = parent.frame()),
  init_call = sys.call()
)
```

Arguments

`fitted_model` Model. A model object, e.g. `lm`.

`keras_model` Keras model. A trained computer vision model.

`data` Data frame. The data used to fit the model.

`node_index` Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.

`env` Environment. The instance environment.

`init_call` Call. Contents of the `..init_call..`. It is recommended to leave it as default.

`keras_model_name` Character. A model name to be used by `get_keras_model()`. See also `list_keras_model()`.

Value

An instance environment.

Functions

- `auto_vi()`: Class constructor, same as `AUTO_VI$instantiate()`.
- `residual_checker()`: Class constructor, same as `AUTO_VI$instantiate()`, but uses the `keras_model_name` argument rather than `keras_model`.

Class information

Parent classes:

- Direct:

  - bandicoot::BASE

New attributes:

- C:
  - AUTO_VI$check_result

New methods:

- A:
  - `AUTO_VI$auxiliary()`
- B:
  - `AUTO_VI$boot_method()`
  - `AUTO_VI$boot_vss()`
- C:
  - `AUTO_VI$check()`
- F:
  - `AUTO_VI$feature_pca()`
  - `AUTO_VI$feature_pca_plot()`
- G:
  - `AUTO_VI$get_data()`
  - `AUTO_VI$get_fitted_and_resid()`
- I:
  - `AUTO_VI$..init..()`
- L:
  - `AUTO_VI$lineup_check()`
  - `AUTO_VI$likelihood_ratio()`
- N:
  - `AUTO_VI$null_method()`
  - `AUTO_VI$null_vss()`
- P:
  - `AUTO_VI$p_value()`
  - `AUTO_VI$plot_pair()`
  - `AUTO_VI$plot_lineup()`
  - `AUTO_VI$plot_resid()`
- R:
  - `AUTO_VI$rotate_resid()`
- S:
  - `AUTO_VI$save_plot()`
  - `AUTO_VI$..str..()`
  - `AUTO_VI$summary()`
  - `AUTO_VI$summary_density_plot()`
  - `AUTO_VI$summary_plot()`
  - `AUTO_VI$summary_rank_plot()`
- V:
  - `AUTO_VI$vss()`

`AUTO_VI$..init..` *Initialization method*

Description

This function will be called after an instance is built. User input will be stored in the environment.

Usage:

```
AUTO_VI$..init..(fitted_model, keras_model = NULL, data = NULL, node_index = 1L)
```

Arguments

`fitted_model`	Model. A model object, e.g. `lm`.
`keras_model`	Keras model. A trained computer vision model.
`data`	Data frame. The data used to fit the model.
`node_index`	Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.

Value

Return the object itself.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
my_vi
```

`AUTO_VI$..str..` *String representation of the object*

Description

This function returns a string representation of the object.

Usage:

```
AUTO_VI$..str..()
```

Value

A string.

Examples

```
AUTO_VI$..str..()

my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
my_vi$..str..()
```

AUTO_VI$auxiliary *Compute auxiliary variables for the keras model*

Description

This function computes auxiliary variables including monotonic measure (measure_monotonic), sparse measure (measure_sparse), splines measure (measure_splines), striped measure (measure_striped), and the number of observation (n). Scagnostics are computed using cassowaryr::sc_monotonic(), cassowaryr::sc_sparse2(), cassowaryr::sc_splines(), and cassowaryr::sc_striped().

If you wish to calculate additional auxiliary variables for your keras model, please override this method. Ensure that it accepts a data frame with columns named .fitted and .resid as input and returns a single row tibble.

Usage:

```
AUTO_VI$auxiliary(data = seflf$get_fitted_and_resid())
```

Arguments

data	Data frame. A data frame containing variables .resid and .fitted. See also AUTO_VI$get_fitted_and_resid().

Value

A tibble.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
my_vi$auxiliary()
```

AUTO_VI$boot_method *Get bootstrapped residuals from a fitted model*

Description

This default method gets bootstrapped residuals from a fitted linear model by sampling the observations with replacement then refit the model. User needs to override this method if a different bootstrapping scheme is needed.

Usage:

```
AUTO_VI$boot_method(
  fitted_model = self$fitted_model,
  data = self$get_data()
)
```

Arguments

`fitted_model`	`lm`. A linear model object.
`data`	Data frame. The data used to fit the model. See also `AUTO_VI$get_data()`.

Value

A tibble with two columns `.fitted` and `.resid`.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
null_resid <- my_vi$boot_method()
my_vi$plot_resid(null_resid)
```

`AUTO_VI$boot_vss` *Predict visual signal strength for bootstrapped residual plots*

Description

This function bootstrap the data and refits the model by using `AUTO_VI$boot_method()`, then predicts the visual signal strength of the bootstrapped residual plots.

Usage:

```
AUTO_VI$boot_vss(
  draws = 100L,
  fitted_model = self$fitted_model,
  keras_model = self$keras_model,
  data = self$get_data(),
  node_index = 1L,
  keep_boot_data = FALSE,
  keep_boot_plot = FALSE,
  extract_feature_from_layer = NULL
)
```

Arguments

`draws`	Integer. Number of simulation draws.
`fitted_model`	Model. A model object, e.g. `lm`.
`keras_model`	Keras model. A trained computer vision model.
`data`	Data frame. The data used to fit the model. See also `AUTO_VI$get_data()`.
`node_index`	Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.
`keep_boot_data`	Boolean. Whether to keep the bootstrapped data.
`keep_boot_plot`	Boolean. Whether to keep the bootstrapped plots.

`extract_feature_from_layer`
Character/Integer. A layer name or an integer layer index for extracting features from a layer.

Value

A tibble.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$boot_vss()
}
```

AUTO_VI$check *Conduct a auto visual inference check with a computer vision model*

Description

This function conducts a visual inference check with a computer vision model. The result will be stored in `self$check_result`.

Usage:

```
AUTO_VI$check(
  null_draws = 100L,
  boot_draws = 100L,
  fitted_model = self$fitted_model,
  keras_model = self$keras_model,
  null_method = self$null_method,
  data = self$get_data(),
  node_index = self$node_index,
  keep_data = FALSE,
  keep_plot = FALSE,
  extract_feature_from_layer = NULL
)
```

Arguments

`null_draws` Integer. Number of simulation draws for `AUTO_VI$null_vss()`.

`boot_draws` Integer. Number of simulation draws for `AUTO_VI$boot_vss()`.

`fitted_model` Model. A model object, e.g. `lm`.

`keras_model` Keras model. A trained computer vision model.

`null_method` Function. A method to simulate residuals from the null hypothesis distribution. For `lm`, the recommended method is residual rotation `AUTO_VI$rotate_resid()`.

`data` Data frame. The data used to fit the model. See also `AUTO_VI$get_data()`.

`node_index` Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.

`keep_data` Boolean. Whether to keep the simulated data.

`keep_plot` Boolean. Whether to keep the simulated plots.

`extract_feature_from_layer`
Character/Integer. A layer name or an integer layer index for extracting features from a layer.

Value

Return the object itself.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$check()
  myvi
}
```

`AUTO_VI$check_result` *List of diagnostic results*

Description

A list, will be initialized after the method `AUTO_VI$check()` is run.

`AUTO_VI$feature_pca` *Conduct principal component analysis for features extracted from keras model*

Description

This function conducts principal component analysis for features extracted from keras model.

Usage:

```
AUTO_VI$feature_pca(
  feature = self$check_result$observed,
  null_feature = self$check_result$null,
  boot_feature = self$check_result$boot,
  center = TRUE,
  scale = TRUE,
  pattern = "^f_.*$"
)
```

Arguments

`feature` Dataframe. A data frame where columns represent features and rows represent observations. It should have only one row.

`null_feature` Dataframe. A data frame where columns represent features and rows represent observations. These features are extracted during the evaluation of null plots.

`boot_feature` Dataframe. A data frame where columns represent features and rows represent observations. These features are extracted during the evaluation of bootstrapped plots.

`center` Boolean. Whether to subtract the mean from the feature.

`scale` Boolean. Whether to divide the feature by its standard deviation.

`pattern` Character. A regrex pattern to search for features in the provided DataFrame. See also `grep()`.

Details

Features need to be extracted while running the method `AUTO_VI$check()` and `AUTO_VI$lineup_check()` by providing the argument `extract_feature_from_layer`. Features with zero variance will be ignored from the analysis. See also `stats::prcomp()`. By default, features are assumed to follow the naming convention "f_(index)", where index is from one to the number of features.

Value

A tibble of the raw features and the rotated features with attributes `sdev` and `rotation` representing the standard deviation of the principal components and the rotation matrix respectively.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$lineup_check(extract_feature_from_layer = "global_max_pooling2d")
  myvi$feature_pca()
}
```

AUTO_VI$feature_pca_plot

Draw a summary Plot for principal component analysis conducted on extracted features

Description

This function draws a summary Plot for principal component analysis conducted on extracted features

Usage:

```
AUTO_VI$feature_pca_plot(
  feature_pca = self$feature_pca(),
  x = PC1,
  y = PC2,
  col_by_set = TRUE)
```

Arguments

`feature_pca`	Dataframe. A data frame containing the rotated features.
`x`	Symbol. The x variable. See also ggplot2::tidyeval.
`y`	Symbol. The y variable. See also ggplot2::tidyeval.
`col_by_set`	Boolena. Whether to color points by sets (observed, null, and boot).

Details

By default, it will visualize PC2 vs PC1. User can choose to visualize other principal components.

Value

A `ggplot`.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$lineup_check(extract_feature_from_layer = "global_max_pooling2d")
  myvi$feature_pca_plot()
}
```

`AUTO_VI$get_data` *Get data out of a model object*

Description

This function gets the data out of a model object by using `stats::model.frame()` if `self$data` is `NULL`.

Usage:

```
AUTO_VI$get_data(fitted_model = self$fitted_model)
```

Arguments

`fitted_model`	Model. A model object, e.g. `lm`.

Value

A tibble.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
my_vi$get_data()
```

`AUTO_VI$get_fitted_and_resid`

Get fitted values and residuals out of a model object

Description

This function gets fitted values and residuals out of a model object by using `stats::fitted()` and `stats::resid()`.

Usage:

```
AUTO_VI$get_fitted_and_resid(fitted_model = self$fitted_model)
```

Arguments

`fitted_model`	Model. A model object, e.g. `lm`.

Value

A tibble.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
my_vi$get_fitted_and_resid()
```

`AUTO_VI$likelihood_ratio` *Compute the likelihood ratio using the simulated result*

Description

This function estimates the likelihood of observing the visual signal strength in terms of the bootstrapped distribution and the simulated null distribution, and computes the ratio between these two likelihood.

Usage:

```
AUTO_VI$likelihood_ratio(
  vss = self$check_result$observed$vss,
  dist_1 = self$check_result$boot$vss,
  dist_2 = self$check_result$null$vss
)
```

Arguments

`vss` Numeric. The observed visual signal strength.

`dist_1` Numeric. A vector of visual signal strength for plots following the first distribution (bootstrap distribution by default).

`dist_2` Numeric. A vector of visual signal strength for plots following the second distribution (null distribution by default).

Value

A named vector with three elements `likelihood_1`, `likelihood_2` and `likelihood_ratio`.

Examples

```
dist_1 <- rnorm(100, 0, 1)
dist_2 <- rnorm(100, 1, 1)
AUTO_VI$likelihood_ratio(0, dist_1, dist_2)
```

`AUTO_VI$lineup_check` *Conduct a auto visual inference lineup check with a computer vision model*

Description

This function conducts a visual inference lineup check with a computer vision model. The result will be stored in `self$check_result`.

Usage:

```
AUTO_VI$lineup_check(
  lineup_size = 20L,
  fitted_model = self$fitted_model,
  keras_model = self$keras_model,
  null_method = self$null_method,
  data = self$get_data(),
  node_index = self$node_index,
  extract_feature_from_layer = NULL
)
```

Arguments

`lineup_size` Integer. Number of plots in a lineup.

`fitted_model` Model. A model object, e.g. `lm`.

`keras_model` Keras model. A trained computer vision model.

`null_method` Function. A method to simulate residuals from the null hypothesis distribution. For `lm`, the recommended method is residual rotation `AUTO_VI$rotate_resid()`.

`data` Data frame. The data used to fit the model. See also `AUTO_VI$get_data()`.

`node_index` Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.

`extract_feature_from_layer`
Character/Integer. A layer name or an integer layer index for extracting features from a layer.

Value

Return the object itself.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$lineup_check()
  myvi
}
```

AUTO_VI$null_method *Get null residuals from a fitted model*

Description

This default method gets rotated residuals from a fitted linear model using AUTO_VI$rotate_resid. User needs to override this method if the fitted model is not a linear regression model.

Usage:

```
AUTO_VI$null_method(fitted_model = self$fitted_model)
```

Arguments

fitted_model	lm. A linear model object.

Value

A tibble with two columns .fitted and .resid.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
null_resid <- my_vi$null_method()
my_vi$plot_resid(null_resid)
```

AUTO_VI$null_vss *Simulate null plots and predict the visual signal strength*

Description

This function simulates null plots from the null hypothesis distribution, and predicts the visual signal strength.

Usage:

```
AUTO_VI$null_vss(
  draws = 100L,
  fitted_model = self$fitted_model,
  keras_model = self$keras_model,
  null_method = self$null_method,
  node_index = self$node_index,
  keep_null_data = FALSE,
  keep_null_plot = FALSE,
  extract_feature_from_layer = NULL
)
```

Arguments

`draws`	Integer. Number of simulation draws.
`fitted_model`	Model. A model object, e.g. `lm`.
`keras_model`	Keras model. A trained computer vision model.
`null_method`	Function. A method to simulate residuals from the null hypothesis distribution. For `lm`, the recommended method is residual rotation `AUTO_VI$rotate_resid()`.
`node_index`	Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.
`keep_null_data`	Boolean. Whether to keep the simulated null data.
`keep_null_plot`	Boolean. Whether to keep the simulated null plots.
`extract_feature_from_layer`	Character/Integer. A layer name or an integer layer index for extracting features from a layer.

Value

A tibble.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$null_vss()
}
```

`AUTO_VI$plot_lineup` *Draw a lineup of standard residual plots*

Description

This function draws a lineup of standard residual plots consisting of a true residual plot and several null residual plots.

Usage:

```
AUTO_VI$plot_lineup(
  lineup_size = 20L,
  data = self$get_fitted_and_resid(),
  null_method = self$null_method,
  theme = ggplot2::theme_light(),
  alpha = 1,
  size = 0.5,
  stroke = 0.5,
  remove_axis = TRUE,
  remove_legend = TRUE,
  remove_grid_line = TRUE,
  add_zero_line = TRUE,
  remove_facet_label = FALSE,
  display_answer = TRUE
)
```

Arguments

`lineup_size` Numeric. Number of plots in a lineup.

`data` Data frame. A data frame containing variables `.resid` and `.fitted`. See also `AUTO_VI$get_fitted_and_resid()`.

`null_method` Function. A function that takes a fitted model as input, and outputs a data frame containing variables `.resid` and `.fitted`. See also `AUTO_VI$null_method()`.

`theme` `ggtheme`. A `ggplot` theme object. See also `ggplot2::theme_light()`.

`alpha` Numeric. Alpha of dot. Value between 0 and 1.

`size` Numeric. Size of dot. Value between 0 and 1.

`stroke` Numeric. Stroke of dot. Value between 0 and 1.

`remove_axis` Boolean. Whether or not to remove the axis.

`remove_legend` Boolean. Whether or not to remove the legend.

`remove_grid_line`
Boolean. Whether or not to remove the grid lines.

`add_zero_line` Boolean. Whether or not to add a zero horizontal line.

`remove_facet_label`
Boolean. Whether or not to remove facet labels.

`display_answer` Boolean. Whether or not to display the answer in title.

Value

A `ggplot`.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
my_vi$plot_lineup()
```

AUTO_VI$plot_pair *Draw a pair of standard residual plots*

Description

This function draws a pair of standard residual plots consisting of a true residual plot and a null residual plot.

Usage:

```
AUTO_VI$plot_pair(
  data = self$get_fitted_and_resid(),
  null_data = self$null_method(),
  theme = ggplot2::theme_light(),
  alpha = 1,
  size = 0.5,
  stroke = 0.5,
  remove_axis = TRUE,
  remove_legend = TRUE,
  remove_grid_line = TRUE,
  add_zero_line = TRUE
)
```

Arguments

`data` Data frame. A data frame containing variables `.resid` and `.fitted`. See also `AUTO_VI$get_fitted_and_resid()`.

`null_data` Data frame. A data frame containing variables `.resid` and `.fitted`. See also `AUTO_VI$null_method()`.

`theme` `ggtheme`. A `ggplot` theme object. See also `ggplot2::theme_light()`.

`alpha` Numeric. Alpha of dot. Value between 0 and 1.

`size` Numeric. Size of dot. Value between 0 and 1.

`stroke` Numeric. Stroke of dot. Value between 0 and 1.

`remove_axis` Boolean. Whether or not to remove the axis.

`remove_legend` Boolean. Whether or not to remove the legend.

`remove_grid_line`
Boolean. Whether or not to remove the grid lines.

`add_zero_line` Boolean. Whether or not to add a zero horizontal line.

Value

A `ggplot`.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
my_vi$plot_pair()
```

AUTO_VI$plot_resid *Draw a standard residual plot*

Description

This function draws a standard residual plot.

Usage:

```
AUTO_VI$plot_resid(
  data = self$get_fitted_and_resid(),
  theme = ggplot2::theme_light(base_size = 11/5),
  alpha = 1,
  size = 0.5,
  stroke = 0.5,
  remove_axis = TRUE,
  remove_legend = TRUE,
  remove_grid_line = TRUE,
  add_zero_line = TRUE
)
```

Arguments

`data`	Data frame. A data frame containing variables `.resid` and `.fitted`. See also `AUTO_VI$get_fitted_and_resid()`.
`theme`	ggtheme. A `ggplot` theme object. See also `ggplot2::theme_light()`.
`alpha`	Numeric. Alpha of dot. Value between 0 and 1.
`size`	Numeric. Size of dot. Value between 0 and 1.
`stroke`	Numeric. Stroke of dot. Value between 0 and 1.
`remove_axis`	Boolean. Whether or not to remove the axis.
`remove_legend`	Boolean. Whether or not to remove the legend.
`remove_grid_line`	Boolean. Whether or not to remove the grid lines.
`add_zero_line`	Boolean. Whether or not to add a zero horizontal line.

Value

A `ggplot`.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
my_vi$plot_resid()
```

AUTO_VI$p_value *Compute the p-value based on the check result*

Description

This function computes the p-value of observing the visual signal strength of the original residual plot based on the null distribution.

Usage:

```
AUTO_VI$p_value(
  vss = self$check_result$observed$vss,
  null_dist = self$check_result$null$vss
)
```

Arguments

vss	Numrice. A single numeric value indicating the visual signal strength for the true residual plot.
null_dist	Numeric. A vector of numeric values indicating the visual signal strength for null residual plots.

Value

A numeric value representing the desired p-value.

Examples

```
vss <- 1
null_dist <- rnorm(100, 0, 1)
AUTO_VI$p_value(vss, null_dist)
```

AUTO_VI$rotate_resid *Get rotated residuals from a fitted linear model*

Description

This function gets rotated residuals from a fitted linear model. The rotated residuals are generated by first regressing random noises on the original regressors, then multiply the obtained residuals by original RSS divided by the current RSS. The results are the rotated residuals.

Usage:

```
AUTO_VI$rotate_resid(fitted_model = self$fitted_mod)
```

Arguments

fitted_model	lm. A linear model object.

Value

A tibble with two columns `.fitted` and `.resid`.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
rotated_resid <- my_vi$rotate_resid()
my_vi$plot_resid(rotated_resid)
```

`AUTO_VI$save_plot` *Save plot(s)*

Description

This is the default method of saving plot(s). It will use `save_plot()` to save the ggplot to a 420 (width) * 525 (height) PNG file. If the trained images are generated differently, one can override this method using `bandicoot::register_method()`.

Usage:

```
AUTO_VI$save_plot(p, path = NULL)
```

Arguments

`p`	ggplot. A plot or a list of plots.
`path`	Character. Character. Path(s) to save the image.

Value

The image path.

Examples

```
my_vi <- auto_vi(fitted_model = lm(speed ~ dist, data = cars))
p <- my_vi$plot_resid()
my_vi$save_plot(p)
```

AUTO_VI$summary *Summary of the object*

Description

The AUTO_VI$..str..() method provides a string representation of the object. If a check is performed, the string will contain some simple statistics of the check result. This method does this same thing as AUTO_VI$..str..(), but it returns an AUTO_VI_SUMMARY object which stores those statistics, such as sample quantiles of the distribution of null visual signal strength, in the object.

Usage:

```
AUTO_VI$summary()
```

Value

An AUTO_VI_SUMMARY object.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$check()
  myvi_summary <- myvi$summary()
  print(myvi_summary)
  names(myvi_summary)
}
```

AUTO_VI$summary_density_plot *Draw a summary density plot for the result*

Description

This function draws a summary density plot for the result.

Usage:

```
AUTO_VI$summary_plot(
  vss = self$check_result$observed$vss,
  null_dist = self$check_result$null$vss,
  boot_dist = self$check_result$boot$vss,
  p_value = self$check_result$p_value,
  likelihood_ratio = self$check_result$likelihood_ratio,
  density_alpha = 0.6
)
```

Arguments

`vss` Numeric. Observed visual signal strength.

`null_dist` Numeric. Null visual signal strength.

`boot_dist` Numeric. Bootstrapped visual signal strength.

`p_value` Numeric. P-value of the visual test. See also `AUTO_VI$p_value()`.

`likelihood_ratio`
Numeric. Likelihood ratio of the visual test. See also `AUTO_VI$likelihood_ratio()`.

`density_alpha` Numeric. Alpha value for the density.

Value

A `ggplot`.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$check()
  myvi$summary_density_plot()
}
```

`AUTO_VI$summary_plot` *Draw a summary plot for the result*

Description

This function draws a summary plot for the result.

Usage:

```
AUTO_VI$summary_plot(type = "auto", ...)
```

Arguments

`type` Character. Either "auto", "density" or "rank". Option "auto" will use the Boolean flag `self$check_result$lineup_check` to determine the correct option. See also `AUTO_VI$summary_density_plot()` and `AUTO_VI$summary_rank_plot()`.

`...` Arguments passed to `AUTO_VI$summary_density_plot()` or `AUTO_VI$summary_rank_plot()`.

Value

A `ggplot`.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$lineup_check()
  myvi$summary_plot()
}
```

AUTO_VI$summary_rank_plot

Draw a summary rank plot for the result

Description

This function draws a summary rank plot for the result.

Usage:

```
AUTO_VI$summary_plot(
  vss = self$check_result$observed$vss,
  null_dist = self$check_result$null$vss,
  p_value = self$check_result$p_value
)
```

Arguments

vss	Numeric. Observed visual signal strength.
null_dist	Numeric. Null visual signal strength.
p_value	Numeric. P-value of the visual test. See also AUTO_VI$p_value().

Value

A ggplot.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$lineup_check()
  myvi$summary_rank_plot()
}
```

AUTO_VI$vss *Predict the visual signal strength*

Description

This function predicts the visual signal strength.

Usage:

```
AUTO_VI$vss(
  p = self$plot_resid(),
  auxiliary = NULL,
  keras_model = self$keras_model,
  node_index = self$node_index,
  extract_feature_from_layer = NULL
)
```

Arguments

`p` ggplot/List/Data.frame/Array/Numpy array/String. The input can be

1. a `ggplot`,
2. a list of `ggplot`,
3. a data.frame containing `.resid` (residuals) and `.fitted` (fitted values) that can be passed to `AUTO_VI$plot_resid()`,
4. a 3D array representing an image,
5. a 4D array representing one or more images,
6. a path to an image,
7. a vector or a list of paths to images,
8. a numpy array.

`auxiliary` Dataframe. A dataframe of auxiliary values. This is only used when the keras model has multiple inputs. If it is not provided, the values will be automatically computed based on the residual plot of the fitted model. See also `AUTO_VI$auxiliary()`.

`keras_model` Keras model. A trained computer vision model.

`node_index` Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.

`extract_feature_from_layer` Character/Integer. A layer name or an integer layer index for extracting features from a layer.

Value

A tibble. The first column is `vss` which is the prediction, the rest of the columns are features extracted from a layer.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  myvi <- auto_vi(lm(dist ~ speed, data = cars), keras_model)

  myvi$vss()
}
```

check_python_library_available

Check python library availability

Description

This function checks if a python library is available. If the library can not be found by the importlib.util.find_spec method, then an error will be throw.

Usage

```
check_python_library_available(lib_name)
```

Arguments

lib_name	Character. A library name.

Value

No return. Called for side-effect.

Examples

```
try(check_python_library_available("numpy"))
```

get_keras_model

Download and load the keras model

Description

This functions download the keras model from the TengMCing/autovi_data Github repo using download.file() and load the model.

Usage

```
get_keras_model(model_name, format = "npz")
```

Arguments

`model_name`	String. The model name. See also `list_keras_model()`.
`format`	String. The model format to download. Either "npz", "SavedModel" or "keras".

Details

Note that the "SavedModel" and "keras" formats are not supported in `tensorflow` versions above 2.15, as `reticulate::import("tensorflow")$keras$models$load_model` encounters issues when loading models saved with the Keras 2 API. Instead, using the "npz" format allows for rebuilding the model from scratch and loading the weights from a ".npz" file, offering a more reliable alternative.

Value

A keras model.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) keras_model$summary()
```

KERAS_WRAPPER *KERAS_WRAPPER class environment*

Description

This is the class of keras wrapper, inherited from bandicoot::BASE. It is an environment with S3 class `bandicoot_oop`.

Usage

```
keras_wrapper(
  keras_model = NULL,
  node_index = 1L,
  env = new.env(parent = parent.frame()),
  init_call = sys.call()
)
```

Arguments

`keras_model`	Keras model. A trained computer vision model.
`node_index`	Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.
`env`	Environment. The instance environment.
`init_call`	Call. Contents of the `..init_call..`. It is recommended to leave it as default.

Value

An instance environment.

Functions

- keras_wrapper(): Class constructor, same as KERAS_WRAPPER$instantiate().

Class information

Parent classes:

- Direct:
  - bandicoot::BASE

New methods:

- G:
  - KERAS_WRAPPER$get_input_height()
  - KERAS_WRAPPER$get_input_width()
- I:
  - KERAS_WRAPPER$image_to_array()
  - KERAS_WRAPPER$..init..()
- L:
  - KERAS_WRAPPER$list_layer_name()
- P:
  - KERAS_WRAPPER$predict()
- S:
  - KERAS_WRAPPER$..str..()

KERAS_WRAPPER$..init..

Initialization method

Description

This function will be called after an instance is built. User input will be stored in the environment.

Usage:

```
KERAS_WRAPPER$..init..(keras_mod = NULL, node_index = 1L)
```

Arguments

keras_mod	Keras model. A trained computer vision model.
node_index	Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.

Value

Return the object itself.

Examples

```
keras_wrapper()
```

KERAS_WRAPPER$..str.. *String representation of the object*

Description

This function returns a string representation of the object.

Usage:

```
KERAS_WRAPPER$..str..()
```

Value

A string.

Examples

```
KERAS_WRAPPER$..str..()

wrapper <- keras_wrapper()
wrapper$..str..()
```

KERAS_WRAPPER$get_input_height *Get keras model input image height*

Description

This function get the input image height (the input shape is (batch_size, height, width, channels)) of a keras model.

Usage:

```
KERAS_WRAPPER$get_input_height(keras_model = self$keras_model)
```

Arguments

keras_model	Keras model. A trained computer vision model.

Value

An integer.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  keras_wrapper(keras_model)$get_input_height()
}
```

KERAS_WRAPPER$get_input_width

Get keras model input image width

Description

This function get the input image width (the input shape is (batch_size, height, width, channels)) of a keras model.

Usage:

```
KERAS_WRAPPER$get_input_width(keras_model = self$keras_model)
```

Arguments

keras_model	Keras model. A trained computer vision model.

Value

An integer.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  keras_wrapper(keras_model)$get_input_width()
}
```

KERAS_WRAPPER$image_to_array

Load an image as numpy array

Description

This function loads an image from file and convert it to a numpy array.

Usage:

```
KERAS_WRAPPER$image_to_array(
  path,
  height = self$get_input_height(),
  width = self$get_input_width()
)
```

Arguments

path	Character. Path to the image.
height	Integer. Target height of the image.
width	Integer. Target width of the image.

Value

A numpy array.

Examples

```
p <- ggplot2::ggplot(cars) + ggplot2::geom_point(ggplot2::aes(dist, speed))
path <- save_plot(p)
result <- try(KERAS_WRAPPER$image_to_array(path, 32L, 32L))
if (!inherits(result, "try-error")) {
  result
}
```

KERAS_WRAPPER$list_layer_name

List all layer names

Description

This function list all layer names of a keras model.

Usage:

```
KERAS_WRAPPER$list_layer_name(keras_model = self$keras_model)
```

Arguments

keras_model Keras model. A trained computer vision model.

Value

A vector of strings.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  keras_wrapper(keras_model)$list_layer_name()
}
```

KERAS_WRAPPER$predict *Predict visual signal strength*

Description

This function predicts the visual signal strength using the provided keras model, input array and optional auxiliary input array.

Usage:

```
KERAS_WRAPPER$predict(
  input_array,
  auxiliary = NULL,
  keras_model = self$keras_model,
  node_index = self$node_index,
  extract_featrue_from_layer = NULL
)
```

Arguments

input_array Array/Numpy array. An input array, usually of the shape (batch_size, height, width, channels).

auxiliary Array/Data frame. An auxiliary input array of the shape (batch_size, number_of_auxiliary_inputs). This is only needed if the keras model takes multiple inputs.

keras_model Keras model. A trained computer vision model.

node_index Integer. An index indicating which node of the output layer contains the visual signal strength. This is particularly useful when the keras model has more than one output nodes.

extract_feature_from_layer
Character/Integer. A layer name or an integer layer index for extracting features from a layer.

Value

A tibble. The first column is vss which is the prediction, the rest of the columns are features extracted from a layer.

Examples

```
keras_model <- try(get_keras_model("vss_phn_32"))
if (!inherits(keras_model, "try-error")) {
  wrapper <- keras_wrapper(keras_model)

  # Provide one 32 * 32 RGB image and one vector of length 5 as input
  wrapper$predict(input_array = array(255, dim = c(1, 32, 32, 3)),
                  auxiliary = matrix(1, ncol = 5))
}
```

list_keras_model *List all available pre-trained computer vision models*

Description

This function gets a table of available pre-trained computer vision models for predicting visual signal strength.

Usage

```
list_keras_model()
```

Value

A tibble of available model names and paths.

Examples

```
list_keras_model()
```

`remove_plot` *Remove a plot*

Description

This function removes a plot from a provided path.

Usage

```
remove_plot(path, check_ext = TRUE)
```

Arguments

`path`	Character. Path to the image.
`check_ext`	Boolean. Whether to check the file extension.

Value

No return. Called for side-effect.

Examples

```
p <- ggplot2::ggplot(cars) + ggplot2::geom_point(ggplot2::aes(dist, speed))
path <- save_plot(p)
remove_plot(path)
```

`save_plot` *Save plot(s)*

Description

This function save a plot of a list of plots to provided path(s).

Usage

```
save_plot(p, path = NULL, width = 7/5, height = 7/4, ...)
```

Arguments

`p`	`ggplot`. A plot.
`path`	Character. Path(s) to save the image.
`width`	Numeric. Width of the image.
`height`	Numeric. Height of the image.
`...`	Other arguments passed to `ggplot2::ggsave()`.

Value

The image path(s).

Examples

```
p <- ggplot2::ggplot(cars) + ggplot2::geom_point(ggplot2::aes(dist, speed))
save_plot(p)
```

Index