Package 'mrIML'

July 28, 2025

Type Package

Title Multi-Response (Multivariate) Interpretable Machine Learning

Version 2.1.0

Description Builds and interprets multi-response machine learning models using 'tidymodels' syntax. Users can supply a tidy model, and 'mrIML' automates the process of fitting multiple response models to multivariate data and applying interpretable machine learning techniques across them. For more details see Fountain-Jones (2021) <doi:10.1111/1755-0998.13495> and Fountain-Jones et al. (2024) <doi:10.22541/au.172676147.77148600/v1>.

Depends R (>= 3.5.0)

Imports dplyr, magrittr, rlang, ggplot2, patchwork, purrr, recipes, rsample, tibble, tidyr, tidyselect, tune, workflows, yardstick, flashlight, future.apply, MetricsWeighted, finetune, hstats

Suggests knitr, rmarkdown, testthat (>= 3.0.0), ape, vegan, hardhat, ggrepel, themis, MRFcov, lme4, randomForest, ggnetwork, igraph, tidymodels, tidyverse, parsnip, gridExtra, future, generics, missForest, kernelshap, shapviz

License MIT + file LICENSE

LazyData true

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.3.2

URL https://github.com/nickfountainjones/mrIML

BugReports https://github.com/nickfountainjones/mrIML/issues

Config/testthat/edition 3

Encoding UTF-8

Config/Needs/website rmarkdown

NeedsCompilation no

Author Nick Fountain-Jones [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-9248-8493>), Ryan Leadbetter [aut] (ORCID: <https://orcid.org/0000-0002-1942-3121>), Gustavo Machado [aut] (ORCID: <https://orcid.org/0000-0001-7552-6144>),

Chris Kozakiewicz [aut],
Nick Clark [aut]

Maintainer Nick Fountain-Jones <nick.fountainjones@utas.edu.au>

Repository CRAN

Date/Publication 2025-07-28 18:40:02 UTC

Contents

`filterRareCommon` *Filter rare response variables from the data*

Description

Filter rare response variables from the data

Usage

```
filterRareCommon(X, lower = lower, higher = higher)
```

Arguments

`X`	is a data.frame with rows as sites or individuals or populations and columns as loci or species OTUs.
`lower`	is the lower threshold value in which response varialkes are removed from the data.frame.
`higher`	is the upper threshold value in which response varialkes are removed from the data.frame.

Details

This function allows you to remove response units (OTUs or SNPs or species) from your response data as a preprocessing step. Suitable when the response is a binary outcome.

Value

A filtered tibble.

Examples

```
X <- filterRareCommon(Responsedata, lower = 0.4, higher = 0.7)
```

`mrBootstrap` *Bootstrap mrIML model predictions*

Description

This function bootstraps model predictions and generates variable profiles for each response variable.

Usage

```
mrBootstrap(mrIMLobj, num_bootstrap = 10, downsample = FALSE)
```

Arguments

`mrIMLobj`	A list object output by `mrIMLpredicts()`.
`num_bootstrap`	The number of bootstrap samples to generate (default: 10).
`downsample`	Logical. Should the bootstrap samples be downsampled? (default: FALSE).

Value

A list containing bootstrap samples of variable profiles for each response variable.

Examples

```
# Specify a random forest tidy model
mrIML_rf <- mrIML::mrIML_bird_parasites_RF

#future::plan(multisession, workers = 4)

mrIML_bootstrap <- mrIML_rf %>%
  mrBootstrap(
    num_bootstrap = 50
  )
```

`mrCoOccurNet` *Generate a MrIML co-occurrence network*

Description

This function generates a co-occurrence network from a provided list and calculates strength and directionality of the relationships. The output can be passed to **igraph** to plot a directed acyclic graph (DAG).

Usage

```
mrCoOccurNet(mrBootstrap_obj)
```

Arguments

`mrBootstrap_obj`
: A list of bootstrapped partial dependencies output from `mrBootstrap()`.

Value

A data frame representing the co-occurrence network with edge strengths and directionality.

Examples

```
library(tidymodels)
library(igraph)
library(ggnetwork)

mrIML_rf <- mrIML::mrIML_bird_parasites_RF

mrIML_rf_boot <- mrIML_rf %>%
  mrBootstrap()

assoc_net_filtered <- mrIML_rf_boot %>%
  mrCoOccurNet() %>%
  filter(mean_strength > 0.05)

# Convert to igraph
g <- graph_from_data_frame(
  assoc_net_filtered,
  directed = TRUE,
  vertices = names(mrIML_rf$Data$Y)
)
E(g)$Value <- assoc_net_filtered$mean_strength
E(g)$Color <- ifelse(
  assoc_net_filtered$direction == "negative",
  "blue", "red"
)
# Convert the igraph object to a ggplot object with NMDS layout
gg <- ggnetwork(g)
```

```
# Plot the graph
ggplot(
  gg,
  aes(x = x, y = y, xend = xend, yend = yend)
) +
  geom_edges(
    aes(color = Color, linewidth = Value),
    curvature = 0.2,
    arrow = arrow(length = unit(5, "pt"), type = "closed")
  ) +
  geom_nodes(
    color = "gray",
    size = degree(g, mode = "out") / 2
  ) +
  scale_color_identity() +
  theme_void() +
  theme(legend.position = "none") +
  geom_nodelabel_repel(
    aes(label = name),
    box.padding = unit(0.5, "lines"),
    size = 2,
    segment.colour = "black",
    colour = "white",
    fill = "grey36"
  )
```

`mrCovar` *Investigate partial dependencies of a covariate for mrIML JSDMs (Joint Species Distribution Models)*

Description

This function is a wrapper around `mrFlashlight()` that plots the covariate partial dependencies for a specified environmental/host variable. It also filters the taxa based on standard deviation thresholds.

Usage

```
mrCovar(mrIMLobj, var, sdthresh = 0.05, ...)
```

Arguments

`mrIMLobj`	A list object output by `mrIMLpredicts()`.
`var`	The variable of interest for calculating the profile.
`sdthresh`	The standard deviation threshold for filtering taxa (default: 0.05).
`...`	Arguments passed to `flashlight::light_profile()`

Value

A list of figures:

- `$partial_dep_curves`: The covariate partial dependence profiles for those models that meet the `sdthresh` requirement.
- `$partial_dep_avg`: The average partial dependence profile for all models. All individual model partial dependence profiles are silhouetted in the background.
- `$partial_dep_diff`: The distribution of the rates of change in probability for the specified variable (the derivatives of the PD curves). Useful to identify key threshold values in the variable.

Examples

```
mrIML_rf <-  mrIML::mrIML_bird_parasites_RF

covar_results <- mrIML_rf %>%
  mrCovar(var = "scale.prop.zos", sdthresh = 0.05)
```

---

mrFlashlight *Convert mrIML object into a* **flashlight** *object*

Description

A wrapper function around `flashlight::flashlight()` to run multi-response model-agnostic interpretable machine learning analyses. The output can be interrogated using the core functionality of **flashlight**: see `vignette("flashlight", package = "flashlight")`.

Usage

```
mrFlashlight(mrIMLobj, response = "multi", index = 1, predict_function = NULL)
```

Arguments

`mrIMLobj`	A list object output by `mrIMLpredicts()`.
`response`	A character string indicating the type of response: `"single"` selects one response (indicated by `index`) and `"multi"` selects all responses.
`index`	A numeric value used when `response` is `"single"` to select which response column in the data to create a flashlight object for.
`predict_function`	A function specifying a user-defined prediction function (optional).

Value

A flashlight or multi-flashlight object.

Examples

```
library(flashlight)
library(ggplot2)

mrIML_rf <- mrIML::mrIML_bird_parasites_RF

fl <- mrFlashlight(
  mrIML_rf,
  response = "multi",
  index = 1
)

# Performance comparison
fl %>%
  light_performance(
    metrics = list(`ROC AUC` = MetricsWeighted::AUC)
  ) %>%
  plot() +
  ylim(0, 1)

# Partial dependence curves
fl %>%
  light_profile(data = cbind(mrIML_rf$Data$X, mrIML_rf$Data$Y), "scale.prop.zos") %>%
  plot()

# Two-way partial dependence
fl %>%
  light_profile2d(c("scale.prop.zos", "Plas")) %>%
  plot()
```

`mrIMLperformance` *Calculate general performance metrics of a mrIML model*

Description

Summarizes the performance of a `mrIML` object created using `mrIMLpredicts()` in a way that allows for easy comparison of different models. For regression models, root mean squared error (RMSE) and R-squared are reported, while for classification models, area under the ROC curve (AUC), Matthews correlation coefficient (MCC), positive predictive value (PPV), specificity, and sensitivity are reported.

Usage

```
mrIMLperformance(mrIMLobj)
```

Arguments

`mrIMLobj`	A list object created by `mrIMLpredicts()` containing multi-response models.

Value

A list with two slots:

- `$model_performance`: A **tibble** of commonly used metrics that can be used to compare model performance of classification models. Performance metrics are based on the test data defined during `mrIMLpredicts()`.
- `$global_performance_summary`: A global performance metric: the average of a performance metric over all response models. MCC is used for classification models and RMSE for regression models.

Examples

```
mrIML_rf <- mrIML::mrIML_bird_parasites_RF

perf <- mrIMLperformance(mrIML_rf )
perf[[1]]
perf[[2]]
```

`mrIMLpredicts` *Generates a multi-response predictive model*

Description

This function fits separate classification/regression models, specified in the **tidymodels** framework, for each response variable in a data set. This is the core function of `mrIML`.

Usage

```
mrIMLpredicts(
  X,
  X1 = NULL,
  Y,
  Model,
  balance_data = "no",
  dummy = FALSE,
  prop = 0.7,
  tune_grid_size = 10,
  k = 10,
  racing = TRUE
)
```

Arguments

`Y`, `X`, `X1`	Data frames containing the response, predictor, and the joint response variables (i.e. the responses that are also to be used as predictors if fitting GN model) respectively. If `X1` is not provided then a standard multi-response model will be fit to the data (e.g. the response models are independant of one another conditional on the predictors supplied in X). See **Details** section below.

`Model`	Any model from the **tidymodels** package. See **Examples**.
`balance_data`	A character string: • "up": up-samples the data to equal class sizes. • "down": down-samples the data to equal class sizes. • "no": leaves the data as is. "no" is the default value.
`dummy`	A logical value indicating if `recipes::step_dummy()` should be included in the data recipe.
`prop`	A numeric value between 0 and 1. Defines the training-testing data proportion to be used, which defaults to `prop = 0.7`.
`tune_grid_size`	A numeric value that sets the grid size for hyperparameter tuning. Larger grid sizes increase computational time. Ignored if `racing = TRUE`.
`k`	A numeric value. Sets the number of folds in the cross-validation. 10-fold CV is the default.
`racing`	A logical value. If `TRUE`, `mrIML` performs the grid search using the `finetune::tune_race_anova()` method; otherwise, `tune::tune_grid()` is used. `racing = TRUE` is now the default method of tuning.

Details

`mrIMLpredicts` fits the supplied tidy model to each response variable in the data frame `Y`. If only `X` (a data frame of predictors) is supplied, then independent models are fit, i.e., the other response variables are not used as predictors. If `X1` (a data frame of all or select response variables) is supplied, then those response variables are also used as predictors in the response models. For example, supplying `X1` means that a co-occurrence model is fit.

If `balance_data = "up"`, then `themis::step_rose()` is used to upsample the dataset; however, we generally recommend using `balance_data = "no"` in most cases.

Value

A list object with three slots:

- `$Model`: The **tidymodels** object that was fit.
- `$Data`: A list of the raw data.
- `$Fits`: A list of the fitted models for each response variable.

Examples

```
data <- MRFcov::Bird.parasites

# Define the response variables of interest
Y <- data %>%
  dplyr::select(-scale.prop.zos) %>%
  dplyr::select(order(everything()))

# Define the predictors
X <- data %>%
  dplyr::select(scale.prop.zos)
```

```
# Specify a random forest tidy model
model_lm <- parsnip::logistic_reg()

# Fitting independent multi-response model ----------------------------------
MR_model <- mrIMLpredicts(
  X = X,
  Y = Y,
  Model = model_lm,
  prop = 0.7,
  k = 5,
  racing = FALSE
)

# Fitting a graphical network model -----------------------------------------
# Define the dependent response variables (all in this case)
if (identical(Sys.getenv("NOT_CRAN"), "true")) {
X1 <- Y

GN_model <- mrIMLpredicts(
  X = X,
  Y = Y,
  X1 = X1,
  Model = model_lm,
  prop = 0.7,
  k = 5,
  racing = FALSE
)
}
```

`mrIML_bird_parasites_LM` *An example mrIML model fit to MRFcov::Bird.parasites*

Description

```
data <- MRFcov::Bird.parasites
Y <- data %>%
  dplyr::select(-scale.prop.zos) %>%
  dplyr::select(order(everything()))
X <- data %>%
  dplyr::select(scale.prop.zos)
```

Usage

```
mrIML_bird_parasites_LM
```

Format

An object of class `list` of length 3.

Details

model_lm <- logistic_reg() %>% set_engine("glm")

mrIML_bird_parasites_LM <- mrIMLpredicts(X = X, Y = Y, X1 = Y, Model = model_lm, prop = 0.7, k = 2, racing = FALSE)

`mrIML_bird_parasites_RF`

An example mrIML model fit to MRFcov::Bird.parasites

Description

```
data <- MRFcov::Bird.parasites
Y <- data %>%
  dplyr::select(-scale.prop.zos) %>%
  dplyr::select(order(everything()))
X <- data %>%
  dplyr::select(scale.prop.zos)
```

Usage

```
mrIML_bird_parasites_RF
```

Format

An object of class `list` of length 3.

Details

model_rf <- parsnip::rand_forest(trees = 10, # 10 trees are set for brevity. Aim to start with 1000 mode = "classification", mtry = tune::tune(), min_n = tune::tune(), engine = "randomForest")

mrIML_bird_parasites <- mrIMLpredicts(X = X, Y = Y, X1 = Y, Model = model_rf, prop = 0.7, k = 2, racing = FALSE)

`mrInteractions` *Calculate and visualize feature interactions*

Description

A wrapper around `hstats::hstats()`. Calculates and visualizes H-statistics for interactions in the model using bootstrapping. See `help("hstats")` for details on H-statistics.

Usage

```
mrInteractions(mrIMLobj, num_bootstrap = 1, feature = NULL, top_int = 10)
```

Arguments

`mrIMLobj`	A list object output by `mrIMLpredicts()`.
`num_bootstrap`	The number of bootstrap samples to generate (default: 1).
`feature`	The response model for which detailed interaction plots should be generated.
`top_int`	The number of top interactions to display (default: 10).

Value

A list containing:

- `$p_h2`: An ordered bar plot of the variability in each response model that is unexplained by the main effects.
- `$p_h2_overall`: An ordered bar plot of the percentage of prediction variability that can be attributed to interactions with each predictor for the model specified by `feature`.
- `$p_h2_pairwise`: An ordered bar plot of the strength of the two-way interactions in the model specified by `feature`. The strength of an interaction is taken to be the un-normalized square root of the H2-pairwise statistic (which is on the prediction scale).
- `$h2_df`: A data frame of the H2 statistics for each response model, along with bootstraps if applicable.
- `$h2_overall_df`: A data frame of the H2-overall statistics for the variable in each response model, along with bootstraps if applicable.
- `$h2_pairwise_df`: A data frame of the H2-pairwise statistics for the variable in each response model, along with bootstraps if applicable.

Examples

```
mrIML_rf <- mrIML::mrIML_bird_parasites_RF

mrIML_interactions_rf <- mrInteractions(
  mrIML_rf,
  num_bootstrap = 50,
  feature = "Plas"
)
```

```
mrIML_interactions_rf[[1]]
mrIML_interactions_rf[[2]]
mrIML_interactions_rf[[3]]
```

`mrPdPlotBootstrap` *Bootstrap Partial Dependence Plots*

Description

This function extracts and plots the bootrapped partial dependence functions calculated by `mrBootstrap()` for each response variable.

Usage

```
mrPdPlotBootstrap(
  mrIML_obj,
  mrBootstrap_obj,
  vi_obj = NULL,
  target,
  global_top_var = 2
)
```

Arguments

`mrIML_obj`	A list object returned by `mrIMLpredicts()`.
`mrBootstrap_obj`	A list object returned by `mrBootstrap()`.
`vi_obj`	A list object returned by `mrVip()`. If `vi_obj` is not provided, then it is created inside `mrPD_bootstrap` by running `mrVip()`.
`target`	The target variable for generating plots.
`global_top_var`	The number of top variables to consider (default: 2).

Value

A list with two elements:

- `[[1]]`: A data frame of the partial dependence grid for each response model, predictor variable, and bootstrap.
- `[[2]]`: A list of partial dependence plots for each predictor variable in the `target` response model.

Examples

```
mrIML_rf <- mrIML::mrIML_bird_parasites_RF

mrIML_rf_boot <- mrIML_rf %>%
  mrBootstrap(num_bootstrap = 50)

mrIML_rf_PD <- mrPdPlotBootstrap(
  mrIML_rf,
  mrIML_rf_boot,
  target = "Plas",
  global_top_var = 4
)

head(mrIML_rf_PD[[1]])
mrIML_rf_PD[[2]]
```

`mrPerformancePlot` *Plot Model Performance Comparison*

Description

Create visualizations to compare the performance of two models based on their performance metrics generated by mrIMLperformance.

Usage

```
mrPerformancePlot(
  ModelPerf1 = NULL,
  ModelPerf2 = NULL,
  mode = "classification"
)
```

Arguments

`ModelPerf1`, `ModelPerf2`	Two data frames of model performance metrics to compare. The data frames are created by mrIMLperformance, see **Examples**.
`mode`	A character string describing the mode of the models. Should be either "regression" or "classification". The default is "classification".

Value

A list containing:

- `$performance_plot`: A box plot of model performance metrics.
- `$performance_diff_plot`: A bar plot of the differences in performance metrics.
- `$performance_diff_df`: A data frame in wide format containing model performance metrics and their differences.

Examples

```
MR_perf_rf <-  mrIML::mrIML_bird_parasites_RF %>%
  mrIMLperformance()
MR_perf_lm <- mrIML::mrIML_bird_parasites_LM%>%
  mrIMLperformance()

perf_comp <- mrPerformancePlot(
  ModelPerf1 = MR_perf_rf,
  ModelPerf2 = MR_perf_lm
)
```

`mrShapely` *Generate SHAP (SHapley Additive exPlanations) Plots for Multiple Models and Responses*

Description

This function generates SHAP (SHapley Additive exPlanations) plots for multiple models and responses.

Usage

```
mrShapely(
  mrIML_obj,
  taxa = NULL,
  kind = "beeswarm",
  max_display = 15L,
  plot_feature_effects = TRUE,
  plot_dependencies = TRUE,
  plot_2D_dependencies = TRUE
)
```

Arguments

`mrIML_obj`	A list object output by `mrIMLpredicts()`.
`taxa`	An optional character vector specifying which responses to include.
`kind`	A character string passed to `shapviz::sv_importance()` specifying the type of plot parameter (e.g., "beeswarm" for feature effect plot, "bar" for variable importance plot, or "both").
`max_display`	An integer passed to `shapviz::sv_importance()` specifying the maximum number of features to display.
`plot_feature_effects`	A logical indicating whether to generate feature effect plots (default is TRUE).
`plot_dependencies`	A logical indicating whether to generate dependency plots (default is TRUE).
`plot_2D_dependencies`	A logical indicating whether to generate interaction plots (default is TRUE).

Value

A list object where the first element returns the SHAP results, and the following elements contain the feature-effect, 1D-dependencies, and 2D-dependencies if they were set to TRUE in the input.

Examples

```
mrIML_model <- mrIML::mrIML_bird_parasites_RF

shapely_plots_list <- mrShapely(mrIML_model, plot_2D_dependencies = FALSE)
```

---

mrVip *Calculates and helps interpret variable importance for* mrIML *models.*

Description

Summarizes variable importance in a mrIML model at both a global (across all the response models) and local (for individual response models) level. This can be done for a plain mrIML model or bootstrap results obtained from mrBootstrap().

Usage

```
mrVip(
  mrIMLobj,
  mrBootstrap_obj = NULL,
  threshold = 0.1,
  global_top_var = 10,
  local_top_var = 5,
  taxa = NULL,
  model_perf = NULL
)
```

Arguments

mrIMLobj	A list object output by mrIMLpredicts().
mrBootstrap_obj	A list of bootstrap results output by mrBootstrap().
threshold	The performance threshold for response models (AUC for classification and R2 for regression). Only response models that meet this performance criterion are plotted.
global_top_var	The number of top global variables to display (default: 10).
local_top_var	The number of top local variables for each response to display (default: 5).
taxa	A character string identifying which response model should be plotted.
model_perf	A list object containing model performance metrics output by mrIMLperformance(). If not supplied, then mrIMLperformance() is run inside mrvip() to get performance metrics.

Value

A list containing:

- $vi_data: Variable importance data in its raw form (including bootstrap samples if mrBootstrap_obj was supplied).
- $vi_tbl: Variable importance data point estimates.
- $vi_plot: A grouped plot of the most important variables both globally and for the individual response models.

Examples

```
# Without bootstrap
mrIML_rf <- mrIML::mrIML_bird_parasites_RF

vip_results <-mrVip(mrIML_rf, taxa = "Plas")

# With bootstrap

mrIML_rf_boot <- mrIML_rf %>%
  mrBootstrap(num_bootstrap = 5)

mrIML_rf_vip <- mrVip(
  mrIML_rf,
  mrBootstrap_obj = mrIML_rf_boot
)
```

---

mrVipPCA *Principal Component Analysis of mrIML variable importance*

Description

Principal Component Analysis of mrIML variable importance

Usage

```
mrVipPCA(mrVip_obj)
```

Arguments

mrVip_obj	A list returned by mrVip().

Value

A list of PCA results:

- $PCA_plot: Side-by-side plots of the different response models on the first two principal components (PCs) and a Scree plot.
- $PC_outliers: A list of the models flagged as outliers on at least one of the PCs.
- $eigenvalues: The eigenvalues associated with the principal components.
- $PC_scores: The PC scores of each response model.

Examples

```
# Without bootstrap
mrIML_rf <- mrIML::mrIML_bird_parasites_RF

mrIML_rf_vip <- mrVip(mrIML_rf, taxa = "Plas")

vipPCA_results <- mrIML_rf_vip %>%
 mrVipPCA()
```

`readSnpsPed` *Conversion to single column per locus from plink file via LEA functionality*

Description

Conversion to single column per locus from plink file via LEA functionality

Usage

```
readSnpsPed(pedfile, mapfile)
```

Arguments

`pedfile`	A file location.
`mapfile`	A file location.

Details

Function to import SNP data from a plink format into a format suitable for MrIML predicts (presence/absence of an alelle for each locus). Currently if there is missing data (NAs) it either imputes them as the mode or leaves them. A histogram is also produced of the missing data.

Value

A tibble.

Examples

```
snps <- readSnpsPed("FILE_NAME.plink.ped", "FILE_NAME.plink.map.map")
X <- filterRareCommon(snps, lower = 0.4, higher = 0.7)
```

`resist_components` *Calculates resistance components from a list of pairwise resistance surfaces*

Description

Calculates resistance components from a list of pairwise resistance surfaces

Usage

```
resist_components(foldername = foldername, p_val = p_val, cl = NULL)
```

Arguments

`foldername`	A `character` this is the location where the resistance surfaces are stored.
`p_val`	A `numeric` this sets the significance threshold for axes in explaining variance in the original resistance matrix based on redundancy analysis. In effect this filters out axes that don't explain variance.
`cl`	A parallel argument to be passed to `vegan::capscale()` if parallel compute is wanted.

Details

Outputs a data frame of significant resistance components for each matrix in the target folder. These data can be combined with non-pairwise matrix data.

Value

A data frame.

Examples

```
Y <- resist_components(filename = 'FILE_PATH', p_val = 0.01)
```

Index