Package ‘APS’

May 6, 2026

Title Analysing Prediction Stability of Non-Deterministic Prediction Models

Version 1.0.1

Maintainer Thomas Martin Lange <thomas.lange@uni-goettingen.de>

Description Provides methods to analyse the stability of non-deterministic prediction models. Prediction stability is quantified either as data-based prediction stability (phi) or as model-based prediction stability (psi). The package implements measures for categorical, ordinal, and metric predictions based on repeated model fitting and corresponding predictions. Methods are based on Lange et al. (2025) <doi:10.1186/s12859-025-06097-1>.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.6.0)

Imports stats

Suggests ranger, covr, rmarkdown, spelling, testthat

NeedsCompilation no

Author Thomas Martin Lange [cre, aut] (ORCID: <https://orcid.org/0000-0003-4351-7950>), Felix Heinrich [ctb] (ORCID: <https://orcid.org/0000-0002-6093-8522>)

LazyData true

Repository CRAN

Date/Publication 2026-04-16 09:02:07 UTC

Contents

calculate_phi 2
calculate_psi 3
example_prediction_matrix 5
MaizeBreedingTrial 6

Index **8**

`calculate_phi` *Calculate the data-based prediction stability (phi)*

Description

Calculate the data-based prediction stability (phi) for a non-deterministic prediction model

Usage

```
calculate_phi(prediction_matrix, training_observations, scale, ordinal_levels = NULL)
```

Arguments

`prediction_matrix`	A matrix or data frame containing repeated predictions for the objects in the test data set. Rows correspond to objects (e.g., samples or individuals), columns correspond to predictions for same set of objects in repetitions of the model training and the prediction process using the same non-deterministic prediction model.
`training_observations`	A vector with the observations in the training data set used to train the prediction model.
`scale`	A value specifying the measurement scale of the predictions. Allowed values are "metric", "ordinal", and "categorical".
`ordinal_levels`	If an ordinal scale is used, the levels of the ordinal variable need to be given as a vector in the correct order

Value

The function returns a list containing the calculated prediction stability, describing the derived measure (phi), as well as the calculated observed and expected instability of the prediction model.

Examples

```
# Load example data included in the package
data(example_prediction_matrix)
data(phenotype)
y_Training <- phenotype[1:200, 2]

# Calculate the data-based prediction stability (phi)
stability_results = calculate_phi(example_prediction_matrix, y_Training, scale = "metric")

# Print the results
stability_results

# Extract the prediction stability value
stability_results$prediction_stability
```

```
# Example: Constructing a prediction matrix using using random forest as example

if(requireNamespace("ranger", quietly = TRUE)){

  # Load the ranger package for random forest
  library(ranger)

  # Load the data sets included in the package
  data(genotype)
  data(phenotype)
  test_IDs <- genotype[201:250, 1]
  X_Test <- genotype[201:250, -1]
  X_Training <- genotype[1:200, -1]
  y_Training <- phenotype[1:200, 2]

  # Create a data frame to save the results
  D_preds_test <- data.frame(test_IDs)

  # Set a seed for reproducibility
  set.seed(123)
  for(rep in 1:10){

    # Repeatedly fit the model to the training data
    RFmodel <- ranger(x = X_Training, y = y_Training, verbose = FALSE)

    # Predict the response of the test objects
    test_predictions <- predict(RFmodel, data = X_Test)$predictions

    # Save the predictions for the test objects
    D_preds_test <- cbind(D_preds_test, test_predictions)
    names(D_preds_test)[rep+1] = paste0("predictions_", sprintf("%02d", rep))
  }

  # Create the prediction matrix
  prediction_matrix <- as.matrix(D_preds_test[, -1])

  # Run the calculate_phi function and store the results
  stability_results = calculate_phi(prediction_matrix, y_Training, scale = "metric")

  # Print the results
  stability_results

  # Extract the prediction stability value
  stability_results$prediction_stability
  }
```

`calculate_psi` *Calculate the model-based prediction stability (psi)*

Description

Calculate the model-based prediction stability (psi) for a non-deterministic prediction model

Usage

```
calculate_psi(prediction_matrix, prediction_space, scale, ordinal_levels = NULL)
```

Arguments

`prediction_matrix`	A matrix or data frame containing repeated predictions for the objects in the test data set. Rows correspond to objects (e.g., samples or individuals), columns correspond to predictions for same set of objects in repetitions of the model training and the prediction process using the same non-deterministic prediction model.
`prediction_space`	A vector, matrix, or data frame containing out-of-sample predictions for the training data (e.g., from cross-validation or OOB).
`scale`	A value specifying the measurement scale of the predictions. Allowed values are "metric", "ordinal", and "categorical".
`ordinal_levels`	If an ordinal scale is used, the levels of the ordinal variable need to be given as a vector in the correct order

Value

The function returns a list containing the calculated prediction stability, describing the derived measure (phi or psi), as well as the calculated observed and expected instability of the prediction model.

Examples

```
# Load example data included in the package
data(example_prediction_matrix)
data(example_prediction_space)

# Calculate the model-based prediction stability (psi)
stability_results = calculate_psi(example_prediction_matrix,
                                  example_prediction_space, scale = "metric")

# Print the results
stability_results

# Extract the prediction stability value
stability_results$prediction_stability


# Constructing a prediction matrix and training prediction matrix using random forest as example

if(requireNamespace("ranger", quietly = TRUE)){

  # Load the ranger package for random forest
```

```
library(ranger)

# Load the data sets included in the package
data(genotype)
data(phenotype)
test_IDs <- genotype[201:250, 1]
training_IDs = genotype[1:200, 1]
X_Test <- genotype[201:250, -1]
X_Training <- genotype[1:200, -1]
y_Training <- phenotype[1:200, 2]

# Create the data frames to save the results
D_preds_test <- data.frame(test_IDs)
D_preds_train = data.frame(training_IDs)

# Set a seed for reproducibility
set.seed(123)
for(rep in 1:10){

  # Repeatedly fit the model to the training data
  RFmodel <- ranger(x = X_Training, y = y_Training, verbose = FALSE)

  # Save the predictions for the test objects
  test_predictions <- predict(RFmodel, data = X_Test)$predictions
  D_preds_test <- cbind(D_preds_test, test_predictions)
  names(D_preds_test)[rep+1] = paste0("predictions_", sprintf("%02d", rep))

  # Save the OOB predictions from random forest
  training_predictions <- RFmodel$predictions
  D_preds_train <- cbind(D_preds_train, training_predictions)
  names(D_preds_train)[rep+1] = paste0("predictions_", sprintf("%02d", rep))
}

# Create the prediction matrices
prediction_matrix <- as.matrix(D_preds_test[, -1])
prediction_space <- as.matrix(D_preds_train[, -1])

# Run the calculate_psi function and store the results
stability_results = calculate_psi(prediction_matrix,
                                  prediction_space, scale = "metric")

# Print the results
stability_results

# Extract the prediction stability value
stability_results$prediction_stability
}
```

`example_prediction_matrix` *Example prediction matrices for non-deterministic models*

Description

Precomputed prediction matrices generated from repeated predictions of a non-deterministic model (random forest) applied to the `genotype` and `phenotype` data sets included in the package.

Usage

```
example_prediction_matrix
```

Format

example_prediction_matrix A numeric matrix with 50 rows and 10 columns.

example_prediction_space A numeric matrix with 200 rows and 10 columns.

Details

- `example_prediction_matrix`: Predictions for test objects. The matrix has 50 rows (one per test object) and 10 columns (one per repeated prediction). - `example_prediction_space`: Predictions for training objects. The matrix has 200 rows (one per training object) and 10 columns (one per repeated prediction).

Source

Created from the `genotype` and `phenotype` data sets included in the package, using repeated random forest predictions (10 repetitions, seed set to 123 for reproducibility).

`MaizeBreedingTrial` *Maize breeding trial data*

Description

A simulated maize breeding data set consisting of phenotypic and genotypic information for a set of individuals. The data are split into two objects: `phenotype` and `genotype` which are intended to be used together.

Format

`phenotype` A data frame with 250 rows and 4 variables:

- **ID** Genotype identifier
- **yield** Simulated maize yield
- **leafblight** Simulated scoring data regarding the infection of maize genotypes with leaf blight
- **maturityclass** Simulated maturity class of the genotypes

`genotype` A data frame with 250 rows and 5001 variables:

- **ID** Genotype identifier
- **SNP_YLD_*** Genomic markers affecting the maize yield
- **SNP_LBS_*** Genomic markers affecting the leaf blight scoring
- **SNP_MAT_*** Genomic markers affecting the maturity class
- **SNP_LD_*** Genomic markers being highly correlated with the causal genomic markers
- **SNP_RND_*** Genomic markers containing random values

Details

The `phenotype` data frame contains observed trait values while `genotype` contains SNP marker information for the same individuals. Rows describe the same individuals across both data sets. The datasets can be accessed directly after loading the package: `phenotype` and `genotype`.

References

This artificial data set was created for the APS package.

Examples

```
head(phenotype)
genotype[1:5,1:5]
```

Index

∗ **datasets**
    example_prediction_matrix, 6
    MaizeBreedingTrial, 6

calculate_phi, 2
calculate_psi, 3

example_prediction_matrix, 5
example_prediction_space
        *(*example_prediction_matrix*)*, 6

genotype *(*MaizeBreedingTrial*)*, 6

MaizeBreedingTrial, 6

phenotype *(*MaizeBreedingTrial*)*, 6