Package 'dsos'

July 22, 2025

Title Dataset Shift with Outlier Scores

Version 0.1.2

Description Test for no adverse shift in two-sample comparison when we have a training set, the reference distribution, and a test set. The approach is flexible and relies on a robust and powerful test statistic, the weighted AUC. Technical details are in Kamulete, V. M. (2021) <doi:10.48550/arXiv.1908.04000>. Modern notions of outlyingness such as trust scores and prediction uncertainty can be used as the underlying scores for example.

License GPL (>= 3)

URL https://github.com/vathymut/dsos

BugReports https://github.com/vathymut/dsos/issues

Imports data.table (>= 1.14.6), future.apply (>= 1.10.0), ggplot2 (>= 3.4.0), scales (>= 1.2.1), simctest (>= 2.6), stats (>= 4.2.1)

Suggests fdrtool (>= 1.2.17), knitr (>= 1.42), rmarkdown (>= 2.20), testthat (>= 3.1.6)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.2.3

NeedsCompilation no

Author Vathy M. Kamulete [aut, cre] (ORCID: <https://orcid.org/0000-0002-4451-3743>), Royal Bank of Canada (RBC) [cph] (Research supported and funded by RBC)

Maintainer Vathy M. Kamulete <vathymut@gmail.com>

Repository CRAN

Date/Publication 2023-02-19 07:30:06 UTC

Contents

`as_bf` *Convert P-value to Bayes Factor*

Description

Convert P-value to Bayes Factor

Usage

```
as_bf(pvalue)
```

Arguments

`pvalue`	P-value.

Value

Bayes Factor (scalar value).

References

Marsman, M., & Wagenmakers, E. J. (2017). *Three insights from a Bayesian interpretation of the one-sided P value*. Educational and Psychological Measurement, 77(3), 529-539.

See Also

[as_pvalue()] to convert Bayes factor to p-value.

Other bayesian-test: `as_pvalue()`, `bf_compare()`, `bf_from_os()`

Examples

```
library(dsos)
bf_from_pvalue <- as_bf(pvalue = 0.5)
bf_from_pvalue
```

`as_pvalue` *Convert Bayes Factor to P-value*

Description

Convert Bayes Factor to P-value

Usage

```
as_pvalue(bf)
```

Arguments

`bf`	Bayes factor.

Value

p-value (scalar value).

References

Marsman, M., & Wagenmakers, E. J. (2017). *Three insights from a Bayesian interpretation of the one-sided P value*. Educational and Psychological Measurement, 77(3), 529-539.

See Also

[as_bf()] to convert p-value to Bayes factor.

Other bayesian-test: `as_bf()`, `bf_compare()`, `bf_from_os()`

Examples

```
library(dsos)
pvalue_from_bf <- as_pvalue(bf = 1)
pvalue_from_bf
```

at_from_os *Asymptotic Test from Outlier Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training set, `x_train` as the as the reference. The method checks whether the test set, `x_test`, is worse off relative to this reference set. The function `scorer` assigns an outlier score to each instance/observation in both training and test set.

Usage

```
at_from_os(os_train, os_test)
```

Arguments

`os_train`	Outlier scores in training (reference) set.
`os_test`	Outlier scores in test set.

Details

Li and Fine (2010) derives the asymptotic null distribution for the weighted AUC (WAUC), the test statistic. This approach does not use permutations and can, as a result, be much faster because it sidesteps the need to refit the scoring function `scorer`. This works well for large samples. The prefix *at* stands for asymptotic test to tell it apart from the prefix *pt*, the permutation test.

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

The outlier scores should all mimic out-of-sample behaviour. Mind that the training scores are not in-sample and thus, biased (overfitted) while the test scores are out-of-sample. The mismatch – in-sample versus out-of-sample scores – voids the test validity. A simple fix for this is to get the training scores from an indepedent (fresh) validation set; this follows the train/validation/test sample splitting convention and the validation set is effectively the reference set or distribution in this case.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

See Also

[at_oob()] for variant requiring a scoring function. [pt_from_os()] for permutation test with the outlier scores.

Other asymptotic-test: `at_oob()`

Examples

```
library(dsos)
set.seed(12345)
os_train <- rnorm(n = 100)
os_test <- rnorm(n = 100)
test_result <- at_from_os(os_train, os_test)
test_result
```

`at_oob` *Asymptotic Test With Out-Of-Bag Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training set, `x_train` as the as the reference. The method checks whether the test set, `x_test`, is worse off relative to this reference set. The function `scorer` assigns an outlier score to each instance/observation in both training and test set.

Usage

```
at_oob(x_train, x_test, scorer)
```

Arguments

`x_train`	Training (reference/validation) sample.
`x_test`	Test sample.
`scorer`	Function which returns a named list with outlier scores from the training and test sample. The first argument to `scorer` must be `x_train`; the second, `x_test`. The returned named list contains two elements: *train* and *test*, each of which is a vector of (outlier) scores. See notes for more information.

Details

Li and Fine (2010) derives the asymptotic null distribution for the weighted AUC (WAUC), the test statistic. This approach does not use permutations and can, as a result, be much faster because it sidesteps the need to refit the scoring function `scorer`. This works well for large samples. The prefix *at* stands for asymptotic test to tell it apart from the prefix *pt*, the permutation test.

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

The scoring function, `scorer`, predicts out-of-bag scores to mimic out-of-sample behaviour. The suffix *oob* stands for out-of-bag to highlight this point. This out-of-bag variant avoids refitting the underlying algorithm from `scorer` at every permutation. It can, as a result, be computationally appealing.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

See Also

[pt_oob()] for (faster) p-value approximation via out-of-bag predictions. [pt_refit()] for (slower) p-value approximation via refitting.

Other asymptotic-test: `at_from_os()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
setosa <- iris[1:50, 1:4] # Training sample: Species == 'setosa'
versicolor <- iris[51:100, 1:4] # Test sample: Species == 'versicolor'

# Using fake scoring function
scorer <- function(tr, te) list(train=runif(nrow(tr)), test=runif(nrow(te)))
oob_test <- at_oob(setosa, versicolor, scorer = scorer)
oob_test
```

bf_compare *Bayesian and Frequentist Test from Outlier Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training (outlier) scores, `os_train`, as the reference. The method checks whether the test scores, `os_test`, are worse off relative to the training set.

Usage

```
bf_compare(os_train, os_test, threshold = 1/12, n_pt = 4000)
```

Arguments

`os_train`	Outlier scores in training (reference) set.
`os_test`	Outlier scores in test set.
`threshold`	Threshold for adverse shift. Defaults to 1 / 12, the asymptotic value of the test statistic when the two samples are drawn from the same distribution.
`n_pt`	The number of permutations.

Details

This compares the Bayesian to the frequentist approach for convenience. The Bayesian test mimics 'bf_from_os()' and the frequentist one, 'pt_from_os()'. The Bayesian test computes Bayes factors based on the asymptotic (defaults to 1/12) and the exchangeable threshold. The latter calculates the threshold as the median weighted AUC (WAUC) after `n_pt` permutations assuming outlier scores are exchangeable. This is recommended for small samples. The frequentist test converts the one-sided (one-tailed) p-value to the Bayes factor - see `as_bf` function.

Value

A list of factors (BF) for 3 different test specifications:

- `frequentist`: Frequentist BF.
- `bayes_noperm`: Bayestion BF test with asymptotic threshold.
- `bayes_perm`: Bayestion BF with exchangeable threshold.

Notes

The outlier scores should all mimic out-of-sample behaviour. Mind that the training scores are not in-sample and thus, biased (overfitted) while the test scores are out-of-sample. The mismatch – in-sample versus out-of-sample scores – voids the test validity. A simple fix for this is to get the training scores from an indepedent (fresh) validation set; this follows the train/validation/test sample splitting convention and the validation set is effectively the reference set or distribution in this case.

See Also

[bf_from_os()] for bayes factor, the Bayesian test. [pt_from_os()] for p-value, the frequentist test.

Other bayesian-test: `as_bf()`, `as_pvalue()`, `bf_from_os()`

Examples

```
library(dsos)
set.seed(12345)
os_train <- rnorm(n = 100)
os_test <- rnorm(n = 100)
bayes_test <- bf_compare(os_train, os_test)
bayes_test
# To run in parallel on local cluster, uncomment the next two lines.
# library(future)
# future::plan(future::multisession)
parallel_test <- bf_compare(os_train, os_test)
parallel_test
```

---

bf_from_os *Bayesian Test from Outlier Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training (outlier) scores, `os_train`, as the reference. The method checks whether the test scores, `os_test`, are worse off relative to the training set.

Usage

```
bf_from_os(os_train, os_test, n_pt = 4000, threshold = 1/12)
```

Arguments

`os_train`	Outlier scores in training (reference) set.
`os_test`	Outlier scores in test set.
`n_pt`	The number of permutations.
`threshold`	Threshold for adverse shift. Defaults to 1 / 12, the asymptotic value of the test statistic when the two samples are drawn from the same distribution.

Details

The posterior distribution of the test statistic is based on `n_pt` (boostrap) permutations. The method uses the Bayesian bootstrap as a resampling procedure as in Gu et al (2008). Johnson (2005) shows to leverage (turn) a test statistic into a Bayes factor. The test statistic is the weighted AUC (WAUC).

Value

A named list of class `outlier.bayes` containing:

- `posterior`: Posterior distribution of WAUC test statistic
- `threshold`: WAUC threshold for adverse shift
- `adverse_probability`: probability of adverse shift
- `bayes_factor`: Bayes factor
- `outlier_scores`: outlier scores from training and test set

Notes

The outlier scores should all mimic out-of-sample behaviour. Mind that the training scores are not in-sample and thus, biased (overfitted) while the test scores are out-of-sample. The mismatch – in-sample versus out-of-sample scores – voids the test validity. A simple fix for this is to get the training scores from an indepedent (fresh) validation set; this follows the train/validation/test sample splitting convention and the validation set is effectively the reference set or distribution in this case.

References

Kamulete, V. M. (2023). *Are you OK? A Bayesian test for adverse shift*. Manuscript in preparation.

Johnson, V. E. (2005). *Bayes factors based on test statistics*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(5), 689-701.

Gu, J., Ghosal, S., & Roy, A. (2008). *Bayesian bootstrap estimation of ROC curve*. Statistics in medicine, 27(26), 5407-5420.

See Also

Other bayesian-test: `as_bf()`, `as_pvalue()`, `bf_compare()`

Examples

```
library(dsos)
set.seed(12345)
os_train <- rnorm(n = 100)
os_test <- rnorm(n = 100)
bayes_test <- bf_from_os(os_train, os_test)
bayes_test
# To run in parallel on local cluster, uncomment the next two lines.
# library(future)
# future::plan(future::multisession)
parallel_test <- bf_from_os(os_train, os_test)
parallel_test
```

plot.outlier.bayes *Plot Bayesian test for no adverse shift.*

Description

Plot Bayesian test for no adverse shift.

Usage

```
## S3 method for class 'outlier.bayes'
plot(x, ...)
```

Arguments

x	A `outlier.bayes` result from test of no adverse shift.
...	Placeholder to be compatible with S3 method `plot`.

Value

A **ggplot2** plot with outlier scores and p-value.

See Also

Other s3-method: `plot.outlier.test()`, `print.outlier.bayes()`, `print.outlier.test()`

Examples

```
set.seed(12345)
os_train <- rnorm(n = 3e2)
os_test <- rnorm(n = 3e2)
test_to_plot <- bf_from_os(os_train, os_test)
plot(test_to_plot)
```

plot.outlier.test *Plot frequentist test for no adverse shift.*

Description

Plot frequentist test for no adverse shift.

Usage

```
## S3 method for class 'outlier.test'
plot(x, ...)
```

Arguments

x	A `outlier.test` result from test of no adverse shift.
...	Placeholder to be compatible with S3 method `plot`.

Value

A **ggplot2** plot with outlier scores and p-value.

See Also

Other s3-method: `plot.outlier.bayes()`, `print.outlier.bayes()`, `print.outlier.test()`

Examples

```
set.seed(12345)
os_train <- rnorm(n = 3e2)
os_test <- rnorm(n = 3e2)
test_to_plot <- at_from_os(os_train, os_test)
# Also: pt_from_os(os_train, os_test) for permutation test
plot(test_to_plot)
```

`print.outlier.bayes` *Print Bayesian test for no adverse shift.*

Description

Print Bayesian test for no adverse shift.

Usage

```
## S3 method for class 'outlier.bayes'
print(x, ...)
```

Arguments

x	A `outlier.test` object from a D-SOS test.
...	Placeholder to be compatible with S3 method `plot`.

Value

Print to screen: display Bayes factor and other information.

See Also

Other s3-method: `plot.outlier.bayes()`, `plot.outlier.test()`, `print.outlier.test()`

Examples

```
set.seed(12345)
os_train <- rnorm(n = 3e2)
os_test <- rnorm(n = 3e2)
test_to_print <- bf_from_os(os_train, os_test)
test_to_print
```

`print.outlier.test` *Print frequentist test for no adverse shift.*

Description

Print frequentist test for no adverse shift.

Usage

```
## S3 method for class 'outlier.test'
print(x, ...)
```

Arguments

`x`	A `outlier.test` object from a D-SOS test.
`...`	Placeholder to be compatible with S3 method `plot`.

Value

Print to screen: display p-value and other information.

See Also

Other s3-method: `plot.outlier.bayes()`, `plot.outlier.test()`, `print.outlier.bayes()`

Examples

```
set.seed(12345)
os_train <- rnorm(n = 3e2)
os_test <- rnorm(n = 3e2)
test_to_print <- at_from_os(os_train, os_test)
# Also: pt_from_os(os_train, os_test) for permutation test
test_to_print
```

pt_from_os *Permutation Test from Outlier Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training (outlier) scores, `os_train`, as the reference. The method checks whether the test scores, `os_test`, are worse off relative to the training set.

Usage

```
pt_from_os(os_train, os_test, n_pt = 2000)
```

Arguments

`os_train`	Outlier scores in training (reference) set.
`os_test`	Outlier scores in test set.
`n_pt`	The number of permutations.

Details

The null distribution of the test statistic is based on `n_pt` permutations. For speed, this is implemented as a sequential Monte Carlo test with the **simctest** package. See Gandy (2009) for details. The prefix *pt* refers to permutation test. This approach does not use the asymptotic null distribution for the test statistic. This is the recommended approach for small samples. The test statistic is the weighted AUC (WAUC).

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

The outlier scores should all mimic out-of-sample behaviour. Mind that the training scores are not in-sample and thus, biased (overfitted) while the test scores are out-of-sample. The mismatch – in-sample versus out-of-sample scores – voids the test validity. A simple fix for this is to get the training scores from an indepedent (fresh) validation set; this follows the train/validation/test sample splitting convention and the validation set is effectively the reference set or distribution in this case.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

See Also

[pt_oob()] for variant requiring a scoring function. [at_from_os()] for asymptotic test with the outlier scores.

Other permutation-test: `pt_oob()`, `pt_refit()`

Examples

```
library(dsos)
set.seed(12345)
os_train <- rnorm(n = 100)
os_test <- rnorm(n = 100)
null_test <- pt_from_os(os_train, os_test)
null_test
```

---

pt_oob *Permutation Test With Out-Of-Bag Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training set, `x_train` as the as the reference. The method checks whether the test set, `x_test`, is worse off relative to this reference set. The function `scorer` assigns an outlier score to each instance/observation in both training and test set.

Usage

```
pt_oob(x_train, x_test, scorer, n_pt = 2000)
```

Arguments

`x_train`	Training (reference/validation) sample.
`x_test`	Test sample.
`scorer`	Function which returns a named list with outlier scores from the training and test sample. The first argument to `scorer` must be `x_train`; the second, `x_test`. The returned named list contains two elements: *train* and *test*, each of which is a vector of corresponding (outlier) scores. See notes below for more information.
`n_pt`	The number of permutations.

Details

The null distribution of the test statistic is based on `n_pt` permutations. For speed, this is implemented as a sequential Monte Carlo test with the **simctest** package. See Gandy (2009) for details. The prefix *pt* refers to permutation test. This approach does not use the asymptotic null distribution for the test statistic. This is the recommended approach for small samples. The test statistic is the weighted AUC (WAUC).

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

The scoring function, `scorer`, predicts out-of-bag scores to mimic out-of-sample behaviour. The suffix *oob* stands for out-of-bag to highlight this point. This out-of-bag variant avoids refitting the underlying algorithm from `scorer` at every permutation. It can, as a result, be computationally appealing.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

See Also

[pt_refit()] for (slower) p-value approximation via refitting. [at_oob()] for p-value approximation from asymptotic null distribution.

Other permutation-test: `pt_from_os()`, `pt_refit()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
idx <- sample(nrow(iris), 2 / 3 * nrow(iris))
iris_train <- iris[idx, ]
iris_test <- iris[-idx, ]
# Use a synthetic (fake) scoring function for illustration
scorer <- function(tr, te) list(train=runif(nrow(tr)), test=runif(nrow(te)))
pt_test <- pt_oob(iris_train, iris_test, scorer = scorer)
pt_test
```

pt_refit *Permutation Test By Refitting*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training set, `x_train` as the as the reference. The method checks whether the test set, `x_test`, is worse off relative to this reference set. The function `scorer` assigns an outlier score to each instance/observation in both training and test set.

Usage

```
pt_refit(x_train, x_test, scorer, n_pt = 2000)
```

Arguments

`x_train`	Training (reference/validation) sample.
`x_test`	Test sample.
`scorer`	Function which returns a named list with outlier scores from the training and test sample. The first argument to `scorer` must be `x_train`; the second, `x_test`. The returned named list contains two elements: *train* and *test*, each of which is a vector of corresponding (outlier) scores. See notes below for more information.
`n_pt`	The number of permutations.

Details

The null distribution of the test statistic is based on `n_pt` permutations. For speed, this is implemented as a sequential Monte Carlo test with the **simctest** package. See Gandy (2009) for details. The prefix *pt* refers to permutation test. This approach does not use the asymptotic null distribution for the test statistic. This is the recommended approach for small samples. The test statistic is the weighted AUC (WAUC).

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

The scoring function, `scorer`, predicts out-of-sample scores by refitting the underlying algorithm from `scorer` at every permutation The suffix *refit* emphasizes this point. This is in contrast to the out-of-bag variant, `pt_oob`, which only fits once. This method can be be computationally expensive.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

See Also

[pt_oob()] for (faster) p-value approximation via out-of-bag predictions. [at_oob()] for p-value approximation from asymptotic null distribution.

Other permutation-test: `pt_from_os()`, `pt_oob()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
setosa <- iris[1:50, 1:4] # Training sample: Species == 'setosa'
versicolor <- iris[51:100, 1:4] # Test sample: Species == 'versicolor'
scorer <- function(tr, te) list(train=runif(nrow(tr)), test=runif(nrow(te)))
pt_test <- pt_refit(setosa, versicolor, scorer = scorer)
pt_test
```

---

wauc_from_os *Weighted AUC from Outlier Scores*

Description

Computes the weighted AUC with the weighting scheme described in Kamulete, V. M. (2021). This assumes that the training set is the reference distribution and specifies a particular functional form to derive weights from threshold scores.

Usage

```
wauc_from_os(os_train, os_test, weight = NULL)
```

Arguments

`os_train`	Outlier scores in training (reference) set.
`os_test`	Outlier scores in test set.
`weight`	Numeric vector of weights of length `length(os_train) + length(os_test)`. The first `length(os_train)` weights belongs to the training set, the rest is for the test set. If `NULL`, the default, all weights are set to 1.

Value

The weighted AUC (scalar value) given the weighting scheme.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Examples

```
library(dsos)
set.seed(12345)
os_train <- rnorm(n = 100)
os_test <- rnorm(n = 100)
test_stat <- wauc_from_os(os_train, os_test)
```

Index