# Package: AEenrich (via r-universe)

September 9, 2024

**Version** 1.1.0

**Title** Adverse Event Enrichment Tests

**Type** Package

**Description** We extend existing gene enrichment tests to perform adverse event enrichment analysis. Unlike the continuous gene expression data, adverse event data are counts. Therefore, adverse event data has many zeros and ties. We propose two enrichment tests. One is a modified Fisher's exact test based on pre-selected significant adverse events, while the other is based on a modified Kolmogorov-Smirnov statistic. We add Covariate adjustment to improve the analysis.``Adverse event enrichment tests using VAERS" Shuoran Li, Lili Zhao (2020) <arXiv:2007.02266>.

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**Biarch** true

**Depends** R (>= 3.5.0)

**Imports** dplyr, magrittr, qvalue, doParallel, tidyr, modelr, foreach, rlang, utils

**URL** https://github.com/umich-biostatistics/AEenrich

**BugReports** https://github.com/umich-biostatistics/AEenrich/issues

**RoxygenNote** 7.1.1

**Roxygen** list(markdown = TRUE)

**Suggests** testthat

**Repository** https://umich-biostatistics.r-universe.dev

**RemoteUrl** https://github.com/umich-biostatistics/aeenrich

**RemoteRef** HEAD

**RemoteSha** 78580f863a71d5fb8412525ca1779a99a0a1019d

# Contents

AEenrich-package . . . . . 2
count_cases . . . . . 3
covid1 . . . . . 4
covid2 . . . . . 5
enrich . . . . . 6
group . . . . . 8

**Index** **9**

---

`AEenrich-package` *AEenrich: Adverse Event Enrichment Tests*

---

## Description

The count_cases function is used to convert data on the report level to aggregated data, grouping by specified covariates.

Use the function `count_cases` to convert report level data into aggregated data.

See our Github home page or run ?count_cases for examples.

Perform Adverse Event Enrichment Tests The enrich function is used to perform Adverse event (AE) enrichment analysis. Unlike the continuous gene expression data, AE data are counts. Therefore, AE data has many zeros and ties. We propose two enrichment tests. AEFisher is a modified Fisher's exact test based on pre-selected significant AEs, while AEKS is based on a modified Kolmogorov-Smirnov statistic.

Use the function `enrich` to fit models and inspect results.

See our Github home page or run ?enrich for examples.

## Author(s)

**Maintainer**: Michael Kleinsasser <mkleinsa@umich.edu>

Authors:

- Shuoran Li <shuoranl@umich.edu>
- Hongfan Chen <chenhf@umich.edu>
- Lili Zhao <zhaolili@med.umich.edu>

**Maintainer**: Michael Kleinsasser <mkleinsa@umich.edu>

Authors:

- Shuoran Li <shuoranl@umich.edu>
- Hongfan Chen <chenhf@umich.edu>
- Lili Zhao <zhaolili@med.umich.edu>

## See Also

Useful links:

- https://github.com/umich-biostatistics/AEenrich
- Report bugs at https://github.com/umich-biostatistics/AEenrich/issues

Useful links:

- https://github.com/umich-biostatistics/AEenrich
- Report bugs at https://github.com/umich-biostatistics/AEenrich/issues

---

`count_cases` *Convert data on the report level to aggregated data.*

---

## Description

The count_cases function is used to convert data on the report level to aggregated data, grouping by specified covariates.

## Usage

```
count_cases(
  data,
  drug.case = drug.case,
  drug.control = NULL,
  covar_disc = NULL,
  covar_cont = NULL,
  breaks = NULL,
  cores = detectCores(),
  min_AE = 10
)
```

## Arguments

| | |
|---|---|
| `data` | a data.frame with at least 3 columns, consisting data on the report level, having ID, Drug type and AE name as the first 3 columns with covariates(optional) followed. The order of columns is not interchangeable. |
| `drug.case` | a character string for the target drug of interest. |
| `drug.control` | a character string for the reference drug. If NULL(default), all other drugs combined are the reference. |
| `covar_disc` | a character vector of categorical covariates. |
| `covar_cont` | a character vector of continuous covariates. |

| | |
|---|---|
| breaks | a list consists of vectors used for creating specific bins to transform continuous covariates into categorical. Breaks Should have the same length as covar_cont. Given a vector of non-decreasing breakpoints in `breaks[i]`, find the interval containing each element of `covar_cont[i]`; i.e., for each index j in `breaks[i]`, value j is assigned to `covar_cont[i]` if and only if `breaks[i][j] <= covar_cont[i] < breaks[i][j+1]`. |
| cores | the number of cores to use for parallel execution. |
| min_AE | the minimum number of cases required to start counting for a specific AE. Default 10. |

## Value

A **data.frame** consists of aggregated data.

The returned data.frame contains the following columns:

- DRUG_TYPE: type of the drug, DrugYes for target drug and DrugNo for referenced drug
- AE_NAME: the name of the adverse event
- AEyes: number of observations that have this AE
- AEno: number of observations that do not have this AE
- covariates: covariates specifed by user

## Examples

```
# count_cases(data = covid1, drug.case = "COVID19", drug.control = "OTHER",
#             covar_cont = c("AGE"), covar_disc = c("SEX"),
#             breaks = list(c(16,30,50,65,120)))
```

---

# covid1 *Covid Vaccine Adverse Event Data*

---

## Description

Adverse event data in the long format. Each row is a single adverse event, along with covariates.

## Usage

```
covid1
```

## Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 12500 rows and 5 columns.

### Details

- VAERS_ID Event ID
- VAX_LABEL Vaccine type
- AE_NAME Adverse event name
- AGE covariate
- SEX covariate

---

`covid2` *Covid Vaccine Adverse Event Data*

---

### Description

Adverse event data in the short format. Each row is a count of adverse events with the given name.

### Usage

```
covid2
```

### Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 2656 rows and 6 columns.

### Details

- DRUG_TYPE Vaccine type
- AE_NAME Adverse event name
- AEYes Number of observations that have this AE
- AENo Number of observations that do not have this AE
- AGE covariate
- SEX covariate

enrich *Perform Adverse Event Enrichment Tests*

## Description

The enrich function is used to perform Adverse event (AE) enrichment analysis. Unlike the continuous gene expression data, AE data are counts. Therefore, AE data has many zeros and ties. We propose two enrichment tests. AEFisher is a modified Fisher's exact test based on pre-selected significant AEs, while AEKS is based on a modified Kolmogorov-Smirnov statistic.

## Usage

```
enrich(
  data,
  dd.group,
  drug.case,
  drug.control = NULL,
  method = "aeks",
  n_perms = 1000,
  covar = NULL,
  p = 0,
  q.cut = 0.1,
  or.cut = 1.5,
  zero = FALSE,
  min_size = 5,
  min_AE = 10,
  cores = detectCores()
)
```

## Arguments

| | |
|---|---|
| data | a data.frame. Two data types are allowed. Type I data consisting data on the report level, having ID, Drug type and AE name as the first 3 columns with covariates(optional) followed. Type II data have drug type and AE name as the first two columns, with the 3rd and 4th Columns giving the numbers of successes(have AE) and failures(Do not have AE) respectively, then followed by covariates. See example data for details. |
| dd.group | a data.frame with AE name and Group name. This data.frame have the group information for each individual AE. |
| drug.case | a character string for the target drug of interest. |
| drug.control | a character string for the reference drug. If NULL(default), all other drugs combined are the reference. |
| method | a character string specifying the method for the enrichment test. It must take "aeks" (default) or "aefisher"; "aeks" is the rank-based enrichment test, and "aefisher" is the Fisher enrichment test. See details described in the paper (see reference section of this document). |

| | |
|---|---|
| n_perms | an integer value specifying the number of permutations in permutation test. |
| covar | a character vector specifying the columns of covariates, default NULL. |
| p | a numerical value to control the weight of the step, can take any value between 0 and 1. If 0(default), reduces to the standard Kolmogorov-Smirnov statistics. |
| q.cut | a numerical value specifying the significance cut for q value of AEs in aefisher. |
| or.cut | a numerical value specifying the significance cut for odds ratio of AEs in aefisher. |
| zero | logical, default FALSE.If TRUE, add zero indicator to enrichment score. |
| min_size | the minimum size of group required for enrichment analysis. |
| min_AE | the minimum number of cases required to start counting for a specific AE. |
| cores | the number of cores to use for parallel execution. |

## Value

A list containing 2 data.frames named **Final_result** and **AE_info**.

The **Final_result** data.frame contains the following columns:

- GROUP_NAME: AE group names
- ES: enrichment score
- p_value: p value of the enrichment test
- GROUP_SIZE: number of AEs per group

The **AE_info** contains the following columns:

- AE_NAME: AE names
- OR: odds ratio for each individual AE
- p_value: p value for AE-drug association
- 95Lower: lower bound of 95 percent confidence interval of odds ratio
- 95Lower: upper bound of 95 percent confidence interval of odds ratio
- se(logOR): standard error of log odds ratio

## References

Li, S. and Zhao, L. (2020). Adverse event enrichment tests using VAERS. arXiv:2007.02266.

Subramanian, A.e.a. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci U S A. Proceedings of the National Academy of Sciences. 102. 15545-15550.

Tian, Lu & Greenberg, Steven & Kong, Sek Won & Altschuler, Josiah & Kohane, Isaac & Park, Peter. (2005). Discovering statistically significant pathways in expression profiling studies. Proceedings of the National Academy of Sciences of the United States of America. 102. 13544-9. 10.1073/pnas.0506577102.

## Examples

```
# AEKS

### Type I data: data on report level
# enrich(data = covid1, covar = c("SEX", "AGE"), p = 0, method = "aeks",
#        n_perms = 1000, drug.case = "COVID19", dd.group = group, cores = 2,
#        drug.control = "OTHER", min_size = 5, min_AE = 10, zero = FALSE)

## Type II data: aggregated data
# enrich(data = covid2, covar = c("SEX", "AGE"), p = 0, method = "aeks",
#        n_perms = 1000, drug.case = "DrugYes", dd.group = group, cores = 2,
#        drug.control = "DrugNo", min_size = 5, min_AE = 10)

# AEFISHER
## Type I data: data on report level
# enrich(data = covid1, covar = c("SEX", "AGE"), p = 0, method = "aefisher",
#        n_perms = 1000, drug.case = "COVID19", dd.group = group,
#        drug.control = "OTHER", min_size = 5, min_AE = 10, q.cut = 0.05,
#        or.cut = 1.5, cores = 2)

## Type II data: aggregated data
# enrich(data = covid2, covar = c("SEX", "AGE"), p = 0, method = "aefisher",
#        n_perms = 1000, drug.case = "DrugYes", dd.group = group,
#        drug.control = "DrugNo", min_size = 5, min_AE = 10, cores = 2)
```

---

# group *Group Structure Data*

---

## Description

Identifies which group each set of adverse events belongs.

## Usage

```
group
```

## Format

An object of class `NULL` of length 0.

## Details

- AE_NAME Adverse event name
- GROUP_NAME Group name

# Index

∗ **datasets**
covid1, 4
covid2, 5
group, 8

AEenrich *(*AEenrich-package*)*, 2
AEenrich-package, 2

count_cases, 3
covid1, 4
covid2, 5

enrich, 6

group, 8