Package: MDDC (via r-universe)

March 5, 2025

Type Package

Title Modified Detecting Deviating Cells Algorithm in Pharmacovigilance

Version 1.0.0

Maintainer Anran Liu <anranliu@buffalo.edu>

Description Methods for detecting signals related to (adverse event, medical product e.g. drugs, vaccines) pairs, a data generation function for simulating pharmacovigilance datasets, and various utility functions. For more details please see Liu A., Mukhopadhyay R., and Markatou M. <doi:10.48550/arXiv.2410.01168>.

URL https://github.com/niuniular/MDDC

BugReports https://github.com/niuniular/MDDC/issues

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Imports doParallel, foreach, ggplot2, MASS, Rcpp, stats

Suggests knitr, rmarkdown, roxygen2, spelling, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppEigen

VignetteBuilder knitr

Depends R (>= 3.5.0)

Config/testthat/edition 3

Language en-US

Repository https://niuniular.r-universe.dev

RemoteUrl https://github.com/niuniular/mddc

RemoteRef HEAD

RemoteSha 925f9e24a2e013b76e0f046b1296ec5269dc7ece

Contents

`MDDC-package` *Modified Detecting Deviating Cells Algorithm in Pharmacovigilance*

Description

In this package, we present the Modified Detecting Deviating Cells (MDDC) algorithm for adverse event identification. For a certain time period, the spontaneous reports can be extracted from the safety database and depicted as an $I \times J$ contingency table, where:

- I denotes the total number of adverse events (AEs).
- J denotes the total number of drugs or vaccines.

The cell counts n_{ij} represent the total number of reported cases corresponding to the j-th drug/vaccine and the i-th AE. We are interested in identifying which (AE, drug or vaccine) pairs are signals. Signals refer to potential adverse events that may be caused by a drug or vaccine. In the contingency table setting, signals refer to the cells where n_{ij} is abnormally higher than the expected values.

The Detecting Deviating Cells (DDC) algorithm, originally proposed by Rousseeuw and Bossche (2018), was designed for outlier identification in multivariate datasets. However, the original DDC algorithm assumes multivariate normality of the data, with cutoffs based on this assumption. In contrast, the MDDC algorithm is designed for the discrete nature of adverse event data in pharmacovigilance, which clearly do not follow a multivariate normal distribution.

Our Modified Detecting Deviating Cells (MDDC) algorithm has the following characteristics:

1. It is easy to compute.
2. It considers AE relationships.
3. It uses data-driven cutoffs.

4. It is independent of the use of ontologies.

The MDDC algorithm consists of five steps. The first two steps identify univariate outliers via cutoffs, and the next three steps evaluate the signals using AE correlations. More details can be found in the MDDC algorithm documentation.

For an introduction to the 'MDDC' package, see the vignette: Usage Examples for MDDC in R.

Details

This work has been supported by the Food and Drug Administration and the Kaleida Health Foundation.

Author(s)

Anran Liu, Raktim Mukhopadhyay, and Marianthi Markatou

Maintainer: Anran Liu <anranliu@buffalo.edu>

References

Liu, A., Mukhopadhyay, R., and Markatou, M. (2024). MDDC: An R and Python package for adverse event identification in pharmacovigilance data. arXiv preprint. arXiv:2410.01168.

Liu, A., Markatou, M., Dang, O., and Ball, R. (2024). Pattern discovery in pharmacovigilance through the Modified Detecting Deviating Cells (MDDC) algorithm. Technical Report, Department of Biostatistics, University at Buffalo.

Rousseeuw, P. J., and Van den Bossche, W. (2018). Detecting deviating data cells. Technometrics, 60(2), 135-145.

See Also

Useful links:

- https://github.com/niuniular/MDDC
- Report bugs at https://github.com/niuniular/MDDC/issues

`betablocker500` *FDA dataset for beta blockers with 500 adverse events*

Description

A 501 by 9 data matrix of a contingency table processed from the FDA Adverse Event Reporting System (FAERS) database. This dataset covers a specific period from Q1 2021 to Q4 2023.

Usage

```
betablocker500
```

Format

A data matrix with 501 rows and 9 columns.

Details

A 501 by 9 data matrix of a contingency table from the FDA Adverse Event Reporting System (FAERS) database, covering the period from Q1 2021 to Q4 2023.

The 500 rows correspond to the AEs with the highest overall frequency (row marginals) reported during the period, and 1 row for Other AEs. The reported AEs - "Off label use" and "Drug ineffective" have been excluded.

The dataset includes the following 9 columns: Acebutolol, Atenolol, Bisoprolol, Carvedilol, Metoprolol, Nadolol, Propranolol, Timolol, and Other.

The marginal totals for each column are as follows: Acebutolol: 62,164, Atenolol: 36,619, Bisoprolol: 134,297, Carvedilol: 35,922, Metoprolol: 88,387, Nadolol: 11,191, Propranolol: 56,444, Timolol: 16,077, Other: 76,926,859.

Also refer to the supplementary material of:

Ding, Y., Markatou, M., & Ball, R. (2020). An evaluation of statistical approaches to postmarketing surveillance. Statistics in Medicine, 39(7), 845-874

for the data generation process. The quarterly files can be found in https://fis.fda.gov/extensions/FPD-QDE-FAERS/FPD-QDE-FAERS.html.

Examples

```
data(betablocker500)
head(betablocker500)
```

---

check_and_fix_contin_table

Verifying and correcting the input I by J contingency table

Description

Verify and correct the input I by J contingency table to ensure it is properly formatted as a data matrix with row (adverse event) and column (drug) names.

Usage

```
check_and_fix_contin_table(contin_table)
```

Arguments

contin_table	A data matrix or a data frame of an I by J contingency table with or without prespecified row and column names.

Value

A data matrix of an I by J contingency table with row and column names.

Examples

```
# Create a 6 by 4 data matrix
set.seed(42)
dat_mat <- matrix(rpois(6 * 4, 20), nrow = 6)
dat_mat

# Check the format of the data matrix and assign row and column names
contin_table <- check_and_fix_contin_table(dat_mat)
contin_table
```

`find_optimal_coef` *Find Adaptive Boxplot Coefficient 'coef' via Grid Search*

Description

This function performs a grid search to determine the optimal adaptive boxplot coefficient 'coef' for each column of a contingency table, ensuring the target false discovery rate (FDR) is met.

Usage

```
find_optimal_coef(
  contin_table,
  n_sim = 1000,
  target_fdr = 0.05,
  grid = 0.1,
  col_specific_cutoff = TRUE,
  exclude_small_count = TRUE
)
```

Arguments

`contin_table`	A matrix representing the $I \times J$ contingency table.
`n_sim`	An integer specifying the number of simulated tables under the assumption of independence between rows and columns. Default is 1000.
`target_fdr`	A numeric value specifying the desired level of false discovery rate (FDR). Default is 0.05.
`grid`	A numeric value representing the size of the grid added to the default value of `coef = 1.5` as suggested by Tukey. Default is 0.1.
`col_specific_cutoff`	Logical. If TRUE, then a single value of the coefficient is returned for the entire dataset, else when FALSE specific values corresponding to each of the columns are returned.

exclude_small_count
: A logical indicating whether to exclude cells with counts smaller than or equal to five when computing boxplot statistics. Default is TRUE.

Value

A list with the following components:

coef: A numeric vector containing the optimal coefficient 'coef' for each column of the input contingency table.

FDR: A numeric vector with the corresponding false discovery rate (FDR) for each column.

Examples

```
# This example uses the statin49 data
data(statin49)
find_optimal_coef(statin49)
```

---

generate_contin_table_with_clustered_AE

Generate simulated contingency tables with the option of incorporating adverse event correlation within clusters.

Description

Generate simulated contingency tables with the option of incorporating adverse event correlation within clusters.

Usage

```
generate_contin_table_with_clustered_AE(
  row_marginal,
  column_marginal,
  signal_mat,
  contin_table = NULL,
  AE_idx = NULL,
  n_rep = 1,
  rho = NULL,
  seed = NULL
)
```

Arguments

row_marginal
: Marginal sums for the rows of the contingency table.

column_marginal
: Marginal sums for the columns of the contingency table.

signal_mat	A data matrix of the same dimension as the contingency table with entries representing the signal strength. The values should be greater or equal to 1, where 1 indicates no signal, and values greater than 1 indicate signal.
contin_table	A data matrix of an I x J contingency table with row (adverse event) and column (drug or vaccine) names, of which the row and column marginals are used to generate the simulated data. Please first check the input contingency table using the function check_and_fix_contin_table(). Default is NULL.
AE_idx	A data frame or list. In case of data frame it must contain two variables idx and AE, where idx indicates the cluster index (a number), and AE lists the adverse event names. See the statin49_AE_idx for statin49 data as an example. In case of a list, make sure the cluster indices are aligned with the corresponding row marginal.
n_rep	Number of contingency tables to be generated.
rho	A numeric value, matrix, or NULL indicating the correlation structure.If a numeric value (float or int) is provided, it represents the correlation value rho to be used between all elements within each cluster specified by AE_idx. If a matrix is provided, it must be a square matrix with dimensions equal to the number of rows in contin_table. In this case, rho defines the correlation structure directly, and AE_idx is not used. If rho is NULL, a covariance matrix is generated based on the correlation coefficients of contin_table.
seed	An optional integer to set the seed for reproducibility. If NULL, no seed is set.

Value

A list of n_rep simulated contingency tables.

Examples

```
# using statin49 as an example
data(statin49)
data(statin49_AE_idx)

# Prepare a matrix of signal strength with the same dimension as
# statin49, where 1 indicates no signal and values > 1 indicate
# signal
lambda_matrix <- matrix(1, nrow = nrow(statin49), ncol = ncol(statin49))

# Assign the cell (45,1) with signal strength 4
lambda_matrix[45, 1] <- 4

# Generate 5 simulated tables
set.seed(123)
simulated_tables <- generate_contin_table_with_clustered_AE(
  contin_table = statin49,
  n_rep = 5,
  AE_idx = statin49_AE_idx,
  rho = 0.5,
  signal_mat = lambda_matrix
)
```

`generate_contin_table_with_clustered_AE_with_tol` *Generate simulated contingency tables with the option of incorporating adverse event correlation within clusters and tolerance for total report count.*

Description

Generate simulated contingency tables with the option of incorporating adverse event correlation within clusters and tolerance for total report count.

Usage

```
generate_contin_table_with_clustered_AE_with_tol(
  row_marginal,
  column_marginal,
  signal_mat,
  tol = 0.1,
  contin_table = NULL,
  AE_idx = NULL,
  n_rep = 1,
  rho = NULL
)
```

Arguments

`row_marginal`	Marginal sums for the rows of the contingency table.
`column_marginal`	Marginal sums for the columns of the contingency table.
`signal_mat`	A data matrix of the same dimension as the contingency table with entries representing the signal strength. The values should be greater or equal to 1, where 1 indicates no signal, and values greater than 1 indicate signal.
`tol`	Tolerance for the total report count, expressed in terms of the Relative Total Difference (RTD), defined as: $$RTD = \frac{\lvert n_{..}^{orig} - n_{..}^{sim} \rvert}{n_{..}^{orig}} \times 100$$ This represents the difference between the total number of reports in the simulated datasets and the original input total number of reports. Sufficiently low tolerance will generate tables with total report counts equal to the actual supplied value. Default is 0.1.
`contin_table`	A data matrix of an I x J contingency table with row (adverse event) and column (drug or vaccine) names, of which the row and column marginals are used to generate the simulated data. Please first check the input contingency table using the function `check_and_fix_contin_table()`. Default is NULL.

AE_idx	A data frame or list. In case of data frame it must contain two variables idx and AE, where idx indicates the cluster index (a number), and AE lists the adverse event names. See the statin49_AE_idx for statin49 data as an example. In case of a list, make sure the cluster indices are aligned with the corresponding row marginal.
n_rep	Number of contingency tables to be generated.
rho	A numeric value, matrix, or NULL indicating the correlation structure.If a numeric value (float or int) is provided, it represents the correlation value rho to be used between all elements within each cluster specified by AE_idx. If a matrix is provided, it must be a square matrix with dimensions equal to the number of rows in contin_table. In this case, rho defines the correlation structure directly, and AE_idx is not used. If rho is NULL, a covariance matrix is generated based on the correlation coefficients of contin_table.

Value

A list of n_rep simulated contingency tables.

Examples

```
# using statin49 as an example
data(statin49)
data(statin49_AE_idx)

# Prepare a matrix of signal strength with the same dimension as
# statin49, where 1 indicates no signal and values > 1 indicate
# signal
lambda_matrix <- matrix(1, nrow = nrow(statin49), ncol = ncol(statin49))
lambda_matrix[1, 1] <- 4

# Generate 5 simulated tables
simulated_tables <- generate_contin_table_with_clustered_AE_with_tol(
  contin_table = statin49,
  signal_mat = lambda_matrix,
  n_rep = 5,
  AE_idx = statin49_AE_idx,
  rho = 0.5,
  tol = 0.1
)
```

---

get_expected_counts *Compute the Expected Count Matrix from a Contingency Table*

Description

This function computes the expected counts matrix E_{ij} from a given $I \times J$ contingency table using the formula:

$$E_{ij} = \frac{n_{i.} n_{.j}}{n_{..}}$$

where $n_{i\cdot}$ is the sum of the i-th row, $n_{\cdot j}$ is the sum of the j-th column, and $n_{\cdot\cdot}$ is the total sum of the table.

Usage

```
get_expected_counts(continTable)
```

Arguments

continTable	A numeric matrix representing the $I \times J$ contingency table.

Value

A numeric matrix of the same dimension as `continTable`, containing the expected counts for each cell (i, j) of the contingency table. The expected counts are based on the row and column marginal sums of the contingency table.

Examples

```
# Create a 6 by 4 contingency table
set.seed(42)
dat_mat <- matrix(rpois(6*4, 20), nrow=6)
dat_mat

# Assign row and column names
contin_table <- check_and_fix_contin_table(dat_mat)
contin_table

# Compute the expected counts of the contingency table
get_expected_counts(contin_table)
```

---

get_std_pearson_res *Computing the standardized Pearson residuals for a given $I \times J$ contingency table*

Description

Compute the standardized Pearson residuals for a given $I \times J$ contingency table.

Usage

```
get_std_pearson_res(contin_table)
```

Arguments

contin_table	A matrix of an $I \times J$ contingency table.

Value

A matrix of the standardized Pearson residuals of the input contingency table.

Examples

```
# Create a 6 by 4 data matrix
set.seed(42)
dat_mat <- matrix(rpois(6 * 4, 20), nrow = 6)
dat_mat

# Check the format of the data matrix and assign row and column names
# to the data matrix
contin_table <- check_and_fix_contin_table(dat_mat)
contin_table

# Compute the standardized Pearson residuals
get_std_pearson_res(contin_table)
```

`mddc_boxplot` *Modified Detecting Deviating Cells (MDDC) algorithm for adverse event signal identification with boxplot method for cutoff selection.*

Description

Modified Detecting Deviating Cells (MDDC) algorithm for adverse event signal identification. Boxplot method is used for cutoff selection in step 2 of the algorithm.

Usage

```
mddc_boxplot(
  contin_table,
  col_specific_cutoff = TRUE,
  separate = TRUE,
  if_col_cor = FALSE,
  cor_lim = 0.8,
  coef = 1.5,
  num_cores = 2
)
```

Arguments

`contin_table`	A data matrix of an I x J contingency table with row (adverse event) and column (drug or vaccine) names. Please first check the input contingency table using the function `check_and_fix_contin_table()`.
`col_specific_cutoff`	Logical. In the second step of the algorithm, whether to apply boxplot method to the standardized Pearson residuals of the entire table, or within each drug or vaccine column. Default is `TRUE`, that is within each drug or vaccine column

	(column specific cutoff). FALSE indicates applying boxplot method on residuals of the entire table.
separate	Logical. In the second step of the algorithm, whether to separate the standardized Pearson residuals for the zero cells and non zero cells and apply boxplot method separately or together. Default is TRUE.
if_col_cor	Logical. In the third step of the algorithm, whether to use column (drug or vaccine) correlation or row (adverse event) correlation. Default is FALSE, that is using the adverse event correlation. TRUE indicates using drug or vaccine correlation.
cor_lim	A numeric value between (0, 1). In the third step, what correlation threshold should be used to select "connected" adverse events. Default is 0.8.
coef	A numeric value or a list of numeric values. If a single numeric value is provided, it will be applied uniformly across all columns of the contingency table. If a list is provided, its length must match the number of columns in the contingency table, and each value will be used as the coefficient for the corresponding column.
num_cores	Number of cores used to parallelize the MDDC Boxplot algorithm. Default is 2.

Value

A list with the following components:

- boxplot_signal returns the signals identified in the second step. 1 indicates signals, 0 for non signal.
- corr_signal_pval returns the p values for each cell in the contingency table in the fifth step, when the r_{ij} values are mapped back to the standard normal distribution.
- corr_signal_adj_pval returns the Benjamini-Hochberg adjusted p values for each cell in the fifth step. We leave here an option for the user to decide whether to use corr_signal_pval or corr_signal_adj_pval, and what threshold for p values should be used (for example, 0.05). Please see the example below.

See Also

find_optimal_coef for finding an optimal value of coef.

Examples

```
# using statin49 data set as an example
data(statin49)

# apply the mddc_boxplot
boxplot_res <- mddc_boxplot(statin49)

# signals identified in step 2 using boxplot method
signal_step2 <- boxplot_res$boxplot_signal

# signals identified in step 5 by considering AE correlations
# In this example, cells with p values less than 0.05 are
```

```
# identified as signals
signal_step5 <- (boxplot_res$corr_signal_pval < 0.05) * 1
```

`mddc_mc` *Modified Detecting Deviating Cells (MDDC) algorithm for adverse event signal identification with Monte Carlo (MC) method for cutoff selection.*

Description

Modified Detecting Deviating Cells (MDDC) algorithm for adverse event signal identification. Monte Carlo (MC) method is used for cutoff selection in the second step of the algorithm.

Usage

```
mddc_mc(
  contin_table,
  quantile = 0.95,
  rep = 10000,
  exclude_same_drug_class = TRUE,
  col_specific_cutoff = TRUE,
  separate = TRUE,
  if_col_cor = FALSE,
  cor_lim = 0.8,
  num_cores = 2,
  seed = NULL
)
```

Arguments

`contin_table`	A data matrix of an I x J contingency table with row (adverse event) and column (drug or vaccine) names. Please first check the input contingency table using the function `check_and_fix_contin_table()`.
`quantile`	In the second step of the algorithm, the quantile of the null distribution obtained via MC method to use as a threshold for identifying cells with high value of the standardized Pearson residuals. Default is 0.95.
`rep`	In the second step, the number of Monte Carlo replications in the MC method. Default is 10000.
`exclude_same_drug_class`	In the second step, when applying Fisher's exact test to cells with a count less than six, a 2 by 2 contingency table needs to be constructed. Does the construction need to exclude other drugs or vaccines in the same class as the drug or vaccine of interest? Default is `TRUE`.

col_specific_cutoff
: Logical. In the second step of the algorithm, whether to apply MC method to the standardized Pearson residuals of the entire table, or within each drug or vaccine column. Default is TRUE, that is within each drug or vaccine column (column specific cutoff). FALSE indicates applying MC method on residuals of the entire table.

separate
: Logical. In the second step of the algorithm, whether to separate the standardized Pearson residuals for the zero cells and non zero cells and apply MC method separately or together. Default is TRUE.

if_col_cor
: Logical. In the third step of the algorithm, whether to use column (drug or vaccine) correlation or row (adverse event) correlation. Default is FALSE, that is using the adverse event correlation. TRUE indicates using drug or vaccine correlation.

cor_lim
: A numeric value between (0, 1). In the third step, what correlation threshold should be used to select "connected" adverse events. Default is 0.8.

num_cores
: Number of cores used to parallelize the MDDC MC algorithm. Default is 2.

seed
: An optional integer to set the seed for reproducibility. If NULL, no seed is set.

Value

A list with the following components:

- mc_pval returns the p values for each cell in the second step. For cells with a count greater than five, the p values are obtained via MC method. For cells with a count less than or equal to five, the p values are obtained via Fisher's exact tests.
- mc_signal returns the signals with a count greater than five and identified in the second step by MC method. 1 indicates signals, 0 for non signal.
- fisher_signal returns the signals with a count less than or equal to five and identified in the second step by Fisher's exact tests. 1 indicates signals, 0 for non signal.
- corr_signal_pval returns the p values for each cell in the contingency table in the fifth step, when the r_{ij} values are mapped back to the standard normal distribution.
- corr_signal_adj_pval returns the Benjamini-Hochberg adjusted p values for each cell in the fifth step. We leave here an option for the user to decide whether to use corr_signal_pval or corr_signal_adj_pval, and what threshold for p values should be used (for example, 0.05). Please see the example below.

Examples

```
# using statin49 data set as an example
data(statin49)

# apply the mddc_mc
mc_res <- mddc_boxplot(statin49)

# signals identified in step 2 using MC method
signal_step2 <- mc_res$mc_signal

# signals identified in step 5 by considering AE correlations
```

```
# In this example, cells with p values less than 0.05 are
# identified as signals
signal_step5 <- (mc_res$corr_signal_pval < 0.05) * 1
```

`plot_heatmap` *Plot Heatmap*

Description

Creates a heatmap plot from a matrix or data frame.

Usage

```
plot_heatmap(data, cell_width = 1, cell_height = 1, ...)
```

Arguments

`data`	A matrix or data frame to be visualized as a heatmap. The row names and column names of the data will be used for labeling the heatmap axes.
`cell_width`	Numeric value indicating the width of each cell in the heatmap. Default is 1.
`cell_height`	Numeric value indicating the height of each cell in the heatmap. Default is 1.
`...`	Additional arguments to be passed to ggplot2 functions.

Value

A ggplot2 object representing the heatmap.

Examples

```
# Example data
data <- matrix(rnorm(100), nrow = 10, ncol = 10)
rownames(data) <- paste0("Row", 1:10)
colnames(data) <- paste0("Col", 1:10)

# Plot heatmap
plot <- plot_heatmap(data)
print(plot)
```

`report_drug_AE_pairs` *Report the potential adverse events for drugs from contingency table*

Description

Report the potential adverse events for drugs from contingency table

Usage

```
report_drug_AE_pairs(
  contin_table,
  contin_table_signal,
  along_rows = "AE",
  along_cols = "Drug"
)
```

Arguments

`contin_table`	A data matrix of an I x J contingency table with row (adverse event) and column (drug or vaccine) names. Please first check the input contingency table using the function `check_and_fix_contin_table()`.
`contin_table_signal`	A data matrix with the same dimension and row and column names as `contin_table`, with entries either 1 (indicating signal) or 0 (indicating non-signal). This data matrix can be obtained via applying the function `mddc_boxplot` or `mddc_mc`.
`along_rows`	Specifies the content along the rows of the `contin_table` (e.g. AE or Drug).
`along_cols`	Specifies the content along the columns of the `contin_table` (e.g. AE or Drug).

Value

A data frame with five variables:

- `drug` the drug name.
- `AE` the potential adverse event for the drug or vaccine.
- `observed_num` the observed count of the (drug or vaccine, AE) pair.
- `expected_num` the expected count of the (drug or vaccine , AE) pair.
- `std_pearson_res` the value of the standardized Pearson residual.

Examples

```
# load statin49 data
data(statin49)

# run mddc boxplot method
test1 <- mddc_boxplot(statin49)
```

```
# get the signals from step 2
contin_table_signal <- test1$boxplot_signal

# get the signals from step 5
contin_table_signal_corr <- test1$corr_signal_pval < 0.05

# identify the (drug, AE) signals for step 2
result_1 <- report_drug_AE_pairs(
  contin_table = statin49,
  contin_table_signal = contin_table_signal
)
result_1

# identify the (drug, AE) signals for step 5
result_2 <- report_drug_AE_pairs(
  contin_table = statin49,
  contin_table_signal = contin_table_signal_corr
)
result_2
```

`sedative1000` *FDA dataset for sedatives with 1000 adverse events*

Description

A 1001 by 11 data matrix of a contingency table processed from the FDA Adverse Event Reporting System (FAERS) database. This dataset covers a specific period from Q1 2021 to Q4 2023.

Usage

```
sedative1000
```

Format

A data matrix with 1001 rows and 11 columns.

Details

A 1001 by 11 data matrix of a contingency table from the FDA Adverse Event Reporting System (FAERS) database, covering a specified period from Q1 2021 to Q4 2023.

The 1000 rows correspond to the AEs with the highest overall frequency (row marginals) reported during the period and 1 row for Other AEs. The reported AEs - "Off label use" and "Drug ineffective" have been excluded.

The dataset includes the following 10 columns: Clonazepam, Dexmedetomidine, Diazepam, Diphenhydramine, Doxepin, Lorazepam, Midazolam, Mirtazapine, Nitrazepam, Temazepam, and an Other column.

The marginal totals for each column are as follows: Clonazepam: 110,453, Dexmedetomidine: 4,262, Diazepam: 74,859 Diphenhydramine: 134,65, Doxepin: 11,795, Lorazepam: 101,969 Midazolam: 26,264, Mirtazapine: 54,273, Nitrazepam: 3,473, Temazepam: 20,523, Other: 77,487,518

Also refer to supplementary material of: Ding, Y., Markatou, M., & Ball, R. (2020). An evaluation of statistical approaches to postmarketing surveillance. Statistics in Medicine, 39(7), 845-874

for the data generation process. The quarterly files can be found in https://fis.fda.gov/extensions/FPD-QDE-FAERS/FPD-QDE-FAERS.html.

Examples

```
data(sedative1000)
head(sedative1000)
```

`statin101` *FDA statin dataset with 101 adverse events*

Description

A 102 by 5 data matrix of a contingency table processed from the FDA Adverse Event Reporting System (FAERS) database from Q1 2021 to Q4 2023.

Usage

```
statin101
```

Format

A data matrix with 102 rows and 5 columns.

Details

A 102 by data matrix of a contingency table from the FDA Adverse Event Reporting System (FAERS) database, covering Q1 2021 to Q4 2023.

The 101 rows correspond to the AEs with the highest overall frequency (row marginals) reported during the period. The reported AEs - "Off label use" and "Drug ineffective" have been excluded.

The 5 columns include 4 statin medications and an "other" column. Marginal totals for each drug: 101,462 for Atorvastatin, 9,203 for Fluvastatin, 130,994 for Rosuvastatin, 87,841 for Simvastatin, and 57,393,834 for Other drugs.

Also refer to supplementary material of:

Ding, Y., Markatou, M., & Ball, R. (2020). An evaluation of statistical approaches to postmarketing surveillance. Statistics in Medicine, 39(7), 845-874

for the data generation process. The quarterly files can be found in https://fis.fda.gov/extensions/FPD-QDE-FAERS/FPD-QDE-FAERS.html.

Examples

```
data(statin101)
head(statin101)
```

statin49 *FDA statin dataset with 49 adverse events*

Description

A 50 by 7 data matrix of a contingency table processed from the FDA Adverse Event Reporting System (FAERS) database from Q3 2014 to Q4 2020.

Usage

```
statin49
```

Format

A data matrix with 50 rows and 7 columns.

Details

A 50 by 7 data matrix of a contingency table from the FDA Adverse Event Reporting System (FAERS) database, covering Q3 2014 to Q4 2020.

The 49 rows represent important adverse events associated with statins, with the final row aggregating the remaining 5,990 events. The 49 AEs are classified into three clusters (see `statin49_AE_idx` for cluster indices):

1) AEs associated with muscle injury signs and symptoms, 2) AEs associated with muscle injury lab tests, and 3) AEs associated with kidney injury and its diagnosis and treatment.

The 7 columns include six statin medications and an "other drugs" column. Marginal totals for each drug: 197,390 for Atorvastatin, 5,742 for Fluvastatin, 3,230 for Lovastatin, 22,486 for Pravastatin, 122,450 for Rosuvastatin, 85,445 for Simvastatin, and 63,539,867 for Other drugs.

Examples

```
data(statin49)
head(statin49)
```

statin49_AE_idx *Cluster index of the FDA statin dataset with 49 adverse events*

Description

A 50 by 2 data frame showing the cluster index of the `statin49` dataset's adverse events.

Usage

```
statin49_AE_idx
```

Format

A data frame with 50 rows and 2 columns: `idx` and `AE`.

Details

A data frame with 50 rows and 2 columns: `idx` and `AE`. `AE` lists the adverse event names in the `statin49` dataset, while `idx` lists the cluster index of each adverse event.

The 49 AEs are classified into three clusters: 1) AEs associated with signs and symptoms of muscle injury, 2) AEs associated with laboratory tests for muscle injury, and 3) AEs associated with kidney injury and its laboratory diagnosis and treatment.

Examples

```
data(statin49_AE_idx)
head(statin49_AE_idx)
```

Index