

Package: MVR (via r-universe)

September 8, 2024

Type Package

Title Mean-Variance Regularization

Version 1.34.0

Date 2020-10-23

Author Jean-Eudes Dazard [aut, cre], Hua Xu [ctb], Alberto Santana [ctb]

Maintainer Jean-Eudes Dazard <jean-eudes.dazard@case.edu>

Description Implements a non-parametric method for joint adaptive mean-variance regularization and variance stabilization of high-dimensional data. It is suited for handling difficult problems posed by high-dimensional multivariate datasets ($p \gg n$ paradigm). Among those are that the variance is often a function of the mean, variable-specific estimators of variances are not reliable, and tests statistics have low powers due to a lack of degrees of freedom. Key features include: (i) Normalization and/or variance stabilization of the data, (ii) Computation of mean-variance-regularized t-statistics (F-statistics to follow), (iii) Generation of diverse diagnostic plots, (iv) Computationally efficient implementation using C/C++ interfacing and an option for parallel computing to enjoy a faster and easier experience in the R environment.

Depends R ($\geq 3.5.0$)

Imports parallel, methods, statmod

NeedsCompilation yes

URL <https://github.com/jedazard/MVR>

Date/Publication 2011-07-26

License GPL (≥ 3) | file LICENSE

Archs i386, x64

Repository <https://jedazard.r-universe.dev>

RemoteUrl <https://github.com/jedazard/mvr>

RemoteRef HEAD

RemoteSha dd6ae1635e7dfae3fac59bc21066db7e87453287

Contents

| | |
|------------------------------------|-----------|
| MVR-package | 2 |
| cluster.diagnostic | 5 |
| mvr | 9 |
| MVR.news | 15 |
| mvr.test | 16 |
| normalization.diagnostic | 23 |
| Real | 26 |
| stabilization.diagnostic | 27 |
| Synthetic | 31 |
| target.diagnostic | 32 |
| Index | 36 |

| | |
|-------------|---|
| MVR-package | <i>Mean-Variance Regularization Package</i> |
|-------------|---|

Description

Implements a non-parametric method for joint adaptive mean-variance regularization and variance stabilization of high-dimensional data (Dazard and Rao, 2012).

It is suited for handling difficult problems posed by high-dimensional multivariate datasets ($p \gg n$ paradigm), such as in omics-type data, among which are that the variance is often a function of the mean, variable-specific estimators of variances are not reliable, and tests statistics have low powers due to a lack of degrees of freedom.

Key features include:

1. Normalization and/or variance stabilization of the data
2. Computation of mean-variance-regularized t -statistics (F -statistics to come)
3. Generation of diverse diagnostic plots
4. Computationally efficient implementation using C/C++ interfacing and an option for parallel computing to enjoy a fast and easy experience in the R environment

Details

The following describes all the end-user functions, and internal R subroutines needed for running a complete MVR procedure. Other internal subroutines are not to be called by the end-user at any time. For computational efficiency, end-user regularization functions offer the option to configure a cluster. This is indicated by an asterisk (* = optionally involving cluster usage). The R functions are categorized as follows:

1. END-USER REGULARIZATION & VARIANCE STABILIZATION FUNCTION
mvr (*) Function for Mean-Variance Regularization and Variance Stabilization.
 End-user function for Mean-Variance Regularization (MVR) and Variance Stabilization by similarity statistic under sample group homoscedasticity or heteroscedasticity assumption.

The function takes advantage of the R package **parallel**, which allows users to create a cluster of workstations on a local and/or remote machine(s), enabling parallel execution of this function and scaling up with the number of CPU cores available.

2. END-USER REGULARIZED TESTS-STATISTICS FUNCTIONS

`mvt.test` (*) Function for Computing Mean-Variance Regularized T-test Statistic and Its Significance.

End-user function for computing MVR t-test statistic and its significance (p-value) under sample group homoscedasticity or heteroscedasticity assumption. The function takes advantage of the R package **parallel**, which allows users to create a cluster of workstations on a local and/or remote machine(s), enabling parallel execution of this function and scaling up with the number of CPU cores available.

3. END-USER DIAGNOSTIC PLOTS FOR QUALITY CONTROL

`cluster.diagnostic` Function for Plotting Summary Cluster Diagnostic Plots.

Plot similarity statistic profiles and the optimal joint clustering configuration for the means and the variances by group. Plot quantile profiles of means and standard deviations by group and for each clustering configuration, to check that the distributions of first and second moments of the MVR-transformed data approach their respective null distributions under the optimal configuration found, assuming independence and normality of all the variables.

`target.diagnostic` Function for Plotting Summary Target Moments Diagnostic Plots.

Plot comparative distribution densities of means and standard deviations of the data before and after Mean-Variance Regularization to check for location shifts between observed first and second moments and their expected target values under a target centered homoscedastic model. Plot comparative QQ scatterplots to look at departures between observed distributions of first and second moments of the MVR-transformed data and their theoretical distributions assuming independence and normality of all the variables.

`stabilization.diagnostic` Function for Plotting Summary Variance Stabilization Diagnostic Plots.

Plot comparative variance-mean plots to check the variance stabilization across variables before and after Mean-Variance Regularization.

`normalization.diagnostic` Function for Plotting Summary Normalization Diagnostic Plots.

Plot comparative Box-Whisker and Heatmap plots of variables across samples check the effectiveness of normalization before and after Mean-Variance Regularization.

4. OTHER END-USER FUNCTIONS

`MVR.news` Display the MVR Package News

Function to display the log file NEWS of updates of the **MVR** package.

5. END-USER DATASETS

A Real dataset coming from a quantitative proteomics experiment, consisting of $n = 6$ samples split into a control ("M") and a treated group ("S") with $p = 9052$ unique peptides or predictor variables. This is a balanced design with two sample groups ($G = 2$), under unequal

sample group variance.

A Synthetic dataset with $n = 10$ observations (samples) and $p = 100$ variables, where $nvar = 20$ of them are significantly different between the two sample groups. This is a balanced design with two sample groups ($G = 2$), under unequal sample group variance.

Known Bugs/Problems : None at this time.

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

References

- Dazard J-E. and J. S. Rao (2010). "Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data." In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization." In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.
- Dazard J-E. and J. S. Rao (2012). "Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data." *Comput. Statist. Data Anal.* 56(7):2317-2333.

See Also

- makeCluster (R package **parallel**)
- justvsN (R package **vsN**) Variance stabilization and calibration for microarray data *Huber, 2002*
- eBayes (R package **limma**) Bayesian Regularized t-test statistic *Smyth, 2004*
- samr (R package **samr**) SAM Regularized t-test statistic *Tusher et al., 2001, Storey, 2003*
- matest (R package **maanova**) James-Stein shrinkage estimator-based Regularized t-test statistic *Cui et al., 2005*
- ebam (R package **siggenes**) Empirical Bayes Regularized z-test statistic *Efron, 2001*
- bayesT Hierarchical Bayesian Regularized t-test statistic *Baldi et al., 2001*

cluster.diagnostic *Function for Plotting Summary Cluster Diagnostic Plots*

Description

Plot similarity statistic profiles and the optimal joint clustering configuration for the means and the variances by group.

Plot quantile profiles of means and standard deviations by group and for each clustering configuration, to check that the distributions of first and second moments of the MVR-transformed data approach their respective null distributions under the optimal configuration found, assuming independence and normality of all the variables.

Usage

```
cluster.diagnostic(obj,
                   span = 0.75,
                   degree = 2,
                   family = "gaussian",
                   title = "Cluster Diagnostic Plots",
                   device = NULL,
                   file = "Cluster Diagnostic Plots",
                   path = getwd(),
                   horizontal = FALSE,
                   width = 8.5,
                   height = 11, ...)
```

Arguments

| | |
|--------|---|
| obj | Object of class "MVR" returned by mvr . |
| title | Title of the plot. Defaults to "Cluster Diagnostic Plots". |
| span | Span parameter of the <code>loess()</code> function (R package stats), which controls the degree of smoothing. Defaults to 0.75. |
| degree | Degree parameter of the <code>loess()</code> function (R package stats), which controls the degree of the polynomials to be used. Defaults to 2. (Normally 1 or 2. Degree 0 is also allowed, but see the "Note" in <code>loess stats</code> package.) |
| family | Family distribution in "gaussian", "symmetric" of the <code>loess()</code> function (R package stats), used for local fitting. If "gaussian" fitting is by least-squares, and if "symmetric" is used, a re-descending M estimator is used with Tukey's biweight function. Defaults to "gaussian". |
| device | Graphic display device in {NULL, "PS", "PDF"}. Defaults to NULL (standard output screen). Currently implemented graphic display devices are "PS" (Postscript) or "PDF" (Portable Document Format). |
| file | File name for output graphic. Defaults to "Cluster Diagnostic Plots". |
| path | Absolute path (without final (back)slash separator). Defaults to working directory path. |

| | |
|------------|---|
| horizontal | Logical scalar. Orientation of the printed image. Defaults to FALSE, that is potrait orientation. |
| width | Numeric scalar. Width of the graphics region in inches. Defaults to 8.5. |
| height | Numeric scalar. Height of the graphics region in inches. Defaults to 11. |
| ... | Generic arguments passed to other plotting functions. |

Details

In a plot of a similarity statistic profile, one checks the goodness of fit of the transformed data relative to the hypothesized underlying reference distribution with mean-0 and standard deviation-1 (e.g. $N(0, 1)$). The red dashed line depicts the LOESS scatterplot smoother estimator. The subroutine internally generates reference null distributions for computing the similarity statistic under each cluster configuration. The optimal cluster configuration (indicated by the vertical red arrow) is found where the similarity statistic reaches its minimum plus/minus one standard deviation (applying the conventional one-standard deviation rule). A smaller cluster number configuration indicates under-regularization, while over-regularization starts to occur at larger numbers. This over/under-regularization must be viewed as a form of over/under-fitting (see *Dazard, J-E. and J. S. Rao (2012)* for more details). The quantile diagnostic plots uses empirical quantiles of the transformed means and standard deviations to check how closely they are approximated by theoretical quantiles derived from a standard normal *equal-mean/homoscedastic* model (solid green lines) under a given cluster configuration. To assess this goodness of fit of the transformed data, theoretical null distributions of the mean and variance are derived from a standard normal *equal-mean/homoscedastic* model with independence of the first two moments, i.e. assuming i.i.d. normality of the raw data. However, we do not require i.i.d. normality of the data in general: these theoretical null distributions are just used here as convenient ones to draw from. Note that under the assumptions that the raw data is i.i.d. standard normal ($N(0, 1)$) with independence of first two moments, the theoretical null distributions of means and standard deviations for each variable are respectively: $N(0, \frac{1}{n})$ and $\sqrt{\frac{\chi_{n-G}^2}{n-G}}$, where G denotes the number of sample groups. The optimal cluster configuration found is indicated by the most horizontal red curve. The single cluster configuration, corresponding to no transformation, is the most vertical curve, while the largest cluster number configuration reaches horizontality. Notice how empirical quantiles of transformed pooled means and standard deviations converge (from red to black) to the theoretical null distributions (solid green lines) for the optimal configuration. One should see a convergence towards the target null, after which overfitting starts to occur (see *Dazard, J-E. and J. S. Rao (2012)* for more details). Both cluster diagnostic plots help determine (i) whether the minimum of the *Similarity Statistic* is observed within the range of clusters (i.e. a large enough number of clusters has been accommodated), and (ii) whether the corresponding cluster configuration is a good fit. If necessary, run the procedure again with larger value of the `nc.max` parameter in the `mvr` as well as in `mvt.test` functions until the minimum of the similarity statistic profile is reached.

Option file is used only if device is specified (i.e. non NULL).

Value

None. Displays the plots on the chosen device.

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Note

End-user function.

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

References

- Dazard J-E. and J. S. Rao (2010). "*Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data.*" In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "*R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization.*" In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.
- Dazard J-E. and J. S. Rao (2012). "*Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data.*" *Comput. Statist. Data Anal.* 56(7):2317-2333.

See Also

loess (R package **stats**) Fit a polynomial surface determined by one or more numerical predictors, using local fitting.

Examples

```
## Not run:
#=====
# Loading the library and its dependencies
#=====
library("MVR")

#=====
# MVR package news
#=====
MVR.news()

#=====
# MVR package citation
```

```

=====
citation("MVR")

=====
# Loading of the Synthetic and Real datasets
# (see description of datasets)
=====
data("Synthetic", "Real", package="MVR")
?Synthetic
?Real

=====
# Mean-Variance Regularization (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
# Without cluster usage
=====
nc.min <- 1
nc.max <- 30
probs <- seq(0, 1, 0.01)
n <- 6
GF <- factor(gl(n = 2, k = n/2, length = n),
             ordered = FALSE,
             labels = c("M", "S"))
mvr.obj <- mvr(data = Real,
              block = GF,
              log = FALSE,
              nc.min = nc.min,
              nc.max = nc.max,
              probs = probs,
              B = 100,
              parallel = FALSE,
              conf = NULL,
              verbose = TRUE,
              seed = 1234)

=====
# Summary Cluster Diagnostic Plots (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
=====
cluster.diagnostic(obj = mvr.obj,
                  title = "Cluster Diagnostic Plots
                           (Real - Multi-Group Assumption)",
                  span = 0.75,
                  degree = 2,
                  family = "gaussian",
                  device = NULL,
                  horizontal = FALSE,
                  width = 8.5,
                  height = 11)

```



```
## End(Not run)
```

| | |
|-----|---|
| mvr | <i>Function for Mean-Variance Regularization and Variance Stabilization</i> |
|-----|---|

Description

End-user function for Mean-Variance Regularization (MVR) and Variance Stabilization by similarity statistic under sample group homoscedasticity or heteroscedasticity assumptions.

Return an object of class "MVR". Offers the option of parallel computation for improved efficiency.

Usage

```
mvr(data,
     block = rep(1, nrow(data)),
     tolog = FALSE,
     nc.min = 1,
     nc.max = 30,
     probs = seq(0, 1, 0.01),
     B = 100,
     parallel = FALSE,
     conf = NULL,
     verbose = TRUE,
     seed = NULL)
```

Arguments

| | |
|--------|--|
| data | numeric matrix of untransformed (raw) data, where samples are by rows and variables (to be clustered) are by columns, or an object that can be coerced to a matrix (such as a numeric vector or a data.frame with all numeric columns). Missing values (NA), NotANumber values (NaN) or Infinite values (Inf) are not allowed. |
| block | character or numeric vector, or factor of group membership indicator variable (grouping/blocking variable) of length the data sample size with as many different values or levels as the number of data sample groups. Defaults to single group situation. See details. |
| tolog | logical scalar. Is the data to be log2-transformed first? Optional, defaults to FALSE. Note that negative or null values will be changed to 1 before taking log2-transformation. |
| nc.min | Positive integer scalar of the minimum number of clusters, defaults to 1 |
| nc.max | Positive integer scalar of the maximum number of clusters, defaults to 30 |
| probs | numeric vector of probabilities for quantile diagnostic plots. Defaults to seq(0, 1, 0.01). |
| B | Positive integer scalar of the number of Monte Carlo replicates of the inner loop of the sim statistic function (see details). |

| | |
|----------|---|
| parallel | logical scalar. Is parallel computing to be performed? Optional, defaults to FALSE. |
| conf | list of 5 fields containing the parameters values needed for creating the parallel backend (cluster configuration). See details below for usage. Optional, defaults to NULL, but all fields are required if used: <ul style="list-style-type: none"> • type : character vector specifying the cluster type ("SOCKET", "MPI"). • spec : A specification (character vector or integer scalar) appropriate to the type of cluster. • homogeneous : logical scalar to be set to FALSE for inhomogeneous clusters. • verbose : logical scalar to be set to FALSE for quiet mode. • outfile : character vector of an output log file name to direct the stdout and stderr connection output from the workernodes. "" indicates no redirection. |
| verbose | logical scalar. Is the output to be verbose? Optional, defaults to TRUE. |
| seed | Positive integer scalar of the user seed to reproduce the results. |

Details

Argument `block` will be converted to a factor, whose levels will match the data groups. It defaults to a single group situation, that is, under the assumption of equal variance between sample groups. All group sample sizes must be greater than 1, otherwise the program will stop.

Argument `nc.max` currently defaults to 30. Empirically, we found that this is enough for most datasets tested. This depends on (i) the dimensionality/sample size ratio $\frac{p}{n}$, (ii) the signal/noise ratio, and (iii) whether a pre-transformation has been applied (see *Dazard, J-É. and J. S. Rao (2012)* for more details). See the cluster diagnostic function `cluster.diagnostic` for more details, whether larger values of `nc.max` may be required.

The function `mvr` relies on the R package **parallel** to create a parallel backend within an R session. This enables access to a cluster of compute cores and/or nodes on a local and/or remote machine(s) and scaling-up with the number of CPU cores available and efficient parallel execution. To run a procedure in parallel (with parallel RNG), argument `parallel` is to be set to TRUE and argument `conf` is to be specified (i.e. non NULL). Argument `conf` uses the options described in function `makeCluster` of the R packages **parallel** and **snow**. **PRIMsrc** supports two types of communication mechanisms between master and worker processes: 'Socket' or 'Message-Passing Interface' ('MPI'). In **PRIMsrc**, parallel 'Socket' clusters use sockets communication mechanisms only (no forking) and are therefore available on all platforms, including Windows, while parallel 'MPI' clusters use high-speed interconnects mechanism in networks of computers (with distributed memory) and are therefore available only in these architectures. A parallel 'MPI' cluster also requires R package **Rmpi** to be installed first. Value `type` is used to setup a cluster of type 'Socket' ("SOCKET") or 'MPI' ("MPI"), respectively. Depending on this type, values of `spec` are to be used alternatively:

- For 'Socket' clusters (`conf$type="SOCKET"`), `spec` should be a character vector naming the hosts on which to run the job; it can default to a unique local machine, in which case, one may use the unique host name "localhost". Each host name can potentially be repeated to the number of CPU cores available on the local machine. It can also be an integer scalar specifying the number of processes to spawn on the local machine; or a list of machine specifications (a character value named `host` specifying the name or address of the host to use).

- For 'MPI' clusters (`conf$type="MPI"`), `spec` should be an integer scalar specifying the total number of processes to be spawned across the network of available nodes, counting the workernodes and masternode.

The actual creation of the cluster, its initialization, and closing are all done internally. For more details, see the reference manual of R package **snow** and examples below.

When random number generation is needed, the creation of separate streams of parallel RNG per node is done internally by distributing the stream states to the nodes. For more details, see the vignette of R package **parallel**. The use of a seed allows to reproduce the results within the same type of session: the same seed will reproduce the same results within a non-parallel session or within a parallel session, but it will not necessarily give the exact same results (up to sampling variability) between a non-parallelized and parallelized session due to the difference of management of the seed between the two (see parallel RNG and value of returned seed below).

Value

| | |
|------------------------|--|
| <code>Xraw</code> | numeric matrix of original data. |
| <code>Xmvr</code> | numeric matrix of MVR-transformed data. |
| <code>centering</code> | numeric vector of centering values for standardization (cluster mean of pooled sample mean). |
| <code>scaling</code> | numeric vector of scaling values for standardization (cluster mean of pooled sample std dev). |
| <code>MVR</code> | list (of size the number of groups) containing for each group: <ul style="list-style-type: none"> • membership numeric vector of cluster membership of each variable • <code>nc</code> Positive integer scalar of number of clusters found in optimal cluster configuration • <code>gap</code> numeric vector of the similarity statistic values • <code>sde</code> numeric vector of the standard errors of the similarity statistic values • <code>mu.std</code> numeric matrix ($K \times p$) of the vector of standardized means by groups (rows), where $K = \#groups$ and $p = \#variables$ • <code>sd.std</code> numeric matrix ($K \times p$) of the vector of standardized standard deviations by groups (rows), where $K = \#groups$ and $p = \#variables$ • <code>mu.quant</code> numeric matrix $(nc.max - nc.min + 1) \times (length(probs))$ of quantiles of means • <code>sd.quant</code> numeric matrix $(nc.max - nc.min + 1) \times (length(probs))$ of quantiles of standard deviations |
| <code>block</code> | Value of argument <code>block</code> . |
| <code>tolog</code> | Value of argument <code>tolog</code> . |
| <code>nc.min</code> | Value of argument <code>nc.min</code> . |
| <code>nc.max</code> | Value of argument <code>nc.max</code> . |
| <code>probs</code> | Value of argument <code>probs</code> . |
| <code>seed</code> | User seed(s) used: integer of a single value, if parallelization is used. integer vector of values, one for each replication, if parallelization is not used. |

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Note

End-user function.

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

References

- Dazard J-E. and J. S. Rao (2010). "*Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data*." In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "*R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization*." In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.
- Dazard J-E. and J. S. Rao (2012). "*Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data*." *Comput. Statist. Data Anal.* 56(7):2317-2333.

See Also

- makeCluster (R package **parallel**).
- justvsn (R package **vsn**) Variance stabilization and calibration for microarray data *Huber, 2002*

Examples

```
#####
# Loading the library and its dependencies
#####
library("MVR")

## Not run:
#####
# MVR package news
#####
MVR.news()
```

```

#####
# MVR package citation
#####
citation("MVR")

#####
# Loading of the Synthetic and Real datasets
# Use help for descriptions
#####
data("Synthetic", "Real", package="MVR")
?Synthetic
?Real

## End(Not run)

#####
# Mean-Variance Regularization (Synthetic dataset)
# Single-Group Assumption
# Assuming equal variance between groups
# Without cluster usage
#####
data("Synthetic", package="MVR")
nc.min <- 1
nc.max <- 10
probs <- seq(0, 1, 0.01)
n <- 10
mvr.obj <- mvr(data = Synthetic,
               block = rep(1,n),
               tolog = FALSE,
               nc.min = nc.min,
               nc.max = nc.max,
               probs = probs,
               B = 100,
               parallel = FALSE,
               conf = NULL,
               verbose = TRUE,
               seed = 1234)

## Not run:
#####
# Examples of parallel backend parametrization
#####
if (require("parallel")) {
  cat("'parallel' is attached correctly \n")
} else {
  stop("'parallel' must be attached first \n")
}
#####
# Ex. #1 - Multicore PC
# Running WINDOWS
# SOCKET communication cluster
# Shared memory parallelization
#####

```

```

cpus <- parallel::detectCores(logical = TRUE)
conf <- list("spec" = rep("localhost", cpus),
            "type" = "SOCKET",
            "homo" = TRUE,
            "verbose" = TRUE,
            "outfile" = "")

#####
# Ex. #2 - Master node + 3 Worker nodes cluster
# All nodes equipped with identical setups of multicores
# (8 core CPUs per machine for a total of 32)
# SOCKET communication cluster
# Distributed memory parallelization
#####
masterhost <- Sys.getenv("HOSTNAME")
slavehosts <- c("compute-0-0", "compute-0-1", "compute-0-2")
nodes <- length(slavehosts) + 1
cpus <- 8
conf <- list("spec" = c(rep(masterhost, cpus),
                        rep(slavehosts, cpus)),
            "type" = "SOCKET",
            "homo" = TRUE,
            "verbose" = TRUE,
            "outfile" = "")

#####
# Ex. #3 - Enterprise Multinode Cluster w/ multicore/node
# Running LINUX with SLURM scheduler
# MPI communication cluster
# Distributed memory parallelization
# Below, variable 'cpus' is the total number of requested
# taks (threads/CPUs), which is specified from within a
# SLURM script.
#####
if (require("Rmpi")) {
  print("Rmpi is loaded correctly \n")
} else {
  stop("Rmpi must be installed first to use MPI\n")
}
cpus <- as.numeric(Sys.getenv("SLURM_NTASKS"))
conf <- list("spec" = cpus,
            "type" = "MPI",
            "homo" = TRUE,
            "verbose" = TRUE,
            "outfile" = "")

#####
# Mean-Variance Regularization (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
#####
data("Real", package="MVR")
nc.min <- 1
nc.max <- 30
probs <- seq(0, 1, 0.01)
n <- 6

```

```
GF <- factor(gl(n = 2, k = n/2, length = n),
             ordered = FALSE,
             labels = c("M", "S"))
mvr.obj <- mvr(data = Real,
              block = GF,
              tolog = FALSE,
              nc.min = nc.min,
              nc.max = nc.max,
              probs = probs,
              B = 100,
              parallel = TRUE,
              conf = conf,
              verbose = TRUE,
              seed = 1234)

## End(Not run)
```

MVR.news

Function to Display the NEWS File

Description

Function to display the NEWS file of the **MVR** package.

Usage

```
MVR.news(...)
```

Arguments

... Further arguments passed to or from other methods.

Value

None.

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Note

End-user function.

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

References

- Dazard J-E. and J. S. Rao (2010). "*Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data*." In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "*R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization*." In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.
- Dazard J-E. and J. S. Rao (2012). "*Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data*." *Comput. Statist. Data Anal.* 56(7):2317-2333.

mvrt.test

Function for Computing Mean-Variance Regularized T-test Statistic and Its Significance

Description

End-user function for computing MVR t-test statistic and its significance (p-value) under sample group homoscedasticity or heteroscedasticity assumption.

Return an object of class "MVRT". Offers the option of parallel computation for improved efficiency.

Usage

```
mvrt.test(data,
           obj=NULL,
           block,
           tolog = FALSE,
           nc.min = 1,
           nc.max = 30,
           pval = FALSE,
           replace = FALSE,
           n.resamp = 100,
           parallel = FALSE,
           conf = NULL,
           verbose = TRUE,
           seed = NULL)
```


Arguments

| | |
|----------|--|
| data | numeric matrix of untransformed (raw) data, where samples are by rows and variables (to be clustered) are by columns, or an object that can be coerced to a matrix (such as a numeric vector or a data.frame with all numeric columns). Missing values (NA), NotANumber values (NaN) or Infinite values (Inf) are not allowed. |
| obj | Object of class "MVR" returned by <code>mvr</code> . |
| block | character or numeric vector, or factor of group membership indicator variable (grouping/blocking variable) of length the data sample size with as many different values or levels as the number of data sample groups. Defaults to single group situation. See details. |
| tolog | logical scalar. Is the data to be log2-transformed first? Optional, defaults to FALSE. Note that negative or null values will be changed to 1 before taking log2-transformation. |
| nc.min | Positive integer scalar of the minimum number of clusters, defaults to 1 |
| nc.max | Positive integer scalar of the maximum number of clusters, defaults to 30 |
| pval | logical scalar. Shall p-values be computed? If not, <code>n.resamp</code> and <code>replace</code> will be ignored. If FALSE (default), t-statistic only will be computed, If TRUE, exact (permutation test) or approximate (bootstrap test) p-values will be computed. |
| replace | logical scalar. Shall permutation test (default) or bootstrap test be computed? If FALSE (default), permutation test will be computed with null permutation distribution, If TRUE, bootstrap test will be computed with null bootstrap distribution. |
| n.resamp | Positive integer scalar of the number of resamplings to compute (default=100) by permutation or bootstrap (see details). |
| parallel | logical scalar. Is parallel computing to be performed? Optional, defaults to FALSE. |
| conf | list of 5 fields containing the parameters values needed for creating the parallel backend (cluster configuration). See details below for usage. Optional, defaults to NULL, but all fields are required if used: <ul style="list-style-type: none"> • <code>type</code> : character vector specifying the cluster type ("SOCKET", "MPI"). • <code>spec</code> : A specification (character vector or integer scalar) appropriate to the type of cluster. • <code>homogeneous</code> : logical scalar to be set to FALSE for inhomogeneous clusters. • <code>verbose</code> : logical scalar to be set to FALSE for quiet mode. • <code>outfile</code> : character vector of an output log file name to direct the stdout and stderr connection output from the workernodes. "" indicates no redirection. |
| verbose | logical scalar. Is the output to be verbose? Optional, defaults to TRUE. |
| seed | Positive integer scalar of the user seed to reproduce the results. |

Details

Argument `block` will be converted to a factor, whose levels will match the data groups. It defaults to a single group situation, that is, under the assumption of equal variance between sample groups. All group sample sizes must be greater than 1, otherwise the program will stop.

Argument `nc.max` currently defaults to 30. Empirically, we found that this is enough for most datasets tested. This depends on (i) the dimensionality/sample size ratio $\frac{p}{n}$, (ii) the signal/noise ratio, and (iii) whether a pre-transformation has been applied (see *Dazard, J-É. and J. S. Rao (2012)* for more details). See the cluster diagnostic function `cluster.diagnostic` for more details, whether larger values of `nc.max` may be required.

To save un-necessary computations, previously computed MVR clustering can be provided through option `obj` (i.e. `obj` is fully specified as a `mvr` object). In this case, arguments `data`, `block`, `tolog`, `nc.min`, `nc.max` are ignored. If `obj` is fully specified (i.e. an object of class "MVR" returned by `mvr`), the the MVR clustering provided by `obj` will be used for the computation of the regularized t-test statistics. If `obj=NULL`, a MVR clustering computation for the regularized t-test statistics and/or p-values will be performed.

The function `mvrt.test` relies on the R package **parallel** to create a parallel backend within an R session, enabling access to a cluster of compute cores and/or nodes on a local and/or remote machine(s) and scaling-up with the number of CPU cores available and efficient parallel execution. To run a procedure in parallel (with parallel RNG), argument `parallel` is to be set to `TRUE` and argument `conf` is to be specified (i.e. non `NULL`). Argument `conf` uses the options described in function `makeCluster` of the R packages **parallel** and **snow**. **PRIMsrc** supports two types of communication mechanisms between master and worker processes: 'Socket' or 'Message-Passing Interface' ('MPI'). In **PRIMsrc**, parallel 'Socket' clusters use sockets communication mechanisms only (no forking) and are therefore available on all platforms, including Windows, while parallel 'MPI' clusters use high-speed interconnects mechanism in networks of computers (with distributed memory) and are therefore available only in these architectures. A parallel 'MPI' cluster also requires R package **Rmpi** to be installed first. Value type is used to setup a cluster of type 'Socket' ("SOCKET") or 'MPI' ("MPI"), respectively. Depending on this type, values of `spec` are to be used alternatively:

- For 'Socket' clusters (`conf$type="SOCKET"`), `spec` should be a character vector naming the hosts on which to run the job; it can default to a unique local machine, in which case, one may use the unique host name "localhost". Each host name can potentially be repeated to the number of CPU cores available on the local machine. It can also be an integer scalar specifying the number of processes to spawn on the local machine; or a list of machine specifications (a character value named `host` specifying the name or address of the host to use).
- For 'MPI' clusters (`conf$type="MPI"`), `spec` should be an integer scalar specifying the total number of processes to be spawned across the network of available nodes, counting the `workernodes` and `masternode`.

The actual creation of the cluster, its initialization, and closing are all done internally. For more details, see the reference manual of R package **snow** and examples below.

When random number generation is needed, the creation of separate streams of parallel RNG per node is done internally by distributing the stream states to the nodes. For more details, see the vignette of R package **parallel**. The use of a seed allows to reproduce the results within the same type of session: the same seed will reproduce the same results within a non-parallel session or within a parallel session, but it will not necessarily give the exact same results (up to sampling variability)

between a non-parallelized and parallelized session due to the difference of management of the seed between the two (see parallel RNG and value of returned seed below).

In case p-values are desired (`pval=TRUE`), the use of a cluster is highly recommended. It is ideal for computing embarrassingly parallel tasks such as permutation or bootstrap resamplings. Note that in case both regularized t-test statistics and p-values are desired, in order to maximize computational efficiency and avoid multiple configurations (since a cluster can only be configured and used one session at a time, which otherwise would result in a run stop), the cluster configuration will only be used for the parallel computation of p-values, but not for the MVR clustering computation of the regularized t-test statistics.

Value

| | |
|-----------|---|
| statistic | vector, of size the number of variables, where entries are the t-statistics values of each variable. |
| p.value | vector, of size the number of variables, where entries are the p-values (if requested, otherwise NULL value) of each variable. |
| seed | User seed(s) used: integer of a single value, if parallelization is used. integer vector of values, one for each replication, if parallelization is not used. |

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Note

End-user function.

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

References

- Dazard J-E. and J. S. Rao (2010). "*Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data.*" In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "*R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization.*" In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.
- Dazard J-E. and J. S. Rao (2012). "*Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data.*" *Comput. Statist. Data Anal.* 56(7):2317-2333.

See Also

- makeCluster (R package **parallel**)
- eBayes (R package **limma**) Bayesian Regularized t-test statistic *Smyth, 2004*
- samr (R package **samr**) SAM Regularized t-test statistic *Tusher et al., 2001, Storey, 2003*
- matest (R package **maanova**) James-Stein shrinkage estimator-based Regularized t-test statistic *Cui et al., 2005*
- ebam (R package **siggenes**) Empirical Bayes Regularized z-test statistic *Efron, 2001*
- bayesT Hierarchical Bayesian Regularized t-test statistic *Baldi et al., 2001*

Examples

```

=====
# Loading the library and its dependencies
=====
library("MVR")

## Not run:
#=====
# MVR package news
#=====
MVR.news()

#=====
# MVR package citation
#=====
citation("MVR")

#=====
# Loading of the Synthetic and Real datasets
# Use help for descriptions
#=====
data("Synthetic", "Real", package="MVR")
?Synthetic
?Real

## End(Not run)

#=====
# Regularized t-test statistics (Synthetic dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
# With option to use prior MVR clustering results
# Without computation of p-values
# Without cluster usage
#=====
data("Synthetic", package="MVR")
nc.min <- 1
nc.max <- 10
probs <- seq(0, 1, 0.01)
n <- 10

```

```

GF <- factor(gl(n = 2, k = n/2, length = n),
             ordered = FALSE,
             labels = c("G1", "G2"))
mvr.obj <- mvr(data = Synthetic,
              block = GF,
              tolog = FALSE,
              nc.min = nc.min,
              nc.max = nc.max,
              probs = probs,
              B = 100,
              parallel = FALSE,
              conf = NULL,
              verbose = TRUE,
              seed = 1234)
mvr.test.obj <- mvr.test(data = NULL,
                       obj = mvr.obj,
                       block = NULL,
                       pval = FALSE,
                       replace = FALSE,
                       n.resamp = 100,
                       parallel = FALSE,
                       conf = NULL,
                       verbose = TRUE,
                       seed = 1234)

## Not run:
#=====
# Examples of parallel backend parametrization
#=====
if (require("parallel")) {
  cat("'parallel' is attached correctly \n")
} else {
  stop("'parallel' must be attached first \n")
}
#=====
# Ex. #1 - Multicore PC
# Running WINDOWS
# SOCKET communication cluster
# Shared memory parallelization
#=====
cpus <- parallel::detectCores(logical = TRUE)
conf <- list("spec" = rep("localhost", cpus),
            "type" = "SOCKET",
            "homo" = TRUE,
            "verbose" = TRUE,
            "outfile" = "")
#=====
# Ex. #2 - Master node + 3 Worker nodes cluster
# All nodes equipped with identical setups of multicores
# (8 core CPUs per machine for a total of 32)
# SOCKET communication cluster
# Distributed memory parallelization
#=====
masterhost <- Sys.getenv("HOSTNAME")

```

```

slavehosts <- c("compute-0-0", "compute-0-1", "compute-0-2")
nodes <- length(slavehosts) + 1
cpus <- 8
conf <- list("spec" = c(rep(masterhost, cpus),
                        rep(slavehosts, cpus)),
            "type" = "SOCKET",
            "homo" = TRUE,
            "verbose" = TRUE,
            "outfile" = "")

#####
# Ex. #3 - Enterprise Multinode Cluster w/ multicore/node
# Running LINUX with SLURM scheduler
# MPI communication cluster
# Distributed memory parallelization
# Below, variable 'cpus' is the total number of requested
# taks (threads/CPU), which is specified from within a
# SLURM script.
#####
if (require("Rmpi")) {
  print("Rmpi is loaded correctly \n")
} else {
  stop("Rmpi must be installed first to use MPI\n")
}
cpus <- as.numeric(Sys.getenv("SLURM_NTASKS"))
conf <- list("spec" = cpus,
            "type" = "MPI",
            "homo" = TRUE,
            "verbose" = TRUE,
            "outfile" = "")

#####
# Mean-Variance Regularization (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
#####
data("Real", package="MVR")
nc.min <- 1
nc.max <- 30
probs <- seq(0, 1, 0.01)
n <- 6
GF <- factor(gl(n = 2, k = n/2, length = n),
            ordered = FALSE,
            labels = c("M", "S"))
mvr.obj <- mvr(data = Real,
              block = GF,
              tolog = FALSE,
              nc.min = nc.min,
              nc.max = nc.max,
              probs = probs,
              B = 100,
              parallel = TRUE,
              conf = conf,
              verbose = TRUE,
              seed = 1234)

```

```

#####
# Regularized t-test statistics (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
# With option to use prior MVR clustering results
# With computation of p-values
#####
mvr.obj <- mvrt.test(data = NULL,
                    obj = mvr.obj,
                    block = NULL,
                    pval = TRUE,
                    replace = FALSE,
                    n.resamp = 100,
                    parallel = TRUE,
                    conf = conf,
                    verbose = TRUE,
                    seed = 1234)

## End(Not run)

```

normalization.diagnostic

Function for Plotting Summary Normalization Diagnostic Plots

Description

Plot comparative Box-Whisker and Heatmap plots of variables across samples check the effectiveness of normalization before and after Mean-Variance Regularization.

Usage

```

normalization.diagnostic(obj,
                        pal,
                        title = "Normalization Diagnostic Plots",
                        device = NULL,
                        file = "Normalization Diagnostic Plots",
                        path = getwd(),
                        horizontal = FALSE,
                        width = 7,
                        height = 8, ...)

```

Arguments

| | |
|-------|--|
| obj | Object of class "MVR" returned by <code>mvr</code> . |
| title | Title of the plot. Defaults to "Normalization Diagnostic Plots". |
| pal | Color palette. |

| | |
|------------|--|
| device | Graphic display device in {NULL, "PS", "PDF"}. Defaults to NULL (standard output screen). Currently implemented graphic display devices are "PS" (Postscript) or "PDF" (Portable Document Format). |
| file | File name for output graphic. Defaults to "Normalization Diagnostic Plots". |
| path | Absolute path (without final (back)slash separator). Defaults to working directory path. |
| horizontal | Logical scalar. Orientation of the printed image. Defaults to FALSE, that is potrait orientation. |
| width | Numeric scalar. Width of the graphics region in inches. Defaults to 7. |
| height | Numeric scalar. Height of the graphics region in inches. Defaults to 8. |
| ... | Generic arguments passed to other plotting functions. |

Details

Option file is used only if device is specified (i.e. non NULL). The argument pal can be any color palette, e.g. as provided by R package **RColorBrewer**.

Value

None. Displays the plots on the chosen device.

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Note

End-user function.

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

References

- Dazard J-E. and J. S. Rao (2010). "*Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data*." In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "*R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization*." In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.

- Dazard J-E. and J. S. Rao (2012). "Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data." *Comput. Statist. Data Anal.* 56(7):2317-2333.

See Also

justvsn (R package **vsn**) Variance stabilization and calibration for microarray data. loess (R package **stats**) Fit a polynomial surface determined by one or more numerical predictors, using local fitting.

Examples

```
## Not run:
#=====
# Loading the library and its dependencies
#=====
library("MVR")
library("RColorBrewer")

#=====
# MVR package news
#=====
MVR.news()

#=====
# MVR package citation
#=====
citation("MVR")

#=====
# Loading of the Synthetic and Real datasets
# (see description of datasets)
#=====
data("Synthetic", "Real", package="MVR")
?Synthetic
?Real

#=====
# Mean-Variance Regularization (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
# Without cluster usage
#=====
nc.min <- 1
nc.max <- 30
probs <- seq(0, 1, 0.01)
n <- 6
GF <- factor(gl(n = 2, k = n/2, length = n),
             ordered = FALSE,
             labels = c("M", "S"))
mvr.obj <- mvr(data = Real,
              block = GF,
              log = FALSE,
```

```

nc.min = nc.min,
nc.max = nc.max,
probs = probs,
B = 100,
parallel = FALSE,
conf = NULL,
verbose = TRUE,
seed = 1234)

#=====
# Summary Normalization Diagnostic Plots (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
#=====
normalization.diagnostic(obj = mvr.obj,
                        title = "Normalization Diagnostic Plots
                                (Real - Multi-Group Assumption)",
                        pal = brewer.pal(n=11, name="RdYlGn"),
                        device = NULL,
                        horizontal = FALSE,
                        width = 7,
                        height = 8)

## End(Not run)

```

Real

Real Proteomics Dataset

Description

The dataset comes from a quantitative Liquid Chromatography/Mass-Spectrometry (LC/MS) shotgun (bottom-up) proteomics experiment. It consists of $n = 6$ independent cell cultures of human Myeloid Dendritic Cells (MDCs) from normal subjects. Samples were split into a control ("M") and a treated group ("S"), stimulated with either media alone or a Toll-Like receptor-3 Ligand respectively. The goal was to identify differentially expressed peptides (or proteins) between the two groups involved in the immune response of human MDCs upon TLR-3 Ligand binding.

The dataset is assumed to have been pre-processed for non-ignorable missing values, leaving a complete dataset with $p = 9052$ unique peptides or predictor variables.

This is a balanced design with two sample groups ($G = 2$), under unequal sample group variance.

Usage

```
data("Real", package="MVR")
```

Format

A numeric matrix containing $n = 6$ observations (samples) by rows and $p = 9052$ variables by columns, named after peptide names ($diffset_1, \dots, diffset_p$). Samples are balanced ($n_1 = 3, n_2 = 3$) between the two groups ("M", "S"). Compressed Rda data file.

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

Source

See real proteomics data application in Dazard et al., 2011, 2012.

References

- Dazard J-E. and J. S. Rao (2010). "*Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data.*" In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "*R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization.*" In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.
- Dazard J-E. and J. S. Rao (2012). "*Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data.*" *Comput. Statist. Data Anal.* 56(7):2317-2333.

stabilization.diagnostic

Function for Plotting Summary Variance Stabilization Diagnostic Plots

Description

Plot comparative variance-mean plots to check the variance stabilization across variables before and after Mean-Variance Regularization.

Usage

```
stabilization.diagnostic(obj,
                          span = 0.5,
                          degree = 2,
                          family = "gaussian",
                          title = "Stabilization Diagnostic Plots",
                          device = NULL,
                          file = "Stabilization Diagnostic Plots",
                          path = getwd(),
                          horizontal = FALSE,
                          width = 7,
                          height = 5, ...)
```

Arguments

| | |
|------------|--|
| obj | Object of class "MVR" returned by <code>mvr</code> . |
| title | Title of the plot. Defaults to "Stabilization Diagnostic Plots". |
| span | Span parameter of the <code>loess()</code> function (R package stats), which controls the degree of smoothing. Defaults to 0.75. |
| degree | Degree parameter of the <code>loess()</code> function (R package stats), which controls the degree of the polynomials to be used. Defaults to 2. (Normally 1 or 2. Degree 0 is also allowed, but see the "Note" in <code>loess stats</code> package.) |
| family | Family distribution in "gaussian", "symmetric" of the <code>loess()</code> function (R package stats), used for local fitting. If "gaussian" fitting is by least-squares, and if "symmetric" a re-descending M estimator is used with Tukey's biweight function. |
| device | Graphic display device in {NULL, "PS", "PDF"}. Defaults to NULL (standard output screen). Currently implemented graphic display devices are "PS" (Postscript) or "PDF" (Portable Document Format). |
| file | File name for output graphic. Defaults to "Stabilization Diagnostic Plots". |
| path | Absolute path (without final (back)slash separator). Defaults to working directory path. |
| horizontal | Logical scalar. Orientation of the printed image. Defaults to FALSE, that is potrait orientation. |
| width | Numeric scalar. Width of the graphics region in inches. Defaults to 7. |
| height | Numeric scalar. Height of the graphics region in inches. Defaults to 5. |
| ... | Generic arguments passed to other plotting functions. |

Details

In the plots of standard deviations vs. means, standard deviations and means are calculated in a feature-wise manner from the expression matrix. The scatterplot allows to visually verify whether there is a dependence of the standard deviation (or variance) on the mean. The black dotted line depicts the LOESS scatterplot smoother estimator. If there is no variance-mean dependence, then this line should be approximately horizontal.

Option `file` is used only if `device` is specified (i.e. non NULL).

Value

None. Displays the plots on the chosen device.

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Note

End-user function.

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

References

- Dazard J-E. and J. S. Rao (2010). "*Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data.*" In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "*R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization.*" In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.
- Dazard J-E. and J. S. Rao (2012). "*Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data.*" *Comput. Statist. Data Anal.* 56(7):2317-2333.

See Also

justvsn (R package **vs**n) Variance stabilization and calibration for microarray data. loess (R package **stats**) Fit a polynomial surface determined by one or more numerical predictors, using local fitting.

Examples

```
## Not run:
#=====
# Loading the library and its dependencies
#=====
library("MVR")

#=====
```

```

# MVR package news
#=====
MVR.news()

#=====
# MVR package citation
#=====
citation("MVR")

#=====
# Loading of the Synthetic and Real datasets
# (see description of datasets)
#=====
data("Synthetic", "Real", package="MVR")
?Synthetic
?Real

#=====
# Mean-Variance Regularization (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
# Without cluster usage
#=====
nc.min <- 1
nc.max <- 30
probs <- seq(0, 1, 0.01)
n <- 6
GF <- factor(gl(n = 2, k = n/2, length = n),
             ordered = FALSE,
             labels = c("M", "S"))
mvr.obj <- mvr(data = Real,
              block = GF,
              log = FALSE,
              nc.min = nc.min,
              nc.max = nc.max,
              probs = probs,
              B = 100,
              parallel = FALSE,
              conf = NULL,
              verbose = TRUE,
              seed = 1234)

#=====
# Summary Stabilization Diagnostic Plots (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
#=====
stabilization.diagnostic(obj = mvr.obj,
                        title = "Stabilization Diagnostic Plots
                                (Real - Multi-Group Assumption)",
                        span = 0.75,
                        degree = 2,
                        family = "gaussian",

```

```
device = NULL,  
horizontal = FALSE,  
width = 7,  
height = 5)  
  
## End(Not run)
```

Synthetic

Multi-Groups Synthetic Dataset

Description

Generation of a synthetic dataset with $n=10$ observations (samples) and $p = 100$ variables, where $nvar = 20$ of them are significantly different between the two sample groups.

This is a balanced design with two sample groups ($G = 2$), under unequal sample group variance.

Usage

```
data("Synthetic", package="MVR")
```

Format

A numeric matrix containing $n = 10$ observations (samples) by rows and $p = 100$ variables by columns, named v_1, \dots, v_p . Samples are balanced ($n_1 = 5, n_2 = 5$) between the two groups (G_1, G_2). Compressed Rda data file.

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

Source

See model #2 in Dazard et al., 2011, 2012.

References

- Dazard J-E. and J. S. Rao (2010). "Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data." In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization." In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.
- Dazard J-E. and J. S. Rao (2012). "Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data." Comput. Statist. Data Anal. 56(7):2317-2333.

| | |
|-------------------|--|
| target.diagnostic | <i>Function for Plotting Summary Target Moments Diagnostic Plots</i> |
|-------------------|--|

Description

Plot comparative distribution densities of means and standard deviations of the data before and after Mean-Variance Regularization to check for location shifts between observed first and second moments and their expected target values under a target centered homoscedastic model.

Plot comparative QQ scatterplots to look at departures between observed distributions of first and second moments of the MVR-transformed data and their theoretical distributions assuming independence and normality of all the variables.

Usage

```
target.diagnostic(obj,
                  title = "Target Moments Diagnostic Plots",
                  device = NULL,
                  file = "Target Moments Diagnostic Plots",
                  path = getwd(),
                  horizontal = FALSE,
                  width = 8.5,
                  height = 6.5, ...)
```

Arguments

| | |
|--------|--|
| obj | Object of class "MVR" returned by <code>mvr</code> . |
| title | Title of the plot. Defaults to "Target Moments Diagnostic Plots". |
| device | Graphic display device in {NULL, "PS", "PDF"}. Defaults to NULL (standard output screen). Currently implemented graphic display devices are "PS" (Postscript) or "PDF" (Portable Document Format). |
| file | File name for output graphic. Defaults to "Target Moments Diagnostic Plots". |
| path | Absolute path (without final (back)slash separator). Defaults to working directory path. |

| | |
|------------|---|
| horizontal | Logical scalar. Orientation of the printed image. Defaults to FALSE, that is potrait orientation. |
| width | Numeric scalar. Width of the graphics region in inches. Defaults to 8.5. |
| height | Numeric scalar. Height of the graphics region in inches. Defaults to 6.5. |
| ... | Generic arguments passed to other plotting functions. |

Details

The plots of the density distribution of means and standard deviations checks that the distributions of means and standard deviations of the MVR-transformed data have correct target first moments, i.e. with mean ~ 0 and mean ~ 1 . The expected target mean and standard deviation are shown in red (before and) after MVR-transformation. Caption shows the p-values from the parametric two-sample two-sided t-tests for the equality of parameters to their expectations (assuming normality since usually sample sizes are large : $p \gg 1$, or a relative robustness to moderate violations of the normality assumption).

In the general case, the variables are not normally distributed and not even independent and identically distributed before and after MVR-transformation. Therefore, the distributions of untransformed first and second moments usually differ from their respective theoretical null distributions, i.e., from $N(0, \frac{1}{n})$ for the means and from $\sqrt{\frac{\chi^2_{n-G}}{n-G}}$ for the standard deviations, where G denotes the number of sample groups (see *Dazard, J-E. and J. S. Rao (2012)* for more details). Also, the observed distributions of transformed first and second moments are unknown. This is reflected in the QQ plots, where theoretical and empirical quantiles do not necessarily align with each other. Caption shows the p-values from the nonparametric two-sample two-sided Kolmogorov-Smirnov tests of the null hypothesis that a parameter distribution differs from its theoretical distribution. Each black dot represents a variable. The red solid line depicts the interquartile line, which passes through the first and third quartiles.

Option `file` is used only if `device` is specified (i.e. non NULL).

Value

None. Displays the plots on the chosen `device`.

Acknowledgments

This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This project was partially funded by the National Institutes of Health (P30-CA043703).

Note

End-user function.

Author(s)

- "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>
- "Hua Xu, Ph.D." <huaxu77@gmail.com>
- "Alberto Santana, MBA." <ahs4@case.edu>

Maintainer: "Jean-Eudes Dazard, Ph.D." <jean-eudes.dazard@case.edu>

References

- Dazard J-E. and J. S. Rao (2010). "Regularized Variance Estimation and Variance Stabilization of High-Dimensional Data." In JSM Proceedings, Section for High-Dimensional Data Analysis and Variable Selection. Vancouver, BC, Canada: American Statistical Association IMS - JSM, 5295-5309.
- Dazard J-E., Hua Xu and J. S. Rao (2011). "R package MVR for Joint Adaptive Mean-Variance Regularization and Variance Stabilization." In JSM Proceedings, Section for Statistical Programmers and Analysts. Miami Beach, FL, USA: American Statistical Association IMS - JSM, 3849-3863.
- Dazard J-E. and J. S. Rao (2012). "Joint Adaptive Mean-Variance Regularization and Variance Stabilization of High Dimensional Data." *Comput. Statist. Data Anal.* 56(7):2317-2333.

See Also

justvsn (R package **vs**n) Variance stabilization and calibration for microarray data. loess (R package **stats**) Fit a polynomial surface determined by one or more numerical predictors, using local fitting.

Examples

```
## Not run:
#####
# Loading the library and its dependencies
#####
library("MVR")
library("RColorBrewer")

#####
# MVR package news
#####
MVR.news()

#####
# MVR package citation
#####
citation("MVR")

#####
# Loading of the Synthetic and Real datasets
# (see description of datasets)
#####
data("Synthetic", "Real", package="MVR")
?Synthetic
?Real

#####
# Mean-Variance Regularization (Real dataset)
# Multi-Group Assumption
# Assuming unequal variance between groups
# Without cluster usage
```


Index

- * **Documentation**
 - MVR-package, 2
 - * **High Performance Computing**
 - mvr, 9
 - MVR-package, 2
 - mvrt.test, 16
 - * **High-Dimensional Data**
 - MVR-package, 2
 - * **Mean-Variance Estimators**
 - cluster.diagnostic, 5
 - mvr, 9
 - MVR-package, 2
 - mvrt.test, 16
 - normalization.diagnostic, 23
 - stabilization.diagnostic, 27
 - target.diagnostic, 32
 - * **Normalization**
 - cluster.diagnostic, 5
 - mvr, 9
 - MVR-package, 2
 - normalization.diagnostic, 23
 - stabilization.diagnostic, 27
 - target.diagnostic, 32
 - * **Parallel Programming**
 - mvr, 9
 - MVR-package, 2
 - mvrt.test, 16
 - * **Regularization**
 - cluster.diagnostic, 5
 - mvr, 9
 - MVR-package, 2
 - normalization.diagnostic, 23
 - stabilization.diagnostic, 27
 - target.diagnostic, 32
 - * **Regularized Test Statistics**
 - MVR-package, 2
 - mvrt.test, 16
 - * **Variance Stabilization**
 - cluster.diagnostic, 5
 - mvr, 9
 - MVR-package, 2
 - normalization.diagnostic, 23
 - stabilization.diagnostic, 27
 - target.diagnostic, 32
 - * **datasets**
 - Real, 26
 - Synthetic, 31
 - * **documentation**
 - MVR.news, 15
- cluster.diagnostic, 3, 5, 10, 18
- MVR (MVR-package), 2
- mvr, 2, 5, 6, 9, 17, 18, 23, 28, 32
- MVR-package, 2
- MVR.news, 3, 15
- mvrt.test, 3, 6, 16
- normalization.diagnostic, 3, 23
- Real, 26
- stabilization.diagnostic, 3, 27
- Synthetic, 31
- target.diagnostic, 3, 32