Package: CJIVE (via r-universe)

August 26, 2024

Type Package

Title Canonical Joint and Individual Variation Explained (CJIVE)

Version 0.1.0

Maintainer Raphiel Murden <`rmurden@emory.edu`>

Description Joint and Individual Variation Explained (JIVE) is a method for decomposing multiple datasets obtained on the same subjects into shared structure, structure unique to each dataset, and noise. The two most common implementations are R.JIVE, an iterative approach, and AJIVE, which uses principal angle analysis. JIVE estimates subspaces but interpreting these subspaces can be challenging with AJIVE or R.JIVE. We expand upon insights into AJIVE as a canonical correlation analysis (CCA) of principal component scores. This reformulation, which we call CJIVE, 1) provides an ordering of joint components by the degree of correlation between corresponding canonical variables; 2) uses a computationally efficient permutation test for the number of joint components, which provides a p-value for each component; and 3) can be used to predict subject scores for out-of-sample observations. Please cite the following article when utilizing this package: Murden, R., Zhang, Z., Guo, Y., & Risk, B. (2022) <doi:10.3389/fnins.2022.969510>.

License MIT + file LICENSE

Encoding UTF-8

Imports rootSolve, ggplot2, reshape2, fields, gplots, psych

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Raphiel Murden [aut, cre] (<`https://orcid.org/0000-0002-6396-9105`>), Benjamin Risk [aut]

Date/Publication 2023-01-20 10:10:13 UTC

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

RemoteUrl https://github.com/cran/CJIVE

RemoteRef HEAD

RemoteSha 4532e2a33f36e4ac3ce38d1b63ef4fdeba0dd0db

Contents

`AdjSigVarExp` *Adjust Signal Variation Explained*

Description

Adjusts the proportion of total variation attributable to each signal component to predetermined values

Usage

```
AdjSigVarExp(J, I, N, JntVarEx, IndVarEx)
```

Arguments

`J`	joint signal matrix of size n-by-p
`I`	individual signal matrix of size n-by-p
`N`	noise/error matrix of size n-by-p
`JntVarEx`	desired proportion of total variation explained by the joint signal
`IndVarEx`	desired proportion of total variation explained by the individual signal

Value

a list of 3 items: 1) adjusted joint signal matrix; 2) adjusted individual signal matrix; 3) data matrix additively comprised of the adjusted signal matrices

---

`cc.jive` *Canonical (Correlation) JIVE*

Description

Performs Canonical JIVE as described in the CJVE manuscript. This method is equivalent to AJIVE for 2 data sets.

Usage

```
cc.jive(
  dat.blocks,
  signal.ranks = NULL,
  joint.rank = 1,
  perc.var = 0.95,
  perm.test = TRUE,
  center = FALSE,
  nperms = 1000
)
```

Arguments

`dat.blocks`	a list of two matrices with samples along rows and features along columns, which contain data on the same n individuals/sampling units
`signal.ranks`	a vector of length two which contains the rank for the signal within each data block. The rank corresponds to the number of principal components (PCs) to be retained within each data block. If NULL, the ranks are determined by the parameter 'perc.var.' Default is NULL
`joint.rank`	The rank of the joint subspace i.e., number of components in the joint subspace
`perc.var`	an alternative to signal.ranks that allows specification of ranks based on the desired proportion of total variation to be retained. F For perc.var = p (where 0<p<1), rank is determined as the minimum number of eigenvalues whose cumulative sum is at least p*(total sum of eigenvalues) Default is 0.95 (i.e. 95% of total variation preserved for each data block).

`perm.test`	logical (TRUE/FALSE) of whether permutation test for joint rank should be performed. Overrides 'joint.rank' parameter if TRUE. Default is TRUE
`center`	logical (TRUE/FALSE) indicating whether data should be column-centered prior to testing. Default is TRUE
`nperms`	integer value indicating the number of permutations that should be performed. Default is 1000

Value

A list of two lists: 1) 'CanCorRes' contains results from the canonical correlation of PC scores including, the joint rank, joint subject sores, canonical correlations (and their respective p-values if perm.test was used), canonical loadings for the joint subspace, and total signal ranks 2) 'sJIVE', i.e. Simple JIVE results, correspond to the AJIVE when all ranks are known; includes the joint and individual signal matrices, concatenated PC scores, and the projection matrix used to project each data block onto the joint subspace

Examples

```
#Assign sample size and the number of features in each dataset
n = 200 #sample size
p1 = 100 #Number of features in data set X1
p2 = 100 #Number of features in data set X2

# Assign values of joint and individual signal ranks
r.J = 1 #joint rank
r.I1 = 2 #individual rank for data set X1
r.I2 = 2 #individual rank for data set X2


# Simulate data sets
ToyDat = GenerateToyData(n = 200, p1 = p1, p2 = p2, JntVarEx1 = 0.05, JntVarEx2 = 0.05,
                    IndVarEx1 = 0.25, IndVarEx2 = 0.25, jnt_rank = r.J, equal.eig = FALSE,
                      ind_rank1 = r.I1, ind_rank2 = r.I2, SVD.plots = TRUE, Error = TRUE,
                         print.cor = TRUE)
# Store simulated data sets in an object called 'blocks'
blocks <- ToyDat$'Data Blocks'

# Save Subject scores as R objects
JntScores = ToyDat[['Scores']][['Joint']]
IndivScore.X = ToyDat[['Scores']][["Indiv_1"]]
IndivScore.Y = ToyDat[['Scores']][["Indiv_2"]]

# Save joint variable loadings as R objects
JntLd.X = t(ToyDat$Loadings$Joint_1)
JntLd.Y = t(ToyDat$Loadings$Joint_2)

# Save individual variable loadings as R objects
IndivLd.X =t(ToyDat$Loadings$Indiv_1)
IndivLd.Y = t(ToyDat$Loadings$Indiv_2)

# Save joint, individual, and noise signal matrices as R objects
```

```
JX = ToyDat[[1]]$J1
JY = ToyDat[[1]]$J2
IX = ToyDat[[1]]$I1
IY = ToyDat[[1]]$I2
EX = ToyDat[[1]]$E1
EY = ToyDat[[1]]$E2


## Check that proportions of variation explained are (approximately) equal to intended values
JVE.X = MatVar(JX)/MatVar(blocks[[1]])
JVE.Y = MatVar(JY)/MatVar(blocks[[2]])

IVE.X = MatVar(IX)/MatVar(blocks[[1]])
IVE.Y = MatVar(IY)/MatVar(blocks[[2]])

TotVE.X = MatVar((JX + IX))/MatVar(blocks[[1]])
TotVE.Y = MatVar((JY + IY))/MatVar(blocks[[2]])


CJIVE.res = cc.jive(blocks, c(r.I1,r.I2)+r.J, r.J, perm.test = FALSE)
# CJIVE signal matrix estimates
J.hat = CJIVE.res$sJIVE$joint_matrices
I.hat = CJIVE.res$sJIVE$indiv_matrices

# CJIVE loading estimates
WJ = lapply(J.hat, function(x) x[['v']])
WI = lapply(I.hat, function(x) x[['v']])

# Plots of CJIVE estimates against true counterparts and include an estimate of their chordal norm
layout(matrix(1:6,2, byrow = TRUE))
plot(JntScores, CJIVE.res$CanCorRes$Jnt_Scores, xlab = "True Joint Scores",
    ylab = "CJIVE Joint Scores",
    sub = paste0("Chordal Norm = ",
                 round(chord.norm.diff(JntScores, CJIVE.res$CanCorRes$Jnt_Scores), 3)))
plot(JntLd.X, WJ[[1]][,1], xlab = "True Joint Loadings X", ylab = "CJIVE Joint Loadings X",
    sub = paste0("Chordal Norm = ", round(chord.norm.diff(JntLd.X, WJ[[1]][,1]), 3)))
plot(JntLd.Y, WJ[[2]][,1], xlab = "True Joint Loadings Y", ylab = "CJIVE Joint Loadings Y",
    sub = paste0("Chordal Norm = ", round(chord.norm.diff(JntLd.Y, WJ[[2]][,1]), 3)))
plot.new(); legend("left", paste("Comp.", 1:2), pch = 1, col  = c("orange", "green"),bty = "n" )
plot(IndivLd.X, WI[[1]][,1:2], xlab = "True Individual Loadings X",
    ylab = "CJIVE Individual Loadings X",
    col = c(rep("orange",p1), rep("green",p2)),
   sub = paste0("Chordal Norm = ", round(chord.norm.diff(IndivLd.X, WI[[1]][,1:2]), 3)))
plot(IndivLd.Y, WI[[2]][,1:2], xlab = "True Individual Loadings Y",
    ylab = "CJIVE Individual Loadings Y",
    col = c(rep("orange",p1), rep("green",p2)),
   sub = paste0("Chordal Norm = ", round(chord.norm.diff(IndivLd.Y, WI[[2]][,1:2]), 3)))
layout(1)
```

`cc.jive.pred` *CJIVE joint subject score prediction*

Description

Predicts joint scores for new subjects based on CJIVE joint scores

Usage

```
cc.jive.pred(
  orig.dat.blocks,
  new.subjs,
  signal.ranks,
  cc.jive.loadings,
  can.cors
)
```

Arguments

`orig.dat.blocks`	list of the two data matrices on which CJIVE was initially conducted
`new.subjs`	list of two data matrices containing information on new subjects
`signal.ranks`	a vector of length two which contains the rank for the signal within each data block. The rank corresponds to the number of principal components (PCs) to be retained within each data block. If NULL, the ranks are determined by the parameter 'perc.var.' Default is NULL
`cc.jive.loadings`	canonical loadings for the joint subspace
`can.cors`	canonical correlations from the PCs of the data on which CJIVE was initially conducted - notated as rho_j in CJIVE manuscript

Value

matrix of joint subject score for new subjects

Examples

```
n = 200 #sample size
p1 = 100 #Number of features in data set X1
p2 = 100 #Number of features in data set X2
# Assign values of joint and individual signal ranks
r.J = 1 #joint rank
r.I1 = 2 #individual rank for data set X1
r.I2 = 2 #individual rank for data set X2
true_signal_ranks = r.J + c(r.I1,r.I2)
# Simulate data sets
ToyDat = GenerateToyData(n = n, p1 = p1, p2 = p2, JntVarEx1 = 0.05, JntVarEx2 = 0.05,
                    IndVarEx1 = 0.25, IndVarEx2 = 0.25, jnt_rank = r.J, equal.eig = FALSE,
                      ind_rank1 = r.I1, ind_rank2 = r.I2, SVD.plots = TRUE, Error = TRUE,
                          print.cor = TRUE)
# Store simulated data sets in an object called 'blocks'
blocks <- ToyDat$'Data Blocks'
# Split data randomly into two subsamples
```

```
rnd.smp = sample(n, n/2)
blocks.sub1 = lapply(blocks, function(x){x[rnd.smp,]})
blocks.sub2 = lapply(blocks, function(x){x[-rnd.smp,]})
# Joint scores for the two sub samples
JntScores.1 = ToyDat[['Scores']][['Joint']][rnd.smp]
JntScores.2 = ToyDat[['Scores']][['Joint']][-rnd.smp]
# Conduct CJIVE analysis on the first sub-sample and store the canonical loadings and canonical
# correlations
cc.jive.res_sub1 = cc.jive(blocks.sub1, signal.ranks = r.J+c(r.I1,r.I2), center = FALSE,
                           perm.test = FALSE, joint.rank = r.J)
cc.ldgs1 = cc.jive.res_sub1$CanCorRes$Loadings
can.cors = cc.jive.res_sub1$CanCorRes$Canonical_Correlations[1:r.J]
# Conduct CJIVE analysis on the second sub-sample. We will predict these joint scores using the
# results above
cc.jive.res_sub2 = cc.jive(blocks.sub2, signal.ranks = true_signal_ranks, center = FALSE,
                          perm.test = FALSE, joint.rank = r.J)
cc.jnt.scores.sub2 = cc.jive.res_sub2$CanCorRes$Jnt_Scores
cc.pred.jnt.scores.sub2 = cc.jive.pred(blocks.sub1, new.subjs = blocks.sub2,
                                      signal.ranks = true_signal_ranks,
                                      cc.jive.loadings = cc.ldgs1, can.cors = can.cors)
# Calculate the Pearson correlation coefficient between predicted and calculated joint scores
# for sub-sample 2
cc.pred.cor = diag(cor(cc.pred.jnt.scores.sub2, cc.jnt.scores.sub2))
print(cc.pred.cor)
# Plots of CJIVE estimates against true counterparts and include an estimate of their chordal
# norm
layout(matrix(1:2, ncol = 2))
plot(JntScores.2, cc.pred.jnt.scores.sub2, ylab = "Predicted Joint Scores",
    xlab = "True Joint Scores",
    col = rep(1:r.J, each = n/2),
    main = paste("Chordal Norm = ",
    round(chord.norm.diff(JntScores.2, cc.pred.jnt.scores.sub2),2)))
legend("topleft", legend = paste("Component", 1:r.J), col = 1:r.J, pch = 1)
plot(cc.jnt.scores.sub2, cc.pred.jnt.scores.sub2, ylab = "Predicted Joint Scores",
   xlab = "Estimated Joint Scores",
    col = rep(1:r.J, each = n/2),
    main = paste("Chordal Norm = ",
                 round(chord.norm.diff(cc.jnt.scores.sub2, cc.pred.jnt.scores.sub2),2)))
layout(1)
```

`chord.norm.diff` *Chordal norm between column-subspaces of two matrices*

Description

Calculates the chordal norm between the column subspaces of two matrices. Matrices must have the same number of rows. Let U_x and U_y represent the singular vectors of matrices X and Y, respectively. The chordal norm can be calculated as the square root of the sum of the singular values of t(U_x)

Usage

```
chord.norm.diff(X, Y, tol = 1e-08)
```

Arguments

`X`	a matrix with the same number of rows as Y and any number of columns
`Y`	a matrix with the same number of rows as X and any number of columns
`tol`	threshold under which singular values of inner product are zeroed out

Value

(Numeric) Chordal norm between column-subspaces of X and Y, scaled to the interval [0,1]

`ConvSims_gg` *Convert simulation study results*

Description

Convert results from simulation study into a form for graphing with ggplot

Usage

```
ConvSims_gg(AllSims)
```

Arguments

`AllSims`	matrix with each row representing results from a replicate in the simulation study described in CJIVE manuscript

Value

list of 2 items: 1) joint ranks determined by each method employed in the simulations study 2) chordal norms between true and estimated joint/individual loadings/scores for each method employed in the simulation study

`create.graph.long` *Function for plotting networks with ggplot*

Description

Convert matrix representation of a network for graphical display via ggplot

Usage

```
create.graph.long(gmatrix, sort_indices = NULL)
```

Arguments

`gmatrix`	square matrix of size p-by-p in which entries represent the strength of (un-directed) edges between the p nodes
`sort_indices`	vector of length p by which nodes are sorted. If NULL, then nodes are not sorted. Default is NULL.

Value

a data frame of three variables: X1, which represents the row from which the edge comes; X2, which represents the column from which the edge comes; 3) value, matrix entry representing the strength of the edge between the nodes represented by X1 and X2

`GenerateToyData` *Generate 'Toy' Data*

Description

Generates two Simulated Datasets that follow JIVE Model using binary subject scores

Usage

```
GenerateToyData(
  n,
  p1,
  p2,
  JntVarEx1,
  JntVarEx2,
  IndVarEx1,
  IndVarEx2,
  jnt_rank = 1,
  equal.eig = FALSE,
  ind_rank1 = 2,
  ind_rank2 = 2,
  SVD.plots = TRUE,
  Error = TRUE,
  print.cor = TRUE
)
```

Arguments

`n`	integer for sample size, i.e. number of subjects
`p1`	integer for number of features/variables in first data set
`p2`	integer for number of features/variables in second data set
`JntVarEx1`	numeric between (0,1) which describes proportion of variance in the first data set which is attributable to the joint signal
`JntVarEx2`	numeric between (0,1) which describes proportion of variance in the second data set which is attributable to the joint signal
`IndVarEx1`	numeric between (0,1) which describes proportion of variance in the first data set which is attributable to the individual signal
`IndVarEx2`	numeric between (0,1) which describes proportion of variance in the second data set which is attributable to the individual signal
`jnt_rank`	integer for rank of the joint signal, i.e., number of joint components
`equal.eig`	logical (TRUE/FALSE) for whether components should contribute equal variance to signal matrices - default is FALSE
`ind_rank1`	integer for rank of the individual signal in first data set, i.e., number of joint components
`ind_rank2`	integer for rank of the individual signal in second data set, i.e., number of joint components
`SVD.plots`	logical (TRUE/FALSE) for whether plots of singular values from signal should be produced - used to confirm number of components
`Error`	logical (TRUE/FALSE) final data sets should be noise contaminated - default is FALSE; use TRUE to obtain pure signal datasets
`print.cor`	logical (TRUE/FALSE) for whether to print matrix of correlations between subject scores)

Value

A 'list' object which contains 1) list of signal matrices which additively comprise the simulated data sets, i.e. joint, individual, and error matrices for each data set; 2) list of simulated data sets (each equal to the sum of the matrices in part 1); 3) list of joint subject scores and individual subject scores for each data set, and 4) lsit of joint and individual loadings for each data set

Examples

```
ToyDat = GenerateToyData(n = 200, p1 = 2000, p2 = 1000, JntVarEx1 = 0.05, JntVarEx2 = 0.05,
                          IndVarEx1 = 0.25, IndVarEx2 = 0.25, jnt_rank = 1, equal.eig = FALSE,
                               ind_rank1 = 2, ind_rank2 = 3, SVD.plots = TRUE, Error = TRUE,
                               print.cor = TRUE)
```

`GetSimResults_Dir` *Retrieve simulation results*

Description

Retrives and compiles results from simulation study which are stored in a directory. A directory should contain separate .csv files (one per replicate), each of which will include all evaluation metrics and most experimental settings for that particular replicate. For the CJIVE manuscript, a directory houses results of all 100 replicates for each combination of experimental factors.

Usage

```
GetSimResults_Dir(sim.dir, p1, p2, Preds = FALSE)
```

Arguments

`sim.dir`	(character string) file path for the directory from which results will be retrieved
`p1`	number of features in data set 1
`p2`	number of features in data set 2
`Preds`	(logical) do the replicate results contain correlations between predicted and true joint subject scores. Default is FALSE

Value

upper triangular p-by-p matrix

`gg.corr.plot` *Function for plotting Pearson correlations between predicted and true subject scores within the simulation study described in CJIVE manuscript*

Description

Graphically displays the center and spread of chordal norms for joint/individual subject score subspaces

Usage

```
gg.corr.plot(cor.dat, cols, show.legend = FALSE, text.size)
```

Arguments

`cor.dat`	data frame with at least the 5 following variables: Norm - the value of the norm for a particular subspace; Type - the subspace for which the norm is given (i.e., joint/individual score/loading for dataset X1 or X2 (except for joint scores)) Method - the method by which the subspace was estimated, e.g. CJIVE, AJIVE, R.JIVE JVE_1 and JVE_2 - labels describing the proportion of joint variation explained in each dataset (and typically the number of variables in dataset X2)
`cols`	a vector of colors, must have length equal to the number of methods used in the simulation
`show.legend`	logical (TRUE/FALSE) for whether a legend should be included in the plot. Default is FALSE
`text.size`	numeric value for the font size

Value

graphical display (via ggplot2)

`gg.load.norm.plot` *Function for plotting chordal norms between estimated and true variable loading subspaces within the simulation study described in CJIVE manuscript*

Description

Graphically displays the center and spread of chordal norms for joint/individual variable loading subspaces

Usage

```
gg.load.norm.plot(
  norm.dat,
  cols,
  show.legend = FALSE,
  text.size,
  lty = 1,
  y.max = 1,
  x.lab.angle = 70
)
```

Arguments

`norm.dat`	data frame with at least the 5 following variables: Norm - the value of the norm for a particular subspace; Type - the subspace for which the norm is given (i.e., joint/individual variable loadings for dataset X1 or X2) Method - the method by which the subspace was estimated, e.g. CJIVE, AJIVE, R.JIVE JVE_1 and JVE_2 - labels describing the proportion of joint variation explained in each dataset (and typically the number of variables in dataset X2)

`cols`	a vector of colors, must have length equal to the number of methods used in the simulation
`show.legend`	logical (TRUE/FALSE) for whether a legend should be included in the plot. Default is FALSE
`text.size`	numeric value for the font size
`lty`	linetype (see ggplot2). Default = 1
`y.max`	maximum value for the horizontal axis of the plot
`x.lab.angle`	angle at which x-axis labels are tilted

Value

graphical display (via ggplot2)

`gg.norm.plot` *Function for plotting chordal norms between estimated and true subspaces within the simulation study described in CJIVE manuscript*

Description

Graphically displays the center and spread of chordal norms for joint/individual score/loading subspaces

Usage

```
gg.norm.plot(
  norm.dat,
  cols,
  show.legend = FALSE,
  text.size,
  lty = 1,
  y.max = 1,
  x.lab.angle = 70
)
```

Arguments

`norm.dat`	data frame with at least the 5 following variables: Norm - the value of the norm for a particular subspace; Type - the subspace for which the norm is given (i.e., joint/individual score/loading for dataset X1 or X2 (except for joint scores)) Method - the method by which the subspace was estimated, e.g. CJIVE, AJIVE, R.JIVE JVE_1 and JVE_2 - labels describing the proportion of joint variation explained in each dataset (and typically the number of variables in dataset X2)
`cols`	a vector of colors, must have length equal to the number of methods used in the simulation
`show.legend`	logical (TRUE/FALSE) for whether a legend should be included in the plot. Default is FALSE

`text.size`	numeric value for the font size
`lty`	linetype (see ggplot2). Default = 1
`y.max`	maximum value for the horizontal axis of the plot
`x.lab.angle`	angle at which x-axis labels are tilted

Value

graphical display (via ggplot2)

`gg.rank.plot` *Function for plotting selected joint ranks*

Description

Graphically displays the count of joint ranks selected by each method employed in the simulation study described in the CJIVE manuscript

Usage

```
gg.rank.plot(rank.dat, cols, show.legend = FALSE, text.size, num.sims)
```

Arguments

`rank.dat`	data frame expected to be built with the functions dplyr::count and tidyr::complete, which should include the following variables Rank - numeric values of the rank selected by each method in each replicate simulation n - the number of times this value was selected as the rank Type - the subspace for which the norm is given (i.e., joint/individual score/loading for dataset X1 or X2 (except for joint scores)) Method - the method by which the subspace was estimated, e.g. CJIVE, AJIVE, R.JIVE JVE_1 and JVE_2 - labels describing the proportion of joint variation explained in each dataset (and typically the number of variables in dataset X2)
`cols`	a vector of colors, must have length equal to the number of methods used in the simulation
`show.legend`	logical (TRUE/FALSE) for whether a legend should be included in the plot. Default is FALSE
`text.size`	numeric value for the font size
`num.sims`	numeric value for the number of replicates evaluated in each full combination of experimental settings

Value

graphical display (via ggplot2)

`gg.score.norm.plot` *Function for plotting chordal norms between estimated and true subject score subspaces within the simulation study described in CJIVE manuscript*

Description

Graphically displays the center and spread of chordal norms for joint/individual subject score subspaces

Usage

```
gg.score.norm.plot(
  norm.dat,
  cols,
  show.legend = FALSE,
  text.size,
  lty = 1,
  y.max = 1,
  x.lab.angle = 70
)
```

Arguments

`norm.dat`	data frame with at least the 5 following variables: Norm - the value of the norm for a particular subspace; Type - the subspace for which the norm is given (i.e., joint and individual subject scores for dataset X1 or X2 (except joint scores, which are for both datasets)) Method - the method by which the subspace was estimated, e.g. CJIVE, AJIVE, R.JIVE JVE_1 and JVE_2 - labels describing the proportion of joint variation explained in each dataset (and typically the number of variables in dataset X2)
`cols`	a vector of colors, must have length equal to the number of methods used in the simulation
`show.legend`	logical (TRUE/FALSE) for whether a legend should be included in the plot. Default is FALSE
`text.size`	numeric value for the font size
`lty`	linetype (see ggplot2). Default = 1
`y.max`	maximum value for the horizontal axis of the plot
`x.lab.angle`	angle at which x-axis labels are tilted

Value

graphical display (via ggplot2)

MatVar — *Matrix variation (i.e. Frobenius norm)*

Description

Calculates the Frobenius norm of a matrix, which can be used as a measure of total variation

Usage

```
MatVar(X)
```

Arguments

`X`	a matrix of any size

Value

The Frobenius norm of the matrix X, calculated as the square root of the sum of squared entries in X

Examples

```
X = matrix(rnorm(10), 5,2)
MatVar(X)
```

MatVar2 — *Alternative calculation - Matrix variation (i.e. Frobenius norm)*

Description

Calculates the Frobenius norm of a matrix, which can be used as a measure of total variation

Usage

```
MatVar2(X)
```

Arguments

`X`	a matrix of any size

Value

The Frobenius norm of the matrix X, calculated as the square root of the trace of t(X)

Examples

```
X = matrix(rnorm(10), 5,2)
MatVar2(X)
```

`Melt.Sim.Cors`	*Converts correlations of predicted to true joint subject scores to a format conducive to ggplot2*

Description

Converts correlations of predicted to true joint subject scores into a format conducive to ggplot2

Usage

```
Melt.Sim.Cors(sim.dat, r.J, p1, p2)
```

Arguments

`sim.dat`	matrix with each row representing results from a replicate in the simulation study described in CJIVE manuscript
`r.J`	(Numeric/integer) the joint rank, i.e. number of components in the joint subspace
`p1`	number of variables/features in data set X1
`p2`	number of variables/features in data set X2

Value

data frame with seven columns: one each for the joint variance explained in each data set, one column containing the method by which predictions were obtained, one column containing the component number (1,...,r.J),

`perm.jntrank`	*Permutation Test for Joint Rank in CJIVE*

Description

Conducts the permutation test for the number of joint components as described in CJIVE manuscript. Briefly, canonical correlations (CC) between principal component vectors of the data are obtained (PC). Then for 1:nperms, the rows of one data set are permuted and CCs between PC vectors are calculated, retaining the maximum CC. These maximum CCs form a null distribution against which the original CCs are tested. The number of original CCs exceeding the (1-alpha)^th percentile is the returned as the joint rank.

Usage

```
perm.jntrank(
  dat.blocks,
  signal.ranks = NULL,
  nperms = 500,
  perc.var = 0.95,
  alpha = 0.05,
  center = TRUE
)
```

Arguments

`dat.blocks`	a list of two matrices with samples along rows and features along columns, which contain data on the same n individuals/sampling units
`signal.ranks`	a vector of length two which contains the rank for the signal within each data block. The rank corresponds to the number of principal components (PCs) to be retained within each data block. If NULL, the ranks are determined by the parameter 'perc.var.' Default is NULL
`nperms`	integer value indicating the number of permutations that should be performed
`perc.var`	numeric value of either a scalar or of length 2: an alternative to signal.ranks that allows specification of signal ranks based on the desired proportion of total variation to be retained in each data block. For perc.var = p (where 0<p<1), rank is determined as the minimum number of eigenvalues whose cumulative sum is at least p*(total sum of eigenvalues). Default is 0.95 (i.e. 95% of total variation preserved for each data block). For p=c(p1,p2) pk is used to determine the rank of block k
`alpha`	nominal type-I error rate
`center`	logical (TRUE/FALSE) indicating whether data should be column-centered prior to testing. Default is TRUE

Value

The Frobenius norm of the matrix X, calculated as the sum of square entries in X

`scale_loadings` *Scale and sign-correct variable loadings to assist interpretation*

Description

Scale loadings for a joint or individual component by its largest absolute value resulting in loadings between -1 and 1. Loadings are also sign-corrected to result in positive skewness

Usage

```
scale_loadings(loading.comp)
```

Arguments

`loading.comp`	numeric vector of variable loadings from a JIVE analysis

Value

numeric vector of loadings which have been scaled and sign-corrected

---

`show.image.2` *Display a heatmap of a matrix (adapted from Erick Lock's show.image function in the r.jive package)*

Description

Visual display of a matrix as a heatmap with colors determined by entry values, and including a colorbar to aid interpretation of the heatmap

Usage

```
show.image.2(
  Image,
  ylab = "",
  xlab = "",
  net = FALSE,
  main = "",
  sub = "",
  colorbar = TRUE
)
```

Arguments

`Image`	matrix to display
`ylab`	lab for y-axis of heatmap
`xlab`	lab for x-axis of heatmap
`net`	logical (TRUE/FALUSE) of whether entries correspond to edges between regions of interest in the Power-264 brain atlas. Default is FALSE
`main`	main title for heatmap
`sub`	subtitle for heatmap
`colorbar`	logical (TRUE/FALUSE) of whether colorabar shouldl be included to aid interpretation. Default is TRUE

Value

graphical display of matrix as a heatmap

`sjive` *Simple JIVE*

Description

Conducts AJIVE estimation under the assumption that all ranks are known and no components are discarded

Usage

```
sjive(blocks, signal_ranks, joint.rank, joint_scores = NULL)
```

Arguments

`blocks`	list of data blocks, i.e. matrices, all having the same number of rows, which correspond to the same sampling units (i.e. study participants, patients, etc.)
`signal_ranks`	numerical vector of the same length as 'blocks' with each entry corresponding to the rank of the respective matrix in 'blocks'
`joint.rank`	integer value corresponding to the rank of the joint signal subspace, i.e. number of components in the signal subspace
`joint_scores`	numerical matrix containing joint subject scores if they were calculated by some other method, e.g. Canonical Correlation of PC scores. Must have the same number of rows as each matrix in 'blocks' and number of columns equal to 'joint_rank'. If NULL, joint scores are calculated and returned. Default is NULL.

Value

list of 4 or 5 items: 1) joint signal matrices, their SVDs, and the proportion of total variation in each matrix that is attributable to the joint signal 2) individual signal matrices, their SVDs, and the proportion of total variation in each matrix that is attributable to the individual signal 3) concatenated PC scores, used to determine joint subspace 4) projection matrix for joint subspace 5) joint subject scores (only returned if not provided initially)

`vec2net.l` *Convert vector to network*

Description

Converts a vector of size p choose 2 into a p-by-p lower triangular matrix

Usage

```
vec2net.l(invector)
```

Arguments

`invector`	numeric vector of size p choose 2

Value

lower triangular p-by-p matrix

`vec2net.u` *Convert vector to network*

Description

Converts a vector of size p choose 2 into a p-by-p upper triangular matrix

Usage

```
vec2net.u(invector)
```

Arguments

`invector`	numeric vector of size p choose 2

Value

upper triangular p-by-p matrix

Index