Package ‘bigutilsr’

July 30, 2025

Title Utility Functions for Large-scale Data

Version 0.3.11

Date 2025-07-30

Description Utility functions for large-scale data. For now, package 'bigutilsr' mainly includes functions for outlier detection and unbiased PCA projection.

License GPL-3

Encoding UTF-8

Language en-US

ByteCompile TRUE

RoxygenNote 7.3.2

URL `https://github.com/privefl/bigutilsr`

BugReports `https://github.com/privefl/bigutilsr/issues`

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Imports bigassertr (>= 0.1.1), bigparallelr (>= 0.2.3), nabor, Rcpp, robustbase, RSpectra, stats

Suggests covr, Gmedian, mvtnorm, rrcov, spelling, testthat (>= 2.1.0)

NeedsCompilation yes

Author Florian Privé [aut, cre]

Maintainer Florian Privé <`florian.prive.21@gmail.com`>

Repository CRAN

Date/Publication 2025-07-30 17:40:02 UTC

Contents

`as_model_matrix` *Transform a data frame*

Description

Transform a data frame into a matrix using one hot encoding.

Usage

```
as_model_matrix(df, intercept = FALSE)
```

Arguments

`df`	A data frame.
`intercept`	Whether to have a column with all `1`s. Default is `FALSE`.

Value

A matrix.

Examples

```
mat <- as_model_matrix(iris)
str(mat)
```

covRob — *Deprecated*

Description

Deprecated

Usage

```
covRob(data, ...)
```

Arguments

`data`	A matrix.
`...`	Not used.

See Also

`covrob_ogk()` `dist_ogk()`

covrob_ogk — *Robust Location and Scatter Estimation - Ortogonalized Gnanadesikan-Kettenring (OGK)*

Description

Computes a robust multivariate location and scatter estimate with a high breakdown point, using the pairwise algorithm proposed by Marona and Zamar (2002) which in turn is based on the pairwise robust estimator proposed by Gnanadesikan-Kettenring (1972).

Usage

```
covrob_ogk(U, niter = 2, beta = 0.9)

dist_ogk(U, niter = 2, beta = 0.9)
```

Arguments

`U`	A matrix with no missing values and at least 2 columns.
`niter`	Number of number of iterations for the first step of the algorithm, usually 1 or 2 since iterations beyond the second do not lead to improvement.
`beta`	Coverage parameter for the final reweighted estimate. Default is `0.9`.

Value

`covrob_ogk()`: list of robust estimates, `$cov` and `$center`.

`dist_ogk()`: vector of robust Mahalanobis (squared) distances.

References

Maronna, R.A. and Zamar, R.H. (2002) Robust estimates of location and dispersion of high-dimensional datasets; *Technometrics* **44**(4), 307–317.

Yohai, R.A. and Zamar, R.H. (1998) High breakdown point estimates of regression by means of the minimization of efficient scale *JASA* **86**, 403–413.

Gnanadesikan, R. and John R. Kettenring (1972) Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics* **28**, 81–124.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47.

See Also

`rrcov::CovOgk()`

`stats::mahalanobis()`

Examples

```
X <- readRDS(system.file("testdata", "three-pops.rds", package = "bigutilsr"))
svd <- svds(scale(X), k = 5)

U <- svd$u
dist <- dist_ogk(U)
str(dist)
```

`geometric_median` *Geometric median*

Description

Compute the geometric median, i.e. the point that minimizes the sum of all Euclidean distances to the observations (rows of `U`).

Usage

```
geometric_median(U, tol = 1e-10, maxiter = 1000, by_grp = NULL)
```

Arguments

`U`	A matrix (e.g. PC scores).
`tol`	Convergence criterion. Default is `1e-10`.
`maxiter`	Maximum number of iterations. Default is `1000`.
`by_grp`	Possibly a vector for splitting rows of `U` into groups before computing the geometric mean for each group. Default is `NULL` (ignored).

Value

The geometric median of all rows of `U`, a vector of the same size as `ncol(U)`. If providing `by_grp`, then a matrix with rows being the geometric median within each group.

Examples

```
X <- readRDS(system.file("testdata", "three-pops.rds", package = "bigutilsr"))
pop <- rep(1:3, c(143, 167, 207))

svd <- svds(scale(X), k = 5)
U <- sweep(svd$u, 2, svd$d, '*')
plot(U, col = pop, pch = 20)

med_all <- geometric_median(U)
points(t(med_all), pch = 20, col = "blue", cex = 4)

med_pop <- geometric_median(U, by_grp = pop)
points(med_pop, pch = 20, col = "blue", cex = 2)
```

`hist_out` *Outlier detection (histogram)*

Description

Outlier detection based on departure from histogram. Suitable for compact values (need a space between main values and outliers).

Usage

```
hist_out(x, breaks = nclass.scottRob, pmax_out = 0.2, nboot = NULL)
```

Arguments

`x`	Numeric vector (with compact values).
`breaks`	Same parameter as for `hist()`. Default uses a robust version of Scott's rule. You can also use `"FD"` or `nclass.FD` for a bit more bins.
`pmax_out`	Percentage at each side that can be considered outliers at each step. Default is `0.2`.
`nboot`	Number of bootstrap replicates to estimate limits more robustly. Default is `NULL` (no bootstrap, even if **I would recommend to use it**).

Value

A list with

- `x`: the initial vector, whose outliers have been removed,
- `lim`: lower and upper limits for outlier removal,
- `all_lim`: all bootstrap replicates for `lim` (if `nboot` not `NULL`).

Examples

```
set.seed(1)
x <- rnorm(1000)
str(hist_out(x))

# Easy to separate
x2 <- c(x, rnorm(50, mean = 7))
hist(x2, breaks = nclass.scottRob)
str(hist_out(x2))

# More difficult to separate
x3 <- c(x, rnorm(50, mean = 6))
hist(x3, breaks = nclass.scottRob)
str(hist_out(x3))
str(hist_out(x3, nboot = 999))
```

`knn_parallel` *Find K nearest neighbours for multiple query points*

Description

Find K nearest neighbours for multiple query points

Usage

```
knn_parallel(data, query = data, k, ..., ncores = bigparallelr::nb_cores())
```

Arguments

`data`	Mxd matrix of M target points with dimension d
`query`	Nxd matrix of N query points with dimension d (nb `data` and `query` must have same dimension). If missing defaults to `data` i.e. a self-query.
`k`	an integer number of nearest neighbours to find
`...`	Arguments passed on to `nabor::knn` `eps` An approximate error bound. The default of 0 implies exact matching. `searchtype` A character vector or integer indicating the search type. The default value of `1L` is equivalent to "auto". See details. `radius` Maximum radius search bound. The default of 0 implies no radius bound.
`ncores`	Number of cores to use. Default uses `bigparallelr::nb_cores()`.

Value

A list with elements `nn.idx` (1-indexed indices) and `nn.dists` (distances), both of which are N x k matrices. See details for the results obtained with1 invalid inputs.

Examples

```
## Not run: knn_parallel(matrix(1:4, 2), k = 2, ncores = 2)
```

`LOF` *Local Outlier Factor (LOF)*

Description

LOF: Identifying Density-Based Local Outliers.

Usage

```
LOF(
  U,
  seq_k = c(4, 10, 30),
  combine = max,
  robMaha = FALSE,
  log = TRUE,
  ncores = 1
)
```

Arguments

`U`	A matrix, from which to detect outliers (rows). E.g. PC scores.
`seq_k`	Sequence of numbers of nearest neighbors to use. If multiple `k` are provided, this returns the combination of statistics. Default is `c(4, 10, 30)` and use `max` to combine (see `combine`).
`combine`	How to combine results for multiple `k`? Default uses `max`.
`robMaha`	Whether to use a robust Mahalanobis distance instead of the normal euclidean distance? Default is `FALSE`, meaning using euclidean.
`log`	Whether to return the logarithm of LOFs? Default is `TRUE`.
`ncores`	Number of cores to use. Default is `1`.

References

Breunig, Markus M., et al. "LOF: identifying density-based local outliers." ACM sigmod record. Vol. 29. No. 2. ACM, 2000.

See Also

`prob_dist()`

Examples

```
X <- readRDS(system.file("testdata", "three-pops.rds", package = "bigutilsr"))
svd <- svds(scale(X), k = 10)

llof <- LOF(svd$u)
hist(llof, breaks = nclass.scottRob)
tukey_mc_up(llof)

llof_maha <- LOF(svd$u, robMaha = TRUE)
hist(llof_maha, breaks = nclass.scottRob)
tukey_mc_up(llof_maha)

lof <- LOF(svd$u, log = FALSE)
hist(lof, breaks = nclass.scottRob)
str(hist_out(lof))
str(hist_out(lof, nboot = 100))
str(hist_out(lof, nboot = 100, breaks = "FD"))
```

`maha_trans` *Transform matrix*

Description

Transform matrix to use Mahalanobis distance instead of Euclidean one.

Usage

```
maha_trans(U, estim = covrob_ogk(U))
```

Arguments

`U`	A matrix (e.g. PC scores).
`estim`	List of location and scatter estimates, `$cov` and `$center`.

Value

`U`, transformed.

Examples

```
X <- readRDS(system.file("testdata", "three-pops.rds", package = "bigutilsr"))
svd <- svds(scale(X), k = 5)

U <- svd$u
dist1 <- dist_ogk(U)

U.maha <- maha_trans(U)
dist2 <- rowSums(U.maha^2)
```

```
all.equal(dist2, dist1)
```

`nclass.scottRob` *Compute the Number of Classes for a Histogram*

Description

Compute the Number of Classes for a Histogram

Usage

```
nclass.scottRob(x)
```

Arguments

`x`	a data vector.

Value

The suggested number of classes.

References

Scott, D. W. (1979). On optimal and data-based histograms. Biometrika, 66, 605–610. doi: 10.2307/2335182.

Examples

```
x <- rnorm(1000)
hist(x, breaks = nclass.scott)
hist(x, breaks = nclass.scottRob)

x2 <- c(x, rnorm(50, mean = 50))
hist(x2, breaks = nclass.scott)
hist(x2, breaks = nclass.scott,    xlim = c(-5, 5))
hist(x2, breaks = nclass.scottRob, xlim = c(-5, 5))
```

`pca_nspike` *Number of spikes in PCA*

Description

Estimate the number of distant spikes based on the histogram of eigenvalues.

Usage

```
pca_nspike(eigval, breaks = "FD", nboot = 100)
```

Arguments

eigval	Eigenvalues (squared singular values).
breaks	Same parameter as for `hist()`. Default uses a robust version of Scott's rule. You can also use `"FD"` or `nclass.FD` for a bit more bins.
nboot	Number of bootstrap replicates to estimate limits more robustly. Default is `100`.

Value

The estimated number of distant spikes.

Examples

```
N <- 400; M <- 2000; K <- 8
U <- matrix(0, N, K); U[] <- rnorm(length(U))
V <- matrix(0, M, K); V[] <- rnorm(length(V))
# X = U V^T + E
X <- tcrossprod(U, V) + 15 * rnorm(N * M)
pca <- prcomp(X)
eigval <- pca$sdev^2
plot(head(eigval, -1), log = "xy", pch = 20)
pca_nspike(eigval)
```

`pca_OADP_proj` *OADP projection*

Description

Online Augmentation, Decomposition, and Procrustes (OADP) projection of PC loadings onto some study data `X`.

Usage

```
pca_OADP_proj(X, loadings, sval)

pca_OADP_proj2(XV, X_norm, sval)
```

Arguments

`X`	Data to get PC loadings into.
`loadings`	PC loadings of the reference PCA to project.
`sval`	Singular values of the reference PCA (sqrt of the eigen values). Only the `ncol(loadings)` first ones will be used.
`XV`	`X %*% loadings`
`X_norm`	Vector of sums of squared rows (e.g. `rowSums(X^2)`).

Value

- `pca_OADP_proj()`: A list with the simple projection `X %*% loadings` and the projection based on OADP.
- `pca_OADP_proj2()`: The projection based on OADP only (a matrix of same size of `XV`).

Examples

```
X <- readRDS(system.file("testdata", "three-pops.rds", package = "bigutilsr"))
N <- 400; M <- ncol(X)
ind <- sample(nrow(X), N)

# Compute SVD using one part of samples
svd <- svds(X[ind, ], k = 5)
U <- sweep(svd$u, 2, svd$d, '*')
col <- 2:3
plot(U[, col])
points(cbind(0, 0), pch = 8, col = "green", cex = 2)

# Projecting other samples
proj <- pca_OADP_proj(X = X[-ind, ], loadings = svd$v, sval = svd$d)
points(proj$simple_proj[, col], col = "red", pch = 20)     # shrunk towards 0
points(proj$OADP_proj[, col], col = "blue", pch = 20)      # unshrunk
```

`predict.Procrustes` *Predict method*

Description

Predict method for class `Procrustes`.

Usage

```
## S3 method for class 'Procrustes'
predict(object, X, ...)
```

Arguments

`object`	Object of class `Procrustes`.
`X`	New matrix to transform.
`...`	Not used.

Value

`X`, transformed.

See Also

`procrustes()`.

`prob_dist` *Probabilistic set distance*

Description

Probabilistic set distance

Usage

```
prob_dist(U, kNN = 5, robMaha = FALSE, ncores = 1)
```

Arguments

`U`	A matrix, from which to detect outliers (rows). E.g. PC scores.
`kNN`	Number of nearest neighbors to use. Default is `5`.
`robMaha`	Whether to use a robust Mahalanobis distance instead of the normal euclidean distance? Default is `FALSE`, meaning using euclidean.
`ncores`	Number of cores to use. Default is `1`.

References

Kriegel, Hans-Peter, et al. "LoOP: local outlier probabilities." Proceedings of the 18th ACM conference on Information and knowledge management. ACM, 2009.

See Also

`LOF()`

Examples

```
X <- readRDS(system.file("testdata", "three-pops.rds", package = "bigutilsr"))
svd <- svds(scale(X), k = 10)
U <- svd$u

test <- prob_dist(U)
plof <- test$dist.self / test$dist.nn
plof_ish <- test$dist.self / sqrt(test$dist.nn)
plot(U[, 1:2], col = (plof_ish > tukey_mc_up(plof_ish)) + 1, pch = 20)
plot(U[, 3:4], col = (plof_ish > tukey_mc_up(plof_ish)) + 1, pch = 20)
plot(U[, 5:6], col = (plof_ish > tukey_mc_up(plof_ish)) + 1, pch = 20)
```

`procrustes` *Procrustes transform*

Description

Procrustes transform Y = pXR (after centering), where p is a scaling coefficient and R is a rotation matrix that minimize ||Y - pXR||_F.

Usage

```
procrustes(Y, X, n_iter_max = 1000, epsilon_min = 1e-07)
```

Arguments

`Y`	Reference matrix.
`X`	Matrix to transform (`ncol(X)` >= `ncol(Y)`).
`n_iter_max`	Maximum number of iterations. Default is `1000`.
`epsilon_min`	Convergence criterion. Default is `1e-7`.

Value

Object of class "procrustes", a list with the following elements:

- `$R`: the rotation matrix to apply to `X`,
- `$rho`: the scaling coefficient to apply to `X`,
- `$c`: the column centering to apply to the resulting matrix,
- `$diff`: the average difference between `Y` and `X` transformed.

You can use method `predict()` to apply this transformation to other data.

Examples

```
A <- matrix(rnorm(200), ncol = 20)
B <- matrix(rnorm(length(A)), nrow = nrow(A))

proc <- procrustes(B, A)
str(proc)
plot(B, predict(proc, A)); abline(0, 1, col = "red")
```

`regul_glasso` *Regularization with the graphical lasso*

Description

Use the graphical lasso algorithm to regularize a square symmetric matrix (e.g. a covariance or correlation matrix) by assuming that its inverse has many zeros.

Usage

```
regul_glasso(
  mat,
  lambda,
  maxiter_outer = 200,
  maxiter_lasso = 200,
  tol = 1e-04,
  verbose = FALSE
)
```

Arguments

`mat`	A square symmetric matrix.
`lambda`	Strength of regularization. It needs to be scaled with `mat`. It should also be the maximum difference between the two matrices.
`maxiter_outer`	Maximum number of iterations of the outer loop. Default is 200.
`maxiter_lasso`	Maximum number of iterations of each lasso solver. Default is 200.
`tol`	Tolerance for assessing convergence. Default is 1e-4 and it needs to be scaled with `mat`.
`verbose`	Whether to print iterations and differences. Default is FALSE.

Value

The regularized matrix, where the diagonal should be the same and zeros should be kept as well. It also returns the `lambda` used as an attribute.

Examples

```
(cov <- cov(iris[1:4]))
lambda <- 1 / sqrt(nrow(iris))
(cov_regul <- regul_glasso(cov, lambda))
```

`rollmean` *Gaussian smoothing*

Description

Gaussian smoothing

Usage

```
rollmean(x, size)
```

Arguments

`x`	Numeric vector.
`size`	Radius of the smoothing (smaller than half of the length of `x`). If using `size = 0`, it returns `x`.

Value

Numeric vector of the same length as `x`, smoothed.

Examples

```
(x <- rnorm(10))
rollmean(x, 3)
```

`solve_linear_system` *Solve (A + lam I) x = b*

Description

Solve (A + lam I) x = b

Usage

```
solve_linear_system(A, b, add_to_diag = 0)
```

Arguments

`A`	A *symmetric* square matrix.
`b`	A vector.
`add_to_diag`	One value to add to the diagonal of A (lam). Default is 0.

Value

The best solution x of this linear system.

Examples

```
A <- matrix(rnorm(4), 2); A[1, 2] <- A[2, 1]  # should be symmetric
x <- rnorm(2)
b <- A %*% x
x2 <- drop(solve(A, b))
x3 <- solve_linear_system(A, b)
rbind(x, x2, x3)
```

---

tukey_mc_up *Outlier detection threshold (upper)*

Description

Outlier detection threshold (upper) based on Tukey's rule, corrected for skewness using the 'medcouple', and possibly corrected for multiple testing.

Usage

```
tukey_mc_up(x, coef = NULL, alpha = 0.05, a = -4, b = 3)
```

Arguments

x	Numeric vector. Should be somewhat normally distributed.
coef	number determining how far 'whiskers' extend out from the box. If NULL (default), this is computed to get an type-I error of alpha, after adjusting for multiple testing. A standard value to use is 1.5.
alpha	See coef. Default is 0.05.
a, b	scaling factors multiplied by the medcouple mc() to determine outlyer boundaries; see the references.

References

Hubert, M. and Vandervieren, E. (2008). An adjusted boxplot for skewed distributions, *Computational Statistics and Data Analysis* **52**, 5186–5201.

See Also

robustbase::adjbox()

Examples

```
hist(x <- c(rnorm(3, m = 6), rnorm(1e4, m = 0)))
(q <- tukey_mc_up(x))
abline(v = q, col = "red")
which(x > q)
```

---

varimax2 *Varimax rotation*

Description

Varimax rotation

Usage

```
varimax2(X, normalize = FALSE, reorder = TRUE, rotmat = FALSE)
```

Arguments

`X`	A matrix with more rows than columns.
`normalize`	Whether to apply Kaiser normalization? See stats::varimax. Default is `FALSE`.
`reorder`	Whether to permute rotation vectors to maximize the conservation of the order of the initial columns of `X`. Default is `TRUE`.
`rotmat`	Whether to return the rotation matrix `rot`, or the rotated matrix `X %*% rot` (the default, `FALSE`).

Value

Either the rotation matrix `rot`, or the rotated matrix `X %*% rot`, depending on `rotmat`.

Examples

```
X <- as.matrix(iris[1:4])
X_rot <- varimax2(X)
X_rot2 <- varimax(X, normalize = FALSE)$loadings[]
all.equal(X_rot2, X_rot[, c(3, 2, 1, 4)], check.attributes = FALSE)
varimax2(X, rotmat = TRUE)

X2 <- prcomp(X)$x
X2_rot <- varimax2(X2)
X2_rot2 <- varimax(X2, normalize = FALSE)$loadings[]
all.equal(X2_rot, X2_rot2, check.attributes = FALSE)
```

Index