# Package: consICA (via r-universe)

May 27, 2026

**Type** Package

**biocViews** Technology, StatisticalMethod, Sequencing, RNASeq, Transcriptomics, Classification, FeatureExtraction

**Title** consensus Independent Component Analysis

**Version** 2.10.0

**Description** consICA implements a data-driven deconvolution method – consensus independent component analysis (ICA) to decompose heterogeneous omics data and extract features suitable for patient diagnostics and prognostics. The method separates biologically relevant transcriptional signals from technical effects and provides information about the cellular composition and biological processes. The implementation of parallel computing in the package ensures efficient analysis of modern multicore systems.

**BugReports** https://github.com/biomod-lih/consICA/issues

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** false

**Imports** fastICA (>= 1.2.1), sm, org.Hs.eg.db, GO.db, stats, SummarizedExperiment, BiocParallel, graph, ggplot2, methods, Rfast, pheatmap, survival, topGO, graphics, grDevices

**Depends** R (>= 4.2.0)

**Suggests** knitr, BiocStyle, rmarkdown, testthat, Seurat

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**Config/pak/sysreqs** make libpng-dev libssl-dev zlib1g-dev

**Repository** https://bioc-release.r-universe.dev

**Date/Publication** 2026-04-28 12:58:59 UTC

**RemoteUrl** https://github.com/bioc/consICA

**RemoteRef** RELEASE_3_23

**RemoteSha** c4a1555952a3cb9f0dd39b38695db2dfafad0b81

# Contents



---

`anovaIC` *ANOVA test for independent component across factors*

---

## Description

ANOVA (ANalysis Of VAriance) test produced for specific independent component across each (clinical) factor as ‘aov(IC ~ factor)‘. Plot distributions of samples’ weight for top 9 significant factors.

## Usage

```
anovaIC(
  cica,
  Var = NULL,
  icomp = 1,
  plot = TRUE,
  mode = "violin",
  color_by_pv = TRUE
)
```

## Arguments

| | |
|---|---|
| `cica` | list compliant to ‘consICA()‘ result |
| `Var` | matrix with samples’ metadata. Samples in rows and factors in columns |
| `icomp` | number of component to analyse |
| `plot` | if plot weights distributions for top factors |
| `mode` | type of plot. Can be ’violin’ or ’box’ |
| `color_by_pv` | if TRUE plots will be colored by p-value ranges |

## Value

a data.frame with

| | |
|---|---|
| factor | name of factor |
| p.value | p-value for ANOVA test for factor |
| p.value_disp | string for p-value printing |

## Examples

```
data("samples_data")
# Var <- data.frame(SummarizedExperiment::colData(samples_data))
# cica <-  consICA(samples_data, ncomp=10, ntry=1, ncores=1, show.every=0)
## Run ANOVA for 4th independent component
# anova <- anovaIC(cica, Var=Var, icomp = 4)
```

---

# consICA *Calculate consensus Independent Component Analysis*

---

## Description

calculate consensus independent component analysis (ICA) Implements efficient ICA calculations.

## Usage

```
consICA(
  X,
  ncomp = 10,
  ntry = 1,
  show.every = 1,
  filter.thr = NULL,
  ncores = 1,
  bpparam = NULL,
  reduced = FALSE,
  fun = "logcosh",
  alg.typ = "deflation",
  verbose = FALSE,
  assay_string = NULL
)
```

## Arguments

| | |
|---|---|
| X | input data with features in rows and samples in columns. Could be a 'SummarizedExperiment' object, matrix or 'Seurat' object. For 'SummarizedExperiment' with multiple assays or 'Seurat' pass the name with 'assay_string' parameter, otherwise the first will be taken. See SummarizedExperiment-class |
| ncomp | number of components |

| | |
|---|---|
| ntry | number of consensus runs. Default value is 1 |
| show.every | numeric logging period in iterations (disabled for 'ncore's > 1). Default value is 1 |
| filter.thr | Filter out genes (rows) with max value lower than this value from 'X' |
| ncores | number of cores for parallel calculation. Default value is 4 |
| bpparam | parameters from the 'BiocParallel' |
| reduced | If TRUE returns reduced result (no 'X', 'i.best', see 'return') |
| fun | the functional form of the G function used in the approximation to neg-entropy in fastICA. Default value is "logcosh" |
| alg.typ | parameter for fastICA(). If alg.typ == "deflation" the components are extracted one at a time. If alg.typ == "parallel" the components are extracted simultaneously. Default value is "deflation" |
| verbose | logic TRUE or FALSE. Use TRUE for print process steps. Default value is FALSE |
| assay_string | name of assay for 'SummarizedExperiment' or 'Seurat' input object 'X'. Default value is NULL |

## Value

a list with

| | |
|---|---|
| X | input object |
| nsamples, nfeatures | dimension of X |
| S, M | consensus metagene and weight matrix |
| ncomp | number of components |
| X_num | input data in matrix format |
| mr2 | mean R2 between rows of M |
| stab | stability, mean R2 between consistent columns of S in multiple tries. Applicable only for 'ntry' > 1 |
| i.best | number of best iteration |

## Author(s)

Petr V. Nazarov

## See Also

fastICA

## Examples

```
data("samples_data")
# Deconvolve into independent components
cica <- consICA(samples_data, ncomp=15, ntry=10, ncores=1, show.every=0)
# X = S * M, where S - independent signals matrix, M - weights matrix
dim(samples_data)
dim(cica$S)
dim(cica$M)
```

---

# enrichGO *Enrichment analysis of GO-terms based on Ensembl IDs*

---

## Description

Enrichment analysis of GO-terms for independent components with Ensembl IDs based on topGO package

## Usage

```
enrichGO(
  genes,
  fdr = NULL,
  fc = NULL,
  ntop = NA,
  thr.fdr = 0.05,
  thr.fc = NA,
  db = "BP",
  genome = "org.Hs.eg.db",
  id = c("entrez", "alias", "ensembl", "symbol", "genename"),
  algorithm = "weight",
  do.sort = TRUE,
  randomFraction = 0,
  return.genes = FALSE
)
```

## Arguments

| | |
|---|---|
| genes | character vector with list of ENSEBML IDs |
| fdr | numeric vector of FDR for each gene |
| fc | numeric vector of logFC for each gene |
| ntop | number of first taken genes |
| thr.fdr | significance threshold for FDR |
| thr.fc | significance threshold for absolute logFC |
| db | name of GO database: "BP","MF","CC" |
| genome | R-package for genome annotation used. For human - 'org.Hs.eg.db' |

| | |
|---|---|
| id | id |
| algorithm | algorithm for ‘runTest()‘ |
| do.sort | if TRUE - resulted functions sorted by p-value |
| randomFraction | for testing only, the fraction of the genes to be randomized |
| return.genes | If TRUE include genes in output. Default value is FALSE |

## Value

list with terms and stats

## Author(s)

Petr V. Nazarov

---

`estimateVarianceExplained` *Estimate the variance explained by the model*

---

## Description

The method estimates the variance explained by the model and by each independent component. We used the coefficient of determination (R2) between the normalized input (X-mean(X)) and (S*M)

## Usage

```
estimateVarianceExplained(cica, X = NULL)
```

## Arguments

| | |
|---|---|
| cica | list compliant to ‘consICA()‘ result |
| X | a ‘SummarizedExperiment‘ object. Assay used for the model. Will be used if consICA$X is NULL, ignore otherwise. |

## Value

a list of:

| | |
|---|---|
| R2 | total variance explained by the model |
| R2_ics | Amount of variance explained by the each independent component |

## Examples

```
data("samples_data")
# cica <- consICA(samples_data, ncomp=15, ntry=10, show.every=0)
# var_ic <- estimateVarianceExplained(cica)
```

getFeatures *Get features from consICA deconvolution result*

## Description

Extract names of features (rows in ‘X‘ and ‘S‘ matrices) and their false discovery rates values

## Usage

```
getFeatures(cica, alpha = 0.05, sort = FALSE)
```

## Arguments

| | |
|---|---|
| cica | list compliant to ‘consICA()‘ result |
| alpha | value in [0,1] interval. Used to filter features with FDR < ‘alpha‘. Default value is 0.05 |
| sort | sort features decreasing FDR. Default is FALSE |

## Value

list of dataframes ‘pos‘ for positive and ‘neg‘ for negative affecting features with columns:

| | |
|---|---|
| features | names of features |
| fdr | false discovery rate value |

## Author(s)

Petr V. Nazarov

## Examples

```
data("samples_data")
# Get deconvolution of X matrix
cica <-  consICA(samples_data, ncomp=10, ntry=1, show.every=0)
# Get features names and FDR for each component
features <- getFeatures(cica)
# Positive affecting features for first components are
ic1_pos <- features$ic.1$pos
```

getGO *Assigns IC signatures to Gene Ontologies*

## Description

Assigns extracted independent components to Gene Ontologies and rotate independent components (‘S‘ matrix) to set most significant Gene Ontologies as positive affecting features. Set ‘ncores‘ param for paralleled calculations.

## Usage

```
getGO(
  cica,
  alpha = 0.05,
  genenames = NULL,
  genome = "org.Hs.eg.db",
  db = c("BP", "CC", "MF"),
  ncores = 4,
  rotate = TRUE
)
```

## Arguments

| | |
|---|---|
| cica | list compliant to ‘consICA()‘ result |
| alpha | value in [0,1] interval. Used to filter features with FDR < ‘alpha‘. Default value is 0.05 |
| genenames | alternative names of genes. If NULL we use rownames of ‘S‘ matrix. We automatically identify type of gene identifier, you can use Ensembl, Symbol, Entrez, Alias, Genename IDs. |
| genome | R-package for genome annotation used. For human - ’org.Hs.eg.db’ |
| db | name of GO database: "BP","MF","CC" |
| ncores | number of cores for parallel calculation. Default value is 4 |
| rotate | rotate components in ‘S‘ and ‘M‘ matricies in ‘cica‘ object to set most significant Gene Ontologies as positive effective features. Default is TRUE |

## Value

rotated (if need) ‘cica‘ object with added ‘GO‘ - list for each db chosen (BP, CC, MM), with dataframes ‘pos‘ for positive and ‘neg‘ for negative affecting features for each component:

| | |
|---|---|
| GO.ID | id of Gene Ontology term |
| Term | name of term |
| Annotated | number of annotated genes |
| Significant | number of significant genes |

| | |
|---|---|
| Expected | estimate of the number of annotated genes if the significant genes would be randomly selected from the gene universe classisFisher F-test |
| FDR | false discovery rate value |
| Score | genes score |

## Author(s)

Petr V. Nazarov

## Examples

```
data("samples_data")
# Calculate ICA (run with ntry=1 for quick test, use more in real analysis)
#cica <- consICA(samples_data, ncomp=2, ntry=1, ncores=1, show.every=0)
# cica <- consICA(samples_data, ncomp=40, ntry=20, show.every=0)

# Annotate independent components with gene ontoligies
# cica <- getGO(cica, db = "BP", ncores=4)
## Positively affected GOs for 2nd independent component
#head(cica$GO$GOBP$ic02$pos)
```

---

# is.consICA *Is the object is consensus ICA compliant?*

---

## Description

Check if the object is a list in the same format as the result of 'consICA()'

## Usage

```
is.consICA(cica)
```

## Arguments

| | |
|---|---|
| cica | list |

## Value

TRUE or FALSE

## Examples

```
# returns TRUE
is.consICA(list("ncomp" = 2, "nsples" = 2, "nfeatures" = 2,
                "S" = matrix(0,2,2),"M" = matrix(0,2,2)))
```

oneICA *Runs* fastICA

## Description

Runs fastICA once and store in a consICA manner

## Usage

```
oneICA(
  X,
  ncomp = 10,
  filter.thr = NULL,
  reduced = FALSE,
  fun = "logcosh",
  alg.typ = "deflation",
  assay_string = NULL
)
```

## Arguments

| | |
|---|---|
| X | input data with features in rows and samples in columns. Could be a 'SummarizedExperiment' object, matrix or 'Seurat' object. For 'SummarizedExperiment' with multiple assays or 'Seurat' pass the name with 'assay_string' parameter, otherwise the first will be taken. See SummarizedExperiment-class |
| ncomp | number of components. Default value is 10 |
| filter.thr | filter rows in input matrix with max value > 'filter.thr'. Default value is NULL |
| reduced | If TRUE returns reduced result (no X, see 'return') |
| fun | the functional form of the G function used in the approximation to neg-entropy in fastICA. Default value is "logcosh" |
| alg.typ | parameter for fastICA(). if alg.typ == "deflation" the components are extracted one at a time. if alg.typ == "parallel" the components are extracted simultaneously. Default value is "deflation" |
| assay_string | name of assay for 'SummarizedExperiment' or 'Seurat' input object 'X'. Default value is NULL |

## Value

a list with

| | |
|---|---|
| X | input 'SummarizedExperiment' object |
| nsamples, nfeatures | dimension of X assay |
| S, M | consensus metagene and weight matrix |
| ncomp | number of components |

### Author(s)

Petr V. Nazarov

### See Also

fastICA

### Examples

```
data("samples_data")
res <- oneICA(samples_data)
```

---

## overlapGO *Similarity of two gene ontologies lists*

---

### Description

Calculate similarity matrix of gene ontilogies (GOs) of independent components. The measure could be cosine similarity or Jaccard index (see details)

### Usage

```
overlapGO(GO1, GO2, method = c("cosine", "jaccard"), fdr = 0.01)
```

### Arguments

| | |
|---|---|
| GO1 | list of GOs for each independent component got from 'getGO()' |
| GO2 | list of GOs for each independent component got from 'getGO()' |
| method | can be 'cosine' for non-parametric cosine similarity or 'jaccard' for Jaccadr index. See details |
| fdr | FDR threshold for GOs that would be used in measures. Default value is 0.01 |

### Details

Jaccard index is a measure of the similarity between two sets of data. It calculated as intersection divided by union

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

Results are from 0 to 1.

Cosine similarity here is calculated in a non-parametric way: for two vectors of gene ontologies, the space is created as a union of GOs in both vectors. Then, two rank vectors in this space created, most enriched GOs get the biggest rank and GOs from space not included in the GO vector get 0. Cosine similarity is calculated between two scaled rank vectors. Such approach allows to take the order of enriched GO into account. Results are from -1 to 1. Zero means no similarity.

## Value

a similarity matrix of cosine or Jaccard values, rows corresponds to independent components in 'GO1', columns to independent components in 'GO2'.

## Author(s)

Maryna Chepeleva

## Examples

```
## Not run:
data("samples_data")
# Calculate ICA (run with ntry=1 for quick test, use more in real analysis)
cica1 <- consICA(samples_data, ncomp=5, ntry=1, show.every=0)
# Search enriched gene ontologies
cica1 <- getGO(cica1, db = "BP", ncores = 1)
# Calculate ICA and GOs for another dataset
cica2 <- consICA(samples_data[,1:100], ncomp=4, ntry=1, show.every=0)
cica2 <- getGO(cica2, db = "BP", ncores = 1)
# Compare two lists of enriched GOs
# Jaccard index
jc <- overlapGO(GO1 = cica1$GO$GOBP, GO2 = cica2$GO$GOBP,
method = "jaccard", fdr = 0.01)
# Cosine similarity
cos_sim <- overlapGO(GO1 = cica1$GO$GOBP, GO2 = cica2$GO$GOBP,
method = "cosine", fdr = 0.01)

## End(Not run)
```

---

# plotICVarianceExplained

*Barplot variance explained by each IC*

---

## Description

Method to plot variance explained (R-squared) by the MOFA model for each view and latent factor. As a measure of variance explained for gaussian data we adopt the coefficient of determination (R2).
For details on the computation see the help of the `estimateVarianceExplained` function

## Usage

```
plotICVarianceExplained(
  cica,
  sort = NULL,
  las = 2,
  title = "Variance explained per IC",
  x.cex = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| cica | consICA compliant list |
| sort | specify the arrangement as 'asc'/'desc'. No sorting if NULL |
| las | orientation value for the axis labels (0 - always parallel to the axis, 1 - always horizontal, 2 - always perpendicular to the axis, 3 - always vertical) |
| title | character string with title of the plot |
| x.cex | specify the size of the tick label numbers/text with a numeric value of length 1 |
| ... | extra arguments to be passed to barplot |

### Value

A numeric vector compliant to 'barplot' output

### Examples

```
data("samples_data")
# cica <- consICA(samples_data, ncomp=15, ntry=10, show.every=0)
# p <- plotICVarianceExplained(cica, sort = "asc")
```

---

## samples_data *Samples of gene expression*

---

### Description

A dataset containing the expression of 2454 genes for 472 samples from skin cutaneous melanoma (SKCM) TCGA cohort, their metadata such as age, gender, cancer type etc. and survival time-to-event data

### Usage

```
data(samples_data)
```

### Format

A SummarizedExperiment object:

**assay** expression matrix with genes by rows and samples by columns

**colData** data frame with sample metadata (clinical variables)

saveReport *Save PDF report with analysis of each independent component*

## Description

Save PDF report with description of each independent component (IC) consists of most affected genes, significant Go terms, survival model for the component, ANOVA analysis for samples characteristics and stability

## Usage

```
saveReport(
  cica,
  Genes = NULL,
  Var = NULL,
  surv = NULL,
  genenames = NULL,
  file = sprintf("report_ICA_%d.pdf", ncol(IC$S)),
  main = "Component # %d (stability = %.3f)",
  show.components = seq.int(1, ncol(cica$S))
)
```

## Arguments

| | |
|---|---|
| cica | list compliant to ‘consICA()‘ result. May include GO list with enrichment analysis appended with ‘getGO()‘ function |
| Genes | features list compilant to ‘getFeatures‘ output (list of dataframes ‘pos‘ for positive and ‘neg‘ for negative affecting features with names of features false discovery rates columns).If NULL will generated automatically |
| Var | matrix with samples metadata |
| surv | dataframe with time and event values for each sample |
| genenames | alternative gene names for printing in the report |
| file | report filename, ends with ".pdf" |
| main | title for each list discribes the component |
| show.components | which compont will be shown |

## Value

TRUE when successfully generate report

## Author(s)

Petr V. Nazarov

## Examples

```
if(FALSE){
  data("samples_data")
  cica <- consICA(samples_data, ncomp=15, ntry=10, ncores = 2, show.every=0)
  if(FALSE){
    cica <- getGO(cica, db = "BP")
  }
  saveReport(cica, Var=samples_data$Var, surv = samples_data$Sur)
}
```

---

`setOrientation` *Set orientation for independent components*

---

## Description

Set orientation for independent components as positive in most enriched direction. Use first element of 'GOs' for direction establishment.

## Usage

```
setOrientation(cica, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `cica` | list compliant to 'consICA()' result. Must contain GO, see 'getGO()' |
| `verbose` | logic TRUE or FALSE. Use TRUE for print process steps. Default is FALSE |

## Value

cica object after rotation, with rotated 'S', 'M' and added 'compsign' which is vector defined rotation: 'S_rot = S * compsign, M_rot = M * compsign, GO_rot = GO * compsign'

## Note

Implemented inside 'getGO()' in version >= 1.1.1.

## Author(s)

Petr V. Nazarov

## Examples

```
## Not run:
data("samples_data")
# Get deconvolution of X matrix
#cica <-  consICA(samples_data, ncomp=10, ntry=1, show.every=0)
cica <- consICA(samples_data, ncomp=2, ntry=1, show.every=0) # timesaving
example
```

```
GOs <- getGO(cica, db = "BP")
# Get already rotated S matrix and Gene Ontologies
cica <- getGO(cica, db = "BP")

# Get Gene Ontologies without rotation (actually, you don't need to do this)
# This may used for GO generated with version < 1.1.1. Add GO to cica list.
cica <- getGO(cica, db = "BP", rotate = FALSE)
# Rotate components
cica <- setOrientation(cica, verbose = T)
# Which components was rotated
which(cica$compsign == -1)

## End(Not run)
```

---

`sortDataFrame` *Sort dataframe*

---

## Description

Sort dataframe, adapted from http://snippets.dzone.com/user/r-fanatic

## Usage

```
sortDataFrame(x,key, ...)
```

## Arguments

| | |
|---|---|
| `x` | a data.frame |
| `key` | sort by this column |
| `...` | other parameters for ‘order‘ function (e. g. ‘decreasing‘) |

## Value

sorted dataframe

## Examples

```
df <- data.frame("features" = c("f1", "f2", "f3"), fdr = c(0.02, 0.002, 1))
sortDataFrame(df, "fdr")
```

`survivalAnalysis` *Survival analysis based on significant IC*

## Description

Cox regression (based on R package 'survival') on the weights of independent components with significant contribution in individual risk model. For more see Nazarov et al. 2019 In addition the function plot Kaplan-Meier diagram.

## Usage

```
survivalAnalysis(cica, surv = NULL, time = NULL, event = NULL, fdr = 0.05)
```

## Arguments

| | |
|---|---|
| `cica` | list compliant to 'consICA()' result |
| `surv` | dataframe with time and event values for each sample. Use this parameter or 'time' and 'event' |
| `time` | survival time value for each sample |
| `event` | survival event factor for each sample (TRUE or FALSE) |
| `fdr` | false discovery rate threshold for significant components involved in final model. Default value is 0.05 |

## Value

a list with

| | |
|---|---|
| `cox.model` | an object of class 'coxph' representing the fit. See 'coxph.object' for details |
| `hazard.score` | hazard score for significant components (fdr < 'fdr' in individual cox model) |

## Examples

```
data("samples_data")
# Get deconvolution of X matrix
cica <-  consICA(samples_data, ncomp=10, ntry=1, show.every=0)
surv <- survivalAnalysis(cica,
  surv = SummarizedExperiment::colData(samples_data)[,c("time", "event")])
```

# Index