

# Package ‘consICA’

January 3, 2025

**Type** Package

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

**Title** consensus Independent Component Analysis

**Version** 2.5.0

**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** 7.2.3

**git\_url** <https://git.bioconductor.org/packages/consICA>

**git\_branch** devel

**git\_last\_commit** 7b9bffe

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2025-01-03

**Author** Petr V. Nazarov [aut, cre] (ORCID: <https://orcid.org/0000-0003-3443-0298>),  
 Tony Kaoma [aut] (ORCID: <https://orcid.org/0000-0002-1269-4826>),  
 Maryna Chepeleva [aut] (ORCID: <https://orcid.org/0000-0003-3036-4916>)

**Maintainer** Petr V. Nazarov <petr.nazarov@lih.lu>

## Contents

anovaIC . . . . .	2
consICA . . . . .	3
coreICA . . . . .	5
enrichGO . . . . .	7
estimateVarianceExplained . . . . .	8
getFeatures . . . . .	9
getGO . . . . .	10
get_score . . . . .	11
get_X_num . . . . .	12
is.consICA . . . . .	12
oneICA . . . . .	13
outICA . . . . .	14
overlapGO . . . . .	15
plotICVarianceExplained . . . . .	16
samples_data . . . . .	17
saveReport . . . . .	17
setOrientation . . . . .	19
set_bpparam . . . . .	20
sortDataFrame . . . . .	20
sortFeatures . . . . .	21
survivalAnalysis . . . . .	21
<b>Index</b>	<b>23</b>

---

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,  
  icode = 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
icode	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, icode = 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 <a href="#">SummarizedExperiment-class</a>
ncomp	number of components
ntry	number of consensus runs. Default value is 1
show.every	numeric logging period in iterations (disabled for ‘ncores’ > 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)
```

---

coreICA

*Fast Independent Component Analysis for multi-run mode*

---

**Description**

Adaptation of [fastICA](#) for quick multiple-run calculations for consensus Independent Component Analysis (ICA)

**Usage**

```
coreICA(
  X,
  n.comp,
  preICA = NULL,
  alg.typ = c("parallel", "deflation"),
  fun = c("logcosh", "exp"),
  w.init = NULL,
  alpha = 1,
```

```

    row.norm = FALSE,
    maxit = 200,
    tol = 1e-04,
    verbose = FALSE
  )

```

### Arguments

X	matrix with features in rows and samples in columns
n.comp	number of components.
preICA	output of 'outICA()'. Default is NULL
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"
fun	the functional form of the G function used in the approximation to neg-entropy in fastICA. Default value is "logcosh"
w.init	initial weights
alpha	default is 1
row.norm	set TRUE if the normalization by rows is needed. Default is FALSE
maxit	default is 200
tol	default is 1e-04
verbose	logic TRUE or FALSE. Use TRUE for print process steps. Default value is FALSE

### Value

a list with (compliant to 'fastICA()' output)

X	pre-processed data matrix
K	pre-whitening matrix that projects data onto the first 'n.comp' principal components
W	estimated un-mixing matrix
A	estimated mixing matrix
S	estimated source matrix

### Author(s)

Maryna Chepeleva

---

`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**

<code>genes</code>	character vector with list of ENSEMBL IDs
<code>fdr</code>	numeric vector of FDR for each gene
<code>fc</code>	numeric vector of logFC for each gene
<code>ntop</code>	number of first taken genes
<code>thr.fdr</code>	significance threshold for FDR
<code>thr.fc</code>	significance threshold for absolute logFC
<code>db</code>	name of GO database: "BP","MF","CC"
<code>genome</code>	R-package for genome annotation used. For human - 'org.Hs.eg.db'
<code>id</code>	id
<code>algorithm</code>	algorithm for 'runTest()'
<code>do.sort</code>	if TRUE - resulted functions sorted by p-value
<code>randomFraction</code>	for testing only, the fraction of the genes to be randomized
<code>return.genes</code>	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 ( $R^2$ ) between the normalized input ( $X - \text{mean}(X)$ ) 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	<i>Get features from consICA deconvolution result</i>
-------------	---

---

**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 $FDR < \text{'alpha'}$ . 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' matrices 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
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=4, ntry=1, ncores=1, show.every=0)
# cica <- consICA(samples_data, ncomp=40, ntry=20, show.every=0)

# Annotate independent components with gene ontologies
cica <- getGO(cica, db = "BP", ncores=1)
# Positively affected GOs for 2nd independent component
head(cica$GO$GOBP$ic02$pos)

```

get\_score

*Create score depending on threshold and paradigm***Description**

Create score depending on threshold and paradigm

**Usage**

```
get_score(genes, fc, thr.fc, fdr, thr.fdr, ntop)
```

**Arguments**

genes	character vector with list of ENSEMBL IDs
fc	numeric vector of logFC for each gene
thr.fc	significance threshold for absolute logFC
fdr	numeric vector of FDR for each gene
thr.fdr	significance threshold for FDR
ntop	number of first taken genes

**Value**

numeric score vector

---

get_X_num	<i>Convert input object as numeric matrix</i>
-----------	---

---

**Description**

Convert input object as numeric matrix

**Usage**

```
get_X_num(obj, assay_string = NULL)
```

**Arguments**

obj	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 <a href="#">SummarizedExperiment-class</a>
assay_string	name of assay for ‘SummarizedExperiment’ or ‘Seurat’ input object ‘obj’. Default value is NULL

**Value**

matrix

---

is.consICA	<i>Is the object is consensus ICA compliant?</i>
------------	--

---

**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**

<code>X</code>	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 <a href="#">SummarizedExperiment-class</a>
<code>ncomp</code>	number of components. Default value is 10
<code>filter.thr</code>	filter rows in input matrix with max value > ‘filter.thr’. Default value is NULL
<code>reduced</code>	If TRUE returns reduced result (no X, see ‘return’)
<code>fun</code>	the functional form of the G function used in the approximation to neg-entropy in fastICA. Default value is "logcosh"
<code>alg.typ</code>	parameter for fastICA(). if <code>alg.typ == "deflation"</code> the components are extracted one at a time. if <code>alg.typ == "parallel"</code> the components are extracted simultaneously. Default value is "deflation"
<code>assay_string</code>	name of assay for ‘SummarizedExperiment’ or ‘Seurat’ input object ‘X’. Default value is NULL

**Value**

a list with

<code>X</code>	input ‘SummarizedExperiment’ object
<code>nsamples, nfeatures</code>	dimension of X assay
<code>S, M</code>	consensus metagene and weight matrix
<code>ncomp</code>	number of components

**Author(s)**

Petr V. Nazarov

**See Also**

[fastICA](#)

**Examples**

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

---

outICA

*Outside part of multiple run Independent Component Analysis*

---

**Description**

Calculate a common part for consensus Independent Component Analysis (ICA)

**Usage**

```
outICA(X, n.comp, row.norm = FALSE, verbose = FALSE)
```

**Arguments**

X	matrix with features in rows and samples in columns
n.comp	number of components
row.norm	rows normalization flag. Default value is FALSE
verbose	logic TRUE or FALSE. Use TRUE for print process steps. Default value is FALSE

**Value**

a list with

X	input matrix
X1	interim calculated matrix
K	pre-whitening matrix that projects data onto the first 'n.comp' principal components

**Author(s)**

Maryna Chepeleva

---

`overlapGO`*Similarity of two gene ontologies lists*

---

**Description**

Calculate similarity matrix of gene ontologies (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 Jaccard 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 (R<sup>2</sup>).

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	<i>Samples of gene expression</i>
--------------	-----------------------------------

---

### 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	<i>Save PDF report with analysis of each independent component</i>
------------	--

---

### 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 compliant 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 be 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 describes the component
show.components	which component will be shown

**Value**

TRUE when successfully generate report

**Author(s)**

Petr V. Nazarov

**Examples**

```
data("samples_data")
cica <- consICA(samples_data, ncomp=40, ntry=10, show.every=0)
if(FALSE){
  cica <- getGO(cica, db = "BP")
}
saveReport(cica, Var=samples_data$Var, surv = samples_data$Sur)
```

---

setOrientation	<i>Set orientation for independent components</i>
----------------	---

---

**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)
```

---

set_bpparam	<i>Set up for the parallel computing for biocParallel Adapt from 'FEAST' This function sets up the environment for parallel computing.</i>
-------------	--

---

### Description

Set up for the parallel computing for biocParallel Adapt from 'FEAST' This function sets up the environment for parallel computing.

### Usage

```
set_bpparam(ncores = 0, BPPARAM = NULL)
```

### Arguments

ncores	number of processors
BPPARAM	bpparameter from bpparam

### Value

BAPPARAM settings

---

sortDataFrame	<i>Sort dataframe</i>
---------------	-----------------------

---

### 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")
```

---

sortFeatures	<i>Sort Genes of consICA object</i>
--------------	-------------------------------------

---

**Description**

Sort Genes for independent components

**Usage**

```
sortFeatures(Genes)
```

**Arguments**

Genes                    list compilant to 'getFeatures' output

**Value**

sorted list

**Examples**

```
#features <- list("ic1" = list(
#                "pos" = data.frame("features" = c("f1", "f2", "f3"),
#                                    "fdr" = c(0.0043, 0.4, 0.04)),
#                "neg" = data.frame("features" = c("f1", "f2", "f3"),
#                                    "fdr" = c(0, 0.1, 0.9))))
#sortFeatures(features)
```

---

survivalAnalysis	<i>Survival analysis based on significant IC</i>
------------------	--

---

**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

- \* **datasets**
  - [samples\\_data](#), [17](#)
- \* **internal**
  - [coreICA](#), [5](#)
  - [get\\_score](#), [11](#)
  - [get\\_X\\_num](#), [12](#)
  - [outICA](#), [14](#)
  - [set\\_bpparam](#), [20](#)
  - [sortFeatures](#), [21](#)
- [anovaIC](#), [2](#)
- [barplot](#), [17](#)
- [consICA](#), [3](#)
- [coreICA](#), [5](#)
- [enrichGO](#), [7](#)
- [estimateVarianceExplained](#), [8](#), [16](#)
- [fastICA](#), [5](#), [13](#), [14](#)
- [get\\_score](#), [11](#)
- [get\\_X\\_num](#), [12](#)
- [getFeatures](#), [9](#)
- [getGO](#), [10](#)
- [is.consICA](#), [12](#)
- [oneICA](#), [13](#)
- [outICA](#), [14](#)
- [overlapGO](#), [15](#)
- [plotICVarianceExplained](#), [16](#)
- [samples\\_data](#), [17](#)
- [saveReport](#), [17](#)
- [set\\_bpparam](#), [20](#)
- [setOrientation](#), [19](#)
- [sortDataFrame](#), [20](#)
- [sortFeatures](#), [21](#)
- [survivalAnalysis](#), [21](#)