Package: SurprisalAnalysis (via r-universe)

November 5, 2026

Title Information Theoretic Analysis of Gene Expression Data

Version 0.2

Description Implements Surprisal Analysis for gene expression data such as RNA-seq or microarray experiments. Surprisal Analysis is an information-theoretic method that decomposes gene expression data into a baseline state and constraint-associated deviations, which helps to capture coordinated gene expression patterns under different biological conditions. References where Surprisal analysis has been used for analyzing gene expression data (using the same methodology provided within this R package) are Kravchenko-Balasha et al (2014) <doi:10.1371/journal.pone.0108549>, Zadran et al. (2014) <doi:10.1073/pnas.1414714111>, Su et al. (2019) <doi:10.1371/journal.pcbi.1007034>, Bogaert et al. (2018) <doi:10.1371/journal.pone.0195142>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, pheatmap, peakRAM, data.table

VignetteBuilder knitr

Imports tidyr, dplyr, matlib, org.Mm.eg.db, org.Hs.eg.db, clusterProfiler, AnnotationDbi, tidyverse, shiny, ggplot2, latex2exp, shinythemes, shinyWidgets, shinyjs, shinycssloaders, patchwork, DT, httpuv

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libglpk-dev libglu1-mesa-dev make libharfbuzz-dev texlive libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libgl1-mesa-dev libssl-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-07-08 17:33:31 UTC

RemoteUrl https://github.com/annicenajafi/surprisalanalysis

RemoteRef HEAD

RemoteSha df5cf9e4a8f8d56cd77dc2ecb3fd98f10ccc631c

Contents

GO_analysis_surprisal_analysis 2
runSurprisalApp 3
surprisal_analysis 4

Index **6**

`GO_analysis_surprisal_analysis` *Perform Gene ontology analysis on a pattern of interest*

Description

Perform Gene ontology analysis on a pattern of interest

Usage

```
GO_analysis_surprisal_analysis(
  transcript_weights,
  percentile_GO,
  lambda_no,
  key_type = "SYMBOL",
  flip = FALSE,
  species.db.str = "org.Hs.eg.db",
  ont = "BP",
  pAdjustMethod = "BH",
  top_GO_terms = 15
)
```

Arguments

`transcript_weights`	a dataframe containing the weight of transcripts in each pattern
`percentile_GO`	the percentile of transcript to be used for GO analysis, for example 95 will run GO on transcripts in the 95th percentile and above
`lambda_no`	the lambda pattern the user is interested in analyzing
`key_type`	type of transcripts which can be either SYMBOL, ENTREZID, ENSEMBL, or PROBEID
`flip`	a boolean variable which can either be true or false, if it is set to true, the lambda values will be multiplied by -1
`species.db.str`	the type of species used for GO analysis, by default set to Homo sapiens, can be either 'org.Hs.eg.db' or 'org.Mm.eg.db'
`ont`	the ontology term for GO enrichment analysis. Can be either "BP", "MF" or "CC". They stand for "Biological Process", "Molecular Function" or "Cellular Component". Set to "BP" by default

pAdjustMethod	multiple testing correction method. Could be one of "BH", "bonferroni", "holm", "hochberg", "hommel", "BY", or "none". The default setting is "BH"
top_GO_terms	number of GO terms returns, by default set to 15

Value

dataframe, the important GO terms related to a lambda gene pattern

Examples

```
csv.path <- system.file(
  "extdata", "helper_T_cell_0_test.csv",
  package = "SurprisalAnalysis"
)

expr.df <- utils::read.csv(csv.path, check.names = FALSE)
expr.df[1:700,]->expr.df
sa.res <- surprisal_analysis(expr.df, zero.handling = "log1p")
alph.all <- sa.res[[2]]

go_top <- GO_analysis_surprisal_analysis(
    transcript_weights = alph.all,
    percentile_GO      = 99,
    lambda_no          = "lambda_1",
    key_type           = "SYMBOL",
    flip               = FALSE,
    species.db.str     = "org.Hs.eg.db",
    ont                = "BP",
    pAdjustMethod      = "BH",
    top_GO_terms       = 15
    )
```

---

runSurprisalApp *Launch the SurprisalAnalysis Shiny App*

Description

Launch the SurprisalAnalysis Shiny App

Usage

```
runSurprisalApp(
  port = getOption("shiny.port", 3838),
  host = getOption("shiny.host", "127.0.0.1"),
  launch.browser = getOption("shiny.launch.browser", TRUE),
  run = TRUE,
  ...
)
```

Arguments

port	port to run the app on (passed to shiny::runApp)
host	host to listen on
launch.browser	should launch a browser? set to TRUE by default
run	boolean value, is set to TRUE by default. If set to FALSE it will not launch the graphical user interface
...	Further arguments passed along to shiny::runApp

Value

no return value, running the function will launch an application with graphical user interface

Examples

```
runSurprisalApp(port = httpuv::randomPort(), run = FALSE)
```

---

surprisal_analysis *This function performs surprisal analysis on transcriptomics data*

Description

This function performs surprisal analysis on transcriptomics data

Usage

```
surprisal_analysis(input.data, zero.handling = "pseudocount")
```

Arguments

input.data	transcriptomics data stores as dataframe
zero.handling	zero handling method. Can be either 'pseudocount' or 'log1p'. By default it is set to 'pseudocount'

Value

a list containing two matrix array objects, first one holding the lambda values representing the constraints or Lagrange multipliers and the second one holding the corresponding weights of transcripts stored (G matrix)

Examples

```
expr.df <- data.frame(gene_id = paste0("Gene", 1:6),
S1 = c(0, 12, 3, 0, 50, 7),
S2 = c(5, 0, 2, 9, 0, 4),
S3 = c(8, 15, 0, 1, 25, 0),
S4 = c(0, 7, 6, 0, 40, 3),
check.names = FALSE)
surprisal_analysis(expr.df, zero.handling = "pseudocount")
```

Index

GO_analysis_surprisal_analysis, 2

runSurprisalApp, 3

surprisal_analysis, 4