Package 'SimSeq'

November 23, 2015

Type Package

Title Nonparametric Simulation of RNA-Seq Data

Version 1.4.0

Date 2015-11-22

Author Samuel Benidt

Maintainer Samuel Benidt <sgbenidt@gmail.com>

Depends R (>= 2.10.0)

Imports fdrtool

Description RNA sequencing analysis methods are often derived by relying on hypothetical parametric models for read counts that are not likely to be precisely satisfied in practice. Methods are often tested by analyzing data that have been simulated according to the assumed model. This testing strategy can result in an overly optimistic view of the performance of an RNA-seq analysis method. We develop a data-based simulation algorithm for RNA-seq data. The vector of read counts simulated for a given experimental unit has a joint distribution that closely matches the distribution of a source RNA-seq dataset provided by the user. Users control the proportion of genes simulated to be differentially expressed (DE) and can provide a vector of weights to control the distribution of effect sizes. The algorithm requires a matrix of RNA-seq read counts with large sample sizes in at least two treatment groups. Many datasets are available that fit this standard.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-11-23 12:33:58

R topics documented:

`SimSeq-package` *Nonparametric Simulation of RNA-Seq Data*

Description

RNA sequencing analysis methods are often derived by relying on hypothetical parametric models for read counts that are not likely to be precisely satisfied in practice. Methods are often tested by analyzing data that have been simulated according to the assumed model. This testing strategy can result in an overly optimistic view of the performance of an RNA-seq analysis method. We develop a data-based simulation algorithm for RNA-seq data. The vector of read counts simulated for a given experimental unit has a joint distribution that closely matches the distribution of a source RNA-seq dataset provided by the user. Users control the proportion of genes simulated to be differentially expressed (DE) and can provide a vector of weights to control the distribution of effect sizes. The algorithm requires a matrix of RNA-seq read counts with large sample sizes in at least two treatment groups. Many datasets are available that fit this standard.

Details

Package:	SimSeq
Type:	Package
Version:	1.4.0
Date:	2015-11-22
License:	GPL-2

SimSeq performs data based simulation of RNA-Seq data creating a dataset with a known list of DE and EE genes. The core function that implements of the methodology of SimSeq is the SimData function.

Author(s)

Samuel Benidt

Maintainer: Samuel Benidt <sgbenidt@gmail.com>

`CalcPvalWilcox` *Calculate P-values of Differential Expression*

Description

A function called within SimData. Calculates the P-value of differential expression for each gene under either Wilcoxin signed rank test (paired data) or rank sum test (unpaired data). The vector of differences used in the test is based off of the log of the counts for each gene plus 1 divided by their respective multiplicative normalization factors provided by the user.

Usage

```
CalcPvalWilcox(counts, treatment, replic = NULL, sort.method,
               sorted = FALSE, norm.factors, exact = FALSE)
```

Arguments

`counts` A matrix of counts where each row specifies a gene and each column specifies a replicate.

`treatment` A vector specifying the treatment group for each column of the counts matrix. Only two treatment groups of either paired or unpaired data are allowed.

`replic` A vector specifying the replicate for each column of the counts matrix when there is paired data; optional if data is unpaired.

`sort.method` Character vector specifying one of "unpaired" or "paired", depending on the structure of the data.

`sorted` logical vector specifying whether data has already gone through SortData function. Defaults to FALSE.

`norm.factors` A positive numeric vector of multiplicative normalization factors for each column of the counts matrix.

`exact` Specifies whether an exact signed rank test (paired) or exact ranksum test (unpaired) should be used.

Value

probs: a vector of p-values of differential expression for each gene.

Author(s)

Samuel Benidt <sgbenidt@gmail.com>

`kidney` *Kidney Renal Clear Cell Carcinoma [KIRC] RNA-Seq data*

Description

The KIRC RNA-seq dataset from The Cancer Genome Atlas containing 20531 genes and 72 paired columns of data with rows corresponding to genes and columns corresponding to replicates; replic vector specifies replicates and treatment vector specifies non-tumor and tumor group samples respectively within replicate.

The maximum possible sample size that can be simulated with this dataset is 36 replicates in each of two treatment groups. However, it is recommended to use a sample size of 20 or lower for simulation studies to ensure each simulated dataset is sufficiently different from the other simulated datasets.

Disclaimer: The version of the KIRC dataset provided is: unc.edu_KIRC.IlluminaHiSeq_RNASeqV2.Level_3.1.5.0.

The Cancer Genome Atlas updates its datasets periodically. The latest version of the KIRC dataset can always be downloaded from: https://tcga-data.nci.nih.gov/tcga/.

Please appropriately reference the source below when using this dataset. The source code used to assemble this dataset is provided below.

Usage

```
data(kidney)
```

Format

List containing:

- counts: matrix of RNA-seq data for 20531 sampled genes and 72 paired columns from individuals with Kidney Renal Clear Cell Carcinoma.
- replic: vector detailing which column in counts matrix belongs to each individual.
- treatment: vector detailing whether each column in counts matrix is a non-tumor or tumor sample.

Source

https://tcga-data.nci.nih.gov/tcga/

The Cancer Genome Atlas Research Network (2013). Comprehensive molecular characterization of clear cell renal cell carcinoma. Nature, 499(7456), 43-49.

Examples

```
data(kidney)

## Not run:
### Source code used to assemble KIRC dataset

### load in SimSeq package for sorting counts matrix
library(SimSeq)

### htmlToText function used to scrape barcode data from uuid
htmlToText <- function(input, ...) {
  ###---PACKAGES ---###
  library(RCurl)
  library(XML)

  ###--- LOCAL FUNCTIONS ---###
  # Determine how to grab html for a single input element
  evaluate_input <- function(input) {
    # if input is a .html file
    if(file.exists(input)) {
      char.vec <- readLines(input, warn = FALSE)
      return(paste(char.vec, collapse = ""))
    }
    # if input is html text
    if(grepl("</html>", input, fixed = TRUE)) return(input)

    # if input is a URL, probably should use a regex here instead?
    if(!grepl(" ", input)) {
      # downolad SSL certificate in case of https problem
      if(!file.exists("cacert.perm")) {
       download.file(url = "http://curl.haxx.se/ca/cacert.pem", destfile = "cacert.perm")
      }
      return(getURL(input, followlocation = TRUE, cainfo = "cacert.perm"))
    }

    # return NULL if none of the conditions above apply
    return(NULL)
  }

  # convert HTML to plain text
  convert_html_to_text <- function(html) {
    doc <- htmlParse(html, asText = TRUE)
    text <- xpathSApply(doc, paste0("//text()",
                   "[not(ancestor::script)][not(ancestor::style)]",
                   "[not(ancestor::noscript)][not(ancestor::form)]"), xmlValue)
    return(text)
  }

  # format text vector into one character string
  collapse_text <- function(txt) {
    return(paste(txt, collapse = " "))
  }

  ###--- MAIN ---###
  # STEP 1: Evaluate input
  html.list <- lapply(input, evaluate_input)

  # STEP 2: Extract text from HTML
  text.list <- lapply(html.list, convert_html_to_text)

  # STEP 3: Return text
  text.vector <- sapply(text.list, collapse_text)
  return(text.vector)
}

### Specify path name for folder containing raw counts for each sample
mainDir <- getwd()
folder.path <- "unc.edu_KIRC.IlluminaHiSeq_RNASeqV2.Level_3.1.5.0"

### Determine list of files containing summarized raw counts
file.list <- dir(file.path(mainDir, folder.path))
keep <- grepl("genes.results", file.list)
file.list <- file.list[keep]

### Create summarized count matrix.
### Get n.row and n.col for summarized count matrix number of genes in first
### sample and number of total samples from file.list

file.temp <- file.path(mainDir, folder.path, file.list[1])
n.row <- nrow(read.table(file = file.temp, header = TRUE))
n.col <- length(file.list)

### initialize counts matrix
counts <- matrix(NA, nrow = n.row, ncol = n.col)

### get gene id's
gene.id <- read.table(file.temp, header = TRUE, stringsAsFactors = FALSE)$gene_id

### read in raw read counts from file.list
for(i in 1:n.col){
  file.temp <- file.path(mainDir, folder.path, file.list[i])
  counts[, i] <- read.table(file.temp, header = TRUE)$raw_count
}

### Data was summarized using RSEM software which produces non_integer
### counts for ambiguous reads. Counts are rounded as a preprocessing
### step.
counts <- round(counts)

### Cast counts matrix as integer type
counts <- matrix(as.integer(counts), nrow = nrow(counts), ncol = ncol(counts))

### Get uuid's for each sample
uuid <- substr(file.list, start = 9, stop = 44)

### Create urls from uuid list
urls <- paste(rep("https://tcga-data.nci.nih.gov/uuid/uuidws/mapping/xml/uuid/",
                  length(uuid)), uuid, sep = "")

### Scrape barcodes from urls
l <- length(urls)
barcodes <- vector("character", l)
for(i in 1:l){
  barcodes[i] <- htmlToText(urls[i])
}


barcodes <- substr(barcodes, start = 1, stop = 28)

### Get metadata on which samples were taken from each individual,
### tumor type of sample, etc. from barcodes for each sample
metadata <- data.frame(barcodes, stringsAsFactors = FALSE)

### Study Participant
metadata$participant <- substr(barcodes, start = 9, stop = 12)

### Sample type code. See:
### https://tcga-data.nci.nih.gov/datareports/codeTablesReport.htm?codeTable=Sample%20type
### for full list of codes and details on TCGA barcodes.
### 01: Primary Solid Tumor
### 02: Recurrent Solid Tumor
### 05: Additional New Primary
### 06: Metastatic Tumor
### 11: Solid Tissue Normal
metadata$type <- substr(barcodes, start = 14, stop = 15)

### Only keep Primary Solid Tumor and Solid Tissue Normal
keep.metadata <- metadata$type == ”01” | metadata$type == ”11”
metadata <- metadata[keep.metadata, ]
counts <- counts[, keep.metadata]

### Code from 01 to Tumor and 11 to Non-Tumor for easy identifiability
metadata$tumor <- ”Non-Tumor”
metadata$tumor[metadata$type == ”01”] <- ”Tumor”

### tag participant, type, and tumor as factors
metadata$participant <- as.factor(metadata$participant)
metadata$type <-as.factor(metadata$type)
metadata$tumor <- as.factor(metadata$tumor)

### Sort and subset down to paired data
sorting <-
  SortData(counts, treatment = metadata$tumor,
           replic = metadata$participant, sort.method = ”paired”)$sorting

counts <- counts[, sorting]
metadata <- metadata[sorting, ]
metadata$participant <- factor(metadata$participant)

### Add in attributes of counts matrix
dimnames(counts) <- list(gene.id, metadata$barcodes)
attr(counts, ”uuid”) <- uuid

kidney <- vector(”list”, 3)
kidney[[1]] <- counts
kidney[[2]] <- metadata$participant
kidney[[3]] <- metadata$tumor
names(kidney) <- c(”counts”, ”replic”, ”treatment”)

###Save file
save(kidney, file = ”kidney.rda”)

## End(Not run)
```

SimData *SimData*

Description

Given a matrix of RNA-seq data with large samples sizes in at least two treatment groups, SimData simulates a new matrix of RNA-seq data with a known list of differentially expressed (DE) and equivalently expressed (EE) genes from the original data.

Usage

```
SimData(counts, treatment, replic = NULL, sort.method, k.ind,
  n.genes = NULL, n.diff = NULL, norm.factors = NULL,
  samp.independent = FALSE, genes.select = NULL, genes.diff = NULL, switch.trt = FALSE,
  probs = NULL, weights = NULL, exact = FALSE, power = 1)
```

Arguments

`counts` A matrix of counts where each row specifies a gene and each column specifies a replicate.

`treatment` A vector specifying the treatment group for each column of the counts matrix. Only two treatment groups of either paired or unpaired data are allowed.

`replic` A vector specifying the replicate for each column of the counts matrix when there is paired data; optional if data is unpaired.

`sort.method` One of either "paired" or "unpaired", depending on the structure of the counts matrix.

`k.ind` The number of experimental units to be simulated for each treatment group.

`n.genes` The number of genes to be subsetted from the counts matrix in the simulation. Must be less than the total number of rows in the counts matrix and greater than n.diff. Optional if genes.select vector is specified.

`n.diff` The number of genes simulated to be differentially expressed. Must be less than n.genes. Optional if genes.diff vector is specified.

`norm.factors` A positive numeric vector of multiplicative normalization factors for each column of the counts matrix. Will default to CalcNormFactors function from the package edgeR with method = "TMM".

`samp.independent`
Should columns be resampled for each gene. (Defaults to FALSE - setting to TRUE is not recommended.)

`genes.select` A vector specifying genes to be subsetted from the counts matrix in the simulation. Can be either a logical vector or a numeric vector indicating the rows of the counts matrix to be used. Optional if n.genes is specified.

`genes.diff` A vector specifying genes to be differentially expressed in the simulation. Genes selected to be differentially expressed must be a subset of the genes selected in the genes.select vector. Can be either logical vector with length equal to genes selected or a numeric vector indicating the rows of the counts matrix to be used. Optional if n.diff is specified.

`switch.trt` Logical specifying which treatment group should be sampled from for EE genes. Default is first treatment group.

`probs` Optional vector specifying the p-value of differential expression for each gene to be used in the estimate of empirical Bayes probability for each gene. If not provided and weights are not specified, SimData will perform either a signed rank test (paired case) or a rank sum test (unpaired) case for each gene in the counts matrix.

`weights` Optional vector specifying weights to be used for sampling which genes are to be differentially expressed in the simulation. If null, weights will be calculated using the fdrtool function from the package 'fdrtool' to calculate one minus local fdr. If desired, the sampling of differentially expressed genes can be done without respect to any weights by providing a vector of ones.

`exact` Specifies whether an exact signed rank test (paired) or exact ranksum test (unpaired) should be used.

`power` Transorms the weights for each gene by raising each weights to some power. Must be greater than 0. Default is set to 1.

Details

SimData simulates an RNA-seq matrix of counts from a large source RNA-sequence dataset using a resampling based approach. If you use the Kidney Cancer (KIRC) dataset from the The Cancer Genome Atlas as the source RNA-seq counts matrix in the SimData function, please cite The Cancer Genome Atlas Research Network (2013) in any publications.

Value

List containing:

`counts` matrix of simulated RNA-seq data with known list of DE and EE genes.

`treament` a numeric vector specifying the treatment group structure in the new simulated counts matrix.

`genes.subset` all genes included in the simulated matrix. The i-th entry corresponds to the i-th row of the simulated counts matrix.

`DE.genes` vector of genes used for differential expression in the simulated counts matrix.

`DE.ind` logical vector indicating which genes are differentially expressed in the simulated counts matrix.

`col` vector of lenght 3*k.ind specifying which columns were used in the simulation. The first k.ind columns were used to simulate the first treatment group. The last 2*k.ind columns were used to simulate the second treatment group. If switch.trt equals TRUE, then the first 2*k.ind columns were used to simulate the first treatment group and the last k.ind columns were used to simulate the second treatment group.

Author(s)

Samuel Benidt <sgbenidt@gmail.com>

Source

Benidt, S., and Nettleton, D. (2015). SimSeq: a nonparametric approach to simulation of RNA-sequence datasets. Bioinformatics. 31, 2131-2140.

The Cancer Genome Atlas Research Network (2013). Comprehensive molecular characterization of clear cell renal cell carcinoma. Nature, 499(7456), 43-49.

https://tcga-data.nci.nih.gov/tcga/

Examples

```
    data(kidney)
    counts <- kidney$counts # Matrix of read counts from KIRC dataset
    replic <- kidney$replic # Replic vector indicating paired columns
    treatment <- kidney$treatment # Treatment vector indicating Non-Tumor or Tumor columns

    nf <- apply(counts, 2, quantile, 0.75)

    library(fdrtool)

    ## Not run:

      ### Example 1: Simulate Matrix with 1000 DE genes and 4000 EE genes
      data.sim <- SimData(counts = counts, replic = replic, treatment = treatment,
                          sort.method = "paired", k.ind = 5, n.genes = 5000, n.diff = 1000,
                          norm.factors = nf)

      ### Example 2: Calculate weights vector beforehand to save run time in
      ### repeated simulations
      sort.list <- SortData(counts = counts, treatment = treatment, replic = replic,
                            sort.method = "paired", norm.factors = nf)
      counts <- sort.list$counts
      replic <- sort.list$replic
      treatment <- sort.list$treatment
      nf <- sort.list$norm.factors

      probs <- CalcPvalWilcox(counts, treatment, sort.method = "paired",
                              sorted = TRUE, norm.factors = nf, exact = FALSE)
      weights <- 1 - fdrtool(probs, statistic = "pvalue", plot = FALSE, verbose = FALSE)$lfdr

      data.sim <- SimData(counts = counts, replic = replic, treatment = treatment,
                          sort.method = "paired", k.ind = 5, n.genes = 5000, n.diff = 1000,
                          weights = weights, norm.factors = nf)

      ### Example 3: Specify which genes you want to use in the simulation

      # Randomly sample genes or feed in the exact genes you wish to use
      genes.diff <- sample(1:nrow(counts), size = 1000, prob = weights)
      genes <- c(sample(1:nrow(counts)[-genes.diff], 4000), genes.diff)

      data.sim <- SimData(counts = counts, replic = replic, treatment = treatment,
                          sort.method = "paired", k.ind = 5, genes.select = genes,
                          genes.diff = genes.diff, weights = weights, norm.factors = nf)

      ### Example 4: Simulate matrix with DE genes having log base 2 fold change greater than 1

      # add one to counts matrix to avoid infinities when taking logs
      tumor.mean <- rowMeans(log2((counts[, treatment == "Tumor"] + 1) %*%
        diag(1/nf[treatment == "Tumor"])))
      nontumor.mean <- rowMeans(log2((counts[, treatment == "Non-Tumor"] + 1) %*%
        diag(1/nf[treatment == "Non-Tumor"])))
```

```
lfc <- tumor.mean - nontumor.mean
weights.zero <- abs(lfc) < 1
weights[weights.zero] <- 0

data.sim <- SimData(counts = counts, replic = replic, treatment = treatment,
                    sort.method = "paired", k.ind = 5, n.genes = 5000, n.diff = 1000,
                    weights = weights, norm.factors = nf)

### Example 5: Simulate three treatment groups:
### 3 Different types of Differential Expression Allowed
### First Group Diff, Second and Third group Equal
### Second Group Diff, First and Third group Equal
### Third Group Diff, First and Second group Equal

k <- 5 # Sample Size in Each treatment group

### Sample DE genes beforehand
N <- nrow(counts)
genes.de <- sample(1:N, size = 1000, prob = weights) # Sample all DE genes
DE1 <- genes.de[1:333] # Sample DE genes with first trt diff
DE2 <- genes.de[334:666] # Sample DE genes with sec trt diff
DE3 <- genes.de[667:1000] # Sample DE genes with third trt diff
EE <- sample( (1:N)[-genes.de], size = 4000) #Sample EE genes

genes.tot <- c(EE, genes.de)
genes.de1 <- union(DE2, EE) #Assign DE genes for first sim
genes.de2 <- union(DE2, DE3) #Assign DE genes for second sim

data.sim1 <- SimData(counts = counts, replic = replic, treatment = treatment,
                     sort.method = "paired", k.ind = k, genes.select = genes.tot,
                     genes.diff = genes.de1, weights = weights, norm.factors = nf)

#remove pairs of columns used in first simulation
cols.rm <- c(data.sim1$col[1:(2*k)], data.sim1$col[1:(2*k)] + 1)
counts.new <- counts[, -cols.rm]
nf.new <- nf[-cols.rm]
replic.new <- replic[-cols.rm]
treatment.new <- treatment[-cols.rm]

### Set switch.trt = TRUE for second sim
data.sim2 <- SimData(counts = counts.new, replic = replic.new, treatment = treatment.new,
                     sort.method = "paired", k.ind = k, genes.select = genes.tot,
                     genes.diff = genes.de2, weights = weights, norm.factors = nf.new,
                     switch.trt = TRUE)

### Remove first k.ind entries from first sim and combine two count matrices
counts.sim <- cbind(data.sim1$counts[, -(1:k)],  data.sim2$counts)

### treatment group levels for simulated matrix
trt.grp <- rep(NA, 5000)
trt.grp[is.element(data.sim1$genes.subset, DE1)] <- "DE_First_Trt"
trt.grp[is.element(data.sim1$genes.subset, DE2)] <- "DE_Second_Trt"
trt.grp[is.element(data.sim1$genes.subset, DE3)] <- "DE_Third_Trt"
```

```
  trt.grp[is.element(data.sim1$genes.subset, EE)] <- "EE"

## End(Not run)
```

SortData *SortData*

Description

A function called in SimData used to trim and sort the matrix of counts provided.

Usage

```
SortData(counts, treatment, replic = NULL, sort.method, norm.factors = NULL)
```

Arguments

`counts`	A matrix of counts where each row specifies a gene and each column specifies a replicate.
`treatment`	A vector specifying the treatment group for each column of the counts matrix. Only two treatment groups of either paired or unpaired data are allowed.
`replic`	A vector specifying the replicate for each column of the counts matrix when there is paired data; optional if data is unpaired.
`sort.method`	Character vector specifying one of "unpaired" or "paired", depending on the structure of the data.
`norm.factors`	An optional positive numeric vector of multiplicative normalization factors for each column of the counts matrix.

Value

List containing:

`counts`	sorted and trimmed matrix of counts.
`replic`	sorted and trimmed replic vector.
`treatment`	sorted and trimmed treatment vector.
`norm.factors`	sorted and trimmed offset vector.
`sorting`	sorting vector used to sort and trim.

Author(s)

Samuel Benidt <sgbenidt@gmail.com>

Index