Package 'findGSEP'

May 8, 2026

Type Package

Title Estimate Genome Size of Polyploid Species Using k-Mer Frequencies

Version 1.2.0

Maintainer Laiyi Fu <fulaiyi321@163.com>

Description Provides tools to estimate the genome size of polyploid species using k-mer frequencies. This package includes functions to process k-mer frequency data and perform genome size estimation by fitting k-mer frequencies with a normal distribution model. It supports handling of complex polyploid genomes and offers various options for customizing the estimation process. The basic method 'findGSE' is detailed in Sun, Hequan, et al. (2018) <doi:10.1093/bioinformatics/btx637>.

License GPL (>= 2)

Encoding UTF-8

Depends R(>= 4.2.0),RColorBrewer,ggplot2,grDevices

Imports pracma, scales, dplyr,

RoxygenNote 7.3.1

URL https://github.com/sperfu/findGSEP

BugReports https://github.com/sperfu/findGSEP/issues

Suggests knitr, rmarkdown,

VignetteBuilder knitr

NeedsCompilation no

Author Laiyi Fu [aut, cre],
Hequan Sun [aut]

Repository CRAN

Date/Publication 2024-05-26 17:30:03 UTC

Contents

`error_minimize` *Minimize the Error for K-mer Frequency Fitting*

Description

This function minimizes the error for k-mer frequency fitting by adjusting the mean, standard deviation, and scaling factors.

Usage

```
error_minimize(
  tooptimize,
  x,
  end,
  xfit,
  xfit_left,
  xfit_right,
  d,
  min_valid_pos,
  itr
)
```

Arguments

tooptimize	A numeric vector containing the scale factors to optimize.
x	A numeric vector representing the histogram replicated from d.
end	An integer indicating the right-side position for fitting.
xfit	x-values to get the skew normal distribution
xfit_left	A numeric value for the left-side position to calculate initial mean and standard deviation.
xfit_right	A numeric value for the right-side position to calculate initial mean and standard deviation.
d	A data frame representing the observed k-mer frequencies that will be fitted.
min_valid_pos	An integer indicating the left-side position from which the observed k-mer frequencies will be fitted.
itr	An integer representing the iteration count.

Value

A numeric value representing the minimized error.

Examples

```
tooptimize <- c(1, 1, 1, 1)
x <- rnorm(100)
end <- 100
xfit <- seq(min(x), max(x), length=end)
xfit_left <- min(x)
xfit_right <- max(x)
d <- data.frame(V1=1:100, V2=rnorm(100))
min_valid_pos <- 10
itr <- 100
error <- error_minimize(tooptimize, x, end, xfit, xfit_left, xfit_right, d, min_valid_pos, itr)
print(error)
```

---

error_minimize2 *Tuning Final Fitting for Heterozygous Genomes*

Description

This function tunes the final fitting for heterozygous genomes by adjusting the delta values for heterozygous and homozygous regions.

Usage

```
error_minimize2(tooptimize, h_het, h_hom, h_target)
```

Arguments

tooptimize	A numeric vector containing the scale factors to optimize.
h_het	A numeric vector representing the raw fitting for the heterozygous region.
h_hom	A numeric vector representing the raw fitting for the homozygous region.
h_target	A numeric vector representing the target k-mer frequency.

Value

A numeric value representing the minimized difference.

Examples

```
tooptimize <- c(0.5)
h_het <- rnorm(100)
h_hom <- rnorm(100)
h_target <- rnorm(100)
diff <- error_minimize2(tooptimize, h_het, h_hom, h_target)
print(diff)
```

---

error_minimize2_raw *Tuning Final Fitting for Raw Heterozygous Genomes*

Description

This function tunes the final fitting for raw heterozygous genomes by adjusting the delta values for heterozygous and homozygous regions.

Usage

```
error_minimize2_raw(tooptimize, h_het, h_hom, h_target)
```

Arguments

tooptimize	A numeric vector containing the scale factors to optimize.
h_het	A numeric vector representing the raw fitting for the heterozygous region.
h_hom	A numeric vector representing the raw fitting for the homozygous region.
h_target	A numeric vector representing the target k-mer frequency.

Value

A numeric value representing the minimized difference.

Examples

```
tooptimize <- c(0.5)
h_het <- rnorm(100)
h_hom <- rnorm(100)
h_target <- rnorm(100)
diff <- error_minimize2_raw(tooptimize, h_het, h_hom, h_target)
print(diff)
```

`error_minimize3` *Minimize the Error for Remaining K-mer Frequency*

Description

This function minimizes the error for the remaining k-mer frequency by adjusting the scaling factor.

Usage

```
error_minimize3(
  tooptimize,
  x,
  end,
  xfit_left,
  xfit_right,
  d,
  min_valid_pos,
  itr,
  meanfit,
  sdfit
)
```

Arguments

`tooptimize`	A numeric vector containing the scale factors to optimize.
`x`	A numeric vector representing the histogram.
`end`	An integer indicating the right-side position for fitting.
`xfit_left`	A numeric value for the left-side position to calculate initial mean and standard deviation.
`xfit_right`	A numeric value for the right-side position to calculate initial mean and standard deviation.
`d`	A data frame representing the observed k-mer frequencies that will be fitted.
`min_valid_pos`	An integer indicating the left-side position from which the observed k-mer frequencies will be fitted.
`itr`	An integer representing the iteration count.
`meanfit`	A numeric value representing the initial mean.
`sdfit`	A numeric value representing the initial standard deviation.

Value

A numeric value representing the minimized error.

Examples

```
tooptimize <- c(1, 1, 1, 1)
x <- rnorm(100)
end <- 100
xfit <- seq(min(x), max(x), length=end)
xfit_left <- 20
xfit_right <- 80
d <- data.frame(V1=1:100, V2=rnorm(100))
min_valid_pos <- 10
itr <- 100
meanfit <- 18
sdfit <- 4.21
error <- error_minimize3(tooptimize, x, end, xfit_left, xfit_right, d,
min_valid_pos, itr, meanfit, sdfit)
print(error)
```

---

error_minimize3_raw *Minimize the Error for Raw Remaining K-mer Frequency*

Description

This function minimizes the error for the raw remaining k-mer frequency by adjusting the scaling factor.

Usage

```
error_minimize3_raw(
  tooptimize,
  x,
  end,
  xfit_left,
  xfit_right,
  d,
  min_valid_pos,
  itr
)
```

Arguments

tooptimize	A numeric vector containing the scale factors to optimize.
x	A numeric vector representing the histogram.
end	An integer indicating the right-side position for fitting.

xfit_left	A numeric value for the left-side position to calculate initial mean and standard deviation.
xfit_right	A numeric value for the right-side position to calculate initial mean and standard deviation.
d	A data frame representing the observed k-mer frequencies that will be fitted.
min_valid_pos	An integer indicating the left-side position from which the observed k-mer frequencies will be fitted.
itr	An integer representing the iteration count.

Value

A numeric value representing the minimized error.

Examples

```
tooptimize <- c(1, 1, 1, 1)
x <- rnorm(100)
end <- 100
xfit <- seq(min(x), max(x), length=end)
xfit_left <- min(x) + 1
xfit_right <- max(x) - 1
d <- data.frame(V1=1:100, V2=rnorm(100))
min_valid_pos <- 10
itr <- 100

error <- error_minimize3_raw(tooptimize, x, end, xfit_left, xfit_right, d,
min_valid_pos, itr)
print(error)
```

---

error_minimize_raw *Minimize the Error for Raw K-mer Frequency Fitting*

Description

This function minimizes the error for raw k-mer frequency fitting by adjusting the mean, standard deviation, and scaling factors.

Usage

```
error_minimize_raw(
  tooptimize,
  x,
  end,
  xfit,
  xfit_left,
  xfit_right,
```

```
  d,
  min_valid_pos,
  itr
)
```

Arguments

tooptimize	A numeric vector containing the scale factors to optimize.
x	A numeric vector representing the histogram replicated from d.
end	An integer indicating the right-side position for fitting.
xfit	x-values to get the skew normal distribution
xfit_left	A numeric value for the left-side position to calculate initial mean and standard deviation.
xfit_right	A numeric value for the right-side position to calculate initial mean and standard deviation.
d	A data frame representing the observed k-mer frequencies that will be fitted.
min_valid_pos	An integer indicating the left-side position from which the observed k-mer frequencies will be fitted.
itr	An integer representing the iteration count.

Value

A numeric value representing the minimized error.

Examples

```
tooptimize <- c(1, 1, 1, 1)
x <- rnorm(100)
end <- 100
xfit <- seq(min(x), max(x), length=end)
xfit_left <- min(x)
xfit_right <- max(x)
d <- data.frame(V1=1:100, V2=rnorm(100))
min_valid_pos <- 10
itr <- 100
error <- error_minimize_raw(tooptimize, x, end, xfit, xfit_left, xfit_right, d, min_valid_pos, itr)
print(error)
```

---

filter_peaks *Filter Peaks from K-mer Histogram*

Description

This function filters peaks from a k-mer histogram to find a major peak with enough support information on both sides.

Usage

```
filter_peaks(peaks, histo)
```

Arguments

peaks	A data frame containing the peaks from the histogram.
histo	A data frame representing the k-mer histogram.

Value

A data frame with filtered peaks.

Examples

```
peaks <- data.frame(V1=1:20, V2=sample(1:10, 20, replace=TRUE))
histo <- data.frame(V1=1:100, V2=sample(1:10, 100, replace=TRUE))
filtered_peaks <- filter_peaks(peaks, histo)
print(filtered_peaks)
```

---

filter_peaks_raw *Filter Peaks from Raw K-mer Histogram*

Description

This function filters peaks from a raw k-mer histogram to find a major peak with enough support information on both sides.

Usage

```
filter_peaks_raw(peaks, histo)
```

Arguments

peaks	A data frame containing the peaks from the histogram.
histo	A data frame representing the raw k-mer histogram.

Value

A data frame with filtered peaks.

Examples

```
peaks <- data.frame(V1=1:20, V2=sample(1:10, 20, replace=TRUE))
histo <- data.frame(V1=1:100, V2=sample(1:10, 100, replace=TRUE))
filtered_peaks <- filter_peaks_raw(peaks, histo)
print(filtered_peaks)
```

`findGSEP` *Estimate genome size of polyploid species using k-mer frequencies.*

Description

findGSEP is a function for multiple polyploidy genome size estimation by fitting k-mer frequencies iteratively with a normal distribution model.

To use findGSEP, one needs to prepare a histo file, which contains two tab-separated columns. The first column gives frequencies at which k-mers occur in reads, while the second column gives counts of such distinct k-mers. Parameters k and related histo file are required for any estimation.

Dependencies (R library) required: pracma, fGarch, etc. - see DESCRIPTION for details.

Usage

```
findGSEP(
  path,
  samples,
  sizek,
  exp_hom,
  ploidy,
  range_left,
  range_right,
  xlimit,
  ylimit,
  output_dir = "outfile"
)
```

Arguments

`path`	is the histo file location (mandatory).
`samples`	is the histo file name (mandatory)
`sizek`	is the size of k used to generate the histo file (mandatory). K is involved in calculating heterzygosity if the genome is heterozygous.
`exp_hom`	a rough average k-mer coverage for finding the homozygous regions. In general, one can get peaks in the k-mer frequencies file, but has to determine which one is for the homozygous regions, and which one is for the heterozygous regions. It is optional, however, it must be provided if one wants to estimate size for a heterozygous genome. VALUE for exp_hom must satisfy fp < VALUE < 2*fp, where fp is the freq for homozygous peak. If not provided, 0 by default assumes the genome is homozygous.
`ploidy`	is the number of ploidy. (mandatory).
`range_left`	is the left range for estimation, default is exp_hom*0.2, normally do not need to change this. (optional).
`range_right`	is the right range for estimation, default is exp_hom*0.2, normally do not need to change this. (optional).

xlimit	is the x-axis range, if not given, then it will automatically calculate a proper range, normally do not need to change this. (optional).
ylimit	is the y-axis range, if not given, then it will automatically calculate a proper range, normally do not need to change this. (optional).
output_dir	is the path to write output files (optional). If not specify, will use tempdir() as output directory.

Value

No return value, called for side effects. The function generates PDF, PNG, and CSV files in the specified output directory.

Examples

```
## Not run:
test_histo <- system.file("extdata","example.histo",package = "findGSEP")
path <- dirname(test_histo)
samples <- basename(test_histo)
sizek <- 21
exp_hom <- 200
ploidy <- 3
range_left <- exp_hom*0.2
range_right <- exp_hom*0.2
xlimit <- -1
ylimit <- -1
output_dir <- tempdir()

findGSEP(path, samples, sizek, exp_hom, ploidy, range_left, range_right, xlimit, ylimit, output_dir)

## End(Not run)
```

---

findGSE_raw *Estimate Genome Size Using Raw K-mer Frequencies*

Description

This function estimates the genome size of a species using raw k-mer frequencies.

Usage

```
findGSE_raw(histo = "", sizek = 0, outdir = "", exp_hom = 0, species = "")
```

Arguments

histo	A character string specifying the path to the histogram file.
sizek	An integer indicating the size of k used to generate the histogram.
outdir	A character string specifying the output directory. If not specify, will use tempdir() as output directory.

exp_hom	A numeric value representing the expected average k-mer coverage for the homozygous regions.
species	A character string specifying the species name.

Value

A list containing the estimated genome size and other fitting parameters.

Examples

```
## Not run:

histo <- "sample1.histo"
sizek <- 21
outdir <- tempdir()
exp_hom <- 200
species <- ""
fit_lists <- findGSE_raw(histo, sizek,output_dir, exp_hom, species)

## End(Not run)
```

---

findGSE_sp *Estimate Genome Size Using K-mer Frequencies*

Description

This function estimates the genome size of a species using k-mer frequencies.

Usage

```
findGSE_sp(
  histo = "",
  sizek = 0,
  outdir = "",
  exp_hom = 0,
  species = "",
  ploidy_ind = 2,
  avg_cov = 0,
  left_fit_ratio = 0.835,
  meanfit_old = 0,
  sdfit_old = 0,
  scale_flag = FALSE
)
```

Arguments

histo	A character string specifying the path to the histogram file.
sizek	An integer indicating the size of k used to generate the histogram.
outdir	A character string specifying the output directory. If not specify, will use tempdir() as output directory.
exp_hom	A numeric value representing the expected average k-mer coverage for the homozygous regions.
species	A character string specifying the species name.
ploidy_ind	An integer indicating the ploidy index (default is 2).
avg_cov	A numeric value representing the average coverage.
left_fit_ratio	A numeric value for the left fit ratio (default is 0.835).
meanfit_old	A numeric value representing the previous mean fit.
sdfit_old	A numeric value representing the previous standard deviation fit.
scale_flag	A logical value indicating whether to apply scaling (default is FALSE).

Value

A list containing the estimated genome size and other fitting parameters.

Examples

```
## Not run:

histo <- "sample1.histo"
sizek <- 21
outdir <- tempdir()
exp_hom <- 200
species <- ""
ploidy_ind <- 2
avg_cov <- 0
left_fit_ratio <- 0.835
meanfit_old <- 0
sdfit_old <- 0
scale_flag <- FALSE
fit_lists <- findGSE_sp(path, samples, sizek, exp_hom, ploidy, range_left,
 range_right, xlimit, ylimit, output_dir)

## End(Not run)
```

get_het_pos *Filter Peaks from K-mer Histogram*

Description

This function filters peaks from a k-mer histogram to find a major peak with enough support information on both sides.

Usage

```
get_het_pos(histo_data)
```

Arguments

histo_data	A data frame representing the k-mer histogram.

Value

A data frame with filtered peaks.

Examples

```
x <- seq(-10, 10, length.out = 100)
y <- dnorm(x)
histo_data <- data.frame(V1 = 1:100, V2 = y * 10)
get_het_pos(histo_data)
```

initialize_start_time *Initialize Start Time*

Description

This function initializes the start time for a process.

Usage

```
initialize_start_time()
```

Value

The start time of the process.

initial_count_recover *Recover Initial K-mer Count*

Description

This function recovers the initial k-mer count, making the k-mer frequency consecutive.

Usage

```
initial_count_recover(d0)
```

Arguments

d0	A data frame representing the initial k-mer count from software like Jellyfish.

Value

A data frame with recovered k-mer counts.

Examples

```
d0 <- data.frame(V1 = c(1, 2, 4), V2 = c(100, 200, 300))
dr <- initial_count_recover(d0)
```

initial_count_recover_raw *Recover Initial Raw K-mer Count*

Description

This function recovers the initial raw k-mer count, making the k-mer frequency consecutive.

Usage

```
initial_count_recover_raw(d0)
```

Arguments

d0	A data frame representing the initial raw k-mer count from software like Jellyfish.

Value

A data frame with recovered raw k-mer counts.

Examples

```
d0 <- data.frame(V1 = c(1, 2, 4), V2 = c(100, 200, 300))
dr <- initial_count_recover_raw(d0)
```

`kmer_count_modify` *Modify K-mer Frequency Before Fitting*

Description

This function modifies the k-mer frequency before fitting, adjusting either the left or right part based on the peak position.

Usage

```
kmer_count_modify(start, end, left_right, histx)
```

Arguments

`start`	An integer indicating the starting position (does not include the peak position).
`end`	An integer indicating the ending position (does not include the peak position).
`left_right`	An integer indicating whether to modify the left part (0) or the right part (1).
`histx`	A data frame representing the k-mer histogram.

Value

A modified data frame with adjusted k-mer frequencies.

Examples

```
histx <- data.frame(V1 = 1:10, V2 = c(100, 200, 300, 400, 500, 400, 300, 200, 100, 50))
histx_new <- kmer_count_modify(3, 7, 0, histx)
```

`kmer_count_modify_raw` *Modify Raw K-mer Frequency Before Fitting*

Description

This function modifies the raw k-mer frequency before fitting, adjusting either the left or right part based on the peak position.

Usage

```
kmer_count_modify_raw(start, end, left_right, histx)
```

Arguments

start	An integer indicating the starting position (does not include the peak position).
end	An integer indicating the ending position (does not include the peak position).
left_right	An integer indicating whether to modify the left part (0) or the right part (1).
histx	A data frame representing the raw k-mer histogram.

Value

A modified data frame with adjusted raw k-mer frequencies.

Examples

```
histx <- data.frame(V1 = 1:10, V2 = c(100, 200, 300, 400, 500, 400, 300, 200, 100, 50))
histx_new <- kmer_count_modify_raw(3, 7, 0, histx)
```

Index