Package 'Rariant'

October 8, 2014

Type Package

Title Identification and Assessment of Single Nucleotide Variants through Shifts in Non-Consensus Base Call Frequencies

Version 1.0.0

Date 2014-04-09

Author Julian Gehring, Simon Anders, Bernd Klaus (EMBL Heidelberg)

Maintainer Julian Gehring <julian.gehring@embl.de>

Imports IRanges, ggbio, ggplot2, VariantAnnotation, h5vc, exomeCopy,SomaticSignatures, Rsamtools, shiny, GenomicRanges

Depends R (>= 3.0.2)

Suggests h5vcData, testthat, knitr, BiocStyle, biovizBase, optparse

Description
The 'Rariant' package identifies single nucleotide variants from sequencing data based on the difference of binomially distributed mismatch rates between matched samples.

VignetteBuilder knitr

Encoding UTF-8

ByteCompile TRUE

License GPL-3

LazyLoad yes

biocViews Sequencing, StatisticalMethod, GenomicVariation,SomaticMutation, VariantDetection, Visualization

R topics documented:

`Rariant-package` *Rariant package*

Description

The 'Rariant' package offers the framework to identify and characterize shifts of variant frequencies in a comparative setting from high-throughput short-read sequencing data. It estimates shifts in the non-consensus variant frequency and provides confidence estimates that allow for a quantitative assessment of presence or absence of variants. The vignette accompanying the package gives a detailed explanation and outlines a typical workflow on real data.

Author(s)

Julian Gehring, Simon Anders, Bernd Klaus (EMBL Heidelberg)

Maintainer: Julian Gehring <julian.gehring@embl.de>

See Also

rariant

vignette(package = "Rariant")

Examples

```
help("rariant")

vignette(package = "Rariant")
```

ciAdjust *CI Adjust*

Description

Multiple testing adjustment of confidence levels, as proposed by Benjamini and Yekutieli.

Usage

```
ciAdjustLevel(eta0, conf_level)
```

Arguments

eta0	Estimated fraction of tests that are consistent with the null hypothesis.
conf_level	Unadjusted confidence level

Value

The adjusted confidence level.

References

Benjamini, Yoav, and Daniel Yekutieli. False Discovery Rate–adjusted Multiple Confidence Intervals for Selected Parameters. Journal of the American Statistical Association 100, no. 469 (2005): 71–81.

Examples

```
conf_level = 0.95
eta0 = seq(0, 1, by = 0.02)

conf_level_adj = ciAdjustLevel(eta0, conf_level)

plot(eta0, conf_level_adj, pch = 20, ylim = c(conf_level, 1))
```

ciAssessment *Assessment of CI methods*

Description

Functions to compute the coverage probability of a confidence interval method.

Usage

```
coverageProbability(pars, fun = acCi, n_sample = 1e4, min_k, ...)
```

Arguments

pars	Data frame with parameter combinations [data.frame]
n_sample	Number of assessments per parameter combination [integer(1)].
fun	CI function
min_k	Minimum 'k2' value to use.
...	Additional arguments that are passed on to 'fun'.

Value

The 'data.frame' object 'pars' with additional columns 'cp' for the coverage probability and 'aw' average confidence interval width.

References

Fagerland, Morten W., Stian Lydersen, and Petter Laake. Recommended Confidence Intervals for Two Independent Binomial Proportions. Statistical Methods in Medical Research (2011).

Examples

```
## Define parameter space
pars = expand.grid(k1 = 1:5, k2 = 5, n1 = 30, n2 = 30)
conf_level = 0.95

## Compute coverage probabilities
cp = coverageProbability(pars, fun = acCi, n_sample = 1e2, conf_level = conf_level)
print(cp)
```

---

ciUtils *CI Utils*

Description

Utility functions to find confidence intervals that (a) overlap a certain value ('ciOutside', 'ciCovers') and (b) different confidence intervals overlap ('ciOverlap').

Usage

```
ciOutside(x, delta = 0)

ciCovers(x, delta = 0)

ciOverlap(x, y)

ciWidth(x)
```

Arguments

x, y	CIs, as obtained from e.g. the 'acCi' function.
delta	Variant frequency value to check against [default: 0].

Value

A logical vector, where each elements corresponds to the respective row of 'x' (and 'y').

For 'ciWidth': A numeric vector with the widths of the confidence intervals.

Examples

```
## Generate sample data
counts = data.frame(x1 = 1:5, n1 = 30, x2 = 0:4, n2 = 30)

## Agresti-Caffo
ci_ac = with(counts, acCi(x1, n1, x2, n2))
ci_ac2 = with(counts, acCi(x1, n1, x2, n2, 0.99))

## cover 0
idx_zero = ciCovers(ci_ac)

## cover 1
idx_one = ciCovers(ci_ac, delta = 1)

## overlap
idx_same = ciOverlap(ci_ac, ci_ac2)

## width
width = ciWidth(ci_ac)
```

---

convertUtils *Position converters*

Description

Utility functions to convert between 'GRanges' and 'character' objects.

Usage

```
gr2pos(x)

pos2gr(x)
```

Arguments

x	GRanges or character object.

Value

A GRanges object or character object, with the position.

Examples

```
library(GenomicRanges)

gr = GRanges(1:2, IRanges(1:2, width = 1))

pos = gr2pos(gr)
gr2 = pos2gr(pos)

identical(gr, gr2)
```

`mismatchUtils` *Tally processing low-level functions*

Description

Functions for processing position-specific base count tables (tallies) and extracting mismatches counts.

Usage

```
## low-level functions
selectStrand(x, strand = c("both", "plus", "minus"), idx = 1:ncol(x))

seqDepth(x)

callConsensus(counts, verbose = FALSE)

mismatchCount(counts, consensus, depth = rowSums(counts))
```

Arguments

`x`	Input object
`strand`	Which strand to return?
`idx`	Index of bases to consider (leave as is)
`counts`	Count matrix
`verbose`	Show warnings
`consensus`	Consensus sequence
`depth`	Sequencing depth for counts.

See Also

comparativeMismatch

plotCIs *Plotting Functions*

Description

The 'plotConfidenceIntervals' is a high-level plotting function for visualizing confidence intervals. The 'plotAbundanceShift' function visualizes the shift in mismatch rates between two samples.

Usage

```
plotConfidenceIntervals(x, ylim = c(-1.05, 1.05), color = NULL, ...)

plotAbundanceShift(x, ylim = c(-0.05, 1.05), rates = TRUE, ...)
```

Arguments

x	'GRanges' with mcols of a CI method, or 'data.frame' as returned by one of the CI methods, with the optional column 'start'.
ylim	Limits of the y-axis. Using this instead of using the 'ylim' prevents ugly warnings of 'ggplot2'.
color	Variable that determines the coloring of the confidence axis (character).
rates	Should the non-consensus rates of both samples be visualized as colored end points of the line range? (logical, default: TRUE).
...	Additional plotting arguments that are passed on to ggplot2::geom_pointrange.

Value

For a 'GRanges' input: A 'ggbio' object

For a 'data.frame' input: A 'ggplot' object

Examples

```
## Generate sample data
counts = data.frame(x1 = 1:5, n1 = 30, x2 = 0:4, n2 = 30)

## Agresti-Caffo
ci_ac = with(counts, acCi(x1, n1, x2, n2))

library(GenomicRanges)
gr = GRanges("1", IRanges(start = 1:nrow(counts), width = 1))
mcols(gr) = ci_ac

## GRanges
plotConfidenceIntervals(gr)

## data.frame
plotConfidenceIntervals(ci_ac)
```

```
## abundance shift
plotAbundanceShift(gr)

plotAbundanceShift(ci_ac)
```

---

propCIs *Confidence Interval Functions*

Description

Vectorized implementation of confidence intervals

Usage

```
acCi(x1, n1, x2, n2, conf_level = 0.95, clip = TRUE, split = FALSE)
nhsCi(x1, n1, x2, n2, conf_level = 0.95)
```

Arguments

x1	Mismatch counts in the test sample.
n1	Sequencing depth (total counts) in the test sample.
x2	Mismatch counts in the control sample.
n2	Sequencing depth (total counts) in the control sample.
conf_level	Confidence level $beta$ (default: 0.95).
clip	Should the CIs be clipped to the interval [-1,1] if they exceed this?
split	Should the sample split method be applied? See 'splitSampleBinom' for details.

Details

These functions implement a vectorized version of the two-sided Agresti-Caffo, and Newcombe-Hybrid-Score confidence interval for the difference of two binomial proportions.

Value

A data frame with columns

- dEstimate for the difference of rates 'p1' and 'p2'.
- p1, p2Estimates for the mismatches rates for each sample.
- lower, upperLower and upper bound of the confidence interval.
- wWidth of the confidence interval.

References

Agresti, Alan, and Brian Caffo. Simple and Effective Confidence Intervals for Proportions and Differences of Proportions Result from Adding Two Successes and Two Failures. The American Statistician 54, no. 4 (2000): 280–288

Newcombe, Robert G. Interval Estimation for the Difference between Independent Proportions: Comparison of Eleven Methods. Statistics in Medicine 17, no. 8 (1998): 873–890.

Fagerland, Morten W., Stian Lydersen, and Petter Laake. Recommended Confidence Intervals for Two Independent Binomial Proportions. Statistical Methods in Medical Research (2011).

See Also

nhsCi

splitSampleBinom

binMto::Add4 binMto::NHS

Examples

```
## Generate sample data
counts = data.frame(x1 = 1:5, n1 = 30, x2 = 0:4, n2 = 30)

## Agresti-Caffo
ci_ac = with(counts, acCi(x1, n1, x2, n2))

## Newcombe-Hybrid Score
ci_nhs = with(counts, nhsCi(x1, n1, x2, n2))

print(ci_ac)
```

`propTests` *Testing Functions*

Description

Vectorized implementation of testing functions

Usage

```
scoreTest(x1, n1, x2, n2)

nmTest(x1, n1, x2, n2, delta = 0)

feTest(x1, n1, x2, n2, ...)
```

Arguments

x1	Mismatch counts in the test sample.
n1	Sequencing depth (total counts) in the test sample.
x2	Mismatch counts in the control sample.
n2	Sequencing depth (total counts) in the control sample.
delta	Difference to test against (default: 0).
...	Additional arguments.

Details

These functions implement a vectorized version of the two-sided (a) Score test and (b) Miettinen-Nurminen test for the difference between to Binomial proportions.

Usage of the score test is discouraged in the settings considered here, since it is ill-defined for positions with no mismatches.

Value

A data frame with columns

- dhatEstimate for the difference of rates 'p1' and 'p2'.
- p1, p2Estimates for the mismatches rates for each sample.
- tvalT-value
- pvalP-value

References

Miettinen, Olli, and Markku Nurminen. Comparative Analysis of Two Rates. Statistics in Medicine 4, no. 2 (1985): 213–226. doi:10.1002/sim.4780040211.

See Also

VariantTools package

Examples

```
## Generate sample data
counts = data.frame(x1 = 1:5, n1 = 30, x2 = 0:4, n2 = 30)

## Score test
stat_st = with(counts, scoreTest(x1, n1, x2, n2))

## NM test
stat_nm = with(counts, nmTest(x1, n1, x2, n2))

## Fisher test
stat_fet = with(counts, feTest(x1, n1, x2, n2))
```

```
print(stat_st)

print(stat_nm)

print(stat_fet)
```

`rariant` *Rariant calling functions*

Description

The 'rariant' function screens for variant shifts between a test and control sample. These highlevel functions offers a convenient interface for large-scale identification as well as for reexamination of existing variant calls.

Usage

```
rariant(test, control, region, beta = 0.95, alpha = 1 - beta,
select = TRUE, consensus, resultFile, strand = c("both", "plus", "minus"),
nCycles = 10, minQual = 20, block = 1e4, value = TRUE, criteria = c("both", "any", "fet", "ci"))

rariantStandalone()

readRariant(file)

writeRariant(x, file)
```

Arguments

`test, control`	Test and control BAM files. Other input sources will be supported in the future.
`region`	Region(s) of interest to analyze in the calling [GRanges with one or more entries]. If missing, the entire genomic space, as defined by the BAM headers of the input files, will be covered.
`beta`	Confidence level [numeric in the range [0,1], default: 0.95].
`alpha`	Significance threshold for BH-adjusted p-values of the Fisher's exact test.
`select`	Should only likely variant positions be selected and returned, or the results for all sites be returned.
`consensus`	How to determine the consensus sequence. By default, the consensus is given by the most abundant allele in the control sample. Alternatively, an object with a reference sequence ('BSgenome', 'FaFile') can be passed to define the consensus sequence.
`resultFile`	If not missing, write the results to a tab-delimited file.
`strand`	Which strand should be extracted? By default, the counts of both strands are summed up.

nCycles	Number of sequencing cycles to remove from the beginning and end of each read when creating the base count table. This avoids low quality read positions [default: 10 is reasonable for current Illumina sequencing].
minQual	Minimum base call quality for reads to be considered for the nucleotide count table [default: 20 is reasonable for current Illumina sequencing]. Reads with a lower quality are dropped.
block	Number of the genomic sites to analyze in one chunk. The default is a good compromise between memory usage and speed, and normally does not require changing.
value	Should the results be returned by the function. For calls within R, this is generally set to TRUE and does not need to be changed.
criteria	The criteria to determine significant sites. Criteria are: Fisher's exact test, confidence intervals, any or both [default] of them.
file	Path to output file from a 'rariant' call.
x	Output of 'rariant' call.

Details

The 'rariant' function is the workhorse for the comparative variant calling and assessment. It starts with the aligned reads for the test (e.g. tumor) and the control (e.g. normal) sample in the BAM format; later versions will support additional inputs.

The 'select' parameter determines whether only significant variant sites or all sites are returned. While the first is suitable for detecting variants, the second becomes relevant assessing for example the abundance of variants at particular sites of interest - an example would be to determine the absence of a specific variant.

For analyses over large genomic regions and for use with infrastructure outside of R, initiating the calling from the command line may be a desirable alternative. The 'rariantStandalone' functions returns the full path to a script that can be directly called from the command line. For further details, see the help of the script by calling it with the '-h' option, for example 'rariant -h'.

The 'readRariant' and 'writeRariant' functions allow to import and export the results of a 'rariant' call from and to a file output, and will return the same object.

Value

A 'GRanges' object, with each row corresponding to a genomic site, and columns:

- testMismatch, controlMismatchMismatch counts in the test and control sample.
- testDepth, controlDepthSequencing depth in the test and control sample.
- testRef, testAltReference and alternative allele of the test sample.
- controlRefReference allele of the control sample.
- testRefDepth, testAltDepthSupporting sequencing depth for the reference and alternative allele in the test sample.
- refConsensus allele.
- p1, p2Estimated non-consensus rate in test and control, respectively.

- dEstimated shift in the non-consensus rate between test and control.
- dsEstimated shift in the non-consensus rate between test and control (shrinkage point estimate).
- lower, upperLower and upper bound of the confidence interval for 'd'.
- pval, padjRaw and BH-adjusted p-value of the FET test.
- calledWas the site identified as variant?
- eventTypeThe class of the event: somatic, heterozygous, undecided.
- padjSomatic, padjHeteroBH-adjusted p-values of the binomial tests for 'eventType'.
- pvalSomatic, pvalHeteroRaw p-values of the binomial tests for 'eventType'.

Examples

```
library(GenomicRanges)

control_bam = system.file("extdata", "NRAS.Control.bam", package = "h5vcData", mustWork = TRUE)
test_bam = system.file("extdata", "NRAS.AML.bam", package = "h5vcData", mustWork = TRUE)

roi = GRanges("1", IRanges(start = 115258439, end = 115259089))

vars = rariant(test_bam, control_bam, roi)

vars_all = rariant(test_bam, control_bam, roi, select =
FALSE)

## Not run:
  system2(rariantStandalone(), "-h")

## End(Not run)
```

`rariantInspect` *Interactive inspection*

Description

Interactively inspect variant sites and results of the 'rariant' function.

Usage

```
rariantInspect(x)
```

Arguments

`x`	The return value of the 'rariant' or 'readRariant' function.

Details

With the web interface of 'rariantInspect' can existing variant calls and assessment be explored interactively. It allows to select the genomic region of interest and the type of event. Results are shown as both a confidence interval plot and a results table that can be further filtered and reordered.

Examples

```
example(rariant)

rariantInspect(vars_all)
```

---

splitSample *Split Sample for Binomial Data*

Description

Sample splitting, according to Hall, 2014.

Usage

```
splitSampleBinom(x, n)
```

Arguments

x	Number of successes
n	Number of trials

Details

These functions implement sample splitting of a binomial rate.

Note that the results depend on the state of the random number generator, and are therefore not strictly deterministic.

Value

A vector with the rate $p = \frac{X}{N}$, obtained with sample splitting.

References

Decrouez, Geoffrey, and Peter Hall. "Split Sample Methods for Constructing Confidence Intervals for Binomial and Poisson Parameters." Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2013, n/a–n/a. doi:10.1111/rssb.12051.

Examples

```
n = 10
m = 5
pt = 0.5

x = rbinom(m, n, pt)
p = x/n

ps = splitSampleBinom(x, n)

round(cbind(p, ps), 2)
```

Index