Package ‘QCpipeline’

March 21, 2012

Version 0.5.0

Type Package

Title Utilities for the QC pipeline

Description Configuration and plotting code

Author Stephanie Gogarten, Sarah Nelson, Jess Shen, Leila Zelnick

Maintainer Stephanie Gogarten ⟨sdmorris@u.washington.edu⟩

Depends GWASTools

Imports gplots, grid, gridBase, hexbin, Biostrings

Suggests MSBVAR

License Artistic-2.0

LazyLoad yes

R topics documented:

`bestLegendPos` *Best legend position*

Description

Find the section of a plot with the fewest number of points for placing a legend

Usage

```
bestLegendPos(x, y)
```

Arguments

`x`	Points on the x axis
`y`	Points on the y axis

Value

Returns one of the legend placement strings from the list `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"` and `"center"`.

Author(s)

Stephanie Gogarten

Examples

```
x <- sample(1:100, 50, replace=TRUE)
y <- sample(1:100, 50, replace=TRUE)
plot(x,y)
legend(bestLegendPos(x,y), "points", pch=1)
```

`boxplotMeanSD` *Boxplot with mean and SD*

Description

Boxplot with mean and SD

Usage

```
boxplotMeanSD(x, y, data=NULL, xlab=NULL, ylab=NULL, nSD=1, ...)
```

Arguments

`x` vector or character string denoting column in data

`y` vector or character string denoting column in data

`data` data.frame

`xlab` title for x axis (defaults to x if data is not `NULL`

`ylab` title for y axis (defaults to y if data is not `NULL`

`nSD` number of standard deviations to plot

`...` additional plotting arguments

Author(s)

Jess Shen

Examples

```
age <- sample(25:55, 100, replace=TRUE)
sex <- sample(c("M", "F"), 100, replace=TRUE)
boxplotMeanSD(sex, age)

data <- data.frame(age, sex)
boxplotMeanSD("sex", "age", data)
```

`dbgapAnnotation` *Write annotation files for dbGaP*

Description

`dbgapScanAnnotation` and `dbgapSnpAnnotation` create text files appropriate for posting on dbGaP.

Usage

```
dbgapScanAnnotation(scanAnnot, dir=".",
   consentVar="consent", subjVar="subj.plink", dupVar="dup.post", omitVar="no.post"
   annotationCol="annotation", analysisCol="analysis")

dbgapSnpAnnotation(snpAnnot, dir=".",
   annotationCol="annotation", analysisCol="analysis")
```

Arguments

`scanAnnot` A `ScanAnnotationDataFrame`.

`snpAnnot` A `SnpAnnotationDataFrame`.

`dir` A character string with the directory for file output.

`consentVar` The variable in `scanAnnot` containing consent levels.

subjVar	The logical variable in scanAnnot indicating unique subjects to post.
dupVar	The logical variable in scanAnnot indicating duplicate scans to post.
omitVar	The logical variable in scanAnnot indicating scans to be omitted from posting.
annotationCol	The logical column in the metadata indicating which variables should be included in the annotation files.
analysisCol	The logical column in the metadata indicating which variables should be included in the analysis files.

Details

dbgapScanAnnotation writes the following files to dir:

- Sample_annotation.csv
- Sample_annotation_consent_*.csv
- Sample_annotation_duplicates.csv
- Sample_annotation_duplicates_consent_*.csv
- Sample_annotation_DD.txt
- Sample_analysis.csv
- Sample_analysis_duplicates.csv
- Sample_analysis_DD.txt

dbgapSnpAnnotation writes the following files to dir:

- SNP_annotation.csv
- SNP_annotation_DD.txt
- SNP_analysis.csv
- SNP_analysis_DD.txt

Which variables should be written to the annotation and analysis files are indicated in the metadata columns annotationCol and analysisCol.

The data dictionary files are populated from the metadata. The "type" column is automatically generated from the classes of the variables in scanAnnot and snpAnnot.

Author(s)

Stephanie Gogarten

`make.allele.mappings` *Make allele mapping file*

Description

Function to take Illumina build 37 SNP annotation data file and make an Allele Mappings table

Usage

```
make.allele.mappings(snp.dat)

make.allele.annotation(map, alleles=c("top", "design", "fwd", "plus"))
```

Arguments

`snp.dat` a data frame made from SNP Illumina annotation file (i.e., "HumanOmni2.5-4v1_D.csv"), with following fields: "IlmnID", "Name", "IlmnStrand", "SNP", "GenomeBuild", "SourceStrand", "SourceSeq", "TopGenomicSeq", "RefStrand"

`map` a data frame with allele mappings (output of `make.allele.mappings`), with following fields: "snp", "alle.AB", "alle.design", "alle.top", "alle.fwd", "alle.plus"

`alleles` character string with type of alleles to return in the annotation table ("top", "design", "fwd", or "plus")

Details

In the case of indels, the `make.allele.mappings` codes A or B as "-" and the other allele with the insertion/deletion sequence. `make.allele.annotation` codes indels as "D" and "I" (useful for making PLINK files).

Note that "RefStrand" in build 37 annotations from Illumina is result of BLAST search of DESIGN strand. In previous strand annotations from Illumina for build 36 arrays, "BlastStrand" held +/- result of BLAST search of the SOURCE sequence.

Value

`make.allele.mappings` returns a data frame object with columns ("snp", "alle.AB", "alle.design", "alle.top", "alle.fwd", "alle.plus").

`make.allele.annotation` returns a data frame with columns "snp", "alleleA.*", "alleleB.*" where "*"=`alleles`.

Author(s)

Sarah Nelson

Examples

```
## Not run:
##  Load in Illumina annotation
column.select <- rep("NULL",times=21)
column.select[c(1:4,9:11,16:18,21)] <- NA  ## only read in select columns
snp.dat <- read.csv(file="/projects/geneva/geneva_sata/SNP_annotation/Illumina/HumanOmni2.5_
                    skip=7,colClasses=column.select,nrows=2450000)   ## last rows are for co
map.final <- make.allele.mappings(snp.dat)
write.csv(map.final, file="/projects/geneva/geneva_sata/SNP_annotation/Illumina/HumanOmni2.5
          row.names=FALSE,quote=FALSE)

## Create a data frame with one row per SNP, TOP alleles, and indels coded as D/I
snp.annot <- make.allele.annotation(map.final, alleles="top")

## Example allele mappings table:
#                 snp alle.AB alle.design alle.top alle.fwd alle.plus
#  rs1000000       A           T        A        T          A
#  rs1000000       B           C        G        C          G
#  rs1000002       A           A        A        A          T
#  rs1000002       B           G        G        G          C
#  rs10000023       A           T        A        T          T
#  rs10000023       B           G        C        G          G

## With the following data dictionary:
#  variable        type    description
#  snp     text    rs id and other comparable identifiers for a snp
#  alle.AB text    A or B, per the Illumina genotyping system nomenclature
#  alle.design     text    nucleotide(s) corresponding to A or B allele for design strand
#  alle.top        text    nucleotide(s) corresponding to A or B allele for Illumina TOP str
#  alle.fwd        text    nucleotide(s) corresponding to A or B allele for FORWARD strand,
#  alle.plus       text    nucleotide(s) corresponding to A or B allele
#  for PLUS(+) strand, relative to the forward direction in the human
#  reference genome sequence

## End(Not run)
```

---

plot2DwithHist *Scatterplot with density*

Description

plot2DwithHist produces a scatterplot of y vs x, along with histograms of the marginal distributions of x and y.

Usage

```
plot2DwithHist(x, y, xlab=NULL, ylab=NULL, xlim=NULL, ylim=NULL,
  sublab=NULL, mn=NULL, sd=NULL, ...)
```

Arguments

`x`	vector of x coordinates
`y`	vector of y coordinates
`xlab`	x-axis label (defaults to variable name)
`ylab`	y-axis label (defaults to variable name)
`xlim`	x-axis limits (defaults to [min,max] of X, plus a bit of space
`ylim`	y-axis limits (defaults to [min,max] of Y, plus a bit of space
`sublab`	sub-label (instead of main, since there's no room)
`mn`	2-element vector with mean of x and y
`sd`	2-element vector with sd of x and y
`...`	additional arguments to pass to `points`

Author(s)

Leila Zelnick

Examples

```
library(MSBVAR)
# generate some multivariate normal example data
n <- 5000
mu <- c(0, 2)
vmat <- matrix(c(1, 0.7, 0.7, 1), nrow=2)

dat <- rmultnorm(n, mu, vmat) # generates n multivariate normal obs.
x <- dat[,1]
y <- dat[,2]

plot2DwithHist(x, y, xlab="This is the X variable", ylab="This is the Y variable.",
  sub="Example Plot!")
# defining axis limits
plot2DwithHist(x, y, xlab="This is the X variable", ylab="This is the Y variable.",
  sub="Example Plot!", xlim=c(0,4), ylim=c(-2,2))
```

`readConfig` *Read a configuration file*

Description

Read a configuration file

Usage

```
readConfig(file, ...)
```

Arguments

`file` file where column 1 is parameter name and column 2 is value.

`...` additional arguments to `read.table`

Value

Returns a named character vector of parameter values.

Author(s)

Stephanie Gogarten

Index