Package 'QCpipeline'

April 18, 2012

Version 0.7.0

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, quantsmooth

Suggests MSBVAR, GWASdata

License Artistic-2.0

LazyLoad yes

R topics documented:

QCpipeline-package *QC utility functions*

Description

This package contains functions for the QC pipeline.

Author(s)

Stephanie Gogarten, Sarah Nelson, Jess Shen, Leila Zelnick

Maintainer: Stephanie M. Gogarten <sdmorris@u.washington.edu>

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

ideogram *BAF/LRR plots with chromosome ideograms*

Description

Plot BAF/LRR with chromosome ideograms at the bottom.

Usage

```
chromIntensityPlotIdeogram(intenData, scan.ids, chrom.ids,
                           main=NULL, info=NULL, ...,
                           ideo.zoom=TRUE, ideo.rect=FALSE,
                           cex.axis=1.7, cex.lab=1.7, cex.main=1.7,
                           cex.sub=1.5, cex.leg=1.5)

anomStatsPlotIdeogram(intenData, genoData, anom.stats, snp.ineligible,
                      win=5, main=NULL, info=NULL, ...,
                      ideo.zoom=FALSE, ideo.rect=TRUE,
                      cex.axis=1.7, cex.lab=1.7, cex.main=1.7,
                      cex.leg=1.5)
```

Arguments

intenData	IntensityData object
genoData	GenotypeData object
scan.ids	A vector containing the sample indices of the plots.
chrom.ids	A vector containing the chromosome indices of the plots.
anom.stats	data.frame of chromosome anomalies with statistics, usually the output of anomSegStats. Names must include "anom.id", "scanID", "chromosome", "left.index", "right.index", "method", "nmark.all", "nmark.elig", "left.base", "right.base", "nbase", "non.anom.baf.med", "non.anom.lrr.med", "anom.baf.dev.med", "anom.baf.dev.5", "anom.lrr.med", "nmark.baf", "nmark.lrr". Left and right refer to start and end, respectively, of the anomaly, in position order.
snp.ineligible	vector of ineligible snp ids (e.g., intensity-only, failed snps, XTR and HLA regions). See HLA and pseudoautosomal.
win	size of the window (a multiple of anomaly length) surrounding the anomaly to plot
main	Vector of plot titles. If NULL then the title will include scanID, sex, and chromosome.
info	A character vector containing extra information to include in the main title.
...	additional arguments to chromIntensityPlot or anomStatsPlot
ideo.zoom	logical for whether to zoom in on the ideogram to match the range of the BAF/LRR plots
ideo.rect	logical for whether to draw a rectangle on the ideogram indicating the range of the BAF/LRR plots
cex.axis	cex value for the axis
cex.lab	cex value for the labels

cex.main	cex value for the title
cex.sub	cex value for the subtitle
cex.leg	cex value for the ideogram legend

Details

chromIntensityPlotIdeogram is a wrapper for chromIntensityPlot. anomStatsPlotIdeogram is a wrapper for anomStatsPlot.

These functions call paintCytobands to draw ideograms.

Author(s)

Stephanie Gogarten

See Also

chromIntensityPlot, anomStatsPlot, paintCytobands

Examples

```
library(GWASdata)
data(illuminaScanADF)
data(illuminaSnpADF)
blfile <- system.file("extdata", "illumina_bl.nc", package="GWASdata")
blnc <- NcdfIntensityReader(blfile)
intenData <-  IntensityData(blnc, scanAnnot=illuminaScanADF, snpAnnot=illuminaSnpADF)

genofile <- system.file("extdata", "illumina_geno.nc", package="GWASdata")
genonc <- NcdfGenotypeReader(genofile)
genoData <-  GenotypeData(genonc, scanAnnot=illuminaScanADF, snpAnnot=illuminaSnpADF)

# chromIntensityPlotIdeogram
scanID <- getScanID(illuminaScanADF, index=1:2)
#png("tmp_%03d.png", width=720, height=900)
main <- paste("Sample", c("A", "B"), "- Chromosome X")
chromIntensityPlotIdeogram(intenData=intenData, scan.ids=scanID,
                           chrom.ids=c(23,23), main=main,
                           colorGenotypes=TRUE, genoData=genoData)
#dev.off()

# anomStatsPlotIdeogram
scan.ids <- illuminaScanADF$scanID[1:2]
chrom.ids <- unique(illuminaSnpADF$chromosome)
snp.ids <- illuminaSnpADF$snpID[illuminaSnpADF$missing.n1 < 1]
snp.failed <- illuminaSnpADF$snpID[illuminaSnpADF$missing.n1 == 1]

# example results from anomDetectBAF
baf.anoms <- data.frame("scanID"=scan.ids[1],"chromosome"=21,
  "left.index"=100,"right.index"=200, sex="M", method="BAF",
  anom.id=1, stringsAsFactors=FALSE)

# example results from anomDetectLOH
loh.anoms <- data.frame("scanID"=scan.ids[2],"chromosome"=22,
  "left.index"=400,"right.index"=500, sex="F", method="LOH",
  anom.id=2, stringsAsFactors=FALSE)
```

```
anoms <- rbind(baf.anoms, loh.anoms)
data(centromeres.hg18)
stats <- anomSegStats(intenData, genoData, snp.ids=snp.ids, anom=anoms,
  centromere=centromeres.hg18)

#png("tmp_%03d.png", width=720, height=900)
main <- paste("Sample", c("A", "B"), "- Chromosome", stats$chromosome)
anomStatsPlotIdeogram(intenData, genoData, anom.stats=stats,
  snp.ineligible=snp.failed, centromere=centromeres.hg18, main=main)
#dev.off()

close(genoData)
close(intenData)
```

`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_4v1/HumanOmni2.
             skip=7,colClasses=column.select,nrows=2450000)  ## last rows are for control probes
map.final <- make.allele.mappings(snp.dat)
write.csv(map.final, file="/projects/geneva/geneva_sata/SNP_annotation/Illumina/HumanOmni2.5_4v1/testfn.csv
          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 strand
# alle.fwd     text   nucleotide(s) corresponding to A or B allele for FORWARD strand, with respect to dbSNP re
#  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, ...)

setConfigDefaults(config, required, optional, default)
```

Arguments

file	file where column 1 is parameter name and column 2 is value.
...	additional arguments to read.table
config	named character vector
required	character vector of required parameter names
optional	character vector of optional parameter names
default	character vector of default values for parameters in optional

Value

readConfig returns a named character vector of parameter values.

setConfigDefaults takes a named character vector returned by readConfig and adds additional parameters in optional with values in default. An error will result if a parameter in required is missing from config.

Author(s)

Stephanie Gogarten

Examples

```
f <- tempfile()
write.table(cbind(letters[1:10], 1:10), file=f, quote=FALSE,
            row.names=FALSE, col.names=FALSE)
config <- readConfig(f)

required <- letters[1:5]
optional <- letters[11:15]
default <- 11:15
config <- setConfigDefaults(config, required, optional, default)

unlink(f)
```

Index