Package ‘gQTLstats’

April 23, 2016

Title gQTLstats: computationally efficient analysis for eQTL and allied studies

Version 1.2.0

Author VJ Carey <stvjc@channing.harvard.edu>

Description computationally efficient analysis of eQTL, mQTL, dsQTL, etc.

Suggests geuvPack, geuvStore, Rsamtools, knitr, rmarkdown, ggbio, BiocStyle, Homo.sapiens

Depends R (>= 3.1.0)

Imports methods, snpStats, BiocGenerics, S4Vectors, IRanges, GenomeInfoDb, GenomicRanges, SummarizedExperiment, VariantAnnotation, Biobase, BatchJobs, gQTLBase, limma, gam, dplyr, AnnotationDbi, GenomicFeatures, ggplot2, reshape2, doParallel, foreach, ffbase

Maintainer VJ Carey <stvjc@channing.harvard.edu>

License Artistic-2.0

LazyLoad yes

VignetteBuilder knitr

BiocViews SNP, GenomeAnnotation, Genetics

NeedsCompilation no

R topics documented:

`gQTLstats-package` *gQTLstats: computationally efficient analysis for eQTL and allied studies*

Description

computationally efficient analysis of eQTL, mQTL, dsQTL, etc.

Details

The DESCRIPTION file:

Package:	gQTLstats
Title:	gQTLstats: computationally efficient analysis for eQTL and allied studies
Version:	1.2.0
Author:	VJ Carey <stvjc@channing.harvard.edu>
Description:	computationally efficient analysis of eQTL, mQTL, dsQTL, etc.
Suggests:	geuvPack, geuvStore, Rsamtools, knitr, rmarkdown, ggbio, BiocStyle, Homo.sapiens
Depends:	R (>= 3.1.0)
Imports:	methods, snpStats, BiocGenerics, S4Vectors, IRanges, GenomeInfoDb, GenomicRanges, SummarizedExp
Maintainer:	VJ Carey <stvjc@channing.harvard.edu>
License:	Artistic-2.0
LazyLoad:	yes
VignetteBuilder:	knitr
BiocViews:	SNP, GenomeAnnotation, Genetics

Index of help topics:

```
FDRsupp-class           Class '"FDRsupp"'
cisAssoc                test for variant-expression associations in
                        cis, using VCF
clipPCs                 transformations of expression data in smlSet
                        instances
directPlot              visualize relationship between empirical and
                        modeled FDR based on analysis of a gQTL store
enumerateByFDR          filter a ciseStore instance using an FDR
                        threshold
eqBox2                  visualization of expression or other assay
                        measure against genotypes extracted from VCF
```

`filtFDR`	illustration of FDRsupp class
`gQTLstats-package`	gQTLstats: computationally efficient analysis for eQTL and allied studies
`hmm878`	labeled GRanges with ChromHMM chromatin states for GM12878
`manhWngr`	manhattan plot with named GRanges
`queryVCF`	obtain SnpMatrix from VCF genotypes
`senstab`	create a plottable table for eQTL sensitivity analysis visualization
`setFDRfunc`	estimate and store function relating association scores to approximate plug-in FDR
`storeToQuantiles`	extract a vector from store results as ff (out of memory reference); support statistical reductions
`txsPlot`	visualize transformed FDR against transformed association statistics

This package addresses the management of map-reduce like computations for cis-association tests between DNA variants and genomic features like gene expression measurements. It makes essential use of data structures defined in package gQTLBase.

A number of experimental functions are present in the current version of the package: prep.cisAssocNB (assembles information to assess negative binomial regression in cis association testing), storeToMaxAssocBySNP (progress towards SNP-specific FDR), table_sensobj_thresh (reporting on sensitivity analysis).

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Maintainer: VJ Carey <stvjc@channing.harvard.edu>

`cisAssoc` *test for variant-expression associations in cis, using VCF*

Description

test for variant-expression associations in cis, using VCF and RangedSummarizedExperiment representations

Usage

```
cisAssoc(summex, vcf.tf, rhs = ~1, nperm = 3, cisradius =
                 50000, genome = "hg19", assayind = 1, lbmaf = 1e-06,
                 lbgtf = 1e-06, dropUnivHet = TRUE, infoFields =
                 c("LDAF", "SVTYPE"), simpleSNV = TRUE)
cisCount(summex, vcf.tf, rhs = ~1, cisradius =
                 50000, genome = "hg19", assayind = 1, lbmaf = 1e-06,
                 lbgtf = 1e-06, dropUnivHet = TRUE, infoFields =
                 c("LDAF", "SVTYPE"), simpleSNV = TRUE)
```

Arguments

`summex`	a `RangedSummarizedExperiment` object
`vcf.tf`	instance of `TabixFile`, referring to a tabix-indexed, bgzipped VCF file
`rhs`	formula 'right hand side' for adjustments to be made as `snp.rhs.tests` is run on each expression vector
`nperm`	number of permutations to be used for plug-in FDR computation
`cisradius`	distance in bp around each gene body to be searched for SNP association
`genome`	tag suitable for use in GenomeInfoDb structures
`assayind`	index of `assays(summex)` to use for expression data retrieval
`lbmaf`	lower bound on MAF of SNP to retain for analysis, computed using `col.summary`
`lbgtf`	lower bound on genotype frequency of SNP to retain for analysis
`dropUnivHet`	logical, if TRUE, will check for columns of SnpMatrix instance that possess no values other than "NA" and "A/B". See `http://www.biostars.org/p/117155/#117270`
`infoFields`	character – VCF fields to retain in vcfInfo() part of query
`simpleSNV`	logical – will use simple computation of `isSNV` to filter variants for analysis to SNV

Details

`snp.rhs.tests` is the workhorse for statistical modeling. VCF content is transformed to the byte-code (which allows for uncertain imputation) and used in fast testing.

Value

cisAssoc: a `GRanges-class` instance with mcols including chisq, permScore...

cisCount: enumerate locations in VCF that would be tested

Note

seqlevelsStyle for summex and vcf.tf content must agree

Author(s)

VJ Carey <stvjd@channing.harvard.edu>

Examples

```
require(GenomeInfoDb)
require(geuvPack)
require(Rsamtools)
data(geuFPKM)
lgeu = geuFPKM[ which(seqnames(geuFPKM)=="chr20"), ]
seqlevelsStyle(lgeu) = "NCBI"
tf20 = TabixFile(system.file("vcf/c20exch.vcf.gz", package="GGtools"))
if (require(VariantAnnotation)) scanVcfHeader(tf20)
```

```
 lgeue = clipPCs(lgeu[,which(lgeu$popcode=="CEU")], 1:2)
 set.seed(1234)
 litc = cisAssoc(lgeue[c(162,201),], tf20, nperm=2, lbmaf=.05, cisradius=50000)
 summary(litc$chisq)
# \dontrun{
 litc$pifdr = gQTLstats:::pifdr(litc$chisq, c(litc$permScore_1, litc$permScore_2))
 litc[which(litc$pifdr < .01)]
# }
```

clipPCs *transformations of expression data in smlSet instances*

Description

transformations of expression data in smlSet instances or assay data in RangedSummarizedExperiment

Usage

```
clipPCs(x, inds2drop, center = TRUE)

regressOut(x, rhs, ...)
```

Arguments

`x`	a `RangedSummarizedExperiment` object
`inds2drop`	Vector of PCs to be eliminated by setting the associated diagonal elements in the SVD to zero before recomposing the matrix of expression values. If the value 0 is present in inds2drop, the smlSet is returned unchanged, with a message.
`center`	logical, passed to `prcomp`
`rhs`	formula fragment (no dependent variable) used to form residuals in a reexpression of the expression matrix; variable bindings found in pData of an ExpressionSet or colData of a RangedSummarizedExperiment
`...`	arguments passed to `lmFit`

Details

`clipPCs` is an operation on the n x p transposed matrix X of expression data. The singular value decomposition X = UDV^t is formed, the diagonal elements of D corresponding to `inds2drop` are set to zero yielding the diagonal matrix E, and then Y = UEV^t is computed and transposed to replace the expression data.

`regressOut` obtains residuals after genewise regression of expression on the design matrix specified by the `rhs`; `lmFit` is used to compute coefficients, linear predictions and residuals.

Value

a `RangedSummarizedExperiment` object

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

References

The use of PCA-based adjustments to remove mass extraneous effects from expression matrices has been criticized in work of Oliver Stegle and Jeffrey Leek, who offer Bayesian PEER and SVA respectively as alternative solutions.

Examples

```
if(require(geuvPack)){
 data(geuFPKM)
 cg = clipPCs(geuFPKM, 1:10)
 ro = regressOut(cg, ~popcode)
 ro
 }
```

`directPlot` *visualize relationship between empirical and modeled FDR based on analysis of a gQTL store*

Description

visualize relationship between empirical and modeled FDR based on analysis of a gQTL store

Usage

```
directPlot(FDRsupp)
```

Arguments

`FDRsupp`	instance of `FDRsupp-class`

Details

This plot is used to show the degree of fit between a smooth model relating modeled FDR to empirical FDR, and the empirical FDR themselves. It should be used in conjunction with `txsPlot`.

It is possible for an implausible squiggly model to yield perfect agreement for all empirical FDR estimates. See the example.

Examples

```
data(filtFDR)
directPlot(filtFDR)
```

`enumerateByFDR` *filter a ciseStore instance using an FDR threshold*

Description

filter a ciseStore instance using an FDR threshold

Usage

```
enumerateByFDR(store, fdrsupp, threshold = 0.05, filter=force,
   ids=NULL, trimToUnit=TRUE)
```

Arguments

`store`	instance of `ciseStore-class`
`fdrsupp`	instance of `FDRsupp-class`
`threshold`	upper bound on FDR to be included
`filter`	The FDR can be computed for any association score. To return only records satisfying a given filter, supply the filter function here. It may be desirable to carry a filter function from the storeToFDR stage, and this may be considered in future versions.
`ids`	if NULL, process all results in store, otherwise limit attention to jobs with id values in `ids`
`trimToUnit`	plug-in FDR estimates can sometimes lie outside [0,1] owing to sparsity or defects of extrapolation; if this parameter is TRUE, estimated FDR values outside [0,1] are moved to the nearest boundary

Details

uses `storeApply`, which will use BiocParallel infrastructure when available

Value

A GRanges instance with store contents to which `estFDR` is appended for each range. The estFDR quantity is predicted using the GAM model held in the FDRsupp instance.

Examples

```
require(geuvStore)
require(gQTLBase)
st = makeGeuvStore()
data(filtFDR)
filtEnum = enumerateByFDR( st, filtFDR,
  filter=function(x)x[which(x$mindist <= 500000 & x$MAF >= 0.05)] )
names(metadata(filtEnum))
filtEnum[order(filtEnum$chisq, decreasing=TRUE)[1:2]]
```

eqBox2 *visualization of expression or other assay measure against genotypes extracted from VCF*

Description

visualization of expression or other assay measure against genotypes extracted from VCF

Usage

```
eqBox2(gene, se, tf, snpgr, genome = "hg19", ...)
eqDesc2(gene, se, tf, snpgr, genome = "hg19")
```

Arguments

`gene`	an element of rownames(se) from which a vector of assay values will be created
`se`	a `RangedSummarizedExperiment` object
`tf`	instance of class `TabixFile-class`, defining paths to a tabix-indexed VCF and index file
`snpgr`	instance of `GRanges-class` identifying the SNP to be visualized
`genome`	tag identifying reference genome
`...`	extra arguments passed to boxplot

Examples

```
require(Rsamtools)
require(SummarizedExperiment)
mygr = GRanges("1", IRanges(54683925, width=1))
gene = "ENSG00000231581.1"
library(geuvPack)
data(geuFPKM)
#tf = gtpath(1)
tf = TabixFile(system.file("vcf/small_1.vcf.gz", package="gQTLstats"))
eqBox2(gene, se=geuFPKM, tf, mygr )
eqDesc2(gene, se=geuFPKM, tf, mygr )
```

FDRsupp-class *Class* "FDRsupp"

Description

Support for FDR computations with ciseStore instances

Objects from the Class

Objects can be created by calls of the form `new("FDRsupp", ...)`.

Slots

`tab`**:** Object of class `"data.frame"` a table with association scores and plug-in FDR estimates evaluated on selected score values

`FDRfunc`**:** Object of class `"function"` a function of one argument with input association score and output the corresponding FDR estimate

`FDRmodel`**:** Object of class `"gam"` that was fit to elements of `tab`

`filterUsed`**:** Object of class `"function"` a copy of the function used for filtering the store to create the FDRfunc element.

`sessinfo`**:** sessionInfo() value at time of construction

`theCall`**:** instance of class `"call"` showing call leading to construction

Methods

getFDRfunc `signature(x = "FDRsupp")`: extract the FDR approximating function, a function of one (vector) argument assumed to represent association scores, evaluating to the plug-in FDR estimates corresponding to these scores

getTab `signature(x = "FDRsupp")`: extract the table of association scores and empirical FDR estimates

Note

Typically the `FDRfunc` function is constructed using a smooth model relating the estimated FDR to association scores.

Examples

```
showClass("FDRsupp")
```

`filtFDR` *illustration of FDRsupp class*

Description

illustration of FDRsupp class

Usage

```
data("filtFDR")
```

Format

A FDRsupp object.

Details

filtFDR was constructed on geuvStore contents, filtering to MAF at least five percent and radius at most 500kbp. rawFDR uses the entire geuvStore contents, with 1Mbp radius and 1 percent MAF lower bound

Examples

```
data(filtFDR)
filtFDR
```

---

hmm878 *labeled GRanges with ChromHMM chromatin states for GM12878*

Description

labeled GRanges with ChromHMM chromatin states for GM12878

Usage

```
data(hmm878)
```

Format

The format is:
Formal class ’GRanges’ [package "GenomicRanges"] with 6 slots
..@ seqnames :Formal class ’Rle’ [package "IRanges"] with 4 slots
.. .. ..@ values : Factor w/ 23 levels "chr1","chr2",..: 1 2 3 4 5 6 7 8 9 10 ...
.. .. ..@ lengths : int [1:23] 54467 46499 37617 25155 30071 34846 29420 24506 24123 27263 ...
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata : list()
..@ ranges :Formal class ’IRanges’ [package "IRanges"] with 6 slots
.. .. ..@ start : int [1:571339] 10001 10601 11138 11738 11938 12138 14538 20338 22138 22938 ...
.. .. ..@ width : int [1:571339] 600 537 600 200 200 2400 5800 1800 800 4000 ...
.. .. ..@ NAMES : NULL
.. .. ..@ elementType : chr "integer"
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata : list()
..@ strand :Formal class ’Rle’ [package "IRanges"] with 4 slots
.. .. ..@ values : Factor w/ 3 levels "+","-","*": 3
.. .. ..@ lengths : int 571339
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata : list()
..@ elementMetadata:Formal class ’DataFrame’ [package "IRanges"] with 6 slots
.. .. ..@ rownames : NULL
.. .. ..@ nrows : int 571339
.. .. ..@ listData :List of 4

.. ..$ name : chr [1:571339] "15_Repetitive/CNV" "13_Heterochrom/lo" "8_Insulator" "11_Weak_Txn" ...
.. ..$ score : num [1:571339] 0 0 0 0 0 0 0 0 0 0 ...
.. ..$ itemRgb: chr [1:571339] "#F5F5F5" "#F5F5F5" "#0ABEFE" "#99FF66" ...
.. ..$ thick :Formal class 'IRanges' [package "IRanges"] with 6 slots
.. ..@ start : int [1:571339] 10001 10601 11138 11738 11938 12138 14538 20338 22138 22938 ...
.. ..@ width : int [1:571339] 600 537 600 200 200 2400 5800 1800 800 4000 ...
.. ..@ NAMES : NULL
.. ..@ elementType : chr "integer"
.. ..@ elementMetadata: NULL
.. ..@ metadata : list()
.. .. ..@ elementType : chr "ANY"
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata : list()
..@ seqinfo :Formal class 'Seqinfo' [package "GenomicRanges"] with 4 slots
.. .. ..@ seqnames : chr [1:23] "chr1" "chr2" "chr3" "chr4" ...
.. .. ..@ seqlengths : int [1:23] 249250621 243199373 198022430 191154276 180915260 171115067 159138663 146364022 141213431 135534747 ...
.. .. ..@ is_circular: logi [1:23] FALSE FALSE FALSE FALSE FALSE FALSE ...
.. .. ..@ genome : chr [1:23] "hg19" "hg19" "hg19" "hg19" ...
..@ metadata :List of 1
.. ..$ url: chr "http://genome.ucsc.edu/cgi-bin/hgFileUi?g=wgEncodeBroadHmm&db=hg19"

Details

acquired using rtracklayer import from the bed file given at metadata(hmm878)[["url"]]

Source

see details

References

Ernst J, Kellis M. Discovery and characterization of chromatin states for systematic annotation of the human genome. Nat Biotechnol. 2010 Aug;28(8):817-25.

Ernst J, Kheradpour P, Mikkelsen TS, Shoresh N, Ward LD, Epstein CB, Zhang X, Wang L, Issner R, Coyne M et al. Mapping and analysis of chromatin state dynamics in nine human cell types. Nature. 2011 May 5;473(7345):43-9.

Examples

```
data(hmm878)
table(hmm878$name)
```

`manhWngr` *manhattan plot with named GRanges*

Description

manhattan plot with named GRanges

Usage

```
manhWngr(store, probeid = "ENSG00000183814.10", sym = "LIN9", fdrsupp, namedGR, slstyle = "NCBI", xlab
```

Arguments

`store`	instance of `ciseStore-class`
`probeid`	name of feature identifier to use for cis association
`sym`	symbol for feature identifier
`fdrsupp`	instance of `FDRsupp-class`
`namedGR`	`GRanges` instance with 'name' in mcols element
`slstyle`	seqlevelsStyle
`xlab.in`	x axis label
`ylab.in`	y axis label
`applyFDRfilter`	if TRUE, use the filter defined in the `filterUsed` element of the object supplied as `fdrsupp` on the output
`...`	additional arguments for plotting

Examples

```
require(geuvStore)
require(gQTLBase)
store = makeGeuvStore()
data(hmm878)
data(filtFDR)
manhWngr(store, fdrsupp=filtFDR, namedGR=hmm878)
```

`queryVCF` *obtain SnpMatrix from VCF genotypes*

Description

obtain SnpMatrix from VCF genotypes

Usage

```
queryVCF(gr, vcf.tf, samps, genome = "hg19", getSM = TRUE)
```

Arguments

`gr`	GRanges instance; SNPs lying within will be processed
`vcf.tf`	TabixFile instance pointing to VCF
`samps`	samples to be retained
`genome`	tag identifying build
`getSM`	logical; if FALSE, `genotypeToSnpMatrix` will not be run and only the output of readVcf is returned.

Value

a list of length two

`readout`	output of readVcf
`sm`	output of genotypeToSnpMatrix run on the read result

Examples

```
require(Rsamtools)
tf20 = TabixFile(system.file("vcf/c20exch.vcf.gz", package="GGtools"))
require(geuvPack)
data(geuFPKM)
lgeu = geuFPKM[ which(seqnames(geuFPKM)=="chr20"),
    which(geuFPKM$popcode=="CEU") ]
seqlevelsStyle(lgeu) = "NCBI"
rng = rowRanges(lgeu)[232] # CPNE1
myq = queryVCF( rng, tf20, samps=colnames(lgeu), genome="hg19" )
myq
```

senstab *create a plottable table for eQTL sensitivity analysis visualization*

Description

create a plottable table for eQTL sensitivity analysis visualization

Usage

```
senstab(x, filt = force)
## S3 method for class 'senstab'
plot(x, ...)
```

Arguments

`x`	a list generated by a process analogous to the sensitivity survey exhibited in the example below
`filt`	a function that operates on and returns a data.frame; typically will select rows based on values of fields 'MAF' and 'radius'
`...`	extra arguments passed to plot

Details

`sensByProbe` is a list structure; for information on this and other elements of sensitivity analysis workflow, see extensive non-executed code in example below

Value

an instance of the S3 class 'senstab', 'data.frame'

Examples

```
## Not run:
#
# illustration of sensitivity analysis using BatchJobs
#
# assume the following content in 'parms.R' (uncommented)
# MAFS = c(.03, .04, .05, .075, .10, .125, .15)
# dists = c(5000, 7500, 10000, 15000, 20000,
#    25000, 50000, 100000, 250000, 500000, 750000, 1000000)
# parms = expand.grid(MAFS, dists)
library(BatchJobs)  # for bigStore manip
library(gQTLstats)

# could use multilevel parallelism here
# because it is a somewhat large, fragile job, BatchJobs
# is a relevant tool for iteration.  but storeToFDRByProbe is
# already using bplapply.  so register 3 cores for it
# and specify 15 cpu for BatchJobs in .BatchJobs.R

sens1 = makeRegistry("sens1", file.dir="sens1",
    packages=c("gQTLstats", "dplyr"),
    src.files="parms.R")  # note parms.R

sens4One = function(z) {
      load("../bigStore.rda")  # get a ciseStore instance
      ans = storeToFDRByProbe(bigStore, xprobs=seq(.01,.99,.01), # xprobs
                                           # needs to be chosen with care
      filter=function(x) x[which(x$MAF >= parms[z,1] &
          x$mindist <= parms[z,2])])
      ans = setFDRfunc(ans, span=.35)  # span can be important
      list(fdrsupp=ans, parms=parms[z,])
 }

batchMap(sens1, sens4One, 1:nrow(parms))
submitJobs(sens1)

# now loadResult(sens1) or the equivalent can be the input to senstab()
# as in the example to continue here:

## End(Not run)
library(gQTLstats)
data(sensByProbe)
ptab = t(sapply(sensByProbe, function(x)as.numeric(x[[2]])))
unique(ptab[,1]) # MAFs used
unique(ptab[,2]) # radii used
# here we filter away some extreme values of the design space
tab = senstab(sensByProbe, filt=function(x) {
    x[ x$radius > 10000 & x$ radius < 500000 & x$MAF > .03, ]
    } )
plot(tab)
```

---

setFDRfunc — *estimate and store function relating association scores to approximate plug-in FDR*

Description

estimate and store function relating association scores to approximate plug-in FDR

Usage

```
setFDRfunc(FDRsupp, fudge = 1e-06, zthresh = 30, ...)
```

Arguments

FDRsupp	instance of FDRsupp-class

`fudge`	if FDR is zero, a log or logistic transform will fail; we add the small positive number `fudge` to avoid this
`zthresh`	for association scores greater than this value, a hard value of FDR 0 is assigned
`...`	arguments passed to `lo` for the smooth model relating association score to FDR at selected quantiles of the association score distribution

Value

returns an updated `FDRsupp-class` instance

Examples

```
data(filtFDR)
filtFDR2 = setFDRfunc(filtFDR)
```

`storeToStats` *extract a vector from store results as ff (out of memory reference); support statistical reductions*

Description

extract a vector from store results as ff (out of memory reference); support statistical reductions

Usage

```
storeToQuantiles(store, field,
    probs=c(seq(0,.999,.001), 1-(c(1e-4,1e-6,1e-6,1e-7))),
    ids = NULL, ..., checkField = FALSE, filter=force)
storeToHist(store, getter = function(x)
             as.numeric(S4Vectors::as.matrix(mcols(x)[,
             grep("permScore", names(mcols(x)))])), breaks, ids =
             NULL, filter = force)
storeToFDR(store, xprobs = c(seq(0, 0.999, 0.001), 1 - (c(1e-04,
             1e-06, 1e-06, 1e-07))), xfield = "chisq", getter =
             function(x) as.numeric(S4Vectors::as.matrix(mcols(x)[,
             grep("permScore", names(mcols(x)))])), filter = force)
```

Arguments

`store`	instance of `ciseStore-class`
`field`	character tag, length one, must be name of a numeric field in the result set (typically something like 'chisq' in the GRanges generated by cisAssoc)
`xfield`	as `field`, for FDR computation, see Details.
`ids`	job ids to be used; if NULL, process all jobs
`breaks`	boundaries of histogram bins

`...`	supplied to makeRegistry for a temporary registry: typically will be a vector of package names if additional packages are needed to process results
`checkField`	if TRUE steps will be taken to verify that the tag to which 'field' evaluates is present in result in the first job
`probs`	numeric vector of probabilities with values in [0,1]. See `quantile.ff`.
`xprobs`	percentiles of the empirical distribution of the association statistic at which FDR estimates are recorded.
`getter`	function of a single argument that extracts a numeric vector of association scores obtained under permutation
`x`	instance of FDRsupp
`filter`	function accepting and returning GRanges instance, executed when cisAssoc result is loaded to modify that result, defaults to no-op

Details

uses current BatchJobs configuration to parallelize extraction; reduceResults could be used for a sequential solution

Value

storeToQuantiles and storeToHist return objects analogous to those returned by stats::quantile and graphics::hist.

However, it should be noted that storeToQuantiles will use the `quantile.ff` of ffbase. For vectors of modest length, this can disagree with results of base::quantile by a few percent.

storeToFDR and storeToFDRByProbe return an instance of FDRsupp class

Note

uses ffbase:::c.ff explicitly to concatenate outputs; there is no guarantee of order among elements

Examples

```
stopifnot(require(geuvStore))
require(BatchJobs)
require(gQTLBase)
store = makeGeuvStore()
library(doParallel)
if (.Platform$OS.type == "windows") {
  registerDoSEQ()
  } else registerDoParallel(cores=max(c(detectCores()-1,1)))
smchisq = storeToFf( store, "chisq", ids=store@validJobs[1:3])
smchisq
qs = storeToQuantiles( store, "chisq", ids = store@validJobs[1:5],
  probs=seq(.1,.9,.1) )
qs
hh = storeToHist( store, ids = store@validJobs[1:5], breaks=
  c(0,qs,1e9) )
hh$counts
```

```
fd = storeToFDR( store, xprobs=c(seq(.05,.95,.05),.99,.999) )
tail(getTab(fd),4)
sss = storeToFDRByProbe( store , xprobs=c(seq(.05,.95,.05),.99) )
tail(getTab(sss),4)
```

`txsPlot` *visualize transformed FDR against transformed association statistics*

Description

visualize transformed FDR against transformed association statistics

Usage

```
txsPlot(FDRsupp)
```

Arguments

`FDRsupp`	an instance of `FDRsupp-class`

Examples

```
data(filtFDR)
txsPlot(filtFDR)
```

Index