Package ‘gQTLstats’

April 25, 2017

Title gQTLstats: computationally efficient analysis for eQTL and allied studies

Version 1.8.0

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

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

Suggests geuvPack, geuvStore2, Rsamtools, knitr, rmarkdown, ggbio,
BiocStyle, Homo.sapiens, RUnit, multtest, gwcat

Depends R (>= 3.1.0)

Imports methods, snpStats, BiocGenerics, S4Vectors (>= 0.9.25),
IRanges, GenomeInfoDb, GenomicFiles, GenomicRanges,
SummarizedExperiment, VariantAnnotation, Biobase, BatchJobs,
gQTLBase, limma, mcv, dplyr, AnnotationDbi, GenomicFeatures,
ggplot2, reshape2, doParallel, foreach, ffbase, BBmisc,
beeswarm, HardyWeinberg, graphics, stats, utils, shiny,
ldBlock, plotly, erma

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
- cisAssoc
- clipPCs
- directPlot
- enumerateByFDR
- eqBox2
- FDRsupp-class
- filtFDR
- gQTLs
- hmm878
- manhWngr
- mixedVCFtoSnpMatrix
gQTLstats-package

pifdr .......................................................... 16
qqStore ........................................................ 17
queryVCF ....................................................... 18
senstab ........................................................ 19
setFDRfunc ..................................................... 20
storeToStats ................................................... 21
tqbrowser ....................................................... 23
transAssoc ...................................................... 25
transBrowse .................................................... 26
TransStore ....................................................... 27
TransStore-class ............................................... 28
tsByRank ....................................................... 28
txsPlot ........................................................ 29

Index 31

Index of help topics:

FDRsupp-class Class "FDRsupp"
TransStore Instance constructor for managing trans gQTL results
TransStore-class Class "TransStore"
cisAssoc test for variant-expression associations in cis or generally, using VCF
clipPCs transformations of expression data in smlSet

Description

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

Details

The DESCRIPTION file:

Package: gQTLstats
Title: gQTLstats: computationally efficient analysis for eQTL and allied studies
Version: 1.8.0
Author: VJ Carey <stvjc@channing.harvard.edu>
Description: computationally efficient analysis of eQTL, mQTL, dsQTL, etc.
Suggests: geuvPack, geuvStore2, Rsamtools, knitr, rmarkdown, ggbio, BiocStyle, Homo.sapiens, RUnit, multtest
Depends: R (>= 3.1.0)
Imports: methods, snpStats, BiocGenerics, S4Vectors (>= 0.9.25), IRanges, GenomeInfoDb, GenomicFiles, GenomicRanges, ... doParallel, foreach, ffbase, BBmisc, beeswarm, HardyWeinberg, graphics, stats, utils, shiny, ldblock, plotly, erma
Maintainer: VJ Carey <stvjc@channing.harvard.edu>
License: Artistic-2.0
LazyLoad: yes
VignetteBuilder: knitr
BiocViews: SNP, GenomeAnnotation, Genetics
instances
directPlot visualize relationship between empirical and
directed FDR based on analysis of a gQTL store
enumerateByFDR filter a cisStore instance using an FDR
threshold
eqBox2 visualization of expression or other assay
measure against genotypes extracted from VCF
filtFDR illustration of FDRsupp class
gQTLs use SummarizedExperiment to manage a collection
of gQTL results of interest
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
mixedVCFtoSnpMatrix amalgamate called genotypes and imputed allelic
dosages in VCF to SnpMatrix representation
pifdr utility for computing plug-in FDR
qqStore create a binned QQplot for a sharded store
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
tqbrowser general browsing facility for trans-gQTL
transAssoc compute 'trans' SNP-feature associations by
wrapping AllAssoc
transBrowse shiny app to exhibit genotype:genomic feature
distributions
tsByRank harvest contents of a TransStore by rank in
associations of features to SNP
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
(associates information to assess negative binomial regression in cis association testing), storeToMaxAssocBySNP (progress towards SNP-specific FDR), table_sensobj_thresh (reporting on sen-
sitivity analysis).

Additional experimental functions are available to support scalable trans-gQTL testing TransChunk,
filteredDFwPerm, and transTable operate on output of AllAssoc.

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

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

test for variant-expression associations in cis or generally, using VCF

Description

test for variant-expression associations in cis or generally, 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)
cisEsts(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)
AllAssoc(summex, vcf.tf, variantRange, rhs = ~1, nperm = 3,
  genome = "hg19", assayind = 1, lbmaf = 1e-06, lbgtf = 1e-06,
  dropUnivHet = TRUE, infoFields = c("LDAF", "SVTYPE"))

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
variantRange GRanges instance that defines the scope of the VCF to be used for testing against
  all features on summex
cisAssoc

Details

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

distToGene is a helper function that should be replaced with something from the Bioconductor annotation subsystem

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

```r
require(GenomeInfoDb)
require(geuvPack)
require(Rsamtools)
#
# obtain geuvadis expression measures as FPKM
##
data(geuFPKM)
#
# confine the chromosome 20
##
lgeu = geuFPKM[ which(seqnames(geuFPKM)=="chr20"), ]
seqlevelsStyle(lgeu) = "NCBI"
#
# acquire subset of genotypes on chr20
##
tf20 = TabixFile(system.file("vcf/c20exch.vcf.gz", package="gQTLstats"))
if (require(VariantAnnotation)) scanVcfHeader(tf20)
#
# perform a general technical confounder correction, and confine
# attention to CEU samples
##
lgeue = clipPCs(lgeu[,which(lgeu$popcode=="CEU"),], 1:2)
#
# obtain all score test statistics for SNP:gene pairs at radius 50k
##
set.seed(1234)
litc = cisAssoc(lgeue[c(162,201),], tf20, nperm=2, lbmaf=.05, cisradius=50000)
#
# obtain all estimates for SNP:gene pairs at radius 50k
##
set.seed(1234)
lite = cisEsts(lgeue[c(162,201),], tf20, nperm=2, lbmaf=.05, cisradius=50000)
summary(litc$chisq)
mysr = range(litc)
```
# compute the plug-in FDR
#
# litc$pifdr = gQTLstats:::pifdr(litc$chisq, c(litc$permScore_1, litc$permScore_2))
#
# trans association testing. leave to the user the question of
# whether a test is actually cis
#
# lita = AllAssoc(geuFPKM[1:10,], tf20, mysr)
# lita3 = AllAssoc(geuFPKM[11:20,], tf20, mysr)
# lita5 = AllAssoc(geuFPKM[21:30,], tf20, mysr)
#
# This retains the top 5 (default) associations per SNP
#
# n1 = gQTLstats:::collapseToBuf(lita, lita3)
# n1 = collapseToBuf(n1, lita5)

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 gene-wise 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**

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

```
directPlot(FDRsupp)
```

**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**

```r
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

## Not run:
require(geuvStore2)
require(gQTLBase)
st = makeGeuvStore2()
data(filter)
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]]

## End(Not run) # not really essential
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

```r
eqBox2(gene, se, tf, snpgr, genome = "hg19", forceRs = TRUE, ...)
eqDesc2(gene, se, tf, snpgr, genome = "hg19", forceRs = TRUE)
eqBox3(gene, se, tf, snpgr, geneAnno, genome = "hg19", forceRs = TRUE, ...)
```

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.
- `forceRs`: In the 1000 genomes VCF, there are sometimes variants identified with DELLY that are grabbed by `readVcf` on an SNV address. Set `forceRs` to `TRUE` to retain only variants with 'rs' in the name. Has no effect if `readVcf` extracts only a single variant.
- `geneAnno`: named vector, `geneAnno[gene]` will be used to annotate display.
- `...`: extra arguments passed to `beeswarm`.

Details

In 1.5.4, altered to supply beeswarm data visualization in addition to boxplot. Use additional option `corral="gutter"` to reduce horizontal sprawl in large samples.

Examples

```r
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)
```
**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**

```r
showClass("FDRsupp")
```
filtFDR

**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

gQTLs

**Description**

use SummarizedExperiment to manage a collection of gQTL results of interest

**Usage**

gQTLs(filtgr, se, tf, genome = "hg19", forceRs = TRUE, chunksize = 50)
gQTLswarm(se, ind, covar = NULL, inpch = 19, xlab, ylab, featTag="probeid", ...)

**Arguments**

filtgr  
a GRanges instance typically obtained by filtering a cisStore instance

se  
SummarizedExperiment with individual level expression and sample-level data from which filtgr statistics were derived; for gQTLswarm, output of gQTLs

tf  
TabixFile for VCF on which filtgr statistics are based

gene  
tag for readVcf

forceRs  
if TRUE insist that snp ids include 'rs'

chunksize  
VCF processing proceeds via foreach in chunks of size chunksize
ind  index into rows of se to be used for visualization, must be length 1

covar a character string indicating a variable in colData(se) to be used to color the points

inpch pch setting for dots in swarm

xlab xlabel for beeswarm plot, defaults to snp id as recovered from rowRanges(se)$snp

ylab ylabel for beeswarm plot, defaults to probe id as recovered from rowRanges(se)$probeid

featTag element of mcols(rowRanges(se)) used to find ylab text, defaults to ’probeid’. ’symbol’ is often preferred

... passed to beeswarm

Value

a SummarizedExperiment instance with two assays, the first is genotype the second is expression

Note

very preliminary

Examples

```
require(Rsamtools)
tf = TabixFile(system.file("vcf/litv.vcf.gz", package="gQTLstats"))
data(sigInlit) # 33 loci with significant cis eQTL on a specific filtering
library(geuvPack)
data(geuFPKM)
require(doParallel)
registerDoSEQ()
gdem = gQTLs(sigInlit, geuFPKM, tf, genome = "hg19")
gQTLswarm(gdem, 1, "popcode")
```

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

Source
see details

References
manhWngr


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.in, ylab.in, applyFDRfilter = TRUE)

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(geuvStore2)
require(gQTLBase)
store = makeGeuvStore2()
data(hmm878)
data(filtFDR)
manhWngr(store, fdrsupp=filtFDR, namedGR=hmm878)
mixedVCFtoSnpMatrix

amalgamate called genotypes and imputed allelic dosages in VCF to SnpMatrix representation

Description

amalgamate called genotypes and imputed allelic dosages in VCF to SnpMatrix representation

Usage

mixedVCFtoSnpMatrix(vcf, preferGT = TRUE)

Arguments

vcf object inheriting from CollapsedVCF-class
preferGT logical. VCF allows loci for samples to be reported in various formats, and a given locus can have a call tagged GT and a genotype probability or likelihood representation tagged GP or GL. genotypeToSnpMatrix has an uncertain parameter that, if TRUE, will transform GP or GL content to allelic dose. Note that only the "first" dosage type appearing in the header will be transformed. Thus if GP is first in the header but a given locus is tagged only with GL, the genotype for thus locus will be recorded as NA.

Details

emulates output from genotypeToSnpMatrix

Value

list with elements genotypes and map

Author(s)

VJ Carey

See Also

genoTypeToSnpMatrix

Examples

fn = system.file("vcf/polytypeSNV.vcf", package="gQTLstats")
require("VariantAnnotation")
require("snpStats")
vv = readVcf(fn, genome="hg19") # only 4th SNP will have dosage coding
mixedVCFtoSnpMatrix(vv)$genotypes@.Data
pifdr

utility for computing plug-in FDR

Description

utility for computing plug-in FDR

Usage

pifdr( obs, perms, trimToUnit = TRUE, ... )
pifdr2( obs, perms, trimToUnit = TRUE, expandPerms=TRUE, ... )

Arguments

  obs          observed association scores
  perms        vector of association scores under permutation; length should be integer multiple
                of length(obs)
  trimToUnit   logical, if TRUE, values greater than 1 are replaced by 1. Such values can occur, for
                example, with relatively small sample sizes.
  expandPerms  With certain pair-specific filtering operations, the number of scores obtained
                after permutation may not be a multiple of the number of observed scores. If
                TRUE, the scores obtained under permutation are sampled with replacement to
                simplify computation of plug-in FDR.
  ...          extra arguments ignored

Details

Revised 12/30/13 to employ hist() to rapidly bin the permuted values.

Value

vector of plug-in FDR estimates congruent to obs

References

Hastie Tibshirani and Friedman Elements of Statistical Learning ch 18.7

Examples

set.seed(1234)
op = par(no.readonly=TRUE)
par(mfrow=c(2,2))
X = c(rchisq(30000,1),rchisq(300,10))
Y = rchisq(30300*3,1)
qqplot(Y, X, xlab="null", ylab="observed")
hist(pp <- pifdr(X,Y), xlab="plug-in FDR", main=" ")
library(multtest)
rawp = 1-pchisq(X, 1)
MT <- mt.rawp2adjp(rawp)
MT2 = MT[[1]][order(MT[[2]]),]
plot(MT2[,"BH"], pp, xlab="BH FDR", ylab="plug-in FDR")
par(op)
create a binned QQplot for a sharded store with association and permutation statistics

Usage

```r
qqStore(st, ids = NULL,
    .probs = c(0, seq(0.6, 0.8, 0.2), 0.9, 0.95, 0.99, 0.999, 0.9999, 1),
    xlim.in = c(0.2, 75), lowfac = 0.5, xlab = "Permutation distribution",
    ylab = "Distribution of score statistic", countpos = 50,
    plot.it = TRUE, doab = TRUE, scoreField = "chisq",
    permField = "permScore_1", ...)  
```

Arguments

- `st` instance of `ciseStore-class`
- `ids` optional job id vector; if NULL, all jobs used
- `.probs` vector of probabilities for use with quantile evaluation, as provided in ffbase, using `storeToQuantiles`
- `xlim.in` xlim setting for QQplot
- `lowfac` we use a log-log plot, and the first quantile (as prescribed in `.probs`) is often close to zero; we reassign it to `lowfac*(second quantile)`
- `xlab` label
- `ylab` label
- `countpos` where on the x axis will we stack the information on bin counts
- `plot.it` logical, if FALSE, a list is returned with elements on quantile values and bin counts
- `doab` logical prescribing drawing of line of identity
- `scoreField` tag in store naming the statistic, typically 'chisq', can also be 'tstat' for GTEx
- `permField` tag in store naming the field holding statistics on realizations from permutation distribution
- `...` passed to `storeToQuantiles`

Value

invisibly returns list with elements qx, qy, counts, fracs

Examples

```r
## Not run:
library(geuvStore2)
library(gQTLBase)
gs = makeGeuvStore2()
qqStore(gs) #, ids=partialIds()[1:20])
## End(Not run)
```
queryVCF

obtain SnpMatrix from VCF genotypes

Description

obtain SnpMatrix from VCF genotypes

Usage

queryVCF(gr, vcf.tf, samps, genome = "hg19", getSM = TRUE, 
          snvOnly=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. |
| snvOnly  | logical, if TRUE, will confine results to SNV       |

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="gQTLstats"))
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
```
```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**

```r
setFDRfunc(FDRsupp, fudge = 1e-06, zthresh = 30, maxch = 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
- `maxch` the model for the functional relationship between association and FDR is subset to observations for which association chisq score is no greater than 1.1*maxch
- `...` arguments passed to `s` 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-5,1e-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-05, 1e-06, 1e-07))), xfield = "chisq", getter =  
function(x) as.numeric(S4Vectors::as.matrix(mcols(x)[,  
grep("permScore", names(mcols(x)))])), filter = force,  
.id4coln=1, ids=NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>store</td>
<td>instance of <code>ciseStore-class</code></td>
</tr>
<tr>
<td>field</td>
<td>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)</td>
</tr>
<tr>
<td>xfield</td>
<td>as field, for FDR computation, see Details.</td>
</tr>
<tr>
<td>ids</td>
<td>job ids to be used; if NULL, process all jobs</td>
</tr>
<tr>
<td>breaks</td>
<td>boundaries of histogram bins</td>
</tr>
<tr>
<td>...</td>
<td>supplied to makeRegistry for a temporary registry: typically will be a vector of package names if additional packages are needed to process results</td>
</tr>
<tr>
<td>checkField</td>
<td>if TRUE steps will be taken to verify that the tag to which ‘field’ evaluates is present in result in the first job</td>
</tr>
<tr>
<td>probs</td>
<td>numeric vector of probabilities with values in [0,1]. See <code>quantile.ff</code>.</td>
</tr>
<tr>
<td>xprobs</td>
<td>percentiles of the empirical distribution of the association statistic at which FDR estimates are recorded.</td>
</tr>
<tr>
<td>getter</td>
<td>function of a single argument that extracts a numeric vector of association scores obtained under permutation</td>
</tr>
</tbody>
</table>
storeToStats

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

.id4coln job id to be used for initial probe to determine names of fields in mcols of all jobs

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

## Not run:
stopifnot(require(geuvStore2))
require(BatchJobs)
require(gQTLBase)
store = makeGeuvStore2()
library(doParallel)
if (.Platform$OS.type == "windows") {
  registerDoSEQ()
} else registerDoParallel(cores=max(c(detectCores()-1,1)))
smchisq = storeToFf( store, "chisq", ids=store@validJobs[1:3])
smchisq
if (.Platform$OS.type != "windows") { # avoid timeout
  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)
}

## End(Not run)
tqbrowser

**general browsing facility for trans-gQTL**

**Description**

Provide a general browsing facility for trans-gQTL.

**Usage**

```
tqbrowser(mae, fn, gn, tiling, ts, tsa,
        an, band.init = "6q12", er, gs, ...
```

**Arguments**

- **mae**: Instance of `MultiAssayExperiment-class`
- **fn**: character naming the element of mae holding assay features
- **gn**: character naming the element of mae holding a `VcfStack-class` instance for genotypes
- **tiling**: a tiling of the genome used to partition large genotype resource
- **ts**: an instance of the output of `tsByRankAccum` that collects association statistics and metadata on general searches for genotype-feature association
- **an**: a named character vector mapping between identifiers used to identify features in `experiments(mae)[[fn]]` and tokens to be used in display – the names of an are the rownames to be translated to the associated value in the display.
- **band.init**: an initial tile selection
- **er**: instance of `ErmaSet-class`
- **gs**: instance of `gwaswloc-class`
- **...**: not currently used

**Details**

starts a shiny app

**Author(s)**

VJ Carey <stvjc@channing.harvard.edu>

**Examples**

```
if (interactive()) {
  oa = options()$example.ask
  options(example.ask=FALSE)
  #
  # this example assumes you have a working internet connection
  # it will collect genotype information from a S3 bucket
  # where 1000 genomes VCF resides
  #
  # obtain infrastructure
  #
```
message("note: as of Dec 17 2016 this function will trigger transient errors... ignore them") # solved with req()

suppressPackageStartupMessages(
  r = sapply(packs, require, character.only=TRUE)
)
stopifnot(all(r))

# use S3 bucket to get genotypes, create VcfStack wrapper
message("create VcfStack...")

tf17 = ldblock::s3_1kg("17")
vcst = VcfStack(c("17"=path(tf17)))
seqlevelsStyle(vcst) = "NCBI"

# obtain expression data for GEUVADIS samples
message("obtain expression data...")
if (!exists("geuFPKM")) data(geuFPKM)
data(gen2sym)
seqlevelsStyle(geuFPKM) = "NCBI"

# bind to MAE

el = ExperimentList(list(geu=geuFPKM, vcf=vcst))
message("create MultiAssayExperiment...")
suppressWarnings(" # samples don't line up between expression and genotype, we know this
mae = MultiAssayExperiment(el, pData=colData(el[[1]]))")

# obtain and clean up cytoband representation
# cyto37n created as follows:
# ah = AnnotationHub()
# cyto37 = ah[['AH5012']]  
# seqlevelsStyle(cyto37) = "NCBI"
# cyto37 = as(cyto37, "GRanges")
# sn = as.character(seqnames(cyto37))
# mcols(cyto37)$name = paste0(sn, mcols(cyto37)$name)
# names(cyto37) = mcols(cyto37)$name
# seqlengths(cyto37)["MT"] = 16569
message("obtain cytoband index...")
data(cyto37n)
data(tbgaOrmdl3) # saved output of tsByRankAccum, giving association scores

# obtain gwas catalog...
library(gwascat)
data(ebicat37)

# obtain chromatin state calls from erma
message("obtain chromatin state calls...")
erset = makeErmaSet()

# target and invoke browser
# okba = c("17q12", "17q21.1", "17q21.2")
on.exit(options(example.ask=oa))
transAssoc

compute 'trans' SNP-feature associations by wrapping AllAssoc

Description

compute 'trans' SNP-feature associations by wrapping AllAssoc, retaining only the strongest associations (and similarly filtered association scores computed under permutation)

Usage

transAssoc(variantGR, exSE, vcfgen, bufsize = 10, nperm = 3, exChLen = 2 * bufsize, ...)

Arguments

variantGR
  GRanges instance establishing scope of variants to test

exSE
  SummarizedExperiment instance, all of whose features will be tested for association with all SNP

vcfgen
  a function returning a path to a tabix-indexed VCF file from which SNP genotypes will be extracted

bufsize
  Size of 'buffer' used to retain largest feature association scores encountered during the search. The scores and the names of associated genes are retained in 'scorebuf' and 'elnames' components of output GRanges

nperm
  number of permutations of features against genotypes to be performed for realizing null distribution of association scores

exChLen
  size of chunks of exSE to be tested through calls to AllAssoc; this is intended to allow control of RAM usage

... arguments passed to AllAssoc

Value

a GRanges with mcols including

Examples

```r
## Not run: # requires access to 1KG S3
library(geuvPack)
data(geuFPKM)
seqlevelsStyle(geuFPKM) = "NCBI"
mysr = GRanges("20", IRanges(33000055, 33020055))
genome(mysr) = "hg19"
tt = transAssoc(mysr, geuFPKM[1:16,]
  bufsize=3, exChLen=4, vcfgen=function(x)gtpath(paste0("chr", x))
) colnames(mcols(tt))

table(as.character(mcols(tt)$elnames))
## End(Not run)
```
transBrowse

shiny app to exhibit genotype:genomic feature distributions

Description
exhibit genotype:genomic feature distributions with a shiny app

Usage
transBrowse(tbg, anno, tivcf, se, title = "trans eQTL")
transBrowse2(tbga, annovec, tivcf, se, title = "trans eQTL", maxrank=3)

Arguments
tbg         filtered output of tsByRankAccum, see example
tbga        filtered output of tsByRankAccum, see example
anno        a vector with 'feature symbols' (e.g., gene symbols) as values and 'feature names' (elements of rownames of se, e.g., ENSEMBLE gene ids) as names
annovec     a vector with 'feature symbols' (e.g., gene symbols) as values and 'feature names' (elements of rownames of se, e.g., ENSEMBLE gene ids) as names
tivcf       reference to Tabix-indexed VCF
se           SummarizedExperiment instance with rowname coincident with anno and tbg[["allfeats"]]
title       optional string for title panel
maxrank     transBrowse2 works with the tsByRankAccum function that collects scores down to a specified rank. This parameter specifies the boundary.

Details
This function is under development. The intention is to allow convenient visualization of off-chromosome genotype-feature relationships. AllAssoc collects association scores SNP-wise, and saves the largest "K" scores obtained, along with feature identity and location metadata. The largest score obtained for a given SNP is the rank 1 association, the next largest is rank 2, and so on.

Examples
## Not run:
# consider the following filtering utility
tbfilt = function(tbg, seqnames."17", minMAF=.1, minabsodist = 1e7, nrec=1000) {
  tbg = tbg[ which(as.character(seqnames(tbg)) %in% seqnames.) ]
  tbg = tbg[ which(tbg$MAF > minMAF & abs(tbg$obsdist) > minabsodist) ]
  tbg[ order(tbg$scores, decreasing=TRUE ) ][1:nrec]
}
#
registerDoSEQ()
library(geuvStore2)
rg = g17transRegistry()
g17 = TransStore(list(r17))
tbg = tbfilt(tsByRankAccum(g17, 3, mcol2keep="REF", "snp", "MAF")) # 1000 records
tf17 = ldblock::s3_1kg("17") # uses S3 bucket
require(geuvPack)
require(shiny)
if (!exists("geuFPKM")) data(geuFPKM)
if (!exists("gencodeV12")) data(gencodeV12)
data(gen2sym)
transBrowse2(tbg, gen2sym, tf17, geuFPKM, title="trans GEUV chr17")

## End(Not run) # end dontrun

---

**TransStore**

*Instance constructor for managing trans gQTL results*

**Description**

Instance constructor for managing trans gQTL results

**Usage**

`TransStore(regs, paths = NULL)`

`tsIndex.reg(tsin, ind)`

**Arguments**

- `regs` a list of `Registry` instances, typically one per (variant-oriented) chromosome
- `paths` if desired, paths to folders for which `loadRegistry` succeeds, used instead of `regs`
- `tsin` a `TransStore` instance
- `ind` index of registry to index

**Details**

tsIndex.reg is experimental, producing a hash mapping snps to registry job identifiers, to support rapid store-level retrieval of locus-specific findings.

**Value**

instance of `TransStore-class`

**Examples**

```r
if (require(geuvStore2) && require(doParallel)) {
  registerDoSEQ()
  r17 = g17transRegistry()
  r18 = g18transRegistry()
  g1718 = TransStore(list(r17, r18))
  g1718
}
```
**TransStore-class**  
*Class "TransStore"*

**Description**
Manage collection of related trans-gQTL results in BatchJobs registries, typically one per chromosome.

**Objects from the Class**
Objects can be created by calls of the form `new("TransStore", ...)`.

**Slots**
- `allRegistries`: Object of class "list" containing Registry instances
- `numSubmitted`: Object of class "numeric" records number of jobs submitted for each registry
- `numDone`: Object of class "numeric" records number of jobs completed for each registry
- `nloci`: Object of class "numeric" records number of loci with test results for each registry
- `jobinfos`: Object of class "list" records results of `getJobInfo` for each registry

**Methods**
- `describe` signature(`object = "TransStore"`): summarize information about a store

**Examples**
```r
showClass("TransStore")
```

---

**tsByRank**  
*harvest contents of a TransStore by rank in associations of features to SNP*

**Description**
Harvest contents of a TransStore by rank in associations of features to SNP.

**Usage**
```r
tsByRankAccum(tsin, maxrank = 3, mcol2keep=c("REF", "ALT", "snp", "MAF", "z.HWE"), filt=force)
```

**Arguments**
- `tsin`  
  An instance of `TransStore-class`
- `maxrank`  
  The maximum rank of association scores to retrieve, cumulatively. Each variant has been tested for association with each genomic feature (e.g., gene in a typical expression QTL study), but only the top ranking associations are recorded for each variant. If `maxrank=k`, for each variant, this function retrieves the features exhibiting the kth largest association recorded over all features, along with all k-1 larger association scores.
mcol2keep  a character vector of metadata columns to retain
filt  a function accepting a GRanges and returning a GRanges. The mcols of
the GRanges to be processed will have elements c(mcol2keep, "scorebuf", "elnames", "dist"),
where the latter two are matrices with number of columns equal to the bufsize
of the transAssoc call that generated ts. Only SNP-specific elements can be
used to define the filter.

Details

tsByRankAccum_sing and other functions with suffix _sing were developed for the case of a single
permutation
getTransRegistries() accesses objects packaged for demonstration purposes

Value

A GRanges instance.

Examples

if (require(doParallel)) {
  registerDoSEQ()
  lit = TransStore(getTransRegistries()) # very limited slice
  tbga = tsByRankAccum(lit, maxrank=5)
  plot(ecdf(as.numeric(data.matrix(tbga$permscoresByRank1))), ylim=c(.99,1),
       main="eCDF of permutation dist. of association, by variant rank")
  exr = paste0("permscoresByRank", 2:5)
  for (i in 1:4)
    lines(ecdf(as.numeric(data.matrix(mcols(tbga)[[exr[i]]]))), col=i+1)
  legend(200, .994, lty=1, col=1:5, legend=paste("rank", 1:5))
  plot(ecdf(as.numeric(data.matrix(tbga$permscoresByRank1[,1]))), ylim=c(.99,1),
       main="between-permutation variation")
  lines(ecdf(as.numeric(data.matrix(tbga$permscoresByRank1[,2]))), col=2)
  lines(ecdf(as.numeric(data.matrix(tbga$permscoresByRank1[,3]))), col=3)
  lines(ecdf(as.numeric(data.matrix(tbga$permscoresByRank1[,4]))), col=4)
  lines(ecdf(as.numeric(data.matrix(tbga$permscoresByRank1[,5]))), col=5)
  lines(ecdf(as.numeric(data.matrix(tbga$permscoresByRank1[,6]))), col=6)
  legend(200, .994, col=1:6, lty=1, legend=c("rank 1 (perm 1)", "(2)", "(3)",
          "rank 5 (perm 1)", "(2)", "(3)")
  # head(tbga,2)
  # consider the following filtering utility
  # tbfilter = function(tb, seqnames="17", minMAF=.1, minabsodist = 1e7,
  #   nrec=1000) {
  #    tb = tb[ which(as.character(seqnames(tb)) %in% seqnames.) ]
  #    tb = tb[ which(tb$MAF > minMAF) ]
  #    tb[ order(tb$scores, decreasing=TRUE )][1:nrec]
  #  }
}
Usage

```r
txsPlot(FDRsupp, xmax=50)
```

Arguments

- **FDRsupp**: an instance of `FDRsupp-class`
- **xmax**: upper bound on xlim for display

Examples

```r
data(filtFDR)
txsPlot(filtFDR)
```
Index

* Topic classes
  FDRsupp-class, 10
  TransStore-class, 28

* Topic datasets
  filtFDR, 11
  hmm878, 12

* Topic graphics
  directPlot, 7
  eqBox2, 9
  txsPlot, 29

* Topic hplot
  tqbrowser, 23

* Topic manip
  gQTLs, 11
  TransStore, 27

* Topic models
  cisAssoc, 4
  clipPCs, 6
  enumerateByFDR, 8
  manhWngr, 14
  mixedVCFtoSnpsMatrix, 15
  pifdr, 16
  qqStore, 17
  queryVCF, 18
  senstab, 19
  setFDRfunc, 20
  storeToStats, 21
  tqbrowser, 23
  transAssoc, 25
  transBrowse, 26
  tsByRank, 28

* Topic package
  gQTLstats-package, 2

AllAssoc (cisAssoc), 4
beeswarm, 12

cisAssoc, 4
cisCount (cisAssoc), 4
cisEsts (cisAssoc), 4
clipPCs, 6
clipPCs, RangedSummarizedExperiment, numeric, missing-method (clipPCs), 6
clipPCs, SummarizedExperiment, numeric, logical-method (clipPCs), 6
clipPCs, SummarizedExperiment, numeric, missing-method (clipPCs), 6
col.summary, 4
collapse_multiPerm (cisAssoc), 4
directPlot, 7
describe (TransStore-class), 28
describe,TransStore-method (TransStore-class), 28
directPlot, 7
distToGene (cisAssoc), 4
enumerateByFDR, 8
eqBox2, 9
eqBox3 (eqBox2), 9
eqDesc2 (eqBox2), 9
FDRsupp-class, 10
filteredDFwPerm (gQTLstats-package), 2
filtFDR, 11
genotypeToSnpsMatrix, 15, 18
getFDRfunc (FDRsupp-class), 10
getFDRfunc, FDRsupp-method (FDRsupp-class), 10
getJobInfo, 28
getTab (FDRsupp-class), 10
globTab, FDRsupp-method (FDRsupp-class), 10
getTransRegistries (tsByRank), 28
gQTLs, 11
gQTLstats (gQTLstats-package), 2
gQTLstats-package, 2
gQTLswarm (gQTLs), 11
GRanges, 14, 29

hmm878, 12

isSNV, 4

lmFit, 6

loadRegistry, 27
manhWngr, 14
mixedVCFtoSnpMatrix, 15
pifdr, 16
pifdr2 (pifdr), 16
pifdr3 (pifdr), 16
plot (senstab), 19
prcomp, 6
prep.cisAssocNB (gQTLstats-package), 2
qqStore, 17
quantile.ff, 21, 22
queryVCF, 18

RangedSummarizedExperiment, 4, 6, 7, 9
rawFDR (filtFDR), 11
readVcf, 11
Registry, 27, 28
regrEssOut (clip PCs), 6

s, 20
sensByProbe (senstab), 19
senstab, 19
setFDRfunc, 20
snp.rhs.tests, 4, 5
storeApply, 8
storeToFDR (storeToStats), 21
storeToFDRbyProbe (storeToStats), 21
storeToHist (storeToStats), 21
storeToMaxAssocBySNP
   (gQTLstats-package), 2
storeToQuantiles, 17
storeToQuantiles (storeToStats), 21
storeToStats, 21

TabixFile, 4
table_sensobj_thresh
   (gQTLstats-package), 2
tqbrowser, 23
transAssoc, 25, 29
transBrowse, 26
transBrowse2 (transBrowse), 26
TransChunk (gQTLstats-package), 2
TransChunk-class (gQTLstats-package), 2
TransStore, 27
TransStore-class, 28
transTable (gQTLstats-package), 2
tsByRank, 28
tsByRank_sing (tsByRank), 28
tsByRankAccum, 23, 26
tsByRankAccum (tsByRank), 28
tsByRankAccum_sing (tsByRank), 28
tsIndex.reg (TransStore), 27
txsPlot, 7, 29