Package ‘gQTLBase’

December 21, 2016

Title gQTLBase: infrastructure for eQTL, mQTL and similar studies
Version 1.6.0
Author VJ Carey <stvjc@channing.harvard.edu>
Description Infrastructure for eQTL, mQTL and similar studies.
Suggests geuvStore2, knitr, rmarkdown, BiocStyle, RUNit, GGtools, Homo.sapiens, IRanges, erma, GenomeInfoDb, gwascat, geuvPack
Imports GenomicRanges, methods, BatchJobs, BBmisc, S4Vectors, BiocGenerics, foreach, doParallel, bit, ff, rtracklayer, ffbase, GenomicFiles, SummarizedExperiment
Depends
Maintainer VJ Carey <stvjc@channing.harvard.edu>
License Artistic-2.0
LazyLoad yes
VignetteBuilder knitr
BiocViews SNP, GenomeAnnotation, Genetics, DataImport, FunctionalGenomics
NeedsCompilation no

R topics documented:

gQTLBase-package ...................................................... 2
ciseStore-class ...................................................... 3
describeStore .......................................................... 4
extractByProbes ...................................................... 5
mergeCIstates .......................................................... 6
storeApply .............................................................. 7
storeMapResults ...................................................... 8
storeToFf ................................................................. 10
ufeatByTiling .......................................................... 11

Index 12
gQTLBase-package

Description

Infrastructure for eQTL, mQTL and similar studies.

Details

The DESCRIPTION file:

Package: gQTLBase
Title: gQTLBase: infrastructure for eQTL, mQTL and similar studies
Version: 1.6.0
Author: VJ Carey <stvjc@channing.harvard.edu>
Description: Infrastructure for eQTL, mQTL and similar studies.
Suggests: geuvStore2, knitr, markdown, BiocStyle, RUnit, GGtools, Homo.sapiens, IRanges, erma, GenomeInfoDb
Imports: GenomicRanges, methods, BatchJobs, BBmisc, S4Vectors, BiocGenerics, foreach, doParallel, bit, ff
Depends:
Maintainer: VJ Carey <stvjc@channing.harvard.edu>
License: Artistic-2.0
LazyLoad: yes
VignetteBuilder: knitr
BiocViews: SNP, GenomeAnnotation, Genetics, DataImport, FunctionalGenomics

Index of help topics:

ciseStore-class Class "ciseStore"'
describeStore collect basic descriptive statistics on ciseStore instances
extractByProbes retrieve eqtlTest results from a ciseStore instance
gQTLBase-package gQTLBase: infrastructure for eQTL, mQTL and similar studies
mergeCIsstates merge ChromImpute chromatin states, or GWAS hit indicators, to a GRanges
storeApply apply a function over job results in a ciseStore instance
storeMapResults use batchMapResults infrastructure to process results in a ciseStore instance
storeToff extract a vector from store results as ff (out of memory reference); support statistical reductions
ufeatByTiling split featurenames of SummarizedExperiment according to tiling, or to achieve simple balance within seqnames

Purpose is to define infrastructure on a comprehensive archive of eQTL, mQTL, dsQTL, etc., association statistics.
ciseStore-class

Package will complement gQTLStats. geuvStore2 is a basic illustration relative to GEUVADIS paper.

matprint is exported from package ff.

Author(s)

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

Description

wrap a BatchJobs registry that manages results of a cis-eQTL search

Objects from the Class

Objects can be created by calls of the form new("ciseStore", reg=reg, ...). All arguments must be named.

We can also use ciseStore(reg, validJobs, addProbeMap = TRUE, addRangeMap = TRUE) and the probemap and rangeMap slots will be populated appropriately. If validJobs is missing, the validJobs slot will be populated by findDone(reg). This may be problematic for handcrafted extracts from archives.

Slots

reg: Object of class "Registry" BatchJobs Registry instance
validJobs: Object of class "integer" vector of valid job identifiers for the registry
probemap: Object of class "data.frame" a map from expression probe identifiers to job identifiers where results for the probe are stored
rangeMap: Object of class "GRanges" a map from ranges on chromosomes, to job identifiers, in mcols()$jobid

Methods

show

Function describeStore uses batchMapResults and reduceResults to leverage a parallel environment to collect information on numbers of tests and features. Arguments are described in the associated man page.

Note

the construction of the maps occurs via storeApply, which will use foreach, so that registration of a parallel back end using, e.g., registerDoParallel, will determine the speed of construction

Any registry job results that do not inherit from GRanges are mapped to NULL and will not be present in ultimate maps.
describeStore

collect basic descriptive statistics on ciseStore instances

Description

collect basic descriptive statistics on ciseStore instances

Usage

describeStore(st, genetag = "probeid", snptag = "snp", ids = NULL, resfilter = force, doChecks = TRUE, ...)
describeByFilts(st, filtlist, ...)

Arguments

st instance of ciseStore-class

genetag string for field name for name of quantitatively assayed feature, defaults to "probeid"; for GTEx application "gene" is used

snptag string for field name for name of genotype feature

ids integerish vector of ids, can be left NULL to survey entire store

resfilter function applied to job results prior to summarization, defaults to force()

filtlist a list of functions suitable as resfilter arguments

doChecks logical – if true, will collect information on match between number probes requested and number reported on, and two scans of VCF loci in cis to probes. See details.

... used with describeByFilts, pass to storeApply

Details

uses parallel infrastructure of foreach on contents managed by st@reg

describeByFilts returns a matrix of descriptions with one row per filtlist element

storeDescription holds results of a describe task and includes information on noncongruence of features with cis tests and of results of two distinct scans of VCF: one with readGT on a single sample, the other with readVcf on all samples. If there are discrepancies between features given and tests returned, [storeDescription]@reqfail will give the job ids for these. If there are discrepancies between the numbers of loci retrieved on the two VCF scans, @locfail will tive the job ids for these. @reqfail events may be legitimate when a feature has no SNP in cis at the given radius. @locfail events usually indicate an I/O problem and the jobs should be resubmitted.
Value
list with elements ntests, ngene.uniq, nsnp.uniq

Examples

## Not run:
library(geuvStore2)
mm = makeGeuvStore2()
describeStore(mm, ids=1:10, resfilter=function(x) x$x$mindist < 50000))

## End(Not run)

extractByProbes  retrieve eqtlTest results from a ciseStore instance

Description
retrieve eqtlTest results from a ciseStore instance

Usage

extractByProbes(store, probeids, extractTag = "probeid")
extractByRanges(store, gr)
extractBySymbols(store, symbols, sym2probe, extractTag = "probeid")

Arguments

store  instance of ciseStore-class
probeids, symbols  vector character tokens
gr  instance of GRanges-class
sym2probe  named character vector of probeids with names given by corresponding symbols
extractTag  character atom telling what field in the archived GRanges is regarded as the probe or gene identifier
...  extra arguments to extractByProbes

Details
an index will be searched if created by the ciseStore constructor

Value
a GRanges instance

Author(s)
VJ Carey <stvjc@channing.harvard.edu>
mergeCIstates

mergeChromImpute chromatin states, or GWAS hit indicators, to a GRanges

Description

merge ChromImpute chromatin states, or GWAS hit indicators, to a GRanges

Usage

mergeCIstates(gr, ermaset=NULL, epig, genome = "hg19", importFull=FALSE, useErma = TRUE, stateGR=NULL)

mergeGWhits(gr, gwcat, use=c("both", "addr", "name")[1],
            grSnpField="SNP")

Arguments

gr a GRanges instance

ermaset an instance of ErmaSet-class. if NULL, supply a GRanges as stateGR, with fields states and statecols
gwcat an instance of gwaswloc-class, or any compliant GRanges instance – must have mcols field SNPS with snp identifier
epig the standardized epigenome name of the epigenome to use
genome a tag for genome build
importFull logical, set to TRUE to acquire entire content (for LNG.FET, 800K ranges), to avoid contention for connections in parallel applications
useErma logical – at the moment, must be TRUE; plan is to allow use of elements of AnnotationHub
use character string selecting approach for linking loci in gr to those in gwcat – if "both", coincidence in address or name are both checked and used; if "addr", only address is checked, if "name", only SNP name.
grSnpField character string naming the field in mcols(gr) with SNP id
stateGR a GRanges instance as imported from erma package or from AnnotationHub, with mcols field states denoting chromatin state and statecols the associated colors for rendering
storeApply

apply a function over job results in a ciseStore instance

Description

apply a function over job results in a ciseStore instance

Usage

storeApply(store, f, n.chunks, ids=NULL, ..., verbose = FALSE, flatten1=TRUE)

Arguments

store instance of ciseStore-class
f function on GRanges stored in ciseStore
n.chunks Number of chunks into which the jobs are to be broken; the series of chunks is handed to foreach to extract results and apply f to them. If missing, the value of getDoParWorkers() used.
ids defaults to NULL; if non-null, the jobs to be processed are limited to those identified in this vector.
... additional arguments to foreach
verbose if TRUE will allow progressbars and other messages to display
flatten1 if TRUE will execute unlist(...recursive=FALSE) on output, defaulted to FALSE in previous version

Details

The chunking of job identifiers will determine the degree of parallelization of application, and the form of the list that is returned. flatten1 will eventually default to TRUE.

Value

A list whose structure depends on the chunking of job identifiers. See the examples.

Note

eqtlStore imports BiocParallel’s bpparam function, and this determines in real time the number of workers to be employed by storeApply.

See Also

storeMapResults will apply over the store using the batch jobs submission infrastructure and can target specific results via ids; storeApply uses bplapply over the entire store

Examples

```r
if (require(geuvStore2)) {
  require(BatchJobs)
  store = makeGeuvStore2()
  storeApply(store, length)
  storeApply(store, length, ids=c(1:3,603))
}
```

Description

use batchMapResults infrastructure to process results in a ciseStore instance

Usage

```r
storeMapResults(store, reg2, fun, ...,
               ids = NULL, part = NA_character_, more.args = list())
loadAndFilterResult(reg,
                     id, filter=force, part = NA_character_, missing.ok = FALSE)
```
Arguments

store       an instance of \texttt{cisestore-class}
reg         instance of BatchJobs Registry class
reg2        an empty instance of the Registry class (see \texttt{makeRegistry})
fun         A function to map over results in \texttt{store}, with formals \texttt{(job, res, ...)}.
filter      a function that accepts and returns a \texttt{GRanges} instance, to be applied just after loading a result from the store
...         additional arguments to vectorize over (should be same length as \texttt{length(findDone(store@reg))})
ids         ids of job results to be mapped; if missing, map all job results
id          a single job id
part        see \texttt{batchMapResults}
missing.ok  see \texttt{loadResult}
more.args   a list of other arguments to be passed to \texttt{fun}; default is empty list.

Value

integer vector with job ids. Main purpose is to prepare the registry for \texttt{submitJobs}.

Note

\texttt{loadAndFilterResult} is not intended to be exported and may be removed in future versions.

Author(s)

VJ Carey \texttt{<stvjc@channing.harvard.edu>}

Examples

```r
## Not run:
if (require(geuvStore2)) {
  require(BatchJobs)
  store = makeGeuvStore2()
  fd = tempfile()
  tempreg = makeRegistry("tempSMR", file.dir=fd)
  storeMapResults( store, tempreg, fun=function(job, res, ...) length(res) )
  showStatus(tempreg)
  submitJobs(tempreg, 1:2)
  loadResults(tempreg)
  unlink(fd)
}

## End(Not run)
```
storeToFf

**Description**

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

**Usage**

storeToFf(store, field, ids = NULL, filter=force, ..., checkField = FALSE, ischar=FALSE)

**Arguments**

- **store**: instance of `ciseStore-class`
- **field**: character tag, length one. If name of a numeric field in the result set (typically something like `chisq` in the GRanges generated by cisAssoc), ff is applied directly. Character variables are converted to factors before ff is applied.
- **ids**: job ids to be used; if NULL, process all jobs
- **filter**: function to be applied when GRanges is loaded from results store, should accept and return a GRanges instance
- ... 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
- **ischar**: must be true for character vector to be handled properly as a factor, otherwise NA will be returned

**Details**

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

**Value**

a vector as ff reference

**Note**

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

**Examples**

```r
if (require(geuvStore2)) {
  require(BatchJobs)
  store = makeGeuvStore2()
  smchisq = storeToFf( store, "chisq", ids=store@validJobs[1:3])
  smchisq
}
```
**ufeatByTiling**

split featurenames of `SummarizedExperiment` according to tiling, or to achieve simple balance within `seqnames`

**Description**

split featurenames of `SummarizedExperiment` according to tiling, without redundancies

**Usage**

```r
ufeatByTiling(se, tiling, maxlen=20)
balancedFeatList(se, maxlen=20)
```

**Arguments**

- `se`: instance of `SummarizedExperiment`
- `tiling`: GRanges instance corresponding to a genomic tiling
- `maxlen`: numeric ... list elements longer than maxlen are chopped up to have this length, to foster load balancing

**Details**

`ufeatByTiling` uses findOverlaps, `balancedFeatList` uses split on `seqnames` and `BBmisc::chunk`

**Value**

a list with elements of `names(rowRanges(se))` corresponding to the elements of the tiling

**Examples**

```r
## Not run:
library(geuvPack)
data(geuFPKM)
library(Homo.sapiens)
au = paste0("chr", 1:22)
tg_500k = tileGenome(seqinfo(TxDb(Homo.sapiens))[au,], tilewidth=500000,
cut.last.tile.in.chrom=TRUE)
fn = ufeatByTiling(geuFPKM, tg_500k)
sn = balancedFeatList(geuFPKM)

summary(sapply(fn,length))
summary(sapply(sn,length))
```

## End(Not run)
Index

*Topic classes
  ciseStore-class, 3
*Topic models
  describeStore, 4
electroProbes, 5
mergeCIstates, 6
storeApply, 7
storeMapResults, 8
storeTof, 10
ufeatByTiling, 11
balancedFeatList (ufeatByTiling), 11
batchMapResults, 9
ciseStore (ciseStore-class), 3
ciseStore-class, 3
describeByFilts (describeStore), 4
descrStore, 4
extractByProbes, 5
extractByProbes, ciseStore, character, character-method
  (extractByProbes), 5
extractByProbes, ciseStore, character, missing-method
  (extractByProbes), 5
extractByRanges (extractByProbes), 5
extractByRanges, ciseStore, GRanges-method
  (extractByProbes), 5
extractBySymbols (extractByProbes), 5
extractBySymbols, ciseStore, character, character-method
  (extractByProbes), 5
extractBySymbols, ciseStore, character, character, missing-method
  (extractByProbes), 5
foreach, 3, 7
gQTLBase (gQTLBase-package), 2
gQTLBase-package, 2
loadAndFilterResult (storeMapResults), 8
loadResult, 9
makeRegistry, 9
matprint, 3
matprint (gQTLBase-package), 2