Package ‘gQTLBase’

March 26, 2015

Title gQTLBase: infrastructure for eQTL, mQTL and similar studies

Version 0.1.8

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

Description Infrastructure for eQTL, mQTL and similar studies.

Suggests geuvStore, knitr, rmarkdown, BiocStyle, RUnit, GGtools, Homo.sapiens, IRanges

Imports GenomicRanges, methods, BatchJobs, BBmisc, S4Vectors, ff, ffbase, BiocGenerics, foreach, doParallel

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
extractByProbes ........................................................................... 4
storeApply ....................................................................................... 5
storeMapResults ............................................................................ 6
storeToFf ......................................................................................... 7

Index 9
**gQTLBase-package**

**gQTLBase: infrastructure for eQTL, mQTL and similar studies**

---

**Description**

Infrastructure for eQTL, mQTL and similar studies.

---

**Details**

The DESCRIPTION file:

```
Package: gQTLBase
Title: gQTLBase: infrastructure for eQTL, mQTL and similar studies
Version: 0.1.8
Author: VJ Carey <stvjc@channing.harvard.edu>
Description: Infrastructure for eQTL, mQTL and similar studies.
Suggests: geuvStore, knitr, rmarkdown, BiocStyle, RUnit, GGtools, Homo.sapiens, IRanges
Imports: GenomicRanges, methods, BatchJobs, BBmisc, S4Vectors, ff, ffbase, BiocGenerics, foreach, doParallel
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'
- extractByProbes
  - retrieve eqtlTest results from a ciseStore instance
- gQTLBase-package
  - gQTLBase: infrastructure for eQTL, mQTL and similar studies
- 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

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

Package will complement gQTLStats. geuvStore is a basic illustration relative to GEUVADIS paper. `matprint` is exported from package ff.
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, ...)`. We can also use `ciseStore(reg, addProbeMap = TRUE, addRangeMap = TRUE)` and the probe-map and rangeMap slots will be populated appropriately.

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**

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

Examples
```
showClass("ciseStore")
# get the global assignment back
require(BatchJobs)
if (require(geuvStore)) {
  reg = partialRegistry()
  store = ciseStore(reg, addProbeMap=TRUE, addRangeMap=FALSE)
  store
}
```
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)

Arguments

store instance of ciseStore-class
probeids vector character tokens
gr instance of GRanges-class
extractTag character atom telling what field in the archived GRanges is regarded as the probe or gene identifier

Details

an index will be searched if created by the ciseStore constructor

Value

a GRanges instance

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

if (require(geuvStore)) {
  reg = partialRegistry()
  store = ciseStore(reg, addProbeMap=TRUE, addRangeMap=TRUE)
  ebp = extractByProbes(store, c("ENSG00000183814.10", "ENSG00000174827.9"))
  ebp
  rr = range(ebp)
  ebr = extractByRanges(store, rr)
  ebr
}
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)

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

Details
The chunking of job identifiers will determine the degree of parallelization of application, and the
form of the list that is returned.

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
storeMapResults

Use batchMapResults infrastructure to process results in a ciseStore instance.

Description

use batchMapResults infrastructure to process results in a ciseStore instance

Usage

storeMapResults(store, reg, 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 ciseStore-class
reg instance of BatchJobs Registry class
reg2 an empty instance of the Registry class (see makeRegistry)
fun A function to map over results in store, with formals (job, res, ...).
filter a function that accepts and returns a GRanges instance, to be applied just after loading a result from the store
... additional arguments to vectorize over (should be same length as 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 batchMapResults
missing.ok see loadResult
more.args a list of other arguments to be passed to fun; default is empty list.

Value

integer vector with job ids. Main purpose is to prepare the registry for submitJobs.

Examples

if (require(geuvStore)) {
  require(BatchJobs)
  reg = partialRegistry()
  store = ciseStore(reg, addProbeMap=FALSE, addRangeMap=FALSE)
  storeApply(store, length)
  storeApply(store, length, ids=c(1:3,603))
}
Note

loadAndFilterResult is not intended to be exported and may be removed in future versions.

Author(s)

VJ Carey <stvjc@channing.harvard.edu>

Examples

```r
if (require(geuvStore)) {
  require(BatchJobs)
  reg = partialRegistry()
  store = ciseStore(reg, addProbeMap=FALSE, addRangeMap=FALSE)
  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)
}
```

---

**storeToFf**

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

```r
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(geuvstore)) {
  require(BatchJobs)
  reg = partialRegistry()
  store = ciseStore(reg, addProbeMap=FALSE, addRangeMap=FALSE)
  smchisq = storeToFf( store, "chisq", ids=store@validJobs[1:3])
  smchisq
}
```
Index

*Topic **classes**
ciseStore-class, 3

*Topic **models**
extactByProbes, 4
storeApply, 5
storeMapResults, 6
storeToff, 7

batchMapResults, 6
ciseStore (ciseStore-class), 3
ciseStore-class, 3
extractByProbes, 4
extractByRanges (extractByProbes), 4
foreach, 3, 5
gQTLBase (gQTLBase-package), 2
gQTLBase-package, 2
loadAndFilterResult (storeMapResults), 6
loadResult, 6
makeRegistry, 6
matprint, 2
matprint (gQTLBase-package), 2
registerDoParallel, 3
storeApply, 3, 5
storeMapResults, 5, 6
storeToff, 7