

geuvStore2: sharded storage for cis-association statistics

Vincent J. Carey, stvjc at channing.harvard.edu

February 2015

Contents

1	Introduction	2
2	Illustration	2
2.1	Construction of mediator and indices	2
2.2	Extraction of content	3
2.3	Applicative programming	5
2.4	Visualization support	6
2.5	Origins	6

1 Introduction

The *geuvStore2* package demonstrates an approach to management of large numbers of statistics generated in integrative genomic analyses. The specific use case demonstrated here is cis-eQTL discovery. The following considerations motivated the design used here.

- Cluster computing will typically be used to perform cis-eQTL searches. Scalable performance is greatly aided by the *BatchJobs* infrastructure, which will create an archive of results.
 - This archive includes a database that holds information on job status (including time and memory required to complete) and result location. We consider this information worth saving.
 - The collection of results is, by default, “sharded” into a reasonable number of folders holding serialized R objects. We find this approach useful for supporting parallelizable retrieval of results.
- It makes sense to store results of cis-association analyses so that queries based on genomic addresses are rapidly resolved. Thus all the results are stored in `GRanges` instances, and queries based on `GRanges` are efficiently resolvable if an optional index is prepared before use.

2 Illustration

2.1 Construction of mediator and indices

The most basic entity mediating access to the information is the `BatchJobs` registry object. This is typically not created in a portable format, but includes directory information that we modify during package installation.

```
suppressPackageStartupMessages(library(geuvStore2))
prst = makeGeuvStore2()
prst
## ciseStore instance with 160 completed jobs.
## excerpt from job 1 :
## GRanges object with 1 range and 14 metadata columns:
##      seqnames      ranges strand |      paramRangeID      REF
##      <Rle> <IRanges> <Rle> |      <factor> <DNAStrngSet>
## [1]      1      526736      * | ENSG00000215915.5      C
##      ALT      chisq      permScore_1      permScore_2
##      <CharacterList>      <numeric>      <numeric>      <numeric>
## [1]      G 2.46382903511311 3.14566717550675 0.409225094116621
##      permScore_3      permScore_4      permScore_5
##      <numeric>      <numeric>      <numeric>
## [1] 0.157174317815962 0.0298147140344611 0.164808833821082
##      permScore_6      snp      MAF      probeid
##      <numeric> <character>      <numeric>      <character>
## [1] 0.0123114014465414 rs28863004 0.0910112359550562 ENSG00000215915.5
##      mindist
```

geuvStore2: sharded storage for cis-association statistics

```
##      <numeric>
## [1] 858333
## -----
## seqinfo: 86 sequences from hg19 genome
```

Association statistics were recorded between expression levels of each gene (as recorded in the GEUVADIS FPKM report) and all SNP with MAF $> 10^{-6}$ lying within a radius of 1 million bp upstream or downstream from the gene region. This package provides access to a selection of 160 jobs.

We use the `ciseStore` class to mediate between the user and the results data. This includes optional mappings based on gene identifiers (in the case of this example, these are Ensembl gene IDs) and `GRanges`. We have stored the maps, but they can be computed in real time if need be.

```
library(gQTLBase)
# prstore = ciseStore(prst, addProbeMap=TRUE, addRangeMap=TRUE)
prstore = makeGeuvStore2()
prstore
## ciseStore instance with 160 completed jobs.
## excerpt from job 1 :
## GRanges object with 1 range and 14 metadata columns:
##      seqnames  ranges strand |      paramRangeID      REF
##      <Rle> <IRanges> <Rle> |      <factor> <DNAStrngSet>
## [1]      1      526736      * | ENSG00000215915.5      C
##      ALT      chisq      permScore_1      permScore_2
##      <CharacterList>      <numeric>      <numeric>      <numeric>
## [1]      G 2.46382903511311 3.14566717550675 0.409225094116621
##      permScore_3      permScore_4      permScore_5
##      <numeric>      <numeric>      <numeric>
## [1] 0.157174317815962 0.0298147140344611 0.164808833821082
##      permScore_6      snp      MAF      probeid
##      <numeric> <character>      <numeric>      <character>
## [1] 0.0123114014465414 rs28863004 0.0910112359550562 ENSG00000215915.5
##      mindist
##      <numeric>
## [1] 858333
## -----
## seqinfo: 86 sequences from hg19 genome
```

2.2 Extraction of content

For a vector of gene identifiers, all available results are extracted.

```
head(
  extractByProbes(prstore,
    probeids=c("ENSG00000183814.10", "ENSG00000174827.9"))
)
## Warning: executing %dopar% sequentially: no parallel backend registered
## GRanges object with 6 ranges and 15 metadata columns:
```

geuvStore2: sharded storage for cis-association statistics

```
##      seqnames      ranges strand |      paramRangeID      REF
##      <Rle> <IRanges> <Rle> |      <factor> <DNAStringSet>
## [1]      1 225418903      * | ENSG00000183814.10      G
## [2]      1 225419456      * | ENSG00000183814.10      T
## [3]      1 225419667      * | ENSG00000183814.10      G
## [4]      1 225419982      * | ENSG00000183814.10      T
## [5]      1 225420024      * | ENSG00000183814.10      G
## [6]      1 225420751      * | ENSG00000183814.10      C
##
##              ALT              chisq              permScore_1
##      <CharacterList>      <numeric>      <numeric>
## [1]              A 0.325683225567047 0.0161058137687943
## [2]              C 0.00884049996442606 0.0777047399560453
## [3]              C 0.0593580128976154 0.136407651250872
## [4]              C 1.17690013755912 0.00242560703148352
## [5]              A 0.0010126244243042 1.24509170829122
## [6]              T 0.0664363123485674 2.19846778999689
##
##      permScore_2      permScore_3      permScore_4
##      <numeric>      <numeric>      <numeric>
## [1] 1.40119211379776 0.128681508566043 0.91973019075071
## [2] 0.550387835273064 1.6765889182995 2.21381884148021
## [3] 1.35459418617104 4.6471684405171 1.34142254253323
## [4] 4.28153393912583 3.89279638446672 2.71255679591447
## [5] 0.789927405868421 3.61644922719095 2.1864196065562
## [6] 0.279558123648989 2.54822763666469 0.188942949365943
##
##      permScore_5      permScore_6      snp      MAF
##      <numeric>      <numeric> <character>      <numeric>
## [1] 5.55960779837658 4.88514435664872 rs114086886 0.0303370786516854
## [2] 2.52197917484287 2.0584121044283  rs664855 0.287640449438202
## [3] 0.214913316486647 0.766617881845212  rs665776 0.173033707865169
## [4] 0.00491447963082568 1.37341001350867  rs200681083 0.110112359550562
## [5] 0.00269013209749856 0.308526373143654  rs74968234 0.0573033707865168
## [6] 1.29585321831181 1.20594286324189  rs785167 0.110112359550562
##
##      probeid      mindist      jobid
##      <character> <numeric> <integer>
## [1] ENSG00000183814.10 999947 20
## [2] ENSG00000183814.10 999394 20
## [3] ENSG00000183814.10 999183 20
## [4] ENSG00000183814.10 998868 20
## [5] ENSG00000183814.10 998826 20
## [6] ENSG00000183814.10 998099 20
##
## -----
## seqinfo: 86 sequences from hg19 genome
```

For a request based on genomic coordinates, a `GRanges` can be used to query. `findOverlaps` is used, and all results for genes whose regions overlap the query ranges are returned.

```
head(
  extractByRanges(prstore, GRanges("1", IRanges(146000000, width=1e6)))
)
## GRanges object with 6 ranges and 15 metadata columns:
##      seqnames      ranges strand |      paramRangeID      REF
```

geuvStore2: sharded storage for cis-association statistics

```
##          <Rle> <IRanges> <Rle> |          <factor> <DNAStrngSet>
## [1]          1 146003411      * | ENSG00000174827.9      A
## [2]          1 146003444      * | ENSG00000174827.9      C
## [3]          1 146003808      * | ENSG00000174827.9      G
## [4]          1 146016381      * | ENSG00000174827.9      G
## [5]          1 146016890      * | ENSG00000174827.9      T
## [6]          1 146019838      * | ENSG00000174827.9      T
##          ALT          chisq          permScore_1
## <CharacterList> <numeric> <numeric>
## [1]          G 0.00535324746696908 0.0800433756024206
## [2]          T 0.756450990425591 0.323116198662254
## [3]          A 0.0295406102160008 0.262138948840105
## [4]          A 0.00492718777498228 3.71660492435863e-06
## [5]          C 0.556989918424858 0.0486686642895075
## [6]          C 0.125411083235281 0.63708041523924
##          permScore_2          permScore_3          permScore_4
## <numeric> <numeric> <numeric>
## [1] 0.0381541515236908 0.0363108286733781 0.21059082325948
## [2] 1.32351467309614 0.137257149141042 0.336849679898503
## [3] 0.0274908024013663 2.74684624076025e-05 0.0415062946578329
## [4] 0.479575231882661 0.372731753714077 0.449551258548945
## [5] 0.384799846235676 0.559305218639807 0.00165827722452358
## [6] 0.314092760517727 1.4869224252764 0.0772102279356608
##          permScore_5          permScore_6          snp
## <numeric> <numeric> <character>
## [1] 3.99732963456782 0.236039374838042 rs150635557
## [2] 0.930174554911409 1.11252259429764 rs79556380
## [3] 0.00736063710220451 0.000123050150509449 rs587693118
## [4] 0.961466289056776 0.000589195101921725 rs376735389
## [5] 1.06933537430284 0.000743054968765492 rs199499386
## [6] 0.000325715409018348 1.55675692334389 rs201518173
##          MAF          probeid          mindist          jobid
## <numeric> <character> <numeric> <integer>
## [1] 0.0235955056179775 ENSG00000174827.9 239337 6
## [2] 0.0168539325842697 ENSG00000174827.9 239370 6
## [3] 0.0123595505617977 ENSG00000174827.9 239734 6
## [4] 0.00449438202247188 ENSG00000174827.9 252307 6
## [5] 0.486516853932584 ENSG00000174827.9 252816 6
## [6] 0.240449438202247 ENSG00000174827.9 255764 6
## -----
## seqinfo: 86 sequences from hg19 genome
```

2.3 Applicative programming

The `storeApply` function will be evaluated on all store elements. Iteration is governed by the `foreach` package.

```
lens = storeApply(prstore, length)
summary(unlist(lens))
```

geuvStore2: sharded storage for cis-association statistics

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 19852  32987  37346   38645  42701 131671
```

2.4 Visualization support

As of March 5, 2015 “biocLite('vcitn/gQTLbrowser')” will acquire a package including an interactive visualization function. “example('gQTLbrowser')” will load a queryable interface into the browser, with tooltips on the Manhattan plot for the selected gene.

2.5 Origins

The code used to generate the store follows. The definition of `kpp` actually used is commented out; `data(kpp)` with the installed package will provide the required vector of gene identifiers. is supplied.

```
library(geuvPack)
data(geuFPKM)
seqlevelsStyle(geuFPKM) = "NCBI"
library(GenomeInfoDb)
ok = which(seqnames(geuFPKM) %in% c(1:22, "X"))
geuFPKM = geuFPKM[ok,]

library(gQTLBase)
#load("../INTERACTIVE/geuvExtractStore.rda")
#kpp = geuvExtractStore@probemap[,1]
data("kpp", package="geuvStore2")
geuFPKM = geuFPKM[kpp,]

library(gQTLBase)
featlist = balancedFeatList( geuFPKM[order(rowRanges(geuFPKM)),], max=6 )
lens = sapply(featlist,length)
featlist = featlist[ which(lens>0) ]

library(BatchJobs)
regExtrP6 = makeRegistry("extractP6pop", # tile/cis
  packages=c("GenomicRanges", "gQTLstats", "geuvPack",
    "Rsamtools", "VariantAnnotation"), seed=1234)
myf = function(i) {
  if (!exists("geuFPKM")) data(geuFPKM)
  seqlevelsStyle(geuFPKM) = "NCBI"
  curse = geuFPKM[i,]
  load("gsvs.rda")
  svmat = gsvs$sv
  colnames(svmat) = paste0("SV", 1:ncol(svmat))
  colData(curse) = cbind(colData(curse), DataFrame(svmat))
  fmla = as.formula(paste("~popcode+", paste0(colnames(svmat), collapse="+")))
  curse = regressOut(curse, fmla)
  pn = gtpath( paste0("chr", as.character(seqnames(curse)[1])) )
```

geuvStore2: sharded storage for cis-association statistics

```
tf = TabixFile(pn)
  cisAssoc( curse, tf, cisradius=1000000, nperm=6 )
}
batchMap(regExtrP6, myf, featlist )
submitJobs(regExtrP6, job.delay = function(n,i) runif(1,1,3))
```