geuvStore2: sharded storage for cis-association statistics

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

February 2015

Contents

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> <DNAStringSet>
##   [1]        1 [526736, 526736]      * | ENSG00000215915.5              C
##                   ALT     chisq permScore_1 permScore_2 permScore_3
```

```
##       <CharacterList> <numeric>   <numeric>   <numeric>   <numeric>
##   [1]               G  2.463829    3.145667   0.4092251   0.1571743
##       permScore_4 permScore_5 permScore_6         snp        MAF
##         <numeric>   <numeric>   <numeric> <character>  <numeric>
##   [1]  0.02981471   0.1648088   0.0123114  rs28863004 0.09101124
##                 probeid   mindist
##             <character> <numeric>
##   [1] ENSG00000215915.5    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> <DNAStringSet>
##   [1]        1 [526736, 526736]      * | ENSG00000215915.5              C
##                   ALT     chisq permScore_1 permScore_2 permScore_3
##       <CharacterList> <numeric>   <numeric>   <numeric>   <numeric>
##   [1]               G  2.463829    3.145667   0.4092251   0.1571743
##       permScore_4 permScore_5 permScore_6         snp        MAF
##         <numeric>   <numeric>   <numeric> <character>  <numeric>
##   [1]  0.02981471   0.1648088   0.0123114  rs28863004 0.09101124
##                 probeid   mindist
##             <character> <numeric>
##   [1] ENSG00000215915.5    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:
##       seqnames                ranges strand |       paramRangeID
##          <Rle>             <IRanges>  <Rle> |           <factor>
##   [1]        1 [225418903, 225418903]      * | ENSG00000183814.10
```

```
##   [2]        1 [225419456, 225419456]      * | ENSG00000183814.10
##   [3]        1 [225419667, 225419667]      * | ENSG00000183814.10
##   [4]        1 [225419982, 225419982]      * | ENSG00000183814.10
##   [5]        1 [225420024, 225420024]      * | ENSG00000183814.10
##   [6]        1 [225420751, 225420751]      * | ENSG00000183814.10
##                   REF             ALT       chisq permScore_1 permScore_2
##       <DNAStringSet> <CharacterList>   <numeric>   <numeric>   <numeric>
##   [1]              G               A 0.325683226 0.016105814   1.4011921
##   [2]              T               C 0.008840500 0.077704740   0.5503878
##   [3]              G               C 0.059358013 0.136407651   1.3545942
##   [4]              T               C 1.176900138 0.002425607   4.2815339
##   [5]              G               A 0.001012624 1.245091708   0.7899274
##   [6]              C               T 0.066436312 2.198467790   0.2795581
##       permScore_3 permScore_4 permScore_5 permScore_6         snp
##         <numeric>   <numeric>   <numeric>   <numeric> <character>
##   [1]   0.1286815   0.9197302 5.559607798   4.8851444 rs114086886
##   [2]   1.6765889   2.2138188 2.521979175   2.0584121    rs664855
##   [3]   4.6471684   1.3414225 0.214913316   0.7666179    rs665776
##   [4]   3.8927964   2.7125568 0.004914480   1.3734100 rs200681083
##   [5]   3.6164492   2.1864196 0.002690132   0.3085264  rs74968234
##   [6]   2.5482276   0.1889429 1.295853218   1.2059429    rs785167
##              MAF            probeid   mindist     jobid
##        <numeric>        <character> <numeric> <integer>
##   [1] 0.03033708 ENSG00000183814.10    999947        20
##   [2] 0.28764045 ENSG00000183814.10    999394        20
##   [3] 0.17303371 ENSG00000183814.10    999183        20
##   [4] 0.11011236 ENSG00000183814.10    998868        20
##   [5] 0.05730337 ENSG00000183814.10    998826        20
##   [6] 0.11011236 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
##          <Rle>             <IRanges>  <Rle> |          <factor>
##   [1]        1 [146003411, 146003411]      * | ENSG00000174827.9
##   [2]        1 [146003444, 146003444]      * | ENSG00000174827.9
##   [3]        1 [146003808, 146003808]      * | ENSG00000174827.9
##   [4]        1 [146016381, 146016381]      * | ENSG00000174827.9
##   [5]        1 [146016890, 146016890]      * | ENSG00000174827.9
##   [6]        1 [146019838, 146019838]      * | ENSG00000174827.9
##                   REF             ALT       chisq  permScore_1 permScore_2
##       <DNAStringSet> <CharacterList>   <numeric>    <numeric>   <numeric>
##   [1]              A               G 0.005353247 8.004338e-02  0.03815415
##   [2]              C               T 0.756450990 3.231162e-01  1.32351467
##   [3]              G               A 0.029540610 2.621389e-01  0.02749080
##   [4]              G               A 0.004927188 3.716605e-06  0.47957523
##   [5]              T               C 0.556989918 4.866866e-02  0.38479985
```

```
##   [6]              T               C 0.125411083 6.370804e-01  0.31409276
##       permScore_3 permScore_4  permScore_5  permScore_6         snp
##         <numeric>   <numeric>    <numeric>    <numeric> <character>
##   [1] 3.631083e-02 0.210590823 3.9973296346 0.2360393748 rs150635557
##   [2] 1.372571e-01 0.336849680 0.9301745549 1.1125225943  rs79556380
##   [3] 2.746846e-05 0.041506295 0.0073606371 0.0001230502 rs587693118
##   [4] 3.727318e-01 0.449551259 0.9614662891 0.0005891951 rs376735389
##   [5] 5.593052e-01 0.001658277 1.0693353743 0.0007430550 rs199499386
##   [6] 1.486922e+00 0.077210228 0.0003257154 1.5567569233 rs201518173
##               MAF           probeid   mindist     jobid
##         <numeric>       <character> <numeric> <integer>
##   [1] 0.023595506 ENSG00000174827.9    239337         6
##   [2] 0.016853933 ENSG00000174827.9    239370         6
##   [3] 0.012359551 ENSG00000174827.9    239734         6
##   [4] 0.004494382 ENSG00000174827.9    252307         6
##   [5] 0.486516854 ENSG00000174827.9    252816         6
##   [6] 0.240449438 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))
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19852   32987   37346   38645   42701  131671
```

2.4 Visualization support

As of March 5, 2015 "biocLite('vjcitn/gQTLbrowser')" will acquire a package including an interactive visualization function. "example('gQTLbrowse')" will load a queriable 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])) )
   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))
```