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      * | ENSG00000215915.5              C
##                   ALT     chisq permScore_1 permScore_2 permScore_3 permScore_4
##       <CharacterList> <numeric>   <numeric>   <numeric>   <numeric>   <numeric>
##   [1]               G   2.46383     3.14567    0.409225    0.157174   0.0298147
##       permScore_5 permScore_6         snp       MAF           probeid   mindist
##         <numeric>   <numeric> <character> <numeric>       <character> <numeric>
##   [1]    0.164809   0.0123114  rs28863004 0.0910112 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      * | ENSG00000215915.5              C
##                   ALT     chisq permScore_1 permScore_2 permScore_3 permScore_4
##       <CharacterList> <numeric>   <numeric>   <numeric>   <numeric>   <numeric>
##   [1]               G   2.46383     3.14567    0.409225    0.157174   0.0298147
##       permScore_5 permScore_6         snp       MAF           probeid   mindist
##         <numeric>   <numeric> <character> <numeric>       <character> <numeric>
##   [1]    0.164809   0.0123114  rs28863004 0.0910112 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            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 permScore_2 permScore_3
##       <CharacterList>  <numeric>   <numeric>   <numeric>   <numeric>
##   [1]               A 0.32568323  0.01610581    1.401192    0.128682
##   [2]               C 0.00884050  0.07770474    0.550388    1.676589
```

```
##   [3]              C 0.05935801  0.13640765    1.354594    4.647168
##   [4]              C 1.17690014  0.00242561    4.281534    3.892796
##   [5]              A 0.00101262  1.24509171    0.789927    3.616449
##   [6]              T 0.06643631  2.19846779    0.279558    2.548228
##       permScore_4 permScore_5 permScore_6         snp       MAF
##         <numeric>   <numeric>   <numeric> <character> <numeric>
##   [1]    0.919730  5.55960780    4.885144 rs114086886 0.0303371
##   [2]    2.213819  2.52197917    2.058412    rs664855 0.2876404
##   [3]    1.341423  0.21491332    0.766618    rs665776 0.1730337
##   [4]    2.712557  0.00491448    1.373410 rs200681083 0.1101124
##   [5]    2.186420  0.00269013    0.308526  rs74968234 0.0573034
##   [6]    0.188943  1.29585322    1.205943    rs785167 0.1101124
##                  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
##          <Rle> <IRanges>  <Rle> |          <factor> <DNAStringSet>
##   [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 permScore_2 permScore_3
##       <CharacterList>  <numeric>   <numeric>   <numeric>   <numeric>
##   [1]               G 0.00535325 8.00434e-02   0.0381542 3.63108e-02
##   [2]               T 0.75645099 3.23116e-01   1.3235147 1.37257e-01
##   [3]               A 0.02954061 2.62139e-01   0.0274908 2.74685e-05
##   [4]               A 0.00492719 3.71660e-06   0.4795752 3.72732e-01
##   [5]               C 0.55698992 4.86687e-02   0.3847998 5.59305e-01
##   [6]               C 0.12541108 6.37080e-01   0.3140928 1.48692e+00
##       permScore_4 permScore_5 permScore_6         snp        MAF
##         <numeric>   <numeric>   <numeric> <character>  <numeric>
##   [1]  0.21059082 3.997329635 0.236039375 rs150635557 0.02359551
##   [2]  0.33684968 0.930174555 1.112522594  rs79556380 0.01685393
##   [3]  0.04150629 0.007360637 0.000123050 rs587693118 0.01235955
```

```
##   [4]  0.44955126 0.961466289 0.000589195 rs376735389 0.00449438
##   [5]  0.00165828 1.069335374 0.000743055 rs199499386 0.48651685
##   [6]  0.07721023 0.000325715 1.556756923 rs201518173 0.24044944
##                probeid   mindist     jobid
##            <character> <numeric> <integer>
##   [1] ENSG00000174827.9    239337         6
##   [2] ENSG00000174827.9    239370         6
##   [3] ENSG00000174827.9    239734         6
##   [4] ENSG00000174827.9    252307         6
##   [5] ENSG00000174827.9    252816         6
##   [6] 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))
```