GGtools 2011: leaner software for genetics of gene expression

VJ Carey (stvjc at channing.harvard.edu)

October 31, 2011

Contents

1 Introduction; major changes

Since its introduction in 2006, GGtools has provided a number of data structures and tools for exploratory data analysis and hypothesis testing in expression genetics. Since 2006, Bioconductor's facilities for representing genomes and for exploiting advanced ideas in computing and statistical modeling have evolved considerably, and many components of GGtools/GGBase need to be discarded to promote use of new facilities.

The following major changes have been made.

- `smlSet` instances should not be used for genotyping panels of more than one million loci. A packaging discipline has been introduced. An expression genetics experiment should be managed in a package in which expression data are held in an `ExpressionSet` instance in the `data` folder, and *snpStats* `SnpMatrix` instances are stored in `inst/parts`. After installation, the `getSS` function constructs an `smlSet` instance on the fly – typically with modest memory footprint because only a fraction of available loci are held in memory. `externalize` demonstrates construction of a compliant package for expression genetics data by converting an existing whole- or partial-genome smlSet instance into a package.
- `gwSnpTests` has been retained for focused in-memory test computation, but it will be phased out.
- `eqtlTests` is the primary function for inference in expression genetics. This function takes genotype and phenotype information from an `smlSet`, passes it through testing facilities of *snpStats*, and creates managed out-of-memory storage for voluminous testing results.
- Facilities for metadata handling, e.g., determining SNP locations, gene locations, vocabulary translations, are being phased out. Users must construct location and feature name metadata using up-to-date Bioconductor resources or their own resources.

The remainder of this document illustrates basic activities now supported by the package.

2 A simple exploration of eQTL for a selected gene

2.1 Filtering an smlSet for a chromosome-wide test

We retain a two-chromosome example with expression data derived from the Wellcome Trust GENEVAR project archives, and genotype data from HapMap phase II:

```
> library(GGtools)
> if (!exists("hmceuB36.2021")) hmceuB36.2021 <- getSS("GGtools", c("20", "21"))
> hmceuB36.2021
```

```
SnpMatrix-based genotype set:
number of samples:  90
number of chromosomes present:  2
annotation: illuminaHumanv1.db
Expression data dims: 47293 x 90
Phenodata: An object of class "AnnotatedDataFrame"
  sampleNames: NA06985 NA06991 ... NA12892 (90 total)
  varLabels: famid persid ... male (7 total)
  varMetadata: labelDescription
```

A commonly found gene with eQTL is CPNE1 on chromosome 20. We can determine the probe name in use for CPNE1 as follows:

```
> library("illuminaHumanv1.db")
> cpn = get("CPNE1", revmap(illuminaHumanv1SYMBOL))
> use = intersect(cpn, featureNames(hmceuB36.2021))
> if (length(use) == 0) stop("probe not on array")
> use = use[1]
```

We will now trim down the `smlSet` instance to this gene and SNPs on chr20:

```
> hmlit = hmceuB36.2021[probeId(use),]
> hmlit = hmlit[ chrnum("20"), ]
> hmlit
```

```
SnpMatrix-based genotype set:
number of samples:  90
number of chromosomes present:  1
annotation: illuminaHumanv1.db
Expression data dims: 1 x 90
Phenodata: An object of class "AnnotatedDataFrame"
  sampleNames: NA06985 NA06991 ... NA12892 (90 total)
  varLabels: famid persid ... male (7 total)
  varMetadata: labelDescription
```

2.2 Executing tests and interrogating the results

The preferred testing procedure is:

```
> tname = function() gsub(".*/|.*\\\\", "", tempfile())
> e1 = eqtlTests(hmlit, ~male, targdir=tname(), runname=tname())
> e1
```

```
eqtlTestsManager  computed Mon Oct 31 21:58:26 2011
gene annotation: illuminaHumanv1.db
There are 1 chromosomes analyzed.
some genes (out of 1): GI_23397697-A
some snps (out of 66226): rs4814683 rs6076506 ... rs6062363 rs4809418
```

```
> class(e1)
```

```
[1] "eqtlTestsManager"
attr(,"package")
[1] "GGtools"
```

We support focused queries into the manager of results. For this application we are computing $\chi^2(1)$ statistics for each gene-SNP combination, measuring the fit of an additive genetic model with, in this case, adjustment for gender.

```
> e1[rsid("rs6060535"), probeId("GI_23397697-A")]
```

```
$`20`
          GI_23397697-A
rs6060535         53.62
attr(,"vmode")
[1] "short"
```

We can acquire the most predictive SNP using `topFeats`.

```
> topFeats(probeId(cpn), mgr=e1, ffind=1)
```

```
rs17093026  rs1118233 rs12480408  rs6060535 rs11696527  rs6058303  rs2425078
     56.21      54.10      53.62      53.62      53.62      53.62      53.62
 rs6060578  rs1970357  rs7273815
     53.62      53.62      53.53
```

Here the `ffind` parameter must be set to pick among files that might be managed by the manager bound to `mgr`. See the technical appendix for more details.

2.3 Visualization

To visualize the results, using hg18 locations for the SNP, we must acquire location metadata. Note that we are using the 2009 SNPlocs metadata from Bioconductor; for later builds, an as.GRanges parameter can be set to help avoid some of the conversion tasks.

```
> library(SNPlocs.Hsapiens.dbSNP.20090506)
> c20l = getSNPlocs("chr20")
> c20 = GRanges(IRanges(c20l$loc, width=1), seqnames="chr20")
> rs20 = paste("rs", c20l$RefSNP_id, sep="")
> names(c20) = rs20
> length(c20)

[1] 323041

> c20[1:3,]

GRanges with 3 ranges and 0 elementMetadata values:
             seqnames       ranges strand
                <Rle>    <IRanges>  <Rle>
  rs28753379    chr20 [8572, 8572]      *
  rs28579812    chr20 [8646, 8646]      *
   rs6078030    chr20 [9098, 9098]      *
  ---
  seqlengths:
   chr20
      NA
```

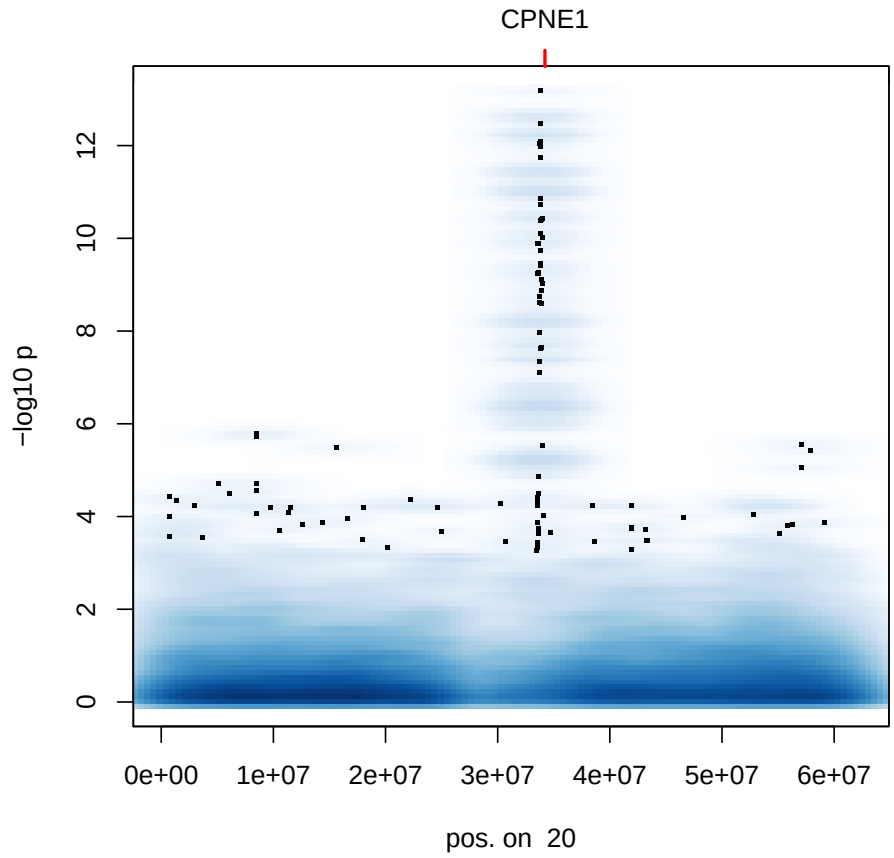
The names of SNP for which tests were computed are

```
> tn = rownames(e1@fflist[[1]])
> tn[1:10]

 [1] "rs4814683"  "rs6076506"  "rs6139074"  "rs1418258"  "rs7274499"
 [6] "rs6116610"  "rs13043000" "rs6054257"  "rs6086616"  "rs6039403"
```

We can now obtain the manhattan plot:

```
> manhPlot(cpn, mgr=e1, ffind=1, locGRanges=c20, cex=3)
```



In general we will prefer to have this information in a browser track. To accomplish this, add the test results to the GRanges instance for locations as 'score' and export a wig file.

We have to align the SNPs with locations with those for which we have tests. First, restrict the locations.

```
> okn = intersect(tn, names(c20))
> c20ok = c20[okn]
```

Now obtain the -log10 p values:

```
> chis1 = e1[, probeId(cpn)][[1]][okn,]
> ml10p = -log10(1-pchisq(chis1, 1))
```

Bind to the location data:

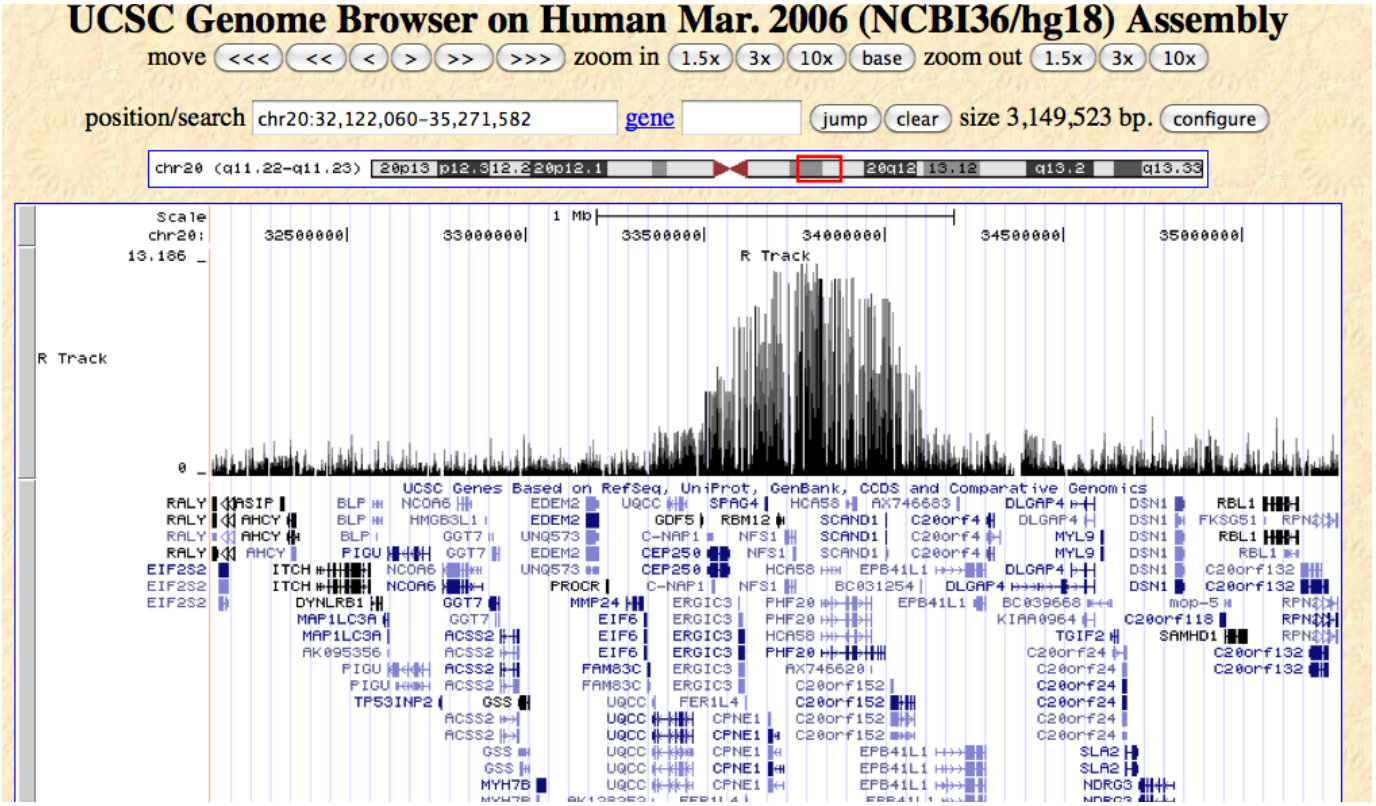
```
> elementMetadata(c20ok)$score = ml10p
> c20ok[1:5,]
```

```
GRanges with 5 ranges and 1 elementMetadata value:
            seqnames         ranges strand |             score
               <Rle>      <IRanges>  <Rle> |         <numeric>
  rs4814683    chr20 [ 9795,  9795]      * | 0.101638648472484
  rs6076506    chr20 [11231, 11231]      * |                 0
  rs6139074    chr20 [11244, 11244]      * | 0.387654128532594
  rs1418258    chr20 [11799, 11799]      * | 0.101638648472484
  rs7274499    chr20 [12150, 12150]      * |                 0
  ---
  seqlengths:
   chr20
      NA
```

and export:

```
> library(rtracklayer)
> export(c20ok, "cpne1.wig")
```

This is imported as a custom track and viewed as:



2.4 Checking coincidence with other genomic features

When we store the statistical results in a `GRanges` instance, we have capacity to quickly assess genomic context of high-scoring SNP. The top 20 SNP and their locations are

```
> top200 = c20ok[order(elementMetadata(c20ok)$score, decreasing=TRUE)[1:200]]
> top200[196:200]
```

```
GRanges with 5 ranges and 1 elementMetadata value:
             seqnames               ranges strand |            score
                <Rle>            <IRanges>  <Rle> |        <numeric>
  rs11697672    chr20 [34088036, 34088036]      * |  6.0071899588606
  rs10439606    chr20 [33531574, 33531574]      * | 5.92144703240679
   rs4911507    chr20 [33539412, 33539412]      * | 5.92144703240679
   rs4911509    chr20 [33550723, 33550723]      * | 5.92144703240679
   rs6086504    chr20 [ 8497393,  8497393]      * | 5.79725821632482
  ---
  seqlengths:
   chr20
      NA
```

We see that we have SNP with p-values at most 10^{-5}. We will now check whether any of these lie in regulatory regions defined by ORegAnno, confined to 10MB near CPNE1 on chr20. Execution suppressed for build failures, 9/15/2011.

```
> library(rtracklayer)
> ss = browserSession()
> genome(ss) = "hg18"
> GG = GRanges(seqnames="chr20", IRanges(30000000,40000000))
> oq = ucscTableQuery(ss, "oreganno", range=GG)
> toq = track(oq)
> ov = subsetByOverlaps(toq, top200)
> ov
```

For further discovery of the regulatory feature characteristics, consider

```
> tableName(oq) = "oregannoAttr"
> featdat = getTable(oq)
> featdat[ featdat$id %in% ov$name, ]
```

Retrieving the table can take a while, so we suppress it in this vignette, but on 8 Mar 2011 the result was:

```
> featdat[ featdat$id %in% ov$name, ]
               id       attribute                                    attrVal
5809  OREG0004040            type                       REGULATORY REGION
5810  OREG0004040            Gene                                   FER1L4
5811  OREG0004040         Dataset                 Stanford ENCODE Dataset
5812  OREG0004040 EvidenceSubtype Transient transfection luciferase assay
5833  OREG0004046            type                       REGULATORY REGION
5834  OREG0004046            Gene                                    RBM12
5835  OREG0004046         Dataset                 Stanford ENCODE Dataset
5836  OREG0004046 EvidenceSubtype Transient transfection luciferase assay
86472 OREG0025894            type                       REGULATORY REGION
86473 OREG0025894            TFbs                                    CTCF
86474 OREG0025894         Dataset          CTCF ChIP-chip sites (Ren lab)
86475 OREG0025894 EvidenceSubtype               ChIP-on-chip (ChIP-chip)
```

enabling us to learn more about the annotations of regions in which high-scoring SNP are found.

Clearly other UCSC-mediated resources can be used similarly to help interpret SNP scores.

3 Supporting comprehensive testing

3.1 The default behavior of `eqtlTests`

`eqtlTests` will, by default, compute scores for all gene-SNP combinations in the `smlSet` instance to which it is applied. With a multicore system, some concurrency can be achieved. To illustrate the activity, we take only 20 genes sampled from the illumina platform and compute all eQTL tests for SNP on chromosomes 20 and 21.

```
> NG = length(featureNames(hmceuB36.2021))
> hmct = hmceuB36.2021[ sample(1:NG, size=20, replace=FALSE), ]
> applier = lapply
> chkmult = function() {length(grep("^multicore$", installed.packages()[,1])) > 0}
> if (chkmult()) {
+   library(multicore)
+   options(cores=min(c(multicore:::volatile$detectedCores, 6)))
+   applier = mclapply
+   }
> unix.time(e2 <- eqtlTests(hmct, ~male, geneApply=applier, targdir=tempdir(), runnam

   user  system elapsed
  8.369   0.936   6.315
```

We are now managing 3.4 million tests. These are stored out of memory using the *ff* system for memory-mapped disk representations of R objects.

```
> e2@fflist[[1]]

ff (closed) short length=1324520 (1324520) dim=c(66226,20) dimorder=c(1,2)

> sapply(e2@fflist,dim)

        20    21
[1,] 66226 35584
[2,]    20    20
```

It is unpleasant to use the at-sign fflist notation but until a richer set of use cases is identified, this primitive idiom will be used.

In general we will break up a comprehensive run by gene sets – typically one run for all genes on a chromosome. In this case, we will generate 22 eqtlTestManager instances; these can be unified with a cisTransDirector or multiCisDirector object. We will discuss that later.

3.2 Reducing the memory footprint for comprehensive 'same chromosome' tests

In the example given above, the data for genotypes on chr20 and chr21 are simultaneously in memory. Up to now, we have often stored all genotype data in memory when using smlSets, leading to multiple Gb of RAM consumption, causing conflicts and loss of speed. We now describe an approach to building and using package-based experiment representations, which allow reduced and more flexible use of resources.

Because this functionality involves use of the file system along with package building and installation, which can lead to subtle portability issues, we do not evaluate the code chunks here.

We can take a unified smlSet instance and turn it into a package using the `externalize` function. We also install it:

```
> edname = paste("exdem", tname(), sep="")
> if (interactive() & file.exists(edname)) stop(paste("will not remove existing '", e
> externalize(hmceuB36.2021, edname)
> tar(tna <- paste(edname,".tar.gz", sep=""), edname, compression="gzip")
> install.packages(tna, repos=NULL, type="source")
> try(system(paste("rm -rf", tna)))
> try(system(paste("rm -rf", edname)))
```

Now we can use `getSS` to create tailored smlSet instances on the fly.

```
> gc()
> s1 = getSS(edname, "20")
> gc()
> rm(s1)
> gc()
```

This technology is used in the makeDiagDirector function. This function requires a list that maps genes to chromosomes. In this example, we take 20 genes from each of chromosome 20 and 21 to define the map. All same-chromosome tests for these 40 genes will be computed. No concurrency is used.

```
> mymap = lapply(c("20", "21"), function(x) get(x, revmap(illuminaHumanv1CHR))[1:20])
> mymap = lapply(mymap, function(x) intersect(x, featureNames(hmceuB36.2021)))
> names(mymap) = c("20", "21")
> unix.time(md <-  makeDiagDirector("GGtools", mymap, rhs=~male, sleeplen=2, runname=
+     targdir=tname()))

   user  system elapsed
 17.873   0.392  22.286

> md

multiCisDirector instance with 2 eqtlTestsManagers.
elements: 20 21
```

It would be typical to save the director object (here named `md`) to disk, so that the ff files that have been computed can be reused conveniently.

3.3 Acquiring test results within specified intervals

Here we will discuss how to use `cisProxScores`. The calling sequence is

```
cisProxScores(smlSet, fmla, dradset, direc = NULL, folder, runname,
      geneApply = mclapply, saveDirector = TRUE, snpGRL=NULL,
      geneGRL=NULL, snpannopack="SNPlocs.Hsapiens.dbSNP.20090506", ffind=NULL, ...)
```

so in fact one can pass an `smlSet` instance as input. However, if a suitable director instance is available, that can be used, avoiding the computation of the tests. If the director is used, we need only specify

- `dradset`: the collection of nested paired intervals around genes within which scores will be collected
- `geneGRL`: a list of GRanges, with one element per chromosome, providing coordinates of gene start and stop locations

- `snpGRL`: a list of GRanges, with one element per chromosome, providing coordinates of SNPs with names
- `ffind`: an index value into the manager-specific lists for the scores we wish to retrieve; in most cases this will be 1

The names of probesets scored in `md` can be obtained:

```
> ps = lapply(md@mgrs, function(x) colnames(x@fflist[[1]]))
> names(ps) = names(md@mgrs)
> ps

$`20`
 [1] "GI_4557248-S"  "GI_15451784-S" "GI_4557678-S"  "GI_9951914-S"
 [5] "GI_21327679-S" "GI_13259532-S" "GI_20336333-A" "GI_20336334-I"
 [9] "GI_13518027-S" "GI_4557368-S"  "GI_4502426-S"  "GI_4502446-S"
[13] "GI_4557422-S"  "GI_23312370-A" "GI_11641410-A" "GI_14589892-S"
[17] "GI_28872795-S" "GI_26105977-S" "GI_4502806-S"  "GI_29570783-S"

$`21`
 [1] "GI_7669476-I"  "GI_7669478-A"  "GI_4557290-I"  "GI_4557294-A"
 [5] "GI_41406053-S" "GI_19913429-S" "GI_41327758-S" "GI_4502352-S"
 [9] "GI_4557346-S"  "GI_11038670-S" "GI_4826650-S"  "GI_40538810-S"
[13] "GI_19923197-S" "GI_4502598-S"  "GI_7108334-S"  "GI_4557414-S"
[17] "GI_15011912-S" "GI_17402874-I" "GI_17402876-A" "GI_17402878-I"
```

The GRanges instance is then obtained as

```
> getgr = function(x,map) abs(sapply(mget(x, map), "[", 1))
> st = lapply(ps, getgr, illuminaHumanv1CHRLOC)
> en = lapply(ps, getgr, illuminaHumanv1CHRLOCEND)
> sp = names(ps)
> grl = list()
> for (i in 1:length(ps)) {
+   grl[[i]] = GRanges(seqnames=sp[i], IRanges(st[[i]], en[[i]]))
+   names(grl[[i]]) = ps[[i]]
+   }
> names(grl) = names(ps)
```

SNP locations from hg18 can be obtained as follows (we did chr20 above):

```
> seqlevels(c20) <- sub("chr", "", seqlevels(c20))
> c21l = getSNPlocs("chr21")  # hg18
> c21 = GRanges(IRanges(c21l$loc, width=1), seqnames="21")
> rs21 = paste("rs", c21l$RefSNP_id, sep="")
> names(c21) = rs21
> srl = list(`20`=c20, `21`=c21)
```

Now compute scores in intervals to 50kb around gene, then from 50kb to 100kb, then from 100kb to 500kb:

```
> csc = cisProxScores( direc=md, dradset = c(0,50000,100000, 500000), snpGRL = srl, g
+   geneApply=lapply, ffind=1 )
```

```
2021202120212021
```

This yields a deeply nested structure; a hint of best scores per gene within intervals comes from

```
> lapply(csc, lapply, sapply,  function(x)max(unlist(x)))
```

```
$`FL0e+00.0e+00`
$`FL0e+00.0e+00`$`20`
 GI_4557248-S GI_15451784-S  GI_4557678-S  GI_9951914-S GI_21327679-S
         5.22          5.23          4.30          2.92          1.26
GI_20336333-A GI_20336334-I GI_13518027-S  GI_4557368-S  GI_4502426-S
         3.76          2.45          2.93          5.85          9.04
 GI_4502446-S  GI_4557422-S GI_23312370-A GI_11641410-A GI_14589892-S
         3.71          6.35          3.21          2.53         10.32
GI_28872795-S GI_26105977-S  GI_4502806-S GI_29570783-S
         0.19          2.33          0.87          5.71

$`FL0e+00.0e+00`$`21`
 GI_7669476-I  GI_7669478-A  GI_4557290-I  GI_4557294-A GI_41406053-S
         1.74          5.28          1.54          1.42         14.65
GI_19913429-S GI_41327758-S  GI_4502352-S GI_11038670-S  GI_4826650-S
         1.76          2.66         10.73          0.74          2.80
GI_40538810-S GI_19923197-S  GI_4502598-S  GI_7108334-S  GI_4557414-S
         2.13          4.03          1.24          0.54          5.62


$`FL0e+00.5e+04`
$`FL0e+00.5e+04`$`20`
 GI_4557248-S GI_15451784-S  GI_4557678-S  GI_9951914-S GI_21327679-S
         4.73          3.25          6.48          2.64          1.42
GI_13259532-S GI_20336333-A GI_20336334-I GI_13518027-S  GI_4557368-S
         4.35          7.82          4.49          3.07          5.85
 GI_4502426-S  GI_4502446-S  GI_4557422-S GI_23312370-A GI_11641410-A
         5.66          7.03          8.35          4.28          6.14
GI_14589892-S GI_28872795-S GI_26105977-S  GI_4502806-S GI_29570783-S
         3.83          2.02          5.03          6.08          3.33
```

```
$`FL0e+00.5e+04`$`21`
 GI_7669476-I  GI_7669478-A  GI_4557290-I  GI_4557294-A GI_41406053-S
         2.12          5.97          3.19          2.00         13.39
GI_19913429-S GI_41327758-S  GI_4502352-S GI_11038670-S  GI_4826650-S
         6.64          6.12          2.23          2.89          3.59
GI_40538810-S GI_19923197-S  GI_4502598-S  GI_7108334-S  GI_4557414-S
         3.60          3.93          0.53          3.58          7.22


$`FL5e+04.1e+05`
$`FL5e+04.1e+05`$`20`
 GI_4557248-S GI_15451784-S  GI_4557678-S  GI_9951914-S GI_21327679-S
         4.19          9.17          1.69          2.64          1.42
GI_13259532-S GI_20336333-A GI_20336334-I GI_13518027-S  GI_4557368-S
         8.52          5.47          2.87          1.83          2.58
 GI_4502426-S  GI_4502446-S  GI_4557422-S GI_23312370-A GI_11641410-A
         5.02          2.21          7.75          3.82          2.87
GI_14589892-S GI_28872795-S GI_26105977-S  GI_4502806-S GI_29570783-S
         7.22          8.70          3.87          6.33          2.94

$`FL5e+04.1e+05`$`21`
 GI_7669476-I  GI_7669478-A  GI_4557290-I  GI_4557294-A GI_41406053-S
         1.11          9.49          5.60          7.43          9.34
GI_19913429-S GI_41327758-S  GI_4502352-S GI_11038670-S  GI_4826650-S
         9.23          5.06          3.48          2.62          5.08
GI_40538810-S GI_19923197-S  GI_4502598-S  GI_7108334-S  GI_4557414-S
         3.67          2.67          0.79          2.08         10.30


$`FL1e+05.5e+05`
$`FL1e+05.5e+05`$`20`
 GI_4557248-S GI_15451784-S  GI_4557678-S  GI_9951914-S GI_21327679-S
         6.34          7.60          8.43          8.49          3.72
GI_13259532-S GI_20336333-A GI_20336334-I GI_13518027-S  GI_4557368-S
         6.95          6.58          5.58          5.22          7.45
 GI_4502426-S  GI_4502446-S  GI_4557422-S GI_23312370-A GI_11641410-A
         8.83          2.96          6.38          5.16          6.04
GI_14589892-S GI_28872795-S GI_26105977-S  GI_4502806-S GI_29570783-S
         5.68          9.63         11.13         10.67          4.84

$`FL1e+05.5e+05`$`21`
 GI_7669476-I  GI_7669478-A  GI_4557290-I  GI_4557294-A GI_41406053-S
```

```
         6.17          12.69           6.25           9.46           6.25
GI_19913429-S GI_41327758-S  GI_4502352-S GI_11038670-S  GI_4826650-S
        13.53           5.94          14.74           5.42          10.01
GI_40538810-S GI_19923197-S  GI_4502598-S  GI_7108334-S  GI_4557414-S
         8.33           6.37           8.01          22.58           8.88
GI_15011912-S
         3.60
```

4 Working with multiple populations

Here we will discuss how to take multiple population-specific directors and obtain a director representing the sums across populations.

5 Session information

```
> sessionInfo()

R version 2.14.0 (2011-10-31)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] tools     splines   stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] multicore_0.1-7
 [2] SNPlocs.Hsapiens.dbSNP.20090506_0.99.6
 [3] illuminaHumanv1.db_1.12.1
 [4] GGtools_4.0.0
 [5] ff_2.2-3
 [6] bit_1.1-7
 [7] GenomicRanges_1.6.0
 [8] org.Hs.eg.db_2.6.4
```

```
 [9] rtracklayer_1.14.0
[10] RCurl_1.6-10
[11] bitops_1.0-4.1
[12] IRanges_1.12.0
[13] annotate_1.32.0
[14] AnnotationDbi_1.16.0
[15] GGBase_3.14.0
[16] genefilter_1.36.0
[17] RSQLite_0.10.0
[18] DBI_0.2-5
[19] snpStats_1.4.0
[20] Matrix_1.0-1
[21] lattice_0.20-0
[22] survival_2.36-10
[23] Biobase_2.14.0

loaded via a namespace (and not attached):
[1] BSgenome_1.22.0   Biostrings_2.22.0 KernSmooth_2.23-6 Rsamtools_1.6.0
[5] XML_3.4-3         grid_2.14.0       xtable_1.6-0      zlibbioc_1.0.0
```