Dataset Merging

UW GCC

June 25, 2015

This document gives an example of how to merge multiple datasets into a single GDS file. It addresses related issues of genome build conversion, retrieving up-to-date rsIDs, and harmonizing SNP allele nomenclature (i.e., genotype coding and strand). Example data are taken from three publicly available datasets: HapMap, 1000 Genomes Project, and the Human Genome Diversity Project.

Contents

1 Update all datasets to a common build 1

2 Choose a common allele coding 5

3 Select samples and SNPs from each dataset 6

4 Merge on common SNPs 8

5 Check against original files 8

6 Combine annotations 9

7 Check duplicate discordance 10

1 Update all datasets to a common build

Dataset 1: HapMap 3 samples genotyped on the Illumina Human 1M array, build 36.

```
library(QCpipeline)

## Loading required package:  GWASTools
## Warning:  package 'GWASTools' was built under R version 3.2.1
## Loading required package:  Biobase
## Loading required package:  BiocGenerics
## Loading required package:  parallel
##
## Attaching package:  'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##    clusterExport, clusterMap, parApply, parCapply, parLapply,
##    parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
##
## The following object is masked from 'package:stats':
##
##     xtabs
##
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, as.vector, cbind,
##     colnames, do.call, duplicated, eval, evalq, Filter, Find, get,
##     grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rep.int, rownames, sapply,
##     setdiff, sort, table, tapply, union, unique, unlist, unsplit
##
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'.  To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
library(QCannot)
dir <- "exampleDatasets"
gdsfile <- file.path(dir, "hapmap3_1M_b36.gds")
scanfile <- file.path(dir, "hapmap3_1M_b36_scanAnnot.RData")
snpfile <- file.path(dir, "hapmap3_1M_b36_snpAnnot.RData")
hm.gds <- GdsGenotypeReader(gdsfile)
hm.scanAnnot <- getobj(scanfile)
hm.snpAnnot <- getobj(snpfile)
hmData <- GenotypeData(hm.gds, scanAnnot=hm.scanAnnot, snpAnnot=hm.snpAnnot)
```

First we need to update this dataset from build 36 to build 37[1]. convertBuild takes a SNP annotation object and uses liftOver to get chromosome and position in the new build. The R package liftOver is an R implementation of the UCSC Genome Browser's liftOver tool[2].

```
## need to specify a chain file for liftOver
## chain files were downloaded from the UCSC Browser ftp site
chain.file <- file.path("/projects/geneva/gcc-fs2/SNP_annotation/UCSC_downloads",
                        "hg18/liftOver/hg18ToHg19.over.chain")
## if converting from build 37 to build 36, the chain file argument would be:
## chain.file <- file.path("/projects/geneva/gcc-fs2/SNP_annotation/UCSC_downloads",
##                           "hg19/liftOver/hg19ToHg18.over.chain")
converted <- convertBuild(hm.snpAnnot, chain.file)
```

```
## 9973 input positions successfully converted; 3 positions failed conversion (will be set
to NA)
## 99.9699% successful conversions
```

```
map <- converted[,c("rsID", "chromosome", "position",
```

[1]Note that there is more than one way to refer to a genome build. Build 36 may be called "NCBI36" or - in UCSC Browser nomenclature - "hg18." Similarly, build 37 may be called "GRCh37" or "hg19," and build 38 may be called "GRCh38." The prefix "GRC" stands for Genome Reference Consortium, a group that essentially took over the curation of the human reference sequence starting with build 37.

[2]http://genome.ucsc.edu/cgi-bin/hgLiftOver

```
                    "rsID", "chromosome.converted", "position.converted")]
names(map) <- c("old.rsID", "old.chromosome", "old.position",
                "new.rsID", "new.chromosome", "new.position")
outPrefix <- file.path(dir, "hapmap3_1M_b37")

## gdsUpdateBuild will create a new GDS file reflecting the updated
## (i.e. build converted) positions
## set remove.unmapped=FALSE to keep unmapped (chromosome=27) SNPs in new file
gdsUpdateBuild(hmData, map, outPrefix=outPrefix, remove.unmapped=FALSE)
## close hmData
close(hmData)
```

A note on the "rsID" SNP identifier and build updates. The convertBuild function converts map information only (chromosome and base pair pair), *not* SNP names. In the example usage of gdsUpdateBuild given above, "old.rsID" and "new.rsID" are the same. However, other sources may also be used for getting updated build information, including array-specific build conversion files provided by Illumina. Illumina's build conversion information includes updates for chromosome, position, and potentailly also rsIDs. Therefore when running gdsUpdateBuild with information from Illumina build conversion files, "old.rsID" and "new.rsID" may potentially hold different information.

To obtain updated rsIDs in a given build, you can use the getSnpNames function. This function will use dbSNP tables downloaded from the UCSC Browser ftp site to look up rsIDs at a given SNP position. While not required for this dataset merging exercise, getSnpNames is illustrated at the end of this section, with the 1000 Genomes dataset (which we know to have some non-rsIDs).

```
## Continuing on from gdsUpdateBuild...
## re-open new files
gdsfile <- file.path(dir, "hapmap3_1M_b37.gds")
snpfile <- file.path(dir, "hapmap3_1M_b37_snpAnnot.RData")
hm.gds <- GdsGenotypeReader(gdsfile)
new.snpAnnot <- getobj(snpfile)
cols <- intersect(varLabels(new.snpAnnot), varLabels(hm.snpAnnot))
varMetadata(new.snpAnnot)[cols,] <- varMetadata(hm.snpAnnot)[cols,,drop=FALSE]
hm.snpAnnot <- new.snpAnnot
hmData <- GenotypeData(hm.gds, scanAnnot=hm.scanAnnot, snpAnnot=hm.snpAnnot)
```

Dataset 2: Human Genome Diversity Project (HGDP) samples genotyped on the Illumina 650Y array, build 37.

```
gdsfile <- file.path(dir, "HGDP_650Y_b37.gds")
scanfile <- file.path(dir, "HGDP_650Y_b37_scanAnnot.RData")
snpfile <- file.path(dir, "HGDP_650Y_b37_snpAnnot.RData")
hg.gds <- GdsGenotypeReader(gdsfile)
hg.scanAnnot <- getobj(scanfile)
hg.snpAnnot <- getobj(snpfile)
hgData <- GenotypeData(hg.gds, scanAnnot=hg.scanAnnot, snpAnnot=hg.snpAnnot)
```

Dataset 3: 1000 Genomes Project samples genotyped on the Illumina Omni2.5M array, build 37.

```
gdsfile <- file.path(dir, "1000G_omni25_b37.gds")
scanfile <- file.path(dir, "1000G_omni25_b37_scanAnnot.RData")
snpfile <- file.path(dir, "1000G_omni25_b37_snpAnnot.RData")
```

```
kg.gds <- GdsGenotypeReader(gdsfile)
kg.scanAnnot <- getobj(scanfile)
kg.snpAnnot <- getobj(snpfile)
kgData <- GenotypeData(kg.gds, scanAnnot=kg.scanAnnot, snpAnnot=kg.snpAnnot)
```

We'll take a quick aside here to give an example of the getSnpNames function, as there are some non-rsID identifiers in these 1000 Genomes data.

```
## Side example of using getSnpNames to obtain updated rsID identifiers:
## preview current rsID field
kg.rsID.curr <- getVariable(kg.snpAnnot,"rsID")
non.rs <- grep("^rs",kg.rsID.curr, invert=TRUE, value=TRUE)
length(non.rs); head(non.rs)

## [1] 4410
## [1] "SNP1-893108"  "SNP1-2381751" "SNP1-3784281" "SNP1-3904110"
## [5] "SNP1-4528430" "SNP1-4751808"

# Open the GDS file containing the dbSNP table information in build 37
library(gdsfmt)

## Warning:  package 'gdsfmt' was built under R version 3.2.1

gds.dbSNP.fn <- file.path("/projects/geneva/gcc-fs2/SNP_annotation/UCSC_downloads",
                          "hg19/database/snp138.gds")
gds.dbSNP <- openfn.gds(gds.dbSNP.fn)
# getSnpNames must be run one chromosome at a time
chrom <- 22
snpChr <- kg.snpAnnot[kg.snpAnnot$chromosome %in% chrom,]
# preview SNP annotation, now in build 37
head(pData(snpChr),3)

##           snpID chromosome position alleleA alleleB      rsID   missing.n1
## 2353540 2353540         22 17677699       C       T rs5747018 0.0000000000
## 2353681 2353681         22 17800472       C       T rs9619055 0.0004710316
## 2353704 2353704         22 17829955       T       C rs2401081 0.0009420631
##            A.freq duplicated alleleA.top alleleB.top
## 2353540 0.2932171      FALSE           G           A
## 2353681 0.6569274      FALSE           G           A
## 2353704 0.7866572      FALSE           A           G

out <- getSnpNames(snpAnnot=snpChr, gds=gds.dbSNP, chromosome=chrom,
                   extraSnptabVars="refNCBI")

## Reading chromosome info...
## Reading SNP info for chromosome 22...
##  name
##  chromStart
##  chromEnd
##  observed
##  strand
##  class
##  exceptions
##  refNCBI
## Selecting SNPs...
```

```
# Returns a list
names(out)

## [1] "merged" "snptab"

# preview the SNP annotation merged with the dbSNP table
snp.names <- out[["merged"]]
# preview where rsID was found for non-rsID variants:
head(snp.names[grep("^rs",snp.names$rsID,invert=TRUE),],3)

##      snpID chromosome position alleleA alleleB           rsID  missing.n1
## 5  2354229         22 18293261       A       G SNP22-16673261 0.000000000
## 8  2355257         22 19553880       T       C SNP22-17933880 0.000000000
## 12 2356009         22 20097730       C       T SNP22-18477730 0.004710316
##       A.freq duplicated alleleA.top alleleB.top chrom       name
## 5  0.6314178      FALSE           A           G chr22  rs1296703
## 8  0.9863401      FALSE           A           G chr22  rs8142170
## 12 0.9176526      FALSE           G           A chr22 rs41281423
##    chromStart observed strand  class exceptions refNCBI snptab.valid
## 5    18293260      A/G      + single                  A         TRUE
## 8    19553879      C/T      + single                  T         TRUE
## 12   20097729      C/T      + single                  C         TRUE
##    snpAnnot.valid
## 5            TRUE
## 8            TRUE
## 12           TRUE

# close GDS with dbSNP table information
closefn.gds(gds.dbSNP)

# optionally -- loop through all chromosomes and update rsID in
# kg.snpAnnot with "name" field returned by getSnpNames.
```

2 Choose a common allele coding

Before we can merge these datasets, we need to make sure that their allele coding is the same. Recall that GDS files represent genotypes as a count (0,1,2) or dosage ([0,2]) of a specified allele, referred to as "alleleA." For Illumina arrays, this count is traditionally of the Illumina A allele, as defined in Illumina's TOP/BOT and A/B nomenclature system[3]. The GDS files we are merging together do not necessarily have to be counting a consistent allele. For example, one file can have alleleA=C and alleleB=T while the other has alleleA=T and alleleB=C. The GDS files do, however, need to have accompanying SNP annotations that define alleles on a consistent strand: e.g., TOP, plus(+), or FORWARD[4]. The GDS merging function will then be able to harmonize the different GDS files to count a consistent allele as "alleleA," by default taking the convention of the first dataset listed.

```
varMetadata(hm.snpAnnot)[grep("allele", varLabels(hm.snpAnnot)),,drop=FALSE]
```

[3] http://res.illumina.com/documents/products/technotes/technote_topbot.pdf

[4] For more on SNP allele nomenclature, see Nelson, S.C., Laurie, C.C., Doheny, K.F. & Mirel, D.B. Is 'forward' the same as 'plus'?...and other adventures in SNP allele nomenclature. Trends in Genetics 28, 361-363 (2012)

```
##         labelDescription
## alleleA     TOP allele A
## alleleB     TOP allele B

varMetadata(hg.snpAnnot)[grep("allele", varLabels(hg.snpAnnot)),,drop=FALSE]

##                        labelDescription
## alleleA     TOP alleleA (coded allele)
## alleleB TOP alleleB (non-coded allele)

varMetadata(kg.snpAnnot)[grep("allele", varLabels(kg.snpAnnot)),,drop=FALSE]

##                                labelDescription
## alleleA     reference allele on the plus strand
## alleleB     alternate allele on the plus strand
## alleleA.top  reference allele on the TOP strand
## alleleB.top alternate allele on the TOP strand)
```

We need to set "alleleA" and "alleleB" to TOP for 1000 Genomes.

```
kg.snpAnnot$alleleA <- kg.snpAnnot$alleleA.top
kg.snpAnnot$alleleB <- kg.snpAnnot$alleleB.top
kgData <- GenotypeData(kg.gds, scanAnnot=kg.scanAnnot, snpAnnot=kg.snpAnnot)
```

3 Select samples and SNPs from each dataset

To merge the datasets with gdsMerge, we make a list containing all GenotypeData objects. The names of the list items will be used to refer to each dataset in the output annotation.

```
genoDataList <- list(hmData, hgData, kgData)
names(genoDataList) <- c("HapMap3", "HGDP", "1000G")
```

We select samples to include from each dataset (the default is to include all). This list must have the same names as genoDataList. Here we include unrelated samples.

```
lapply(genoDataList, getScanVariableNames)

## $HapMap3
##  [1] "coriell.id"   "genorun.id"   "family"       "father"
##  [5] "mother"       "sex"          "pop.group"    "scanID"
##  [9] "subjectID"    "unrelated"    "family.alias"
##
## $HGDP
##  [1] "subjectID"            "scanID"               "sex"
##  [4] "population"           "geographic.origin"    "continental.region"
##  [7] "unrelated.rosenberg" "missing.e1"           "unrelated.deg2"
## [10] "unrelated.deg3"
##
## $`1000G`
##  [1] "family"
##  [2] "Individual.ID"
##  [3] "father"
##  [4] "mother"
##  [5] "Gender"
##  [6] "Phenotype"
##  [7] "Population"
##  [8] "Other.info"
##  [9] "Relationships.from.Pemberton.et.al.AJHG.2010"
## [10] "scanID"
## [11] "sex"
## [12] "missing.e1"
## [13] "missing.e1.auto"
## [14] "missing.e1.xchr"
## [15] "family.inferred"
## [16] "unrelated.2ndDeg"
## [17] "unrelated.3rdDeg"

sampleList <- list(hm.scanAnnot$scanID[hm.scanAnnot$unrelated],
                   hg.scanAnnot$scanID[hg.scanAnnot$unrelated.deg3],
                   kg.scanAnnot$scanID[kg.scanAnnot$unrelated.3rdDeg])
names(sampleList) <- names(genoDataList)
unlist(lapply(sampleList, length))

## HapMap3    HGDP   1000G
##      42      27      44
```

We select SNPs to include from each dataset (the default is to include all matching SNPs). This list must have the same names as genoDataList. Here we select high-quality SNPs from the datasets with quality information and remove duplicate SNPs on the Omni2.5.

```
lapply(genoDataList, getSnpVariableNames)

## $HapMap3
## [1] "snpID"      "chromosome" "position"   "rsID"       "alleleA"
## [6] "alleleB"    "old.snpID"
##
## $HGDP
## [1] "snpID"          "chromosome"     "position"       "rsID"
## [5] "alleleA"        "alleleB"        "missing.n1"     "hwe.pval"
## [9] "quality.filter"
##
## $`1000G`
##  [1] "snpID"       "chromosome"  "position"    "alleleA"     "alleleB"
##  [6] "rsID"        "missing.n1"  "A.freq"      "duplicated"  "alleleA.top"
## [11] "alleleB.top"

snpList <- list(hm.snpAnnot$snpID,
                hg.snpAnnot$snpID[hg.snpAnnot$quality.filter],
                kg.snpAnnot$snpID[kg.snpAnnot$missing.n1 < 0.02 & !kg.snpAnnot$duplicated])
names(snpList) <- names(genoDataList)
unlist(lapply(snpList, length))
```

```
## HapMap3    HGDP   1000G
##    9976    9883    9768
```

4 Merge on common SNPs

We can match SNPs on position, name, or both. If matching on name, the column in each SNP annotation with the SNP identifier must be given in snpNameList (usually rsID or similar). Because the same SNP may be duplicated on an array with multiple names, or rsID may change with annotation updates, we typically choose to match on position only. Alleles are always required to match, while allowing for the the possibilty of reversed A/B alleles. The A/B alleles in the output will be taken from the first dataset.

The default is to sort samples by scanID. If sortByScanID is FALSE, each dataset will be in a contiguous block in the new file. Generating a new set of snpIDs 1:N is also the default. If newSnpID=FALSE, the snpIDs from the first dataset in the list will be used.

```
outPrefix <- file.path(dir, "1000G_HGDP_HapMap3")
gdsMerge(genoDataList, outPrefix=outPrefix,
         sampleList=sampleList, snpList=snpList,
         match.snps.on="position",
         sortByScanID=TRUE, newSnpID=TRUE)

## 42 samples included from HapMap3
## 27 samples included from HGDP
## 44 samples included from 1000G
## Matching SNPs...
## 4922 SNPs matched on chromosome, position, alleles
## 0 alleles swapped for HGDP
## 2617 alleles swapped for 1000G
## Creating new GDS file with 113 samples and 4922 SNPs
## Reading genotypes from HapMap3
## Reading genotypes from HGDP
## Reading genotypes from 1000G
## Cleaning up
## Saving annotation
```

5 Check against original files

After creating the merged file, we check it against the original files with gdsMergeCheck.

```
gdsMergeCheck(genoDataList, outPrefix)

## Reading genotypes from HapMap3
## Reading genotypes from HGDP
## Reading genotypes from 1000G
## All genotypes match!
```

6 Combine annotations

gdsMerge creates sample and SNP annotations with basic information, but we will want to fill in more columns with additional annotation from the three input datasets.

```
scanfile <- paste0(outPrefix, "_scanAnnot.RData")
scanAnnot <- getobj(scanfile)
head(pData(scanAnnot))

##   scanID
## 1 104540
## 2 106991
## 3 112735
## 4 115172
## 5 128911
## 6 142661

## merge sample annotations
hm.samp <- pData(hm.scanAnnot)[,c("scanID", "sex", "coriell.id", "pop.group")]
names(hm.samp)[3:4] <- c("subjectID", "population")
head(hm.samp)

##     scanID sex subjectID population
## 959 104540   M   NA20126        ASW
## 202 106991   F   NA18109        CHD
## 230 112735   F   NA18144        CHD
## 924 115172   M   NA19783        MEX
## 700 128911   M   NA19210        YRI
## 370 142661   F   NA18641        CHB

hg.samp <- pData(hg.scanAnnot)[,c("scanID", "sex", "subjectID", "population")]
head(hg.samp)

##      scanID sex subjectID    population
## 132  613983   M HGDP00226        Pathan
## 807  631393   M HGDP01164        Tuscan
## 1003 655978   F HGDP01369 French_Basque
## 158  661769   F HGDP00298        Kalash
## 957  692745   M HGDP01322          Lahu
## 656  709750   M HGDP00952         Yakut

kg.samp <- pData(kg.scanAnnot)[,c("scanID", "sex", "Individual.ID", "Population")]
names(kg.samp)[3:4] <- c("subjectID", "population")
head(kg.samp)

##     scanID sex subjectID population
## 116 621098   F   HG00268        FIN
## 131 624951   F   HG00285        FIN
## 251 644292   M   HG00530        CHS
## 253 644322   M   HG00532        CHS
## 294 652813   F   HG00597        CHS
## 358 663897   M   HG00731        PUR
```

```
samp <- rbind(hm.samp, hg.samp, kg.samp)
samp <- samp[match(scanAnnot$scanID, samp$scanID),]
dim(samp)

## [1] 113   4

pData(scanAnnot) <- samp
save(scanAnnot, file=scanfile)

snpfile <- paste0(outPrefix, "_snpAnnot.RData")
snpAnnot <- getobj(snpfile)
head(pData(snpAnnot))

##   snpID chromosome position alleleA alleleB snpID.HapMap3 snpID.HGDP
## 1     1          1  2082566       A       G             3         77
## 2     2          1  2280661       A       G             6        117
## 3     3          1  3276374       A       G             9        345
## 4     4          1  3285860       A       G            10        350
## 5     5          1  3312914       A       C            11        368
## 6     6          1  3473345       A       G            13        405
##   snpID.1000G
## 1        1239
## 2        1451
## 3        2619
## 4        2633
## 5        2674
## 6        2944

## get rsID from 1000genomes annotation
snpAnnot$rsID <- kg.snpAnnot$rsID[match(snpAnnot$snpID.1000G, kg.snpAnnot$snpID)]
save(snpAnnot, file=snpfile)
```

7 Check duplicate discordance

As a final check, we run duplicateDiscordance to make sure that duplicate samples across the datasets have concordant genotypes. SNPs with discordant genotypes may be removed from further analysis. SNPs with very high discordance rates may indicate a problem with the SNP annotation (e.g., a probe not actually mapping to the specified position or perhaps A/B alleles incorrectly specified). SNPs with low to moderate discordance rates suggest differences in genotyping performance across datasets, or possibly misidentification of duplicate samples.

```
gdsfile <- paste0(outPrefix, ".gds")
gds <- GdsGenotypeReader(gdsfile)
genoData <- GenotypeData(gds, scanAnnot=scanAnnot, snpAnnot=snpAnnot)
sum(duplicated(scanAnnot$subjectID))

## [1] 17

# The arguemnt "subjName.col" points to the field in the sample annotation
## that should be used to identify duplicate samples. For example, Coriell ID
## ("NAxxxx") for both HapMap and 1000 Genomes samples.
```

```
disc <- duplicateDiscordance(genoData, subjName.col="subjectID", verbose=FALSE)
disc.subj <- unlist(lapply(disc$discordance.by.subject, function(x) x[1,2]))
summary(disc.subj)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0002038 0.0002044 0.0017950 0.0004088 0.0135900

summary(disc$discordance.by.snp$discordant)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.02987 0.00000 2.00000
```

```
close(genoData)
invisible(lapply(genoDataList, close))
sessionInfo()

## R version 3.2.0 (2015-04-16)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.9.5 (Mavericks)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] gdsfmt_1.5.1        QCannot_0.1.0       QCpipeline_0.10.0
## [4] GWASTools_1.15.9    Biobase_2.29.1      BiocGenerics_0.15.2
## [7] knitr_1.10.5
##
## loaded via a namespace (and not attached):
##  [1] SummarizedExperiment_0.2.3 gtools_3.5.0
##  [3] zoo_1.7-12                 ncdf_1.6.8
##  [5] VariantAnnotation_1.15.15  reshape2_1.4.1
##  [7] DNAcopy_1.43.0             splines_3.2.0
##  [9] lattice_0.20-31            GWASExactHW_1.01
## [11] stats4_3.2.0               rtracklayer_1.29.11
## [13] GenomicFeatures_1.21.13    survival_2.38-2
## [15] XML_3.98-1.2               DBI_0.3.1
## [17] BiocParallel_1.3.27        lambda.r_1.1.7
## [19] plyr_1.8.3                 stringr_1.0.0
## [21] zlibbioc_1.15.0            Biostrings_2.37.2
## [23] futile.logger_1.4.1        caTools_1.17.1
## [25] evaluate_0.7               IRanges_2.3.12
## [27] SparseM_1.6                biomaRt_2.25.1
## [29] lmtest_0.9-34              GenomeInfoDb_1.5.8
## [31] quantreg_5.11              AnnotationDbi_1.31.17
## [33] highr_0.5                  Rcpp_0.11.6
## [35] KernSmooth_2.23-14         BSgenome_1.37.2
## [37] gdata_2.16.1                quantsmooth_1.35.0
## [39] S4Vectors_0.7.7             XVector_0.9.1
## [41] gplots_2.17.0               Rsamtools_1.21.10
## [43] stringi_0.5-2               GenomicRanges_1.21.15
## [45] grid_3.2.0                  tools_3.2.0
## [47] bitops_1.0-6                sandwich_2.3-3
## [49] magrittr_1.5                RCurl_1.95-4.6
## [51] RSQLite_1.0.0               futile.options_1.0.0
## [53] logistf_1.21                GenomicAlignments_1.5.9
## [55] compiler_3.2.0
```