yriMulti – HapMap YRI population, multiassay interfaces

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

May 2016

Contents

1 Introduction

The EBV-transformed B-cells from Yoruban donors are assayed for genotype and various genomic features in a number of prominent studies. This package helps work with relevant datasets and data structures as use cases for MultiAssayExperiment package development. A particular concern is accommodation of distributed genotype data, in this case, based on the 1000 genomes VCF files in an S3 bucket.

2 Basic data resources

2.1 Expression data

We will use the RNA-seq expression data in the *geuvPack* package.

```
library(geuvPack)
data(geuFPKM)
```

```
geuFPKM
## class: RangedSummarizedExperiment
## dim: 23722 462
## metadata(3): MIAME constrHist colDataSource
## assays(1): exprs
## rownames(23722): ENSG00000152931.6 ENSG00000183696.9 ...
##   ENSG00000257337.1 ENSG00000177494.5
## rowData names(18): source type ... tag ccdsid
## colnames(462): HG00096 HG00097 ... NA20826 NA20828
## colData names(35): Source.Name Comment.ENA_SAMPLE. ...
##   Factor.Value.laboratory. popcode
```

2.2 Methylation data

We have added 450k data from Banovich, Lan, McVicker, van de Geijn, Degner, Blischak, Roux, Pritchard, and Gilad (2014) paper to the yriMulti package.

```
library(yriMulti)
data(banovichSE)
```

```
banovichSE
## class: RangedSummarizedExperiment
## dim: 329469 64
## metadata(0):
## assays(1): betas
## rownames(329469): cg00000029 cg00000165 ... ch.9.98989607R
##   ch.9.991104F
## rowData names(10): addressA addressB ... probeEnd probeTarget
## colnames(64): NA18498 NA18499 ... NA18489 NA18909
## colData names(35): title geo_accession ... data_row_count naid
```

2.3 DnaseI hypersensitivity data

```
library(dsQTL)
if (!exists("DHStop5_hg19")) data(DHStop5_hg19)
```

```
DHStop5_hg19
## class: RangedSummarizedExperiment
## dim: 1465442 70
## metadata(2): MIAME annotation
## assays(1): scores
## rownames(1465442): dhs_chr1_10402 dhs_chr1_10502 ...
##   dhs_chr22_51228236 dhs_chr22_51234736
## rowData names(0):
## colnames(70): NA18486 NA18498 ... NA19239 NA19257
## colData names(9): naid one ... male isFounder
```

2.4 Genotype data

We take advantage of a function (`gtpath`) that generates paths to S3-resident VCF from the 1000 genomes project.

```
litvcf = readVcf(gtpath(20),
      param=ScanVcfParam(which=GRanges("20",
           IRanges(3.7e7,3.701e7))), genome="hg19")
```

```
litvcf
## class: CollapsedVCF
## dim: 347 2504
## rowRanges(vcf):
##   GRanges with 5 metadata columns: paramRangeID, REF, ALT, QUAL, FILTER
## info(vcf):
##   DataFrame with 27 columns: CIEND, CIPOS, CS, END, IMPRECISE, MC, MEINFO...
## info(header(vcf)):
##                 Number Type    Description
##    CIEND        2      Integer Confidence interval around END for imprec...
##    CIPOS        2      Integer Confidence interval around POS for imprec...
##    CS           1      String  Source call set.
##    END          1      Integer End coordinate of this variant
##    IMPRECISE    0      Flag    Imprecise structural variation
##    MC           .      String  Merged calls.
##    MEINFO       4      String  Mobile element info of the form NAME,STAR...
##    MEND         1      Integer Mitochondrial end coordinate of inserted ...
##    MLEN         1      Integer Estimated length of mitochondrial insert
##    MSTART       1      Integer Mitochondrial start coordinate of inserte...
##    SVLEN        .      Integer SV length. It is only calculated for stru...
##    SVTYPE       1      String  Type of structural variant
##    TSD          1      String  Precise Target Site Duplication for bases...
##    AC           A      Integer Total number of alternate alleles in call...
##    AF           A      Float   Estimated allele frequency in the range (...
```

```
##    NS            1      Integer Number of samples with data
##    AN            1      Integer Total number of alleles in called genotypes
##    EAS_AF        A      Float   Allele frequency in the EAS populations c...
##    EUR_AF        A      Float   Allele frequency in the EUR populations c...
##    AFR_AF        A      Float   Allele frequency in the AFR populations c...
##    AMR_AF        A      Float   Allele frequency in the AMR populations c...
##    SAS_AF        A      Float   Allele frequency in the SAS populations c...
##    DP            1      Integer Total read depth; only low coverage data ...
##    AA            1      String  Ancestral Allele. Format: AA|REF|ALT|Inde...
##    VT            .      String  indicates what type of variant the line r...
##    EX_TARGET     0      Flag    indicates whether a variant is within the...
##    MULTI_ALLELIC 0      Flag    indicates whether a site is multi-allelic
## geno(vcf):
##   SimpleList of length 1: GT
## geno(header(vcf)):
##       Number Type   Description
##    GT 1      String Genotype
length(colnames(litvcf))
## [1] 2504
length(intersect(colnames(litvcf), colnames(banovichSE)))
## [1] 52
length(intersect(colnames(litvcf), colnames(geuFPKM)))
## [1] 445
length(intersect(colnames(litvcf), colnames(DHStop5_hg19)))
## [1] 59
```

3 Some computations focused on methylation-expression association

The `yriMulti` package is currently a scratch-pad for some integrative infrastructure thoughts.

3.1 Gene-centric selection

With `mexGR`, a GRanges instance is formed with methylation scores for CpG near a gene. The assay data are placed in the mcols, with one range devoted to the expression measures.

```
m1 = mexGR(banovichSE, geuFPKM, symbol="ORMDL3")
m1
## mexGR instance with 44 metadata columns:
## NA18498, NA18499, ..., NA18909, type
## and 7 ranges
mcols(m1)[1:4,1:4]
## DataFrame with 4 rows and 4 columns
##                 NA18498      NA18499     NA18502      NA18517
##               <numeric>    <numeric>   <numeric>    <numeric>
## cg01482279 -0.592058106 -0.177502568 -0.00553931 -1.566010118
## cg02305874 -0.463354922  0.287661199 0.553863032 -1.096806505
```

```
## cg04145193  -0.22453404 -1.179221562 1.090713571  0.453933061
## cg12655416 -0.158362114 -0.899115621 1.020870227   0.53997149
table(mcols(m1)$type)
##
## expr meth
##    1    6
```

3.2 DNA-methylation association with expression

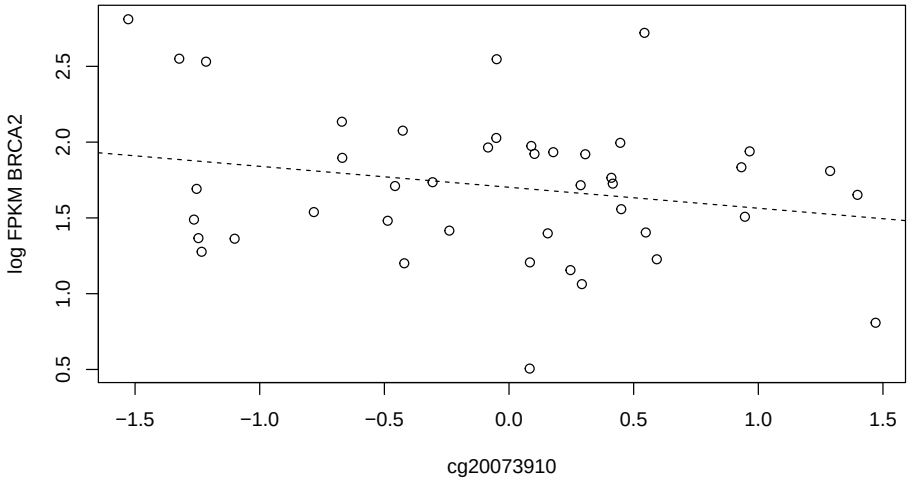
`bindelms` computes the regressions of the selected gene's expression values on the methylation scores. We have options to transform the expression value (parameter `ytx` is a function) and can indicate the radius around the gene coding region to search for CpG (parameter `gradius`).

We'll examine a region around gene BRCA2 for a CpG whose methylation score is negatively associated with BRCA2 expression.

```
b1 = bindelms(geuFPKM, banovichSE, symbol="BRCA2", ytx=log,
   gradius=20000)
b1
## class: RangedSummarizedExperiment
## dim: 29 43
## metadata(6): theCall symbol ... pwd txexpr
## assays(1): betas
## rownames(29): cg00031759 cg00214044 ... cg26458617 cg26941801
## rowData names(15): addressA addressB ... t p
## colnames(43): NA18498 NA18499 ... NA18489 NA18909
## colData names(35): title geo_accession ... data_row_count naid
mcols(b1)[1:3,]
## DataFrame with 3 rows and 15 columns
##              addressA    addressB channel platform percentGC
##            <character> <character>   <Rle>    <Rle> <numeric>
## cg00031759    11602350               Both    HM450       0.5
## cg00214044    33707391               Both    HM450      0.62
## cg00785980    12675375               Both    HM450      0.54
##                        sourceSeq probeType  probeStart    probeEnd
##                      <DNAStringSet>     <Rle> <character> <character>
## cg00031759 CGGGTATTTC...GCATCCCAAC        cg    32889486    32889535
## cg00214044 CGGGCACCAG...ACCCATATTT        cg    32885906    32885955
## cg00785980 GCCCACCTGA...TTCATTCCCG        cg    32984237    32984286
##            probeTarget
##              <numeric>
## cg00031759    32889534
## cg00214044    32885906
## cg00785980    32984237
##
##
## cg00031759          list(coefficients = c(`(Intercept)` = 1.71061243102637, `df[, i]` = 0.0074090671480769
## cg00214044 list(coefficients = c(`(Intercept)` = 1.70619587753478, `df[, i]` = 0.0396189079256385), residu
## cg00785980             list(coefficients = c(`(Intercept)` = 1.71120579164398, `df[, i]` = -0.024122378171
##                          slope                se                 t
```

```
##                      <numeric>          <numeric>          <numeric>
## cg00031759 0.00740906714807691  0.100650797876406 0.0736116086945963
## cg00214044  0.0396189079256385  0.102764594709113  0.385530717440033
## cg00785980 -0.0241223781711239 0.0849465400590847 -0.283971285403096
##                            p
##                    <numeric>
## cg00031759 0.941677399846048
## cg00214044 0.701837353572534
## cg00785980 0.777861586114457
summary(mcols(b1)$t)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.4738 -0.3809  0.3426  0.2091  0.5894  1.6641
mintind = which.min(mcols(b1)$t)
mincpg = names(b1)[mintind]
mincpg
## [1] "cg20073910"
```

```
plotEvM(b1)
```



4 Using the MultiAssayExperiment infrastructure

4.1 Construction of the MultiAssayExperiment

We will use a unified object design to reproduce the BRCA2 display just obtained.

We need a list of relevant objects and a phenodata component.

```
library(MultiAssayExperiment)
myobs = list(geuvRNAseq=geuFPKM, yri450k=banovichSE, yriDHS=DHStop5_hg19)
cold = colData(geuFPKM)
suppressWarnings({
library(MultiAssayExperiment)
mm = MultiAssayExperiment(myobs, as.data.frame(cold))
```

```
})
mm
## A MultiAssayExperiment object of 3 listed
##  experiments with user-defined names and respective classes.
##  Containing an ExperimentList class object of length 3:
##  [1] geuvRNAseq: RangedSummarizedExperiment with 23722 rows and 462 columns
##  [2] yri450k: RangedSummarizedExperiment with 329469 rows and 43 columns
##  [3] yriDHS: RangedSummarizedExperiment with 1465442 rows and 50 columns
## Features:
##  experiments() - obtain the ExperimentList instance
##  colData() - the primary/phenotype DataFrame
##  sampleMap() - the sample availability DataFrame
##  `$`, `[`, `[[` - extract colData columns, subset, or experiment
##  *Format() - convert into a long or wide DataFrame
##  assays() - convert ExperimentList to a SimpleList of matrices
```

4.2 Restriction by range

We compute the BRCA2 'gene range'.

```
library(erma)
brr = range(genemodel("BRCA2"))
## 'select()' returned 1:many mapping between keys and columns
brr
## GRanges object with 1 range and 0 metadata columns:
##       seqnames            ranges strand
##          <Rle>         <IRanges>  <Rle>
##   [1]    chr13 32889617-32973809      +
##   -------
##   seqinfo: 1 sequence from hg19 genome
```

Subset the multiassay structure to features in the vicinity of this range.

```
.subsetByRanges = function(ma, r) {
  subsetByRow(ma,r)
}
newmm = .subsetByRanges(mm, brr+20000)
newmm
## A MultiAssayExperiment object of 3 listed
##  experiments with user-defined names and respective classes.
##  Containing an ExperimentList class object of length 3:
##  [1] geuvRNAseq: RangedSummarizedExperiment with 3 rows and 462 columns
##  [2] yri450k: RangedSummarizedExperiment with 29 rows and 43 columns
##  [3] yriDHS: RangedSummarizedExperiment with 32 rows and 50 columns
## Features:
##  experiments() - obtain the ExperimentList instance
##  colData() - the primary/phenotype DataFrame
##  sampleMap() - the sample availability DataFrame
##  `$`, `[`, `[[` - extract colData columns, subset, or experiment
##  *Format() - convert into a long or wide DataFrame
```

```
##  assays() - convert ExperimentList to a SimpleList of matrices
```

We now have all the relevant features and samples. In fact we have more genes than we really wanted. But we will proceed with this selection.

4.3 All pairwise regressions

We will introduce a formula idiom to specify a collection of models of interest.

```
library(doParallel)
## Loading required package: foreach
## Loading required package: iterators
registerDoSEQ()
allLM_pw = function(fmla, mae, xtx=force, ytx=force) {
#
# formula specifies dependent and independent assays
# form all regressions of ytx(dep) on xtx(indep) for all
# pairs of dependent and independent variables defined by
# assays for samples held in common
#
lf = as.list(fmla)
nms = lapply(lf, as.character)
yel = experiments(mae)[[nms[[2]]]]
xel = experiments(mae)[[nms[[3]]]]
sy = colnames(yel)
sx = colnames(xel)
sb = intersect(sy,sx)
yel = yel[,sb]
xel = xel[,sb]
vdf = as.matrix(expand.grid( rownames(yel),
    rownames(xel), stringsAsFactors=FALSE ))
allf = apply(vdf, 1, function(x) as.formula(paste(x, collapse="~")))
alllm = foreach (i = 1:length(allf)) %dopar% {
  df = data.frame(ytx(assay(yel)[vdf[i,1],]), xtx(assay(xel)[vdf[i,2],]))
  names(df) = vdf[i,]
  lm(allf[[i]], data=df)
  }
names(alllm) = apply(vdf,1,function(x) paste(x, collapse="~"))
allts = lapply(alllm, function(x) summary(x)$coef[2, "t value"])
list(mods=alllm, tslopes=allts)
}
pwplot = function(fmla1, fmla2, mae, ytx=force, xtx=force, ...) {
#
# use fmla1 with assays as components to identify
#    two assays to regard as sources of y and x
# fmla2 indicates which features to plot
#
lf = as.list(fmla1)
nms = lapply(lf, as.character)
yel = experiments(mae)[[nms[[2]]]]
```

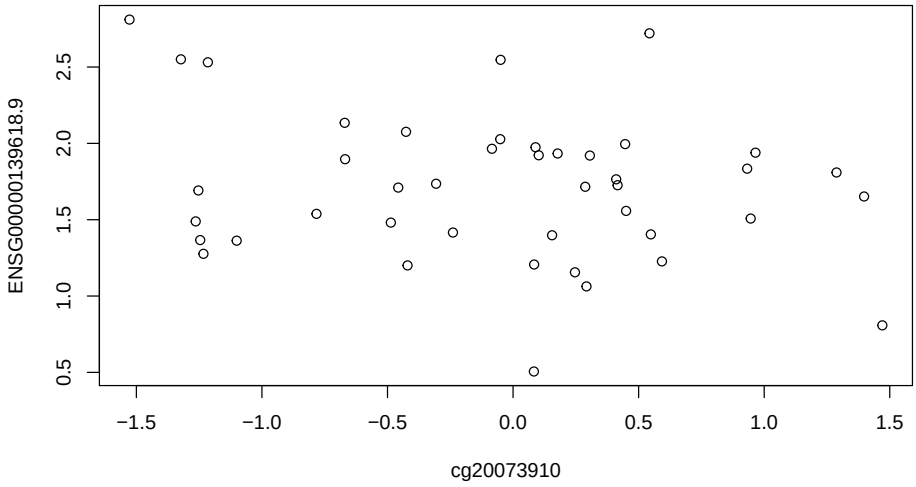
```
xel = experiments(mae)[[nms[[3]]]]
sy = colnames(yel)
sx = colnames(xel)
sb = intersect(sy,sx)
yel = yel[,sb]
xel = xel[,sb]
lf2 = lapply(as.list(fmla2), as.character)
ndf = data.frame( ytx(assay(yel)[ lf2[[2]], ]), xtx(assay(xel)[ lf2[[3]], ]) )
names(ndf) = c(lf2[[2]], lf2[[3]])
plot(fmla2, ndf, ...)
}
```

```
pp = allLM_pw(geuvRNAseq~yri450k, newmm, ytx=log)
## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced
```

```
## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced

## Warning in ytx(assay(yel)[vdf[i, 1], ]): NaNs produced
names(pp)
## [1] "mods"    "tslopes"
summary(pp[[1]][[1]])
##
## Call:
## lm(formula = allf[[i]], data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.20112 -0.30856  0.00553  0.24330  1.10330
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.710612   0.074777  22.876   <2e-16 ***
## cg00031759  0.007409   0.100651   0.074    0.942
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4898 on 41 degrees of freedom
## Multiple R-squared:  0.0001321,  Adjusted R-squared:  -0.02425
## F-statistic: 0.005419 on 1 and 41 DF,  p-value: 0.9417
which.min(unlist(pp[[2]])) # not BRCA2 but FRY
## ENSG00000073910.13~cg05918473
##                            23
```

The formula idiom can be used to isolate assays and features.

```
pwplot(geuvRNAseq~yri450k, ENSG00000139618.9~cg20073910, newmm, ytx=log)
```



4.4 Integrating dense genotypes in a VcfStack instance

We set up a reference to a collection of VCF. We'll use 1000 genomes VCF for chr21, and chr22. At present (16 Oct 2016) chrY must be kept out of the stack as it does not have the full set of samples for other chromosomes.

```
library(gQTLstats)
library(VariantAnnotation)
library(GenomicFiles)
pa = paths1kg(paste0("chr", c(21:22))) #,"Y")))
sn = vcfSamples(scanVcfHeader(TabixFile(pa[1])))
library(Homo.sapiens)  # necessary?
stopifnot(requireNamespace("GenomeInfoDb")) # indexVcf with S3 bucket? ->
ob = RangedVcfStack(VcfStack2(pa, seqinfo(Homo.sapiens)))
cd = DataFrame(id1kg=sn)
rownames(cd) = sn
colData(ob) = cd
```

Now we set up a region of interest and bind it to the stack. This yields an instance of `RangedVcfStack`, for which we have `samples`, `features`, and `assay` methods defined in gQTLstats.

```
myr = GRanges("chr22", IRanges(20e6,20.01e6))
rowRanges(ob) = myr
colData(ob) = DataFrame(colData(ob), zz=runif(nrow(colData(ob))))
hasInternetConnectivity = function()
 !is.null(nsl("www.r-project.org"))
#if (hasInternetConnectivity()) lka = assay(ob)
myobs = list(geuvRNAseq=geuFPKM, yri450k=banovichSE, yriDHS=DHStop5_hg19,
    yriGeno=ob)
suppressWarnings({
mm = MultiAssayExperiment(myobs)
})
mm
## A MultiAssayExperiment object of 4 listed
##  experiments with user-defined names and respective classes.
```

```
##  Containing an ExperimentList class object of length 4:
##  [1] geuvRNAseq: RangedSummarizedExperiment with 23722 rows and 462 columns
##  [2] yri450k: RangedSummarizedExperiment with 329469 rows and 64 columns
##  [3] yriDHS: RangedSummarizedExperiment with 1465442 rows and 70 columns
##  [4] yriGeno: RangedVcfStack with 2 rows and 2504 columns
## Features:
##  experiments() - obtain the ExperimentList instance
##  colData() - the primary/phenotype DataFrame
##  sampleMap() - the sample availability DataFrame
##  `$`, `[`, `[[` - extract colData columns, subset, or experiment
##  *Format() - convert into a long or wide DataFrame
##  assays() - convert ExperimentList to a SimpleList of matrices
```

References

```
## No encoding supplied: defaulting to UTF-8.
```