Preparation of transcript expression and genotypes from the GEUVADIS project for sQTL analysis

Malgorzata Nowicka*, Mark Robinson

October 19, 2016

This vignette describes version 1.2.0 of the *GeuvadisTranscriptExpr* package.

Contents

1 Description of the GEUVADIS project

In the GEUVADIS project [1], 462 RNA-Seq samples from lymphoblastoid cell lines were obtained. The genome sequencing data of the same individuals is provided by the 1000 Genomes Project. The samples in this project come from five populations: CEPH (CEU), Finns (FIN), British (GBR), Toscani (TSI) and Yoruba (YRI).

In the sQTL analysis, we want to identify genetic variants (here, bi-allelic SNPs) that are associated with changes in splicing. Such SNPs are then called splicing quantitative trait locies (sQTLs).

Ideally, we would like to test the associations of every SNP with every gene. However, such an approach would be very costly computationally and in terms of multiple testing correction. Under the assumption that SNPs that directly affect splicing are likely to be placed in the close surrounding of genes, we test only SNPs that are located within the gene body and within some range upstream and downstream of the gene.

*gosia.nowicka@uzh.ch

2 Downloading the GEUVADIS data

In the *GeuvadisTranscriptExpr* package, we use transcript quantification (expected counts from FluxCapacitor) and genotypes available on the GEUVADIS project website http://www.ebi.ac.uk/Tools/geuvadis-das/.

As a reference genome, we use the Gencode v12 gene annotation available on http://www.gencodegenes.org/releases/12.html, which we save in a `gencode.v12.annotation.gtf` file.

From http://www.ebi.ac.uk/arrayexpress/files/E-GEUV-1/analysis_results/, we can download a file with information about samples `E-GEUV-1.sdrf.txt` and a file with transcript quantification `GD660.TrQuantCount.txt`. Let's save them under their original names.

The VCF files with genotypes are available on http://www.ebi.ac.uk/arrayexpress/files/E-GEUV-1/genotypes/ in the compressed format.

The *R* code in this vignette assumes that all the files are saved in a current working directory.

3 Data preprocessing

In the following sections, we shows how you can prepare the GEUVADIS data for the sQTL analysis. We preprocess counts and genotypes for all the populations and chromosomes of the human genome, but the package makes available only the preprocessed data for chromosome 19 from CEU population.

The *R* code in this section does not run when the vignette is generated, mainly because it takes some time to read in the files with transcript counts and genotypes.

3.1 Annotation and metadata

To find out which SNPs should be tested with which genes, we need to know the location of genes, which we get from the GTF file. We can import it into *R* using the *rtracklayer* package. We save the extracted locations of protein coding genes as BED files (about BED format, see https://genome.ucsc.edu/FAQ/FAQformat.html) for every chromosome separately. We also remove the *'chr'* string from the chromosome names. Otherwise, we could not read in the VCF files with genotypes, which originally contain only a chromosome number.

In the *GeuvadisTranscriptExpr* package, you can access the `"genes_chr19.bed"` file.

```
library(GenomicRanges)
library(rtracklayer)
```

```
gtf_dir <- "gencode.v12.annotation.gtf"

gtf <- import(gtf_dir)

### Keep protein coding genes
keep_index <- mcols(gtf)$gene_type == "protein_coding" &
  mcols(gtf)$type == "gene"
gtf <- gtf[keep_index]
### Remove 'chr' from chromosome names
seqlevels(gtf) <- gsub(pattern = "chr", replacement = "", x = seqlevels(gtf))
```

```
genes_bed <- data.frame(chr = seqnames(gtf), start =  start(gtf),
  end = end(gtf), geneId = mcols(gtf)$gene_id,
  stringsAsFactors = FALSE)

for(i in as.character(1:22)){
  genes_bed_sub <- genes_bed[genes_bed$chr == i, ]
  write.table(genes_bed_sub, "genes_chr", i ,".bed", quote = FALSE,
    sep = "\t", row.names = FALSE, col.names = FALSE)
}
```

The metadata information about sample names and the population they originate from can be accessed from the `E-GEUV-1.sdrf.txt` file, and we save it in `samples` data frame. Additionally, we create a variable with shorter sample names because such names are used in the genotype files.

```
library(limma)
```

```
metadata_dir <- "E-GEUV-1.sdrf.txt"

samples <- read.table(metadata_dir, header = TRUE, sep = "\t", as.is = TRUE)

samples <- unique(samples[c("Assay.Name", "Characteristics.population.")])
colnames(samples) <- c("sample_id", "population")

samples$sample_id_short <- strsplit2(samples$sample_id, "\\.")[,1]
```

3.2 Transcript counts

The `GD660.TrQuantCount.txt` file contains quantification for all the populations. We split this table by population and chromosome, and use the short sample names. Then the tables are saved in separate files.

In the *GeuvadisTranscriptExpr* package, you can access the `"TrQuantCount_CEU_chr19.tsv"` file.

```
expr_dir <- "GD660.TrQuantCount.txt"

expr_all <- read.table(expr_dir, header = TRUE, sep="\t", as.is = TRUE)

expr_all <- expr_all[, c("TargetID", "Gene_Symbol", "Chr",
  samples$sample_id)]
colnames(expr_all) <- c("TargetID", "Gene_Symbol", "Chr",
  samples$sample_id_short)

for(j in "CEU"){
  for(i in 1:22){
    expr <- expr_all[expr_all$Chr == i, c("TargetID", "Gene_Symbol",
      samples$sample_id_short[samples$population == j])]
    write.table(expr, paste0("TrQuantCount_", j, "_chr", i, ".tsv"),
      quote = FALSE, sep = "\t", row.names = FALSE, col.names = TRUE)
  }
```

```
}
```

3.3 Genotypes

VCF files can be loaded into *R* using the *VariantAnnotation* package (see the Annotating Genomic Variants Bioconductor workflow https://bioconductor.org/help/workflows/variants/). To do so, we have to first `bgzip` and create indexes with `indexTabix`. Both functions are from the *Rsamtools* package.

```
library(Rsamtools)
```

```
files <- list.files(path = ".", pattern = "genotypes.vcf.gz",
  full.names = TRUE, include.dirs = FALSE)

### bgzip and index the vcf files
for(i in 1:length(files)){
  zipped <- bgzip(files[i])
  idx <- indexTabix(zipped, format = "vcf")
}
```

We are interested in bi-allelic SNPs that lay within 5000 bases of a gene. Additionally, we want to convert the genotype information into 0 for ref/ref, 1 for ref/not ref, 2 for not ref/not ref, -1 or NA for missing values. Newly encoded genotypes are saved as text files, one file for each chromosome and population.

In the *GeuvadisTranscriptExpr* package, you can access the `"genotypes_CEU_chr19.tsv"` file.

```
library(VariantAnnotation)
library(tools)
```

```
### Extended gene ranges
window <- 5000
gene_ranges <- resize(gtf, GenomicRanges::width(gtf) + 2 * window,
  fix = "center")

chr <- gsub("chr", "", strsplit2(files, split = "\\.")[, 2])

for(j in "CEU"){
  for(i in 1:length(files)){
    cat(j, chr[i], fill = TRUE)

    zipped <- paste0(file_path_sans_ext(files[i]), ".bgz")
    idx <- paste0(file_path_sans_ext(files[i]), ".bgz.tbi")
    tab <- TabixFile(zipped, idx)

    ### Explore the file header with scanVcfHeader
    hdr <- scanVcfHeader(tab)
    print(all(samples$sample_id_short %in% samples(hdr)))

    ### Read a subset of VCF file
    gene_ranges_tmp <- gene_ranges[seqnames(gene_ranges) == chr[i]]
```

```
    param <- ScanVcfParam(which = gene_ranges_tmp, samples =
        samples$sample_id_short[samples$population == j])
    vcf <- readVcf(tab, "hg19", param)

    ### Keep only the bi-allelic SNPs
    # width of ref seq
    rw <- width(ref(vcf))
    # width of first alt seq
    aw <- unlist(lapply(alt(vcf), function(x) {width(x[1])}))
    # number of alternate genotypes
    nalt <- elementLengths(alt(vcf))
    # select only bi-allelic SNPs (monomorphic OK, so aw can be 0 or 1)
    snp <- rw == 1 & aw <= 1 & nalt == 1
    # subset vcf
    vcfbi <- vcf[snp,]

    rowdata <- rowData(vcfbi)

    ### Convert genotype into number of alleles different from reference
    geno <- geno(vcfbi)$GT
    geno01 <- geno
    geno01[,] <- -1
    geno01[geno %in% c("0/0", "0|0")] <- 0 # REF/REF
    geno01[geno %in% c("0/1", "0|1", "1/0", "1|0")] <- 1 # REF/ALT
    geno01[geno %in% c("1/1", "1|1")] <- 2 # ALT/ALT
    mode(geno01) <- "integer"

    genotypes <- unique(data.frame(chr = seqnames(rowdata),
      start = start(rowdata), end = end(rowdata), snpId = rownames(geno01),
      geno01, stringsAsFactors = FALSE))

    ### Sort SNPs by position
    genotypes <- genotypes[order(genotypes[ ,2]), ]

    write.table(genotypes, file = paste0("genotypes_", j, "_chr",
      chr[i], ".tsv"), quote = FALSE, sep = "\t", row.names = FALSE,
      col.names = TRUE)

  }
}
```

APPENDIX

A Session information

```
sessionInfo()
## R version 3.3.1 (2016-06-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.1 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C                 LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8      LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                    LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
##  [1] tools     parallel  stats4    stats     graphics  grDevices utils     datasets
##  [9] methods   base
##
## other attached packages:
##  [1] VariantAnnotation_1.20.0   SummarizedExperiment_1.4.0 Biobase_2.34.0
##  [4] Rsamtools_1.26.0           Biostrings_2.42.0          XVector_0.14.0
##  [7] limma_3.30.0               rtracklayer_1.34.0         GenomicRanges_1.26.0
## [10] GenomeInfoDb_1.10.0        IRanges_2.8.0              S4Vectors_0.12.0
## [13] BiocGenerics_0.20.0        knitr_1.14
##
## loaded via a namespace (and not attached):
##  [1] AnnotationDbi_1.36.0     magrittr_1.5             zlibbioc_1.20.0
##  [4] GenomicAlignments_1.10.0 BiocParallel_1.8.0       BSgenome_1.42.0
##  [7] lattice_0.20-34          stringr_1.1.0            highr_0.6
## [10] grid_3.3.1               DBI_0.5-1                digest_0.6.10
## [13] Matrix_1.2-7.1           formatR_1.4              codetools_0.2-15
## [16] bitops_1.0-6             biomaRt_2.30.0           RCurl_1.95-4.8
## [19] RSQLite_1.0.0            evaluate_0.10            stringi_1.1.2
## [22] GenomicFeatures_1.26.0   XML_3.98-1.4             BiocStyle_2.2.0
```

B References

References

[1] T. Lappalainen, M. Sammeth, M. R. Friedländer, P. A. C. 't Hoen, J. Monlong, M. A. Rivas, M. Gonzàlez-Porta, N. Kurbatova, T. Griebel, P. G. Ferreira, M. Barann, T. Wieland, L. Greger, M. van Iterson, J. Almlöf, P. Ribeca, I. Pulyakhina, D. Esser, T. Giger, A. Tikhonov, M. Sultan, G. Bertier, D. G. MacArthur, M. Lek, E. Lizano, H. P. J. Buermans, I. Padioleau, T. Schwarzmayr, O. Karlberg, H. Ongen, H. Kilpinen, S. Beltran,

M. Gut, K. Kahlem, V. Amstislavskiy, O. Stegle, M. Pirinen, S. B. Montgomery, P. Donnelly, M. I. McCarthy, P. Flicek, T. M. Strom, H. Lehrach, S. Schreiber, R. Sudbrak, A. Carracedo, S. E. Antonarakis, R. Häsler, A.-C. Syvänen, G.-J. van Ommen, A. Brazma, T. Meitinger, P. Rosenstiel, R. Guigó, I. G. Gut, X. Estivill, and E. T. Dermitzakis, "Transcriptome and genome sequencing uncovers functional variation in humans.," *Nature*, vol. 501, no. 7468, pp. 506–11, 2013.