Process scRNA-Seq reads in *scruff*

Zhe Wang

2018-05-26

Package

scruff 0.99.11

Contents

1 Introduction

scruff is a toolkit for processing single cell RNA-seq FASTQ reads generated by CEL-Seq and CEL-Seq2 protocols. It does demultiplexing, alignment, Unique Molecular Identifier (UMI) filtering, and transcript counting in an automated fashion and generates the gene count matrix, QC metrics and provides visualizations of data quality. This vignette provides a brief introduction to the *scruff* package by walking through the demultiplexing, alignment, and UMI-counting of a built-in publicly available example dataset (van den Brink, et al. 2017).

2 Quick Start

```
# Run scruff on example dataset
# NOTE: Requires Rsubread index and TxDb objects for the reference genome.
# For generation of these files, please refer to the Stepwise Tutorial.

library(scruff)

# Get the paths to example FASTQ, FASTA, and GTF files.
v1h1R1 <- system.file("extdata",
                      "vandenBrink_1h1_L001_R1_001.fastq.gz",
                      package = "scruff")
v1h1R2 <- system.file("extdata",
                      "vandenBrink_1h1_L001_R2_001.fastq.gz",
                      package = "scruff")
vb1R1 <- system.file("extdata",
                      "vandenBrink_b1_L001_R1_001.fastq.gz",
                      package = "scruff")
vb1R2 <- system.file("extdata",
                      "vandenBrink_b1_L001_R2_001.fastq.gz",
                      package = "scruff")
fasta <- system.file("extdata", "GRCm38_MT.fa", package = "scruff")
gtf <- system.file("extdata", "GRCm38_MT.gtf", package = "scruff")
```

Build Rsubread alignment index. This is for the alignment step. For test porpurse, here we are aligning the example FASTQ files to the genes on mitochondrial chromosome only.

```
# NOTE: Rsubread package does not support Windows environment.
if (!requireNamespace("Rsubread", quietly = TRUE)) {
  message(paste("Package \"Rsubread\" needed for \"alignRsubread\"",
              "function to work.\n",
              "Please install it if you are using Linux or macOS.\n",
              "The function is not available in Windows environment.\n"),
          call. = FALSE)
  } else {
    # Create index files for GRCm38_MT.
    # For details, please refer to Rsubread user manual.
    # Specify the basename for Rsubread index
    indexBase <- "GRCm38_MT"
    Rsubread::buildindex(basename = indexBase,
                         reference = fasta,
                         indexSplit = FALSE)
}
## Package "Rsubread" needed for "alignRsubread" function to work.
##  Please install it if you are using Linux or macOS.
##  The function is not available in Windows environment.
## FALSE
```

Now that everything is ready, we can run *scruff*. In sample 1h1, cell barcodes 95 and 96 are empty well controls. In sample b1, cell barcode 95 is bulk sample containing 300 cells. These information can be set by the `cellPerWell` argument. *scruff* makes use of the *SingleCellExperiment* package. The following command returns a `SingleCellExperiment` object containing UMI filtered count matrix as well as gene and sample annotations and QC metrics.

```
data(barcodeExample, package = "scruff")

if (!requireNamespace("Rsubread", quietly = TRUE)) {
  message(paste("Package \"Rsubread\" needed.\n",
                "Please install it if you are using Linux or macOS systems.\n",
                "The function is not available in Windows environment.\n"),
          call. = FALSE)
  } else {
    sce <- scruff(project = "example",
                  sample = c("1h1", "b1"),
                  lane = c("L001", "L001"),
                  read1Path = c(v1h1R1, vb1R1),
                  read2Path = c(v1h1R2, vb1R2),
                  bc = barcodeExample,
                  index = indexBase,
                  unique = FALSE,
                  nBestLocations = 1,
                  reference = gtf,
                  bcStart = 1,
                  bcStop = 8,
                  umiStart = 9,
                  umiStop = 12,
                  keep = 75,
                  cellPerWell = c(rep(1, 94), 0, 0, rep(1, 94), 300, 1),
                  cores = 2,
                  verbose = TRUE)
}
```

Visualize data quality.

```
data(sceExample, package = "scruff")
qc <- qcplots(sceExample)
```

3 Stepwise Tutorial

3.1 Load Example Dataset

The *scruff* package contains 4 single cell RNA-seq FASTQ example files. Each file has 10,000 sequenced reads.

```
library(scruff)
v1h1R1 <- system.file("extdata",
                      "vandenBrink_1h1_L001_R1_001.fastq.gz",
                      package = "scruff")
v1h1R2 <- system.file("extdata",
                      "vandenBrink_1h1_L001_R2_001.fastq.gz",
                      package = "scruff")
vb1R1 <- system.file("extdata",
                      "vandenBrink_b1_L001_R1_001.fastq.gz",
                      package = "scruff")
vb1R2 <- system.file("extdata",
                      "vandenBrink_b1_L001_R2_001.fastq.gz",
                      package = "scruff")
```

3.2 Demultiplex and Assign Cell Specific Reads

Now the FASTQ files are ready to be demultiplexed. *scruff* package provides built-in predefined cell barcodes `barcodeExample` for demultiplexing the example dataset. In the example FASTQ files, read 1 contains cell barcode and UMI sequence information. Read 2 contains transcript sequences. The barcode sequence of each read starts at base 1 and ends at base 8. The UMI sequence starts at base 9 and ends at base 12. They can be set via `bcStart`, `bcStop`, and `umiStart`, `umiStop` arguments. By default, reads with any nucleotide in the barcode and UMI sequences with sequencing quality lower than 10 (Phred score) will be excluded. The following command demultiplexes the example FASTQ reads and trims reads longer than 75 nucleotides. The command returns a `SingleCellExperiment` object whose `colData` contains the cell index, barcode, reads, percentage of reads assigned, sample, and FASTQ file path information for each cell. By default, the cell specific demultiplexed fastq.gz files are stored in `./Demultiplex` folder.

```
data(barcodeExample, package = "scruff")
de <- demultiplex(project = "example",
                  sample = c("1h1", "b1"),
                  lane = c("L001", "L001"),
                  read1Path = c(v1h1R1, vb1R1),
                  read2Path = c(v1h1R2, vb1R2),
                  barcodeExample,
                  bcStart = 1,
                  bcStop = 8,
                  bcEdit = 0,
                  umiStart = 9,
                  umiStop = 12,
                  keep = 75,
                  minQual = 10,
                  yieldReads = 1e+06,
                  cores = 2,
                  verbose = TRUE,
                  overwrite = TRUE)
```

3.3 Alignment

scruff provides an alignment function `alignRsubread` which is a wrapper function to `align` in `Rsubread` package. It aligns the reads to reference sequence index and outputs sequence alignment map files in "BAM" or "SAM" format. For demonstration purpose, the built-in mitochondrial DNA sequence from GRCm38 reference assembly `GRCm38MitochondrialFasta` will be used to map the reads. First, a *Rsubread* index for the reference sequence needs to be generated.

```
# Create index files for GRCm38_MT. For details, please refer to Rsubread user manual.
fasta <- system.file("extdata", "GRCm38_MT.fa", package = "scruff")
# NOTE: Rsubread package does not support Windows environment.
if (!requireNamespace("Rsubread", quietly = TRUE)) {
  message(paste("Package \"Rsubread\" needed.\n",
                "Please install it if you are using Linux or macOS systems.\n",
                "The function is not available in Windows environment.\n"),
          call. = FALSE)
  } else {
    # Create index files for GRCm38_MT.
    # For details, please refer to Rsubread user manual.
    # Specify the basename for Rsubread index
    indexBase <- "GRCm38_MT"
    Rsubread::buildindex(basename = indexBase,
                         reference = fasta,
                         indexSplit = FALSE)
}
## Package "Rsubread" needed.
##  Please install it if you are using Linux or macOS systems.
##  The function is not available in Windows environment.
## FALSE
```

The following command maps the FASTQ files to GRCm38 mitochondrial reference sequence `GRCm38_MT.fa` and returns a `SingleCellExperiment` object. By default, the files are stored in BAM format in `./Alignment` folder.

```
# Align the reads using Rsubread
if (requireNamespace("Rsubread", quietly = TRUE)) {
  al <- alignRsubread(de,
                      indexBase,
                      unique = FALSE,
                      nBestLocations = 1,
                      format = "BAM",
                      cores = 2,
                      overwrite = TRUE,
                                 verbose = TRUE)
        }
```

3.4 UMI correction and Generation of Count Matrix

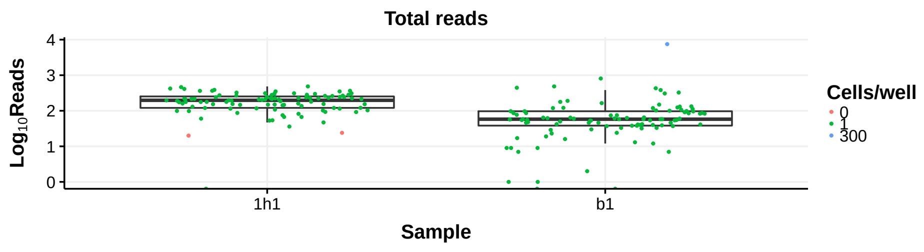
Example GTF file `GRCm38_MT.gtf` will be used for feature counting. Currently, *scruff* applies the union counting mode of the HTSeq Python package. The following command generates the UMI filtered count matrix for the example dataset.

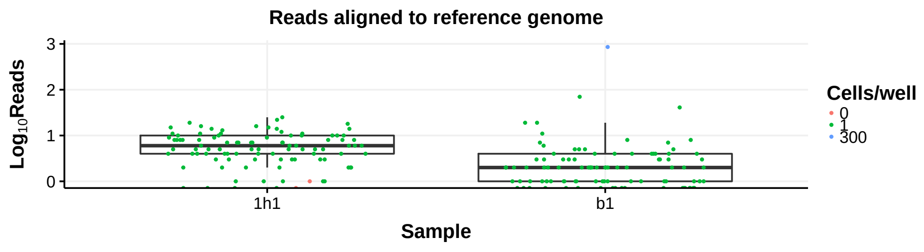
```
gtf <- system.file("extdata", "GRCm38_MT.gtf", package = "scruff")
# get the molecular counts of trancsripts for each cell
# In sample 1h1, cell barcodes 95 and 96 are empty well controls. In sample b1, cell barcode 95 is bulk samp
if (requireNamespace("Rsubread", quietly = TRUE)) {
  sce = countUMI(al,
                 gtf,
                 format = "BAM",
                 cellPerWell = c(rep(1, 94), 0, 0, rep(1, 94), 300, 1),
                 cores = 2,
                 verbose = TRUE)
}
```

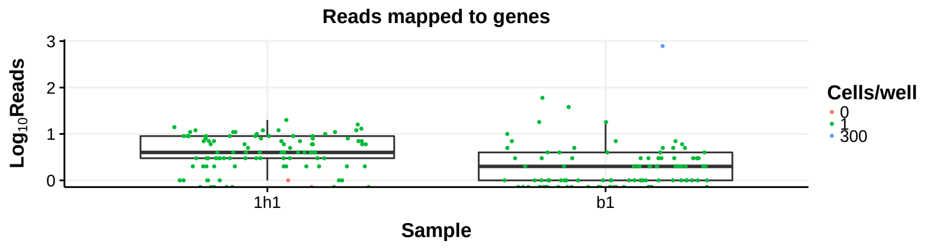
3.5 Visualization of QC metrics

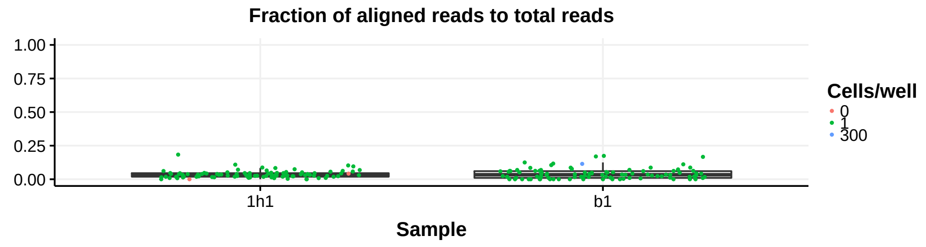
The data quality diagnostic information are contained in the `colData` of the returned `SingleCellExperiment` object `sce`. They can be visualized using the `qcplots` function.

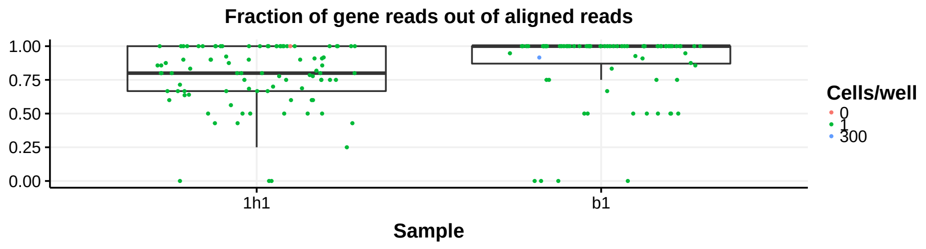
```
data(sceExample, package = "scruff")
qc <- qcplots(sceExample)
qc
```

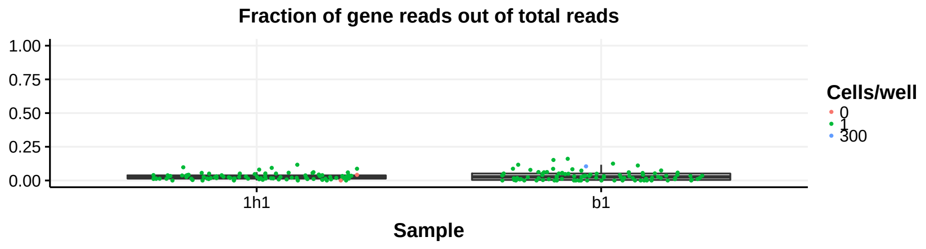


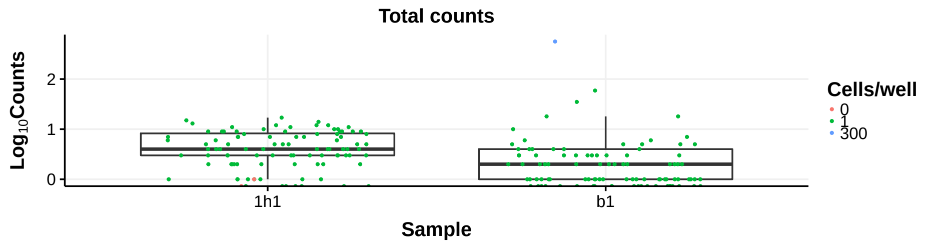


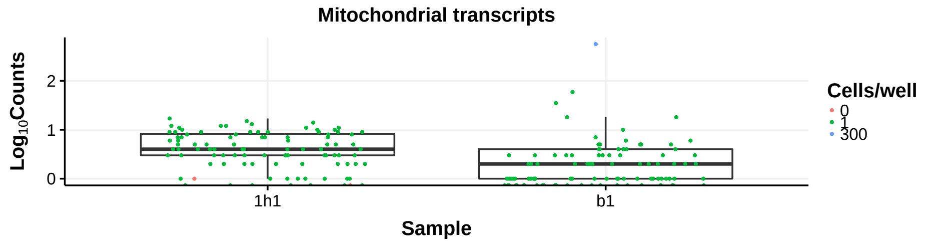

Reads mapped to genes
Log10Reads
3
2
1
0
1h1
b1
Sample
Cells/well
0
1
300

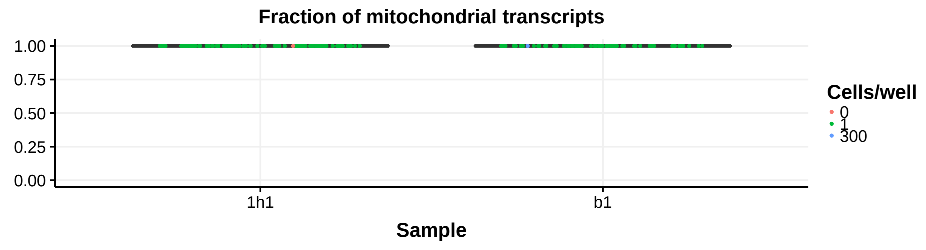

Fraction of aligned reads to total reads
1.00
0.75
0.50
0.25
0.00
1h1
b1
Sample
Cells/well
0
1
300

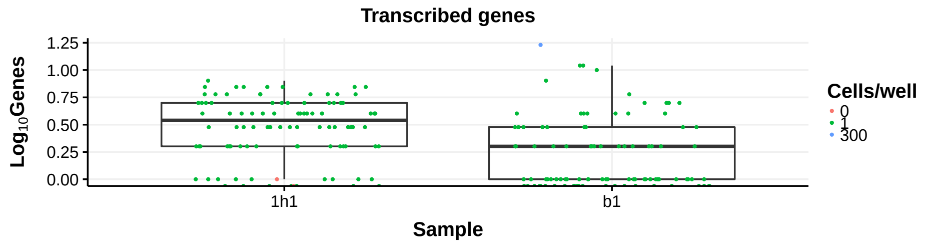

Fraction of gene reads out of aligned reads
1.00
0.75
0.50
0.25
0.00
1h1
b1
Sample
Cells/well
0
1
300

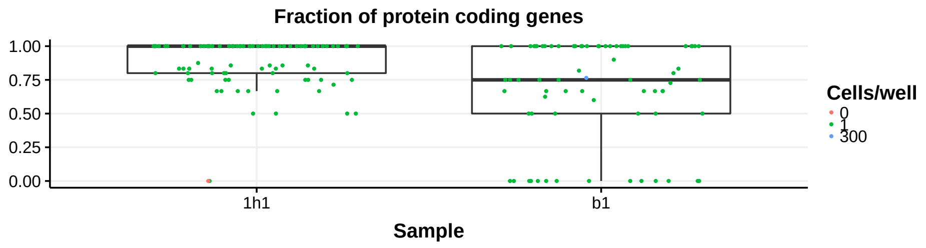

Fraction of gene reads out of total reads
1.00
0.75
0.50
0.25
0.00
1h1
b1
Sample
Cells/well
0
1
300

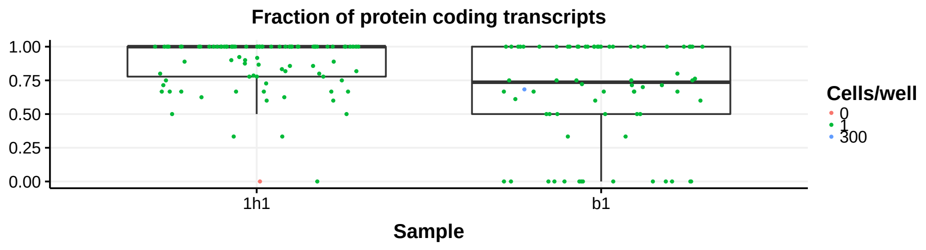

Total counts
Log10Counts
2
1
0
1h1
b1
Sample
Cells/well
0
1
300

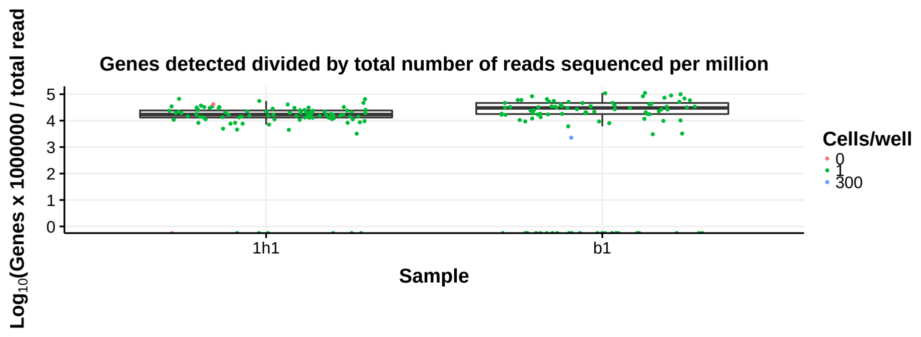

Mitochondrial transcripts
Log10Counts
2
1
0
1h1
b1
Sample
Cells/well
0
1
300











3.6 Visualization of Read alignments

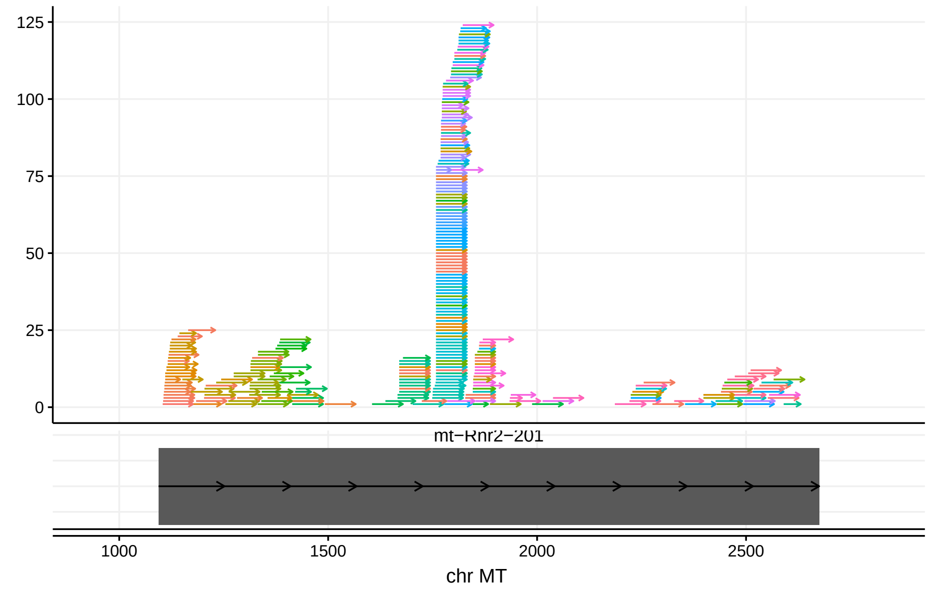
scruff package provides functions to visualize read alignments on the reference genome. Reads are colored by their UMI. The following command visualize the reads mapped to gene mt-Rnr2 for the bulk sample `vandenBrink_b1_cell_0095`.

```
# Visualize the reads mapped to gene "mt-Rnr2" in cell "vandenBrink_b1_cell_0095".
data(bamExample, package = "scruff")

gtfEG = refGenome::ensemblGenome(dirname(gtf))
refGenome::read.gtf(gtfEG, filename = basename(gtf))
```

```
# gene mt-Rnr2 starts at 1094 and ends at 2675
start <- 1094
end <- 2675

g1 <- rview(bamExample, chr = "MT", start = start, end = end)
g2 <- gview(gtfEG, chr = "MT", start = start, end = end)
g <- ggbio::tracks(g1, g2, heights = c(4,1), xlab = "chr MT")
g
```



4 Session Information

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
```

```
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] SingleCellExperiment_1.2.0  SummarizedExperiment_1.10.1
##  [3] DelayedArray_0.6.0          BiocParallel_1.14.1
##  [5] matrixStats_0.53.1          Biobase_2.40.0
##  [7] GenomicRanges_1.32.3        GenomeInfoDb_1.16.0
##  [9] IRanges_2.14.5              S4Vectors_0.18.2
## [11] BiocGenerics_0.26.0         scruff_0.99.11
## [13] BiocStyle_2.8.1
##
## loaded via a namespace (and not attached):
##   [1] colorspace_1.3-2         hwriter_1.3.2
##   [3] rprojroot_1.3-2          biovizBase_1.28.0
##   [5] htmlTable_1.11.2         XVector_0.20.0
##   [7] base64enc_0.1-3          dichromat_2.0-0
##   [9] rstudioapi_0.7           bit64_0.9-7
##  [11] AnnotationDbi_1.42.1     codetools_0.2-15
##  [13] splines_3.5.0            ggbio_1.28.0
##  [15] doParallel_1.0.11        doBy_4.6-1
##  [17] knitr_1.20               Formula_1.2-3
##  [19] Rsamtools_1.32.0         cluster_2.0.7-1
##  [21] graph_1.57.1             compiler_3.5.0
##  [23] httr_1.3.1               backports_1.1.2
##  [25] assertthat_0.2.0         Matrix_1.2-14
##  [27] lazyeval_0.2.1           acepack_1.4.1
##  [29] htmltools_0.3.6          prettyunits_1.0.2
##  [31] tools_3.5.0              bindrcpp_0.2.2
##  [33] gtable_0.2.0             glue_1.2.0
##  [35] GenomeInfoDbData_1.1.0   reshape2_1.4.3
##  [37] dplyr_0.7.4              ggthemes_3.5.0
##  [39] ShortRead_1.38.0         Rcpp_0.12.16
##  [41] Biostrings_2.48.0        rtracklayer_1.40.2
##  [43] iterators_1.0.9          refGenome_1.7.3
##  [45] xfun_0.1                 stringr_1.3.0
##  [47] ensembldb_2.4.1          XML_3.98-1.11
##  [49] zlibbioc_1.26.0          MASS_7.3-49
##  [51] scales_0.5.0             BSgenome_1.48.0
##  [53] VariantAnnotation_1.26.0 BiocInstaller_1.30.0
##  [55] ProtGenerics_1.12.0      RBGL_1.56.0
##  [57] AnnotationFilter_1.4.0   RColorBrewer_1.1-2
##  [59] yaml_2.1.19              curl_3.2
##  [61] memoise_1.1.0            gridExtra_2.3
##  [63] ggplot2_2.2.1            biomaRt_2.36.0
##  [65] rpart_4.1-13             reshape_0.8.7
##  [67] latticeExtra_0.6-28      stringi_1.1.7
##  [69] RSQLite_2.1.1            foreach_1.4.4
##  [71] checkmate_1.8.5          GenomicFeatures_1.32.0
##  [73] rlang_0.2.0              pkgconfig_2.0.1
##  [75] bitops_1.0-6             evaluate_0.10.1
##  [77] lattice_0.20-35          bindr_0.1.1
##  [79] GenomicAlignments_1.16.0 htmlwidgets_1.2
##  [81] labeling_0.3             bit_1.1-12
##  [83] GGally_1.4.0             plyr_1.8.4
##  [85] magrittr_1.5             bookdown_0.7
##  [87] R6_2.2.2                 Hmisc_4.1-1
##  [89] DBI_1.0.0                pillar_1.2.2
##  [91] foreign_0.8-70           survival_2.41-3
##  [93] RCurl_1.95-4.10          nnet_7.3-12
##  [95] tibble_1.4.2             OrganismDbi_1.22.0
##  [97] rmarkdown_1.9            progress_1.1.2
##  [99] grid_3.5.0               data.table_1.11.2
## [101] blob_1.1.1               digest_0.6.15
## [103] munsell_0.4.3
```