Process scRNA-Seq reads in *scruff*

Zhe Wang

2019-07-30

Package

scruff 1.2.3

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)
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

# 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 purpose, 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("Package \"Rsubread\" needed for \"alignRsubread\"",
        " function to work. ",
        "Please install it if you are using Linux or macOS systems. ",
        "The function is not available in Windows environment.")
} 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)
}
```

Now that everything is ready, we can run *scruff*. In experiment 1h1, cell barcodes 95 and 96 are empty well controls. In experiment b1, cell barcode 95 is bulk sample containing 300 cells. These information can be set by the `cellPerWell` argument. We apply cell barcode and UMI correction by setting `bcEdit` to 1 and `umiEdit` to 1. *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("Package \"Rsubread\" needed. ",
        "Please install it if you are using Linux or macOS systems. ",
        "The function is not available in Windows environment.")
} else {
    sce <- scruff(project = "example",
        experiment = 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,
        bcEdit = 1,
        umiStart = 9,
        umiStop = 12,
        umiEdit = 1,
        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 For CEL-Seq Samples

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. Cell barcode correction can be set by `bcEdit` parameter. 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",
    experiment = c("1h1", "b1"),
    lane = c("L001", "L001"),
    read1Path = c(v1h1R1, vb1R1),
    read2Path = c(v1h1R2, vb1R2),
    barcodeExample,
    bcStart = 1,
    bcStop = 8,
    bcEdit = 1,
    umiStart = 9,
    umiStop = 12,
    keep = 75,
```

```
        minQual = 10,
        yieldReads = 1e+06,
        verbose = TRUE,
        overwrite = TRUE,
        cores = 2)
```

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("Package \"Rsubread\" needed.",
        " Please install it if you are using Linux or macOS systems.",
        " The function is not available in Windows environment.\n")
} 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)
}
```

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",
        overwrite = TRUE,
        verbose = TRUE,
        cores = 2)
}
```

3.4 UMI correction and Generation of Count Matrix

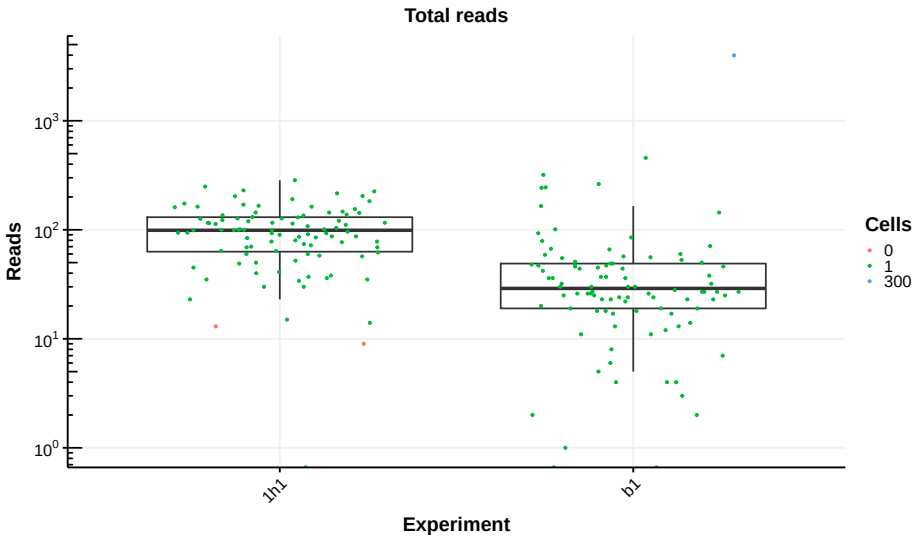
Example GTF file `GRCm38_MT.gtf` will be used for feature counting. Currently, *scruff* applies the union counting mode of the HTSeq Python package. The following command generates the UMI corrected count matrix for the example dataset by allowing correction of UMIs with Hamming distances smaller than 1 for each gene in each cell.

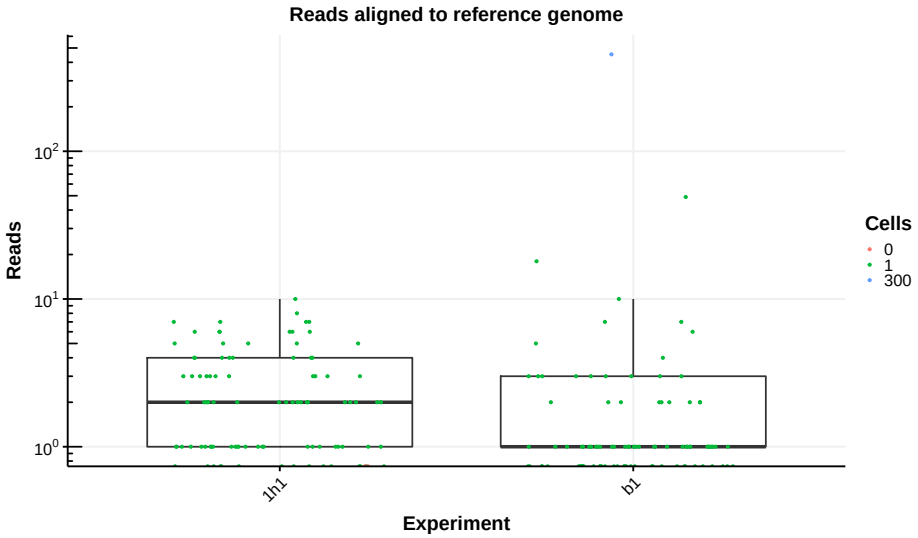
```
gtf <- system.file("extdata", "GRCm38_MT.gtf", package = "scruff")
# get the molecular counts of trancsripts for each cell
# In experiment 1h1, cell barcodes 95 and 96 are empty well controls.
# In experiment b1, cell barcode 95 is bulk sample containing 300 cells.
if (requireNamespace("Rsubread",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

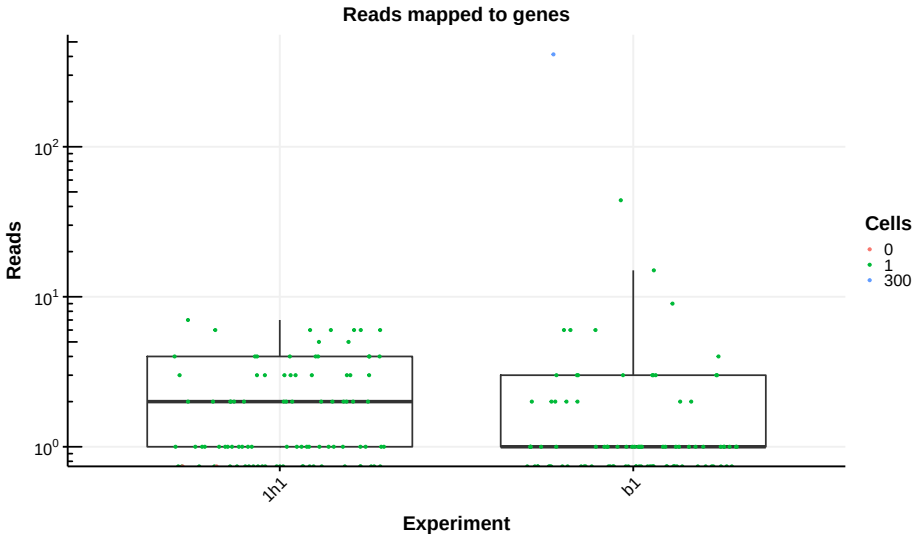
3.5 Visualization of QC metrics

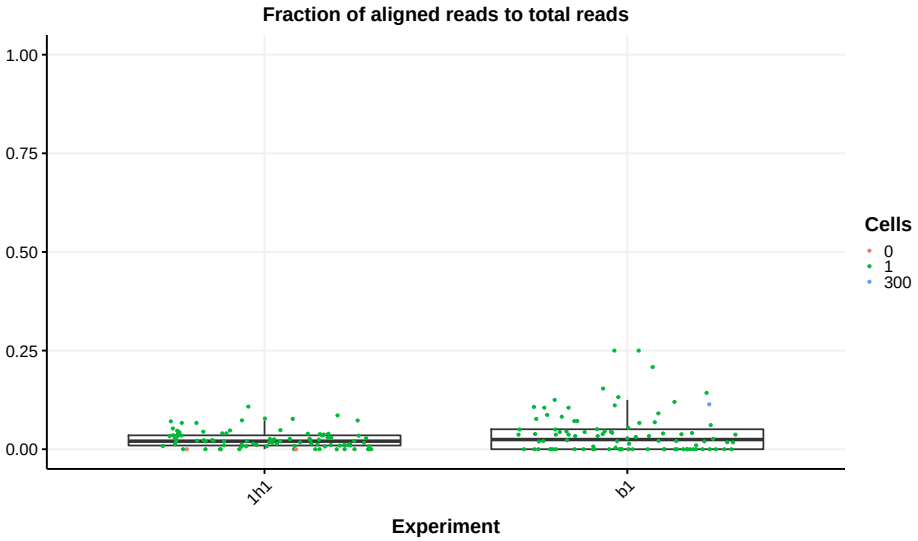
The data quality diagnostic information are contained in the `colData` of the returned `SingleCellExperiment` object `sce`. They can be visualized using the `qcplots` function.

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

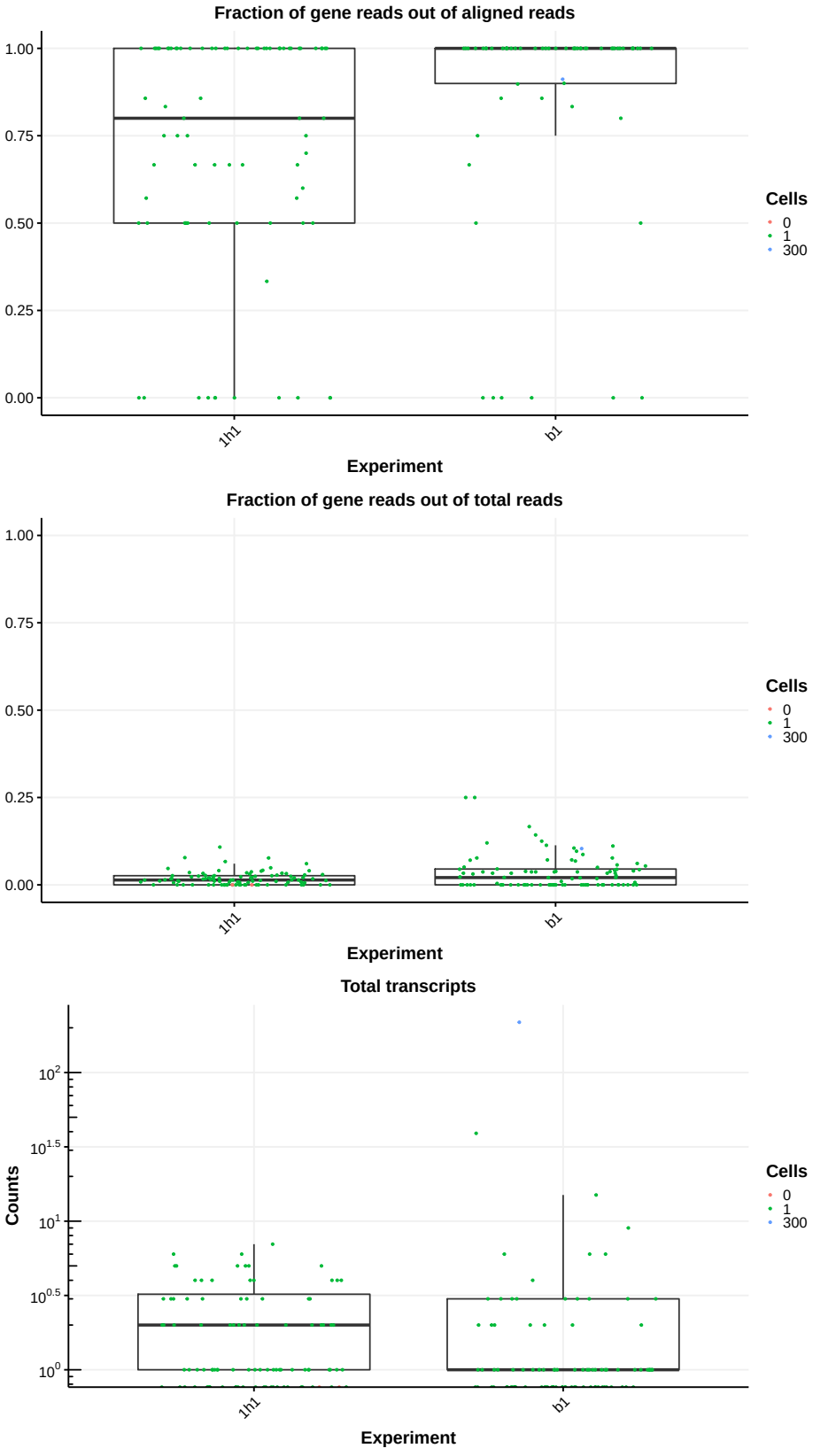


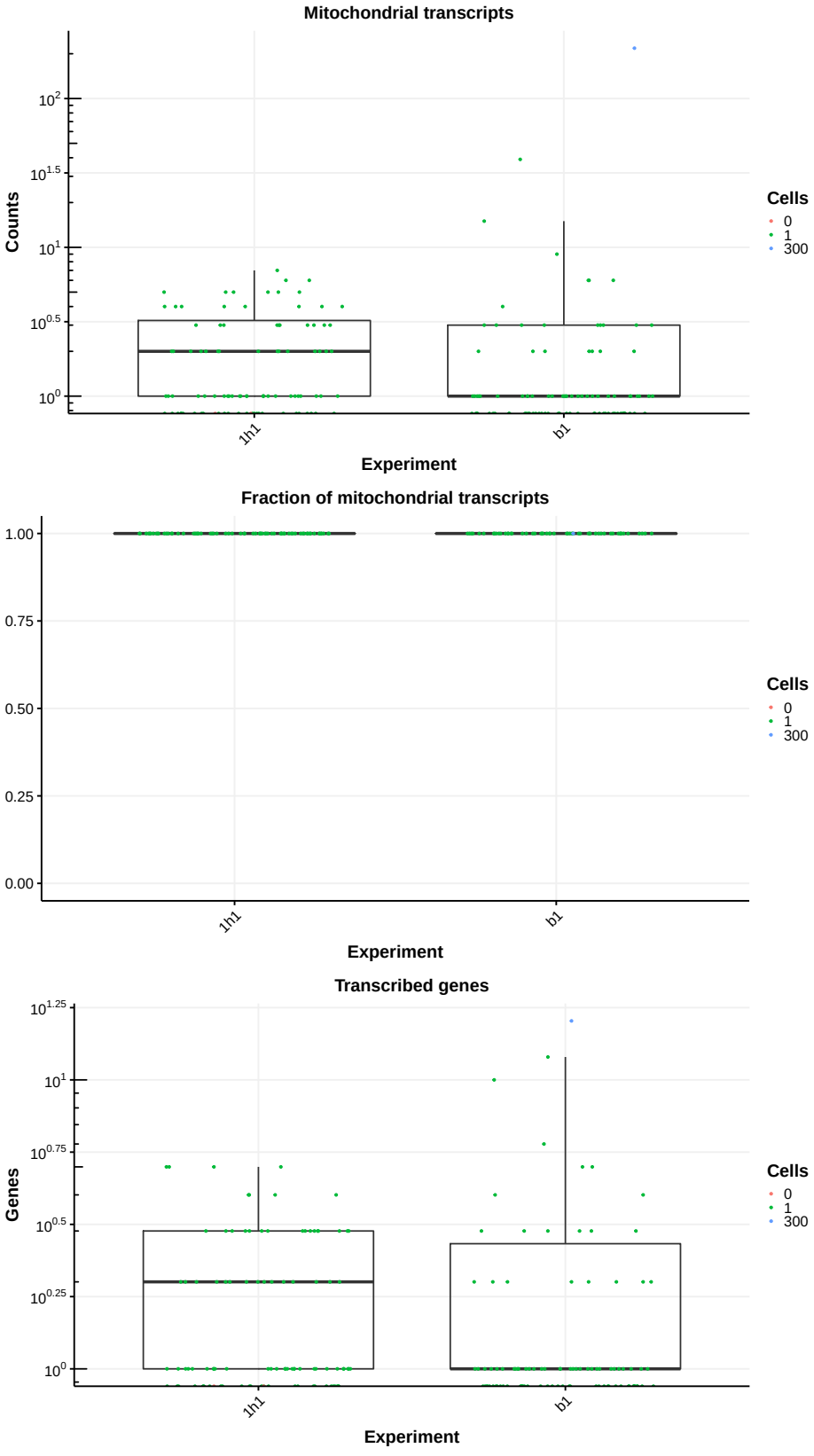

Reads aligned to reference genome
Reads
10^2
10^1
10^0
1h1
b1
Experiment
Cells
0
1
300


Reads mapped to genes
Reads
10^2
10^1
10^0
1h1
b1
Experiment
Cells
0
1
300

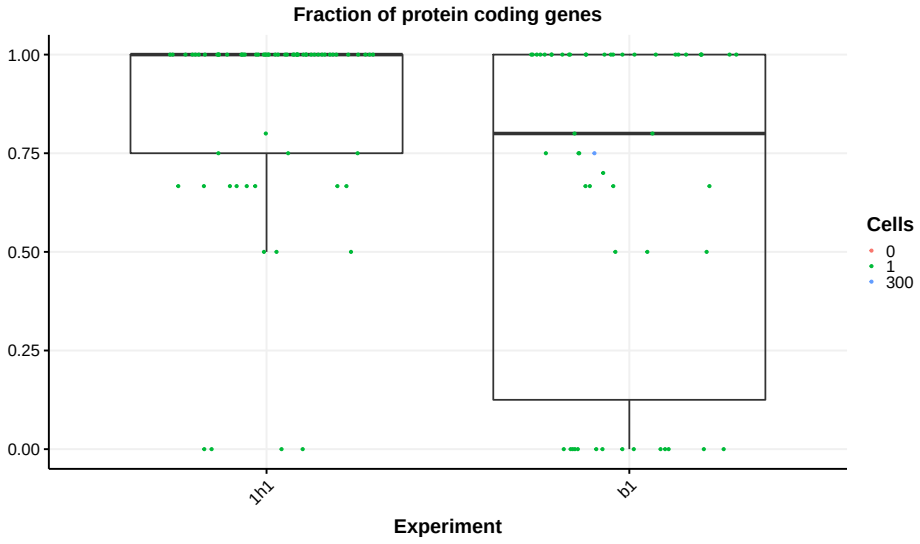

Fraction of aligned reads to total reads
1.00
0.75
0.50
0.25
0.00
1h1
b1
Experiment
Cells
0
1
300

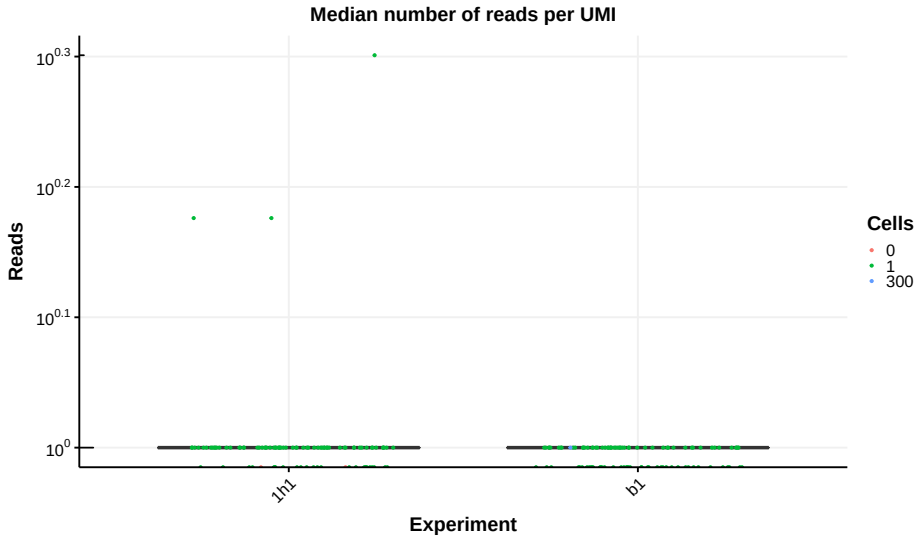
Process scRNA-Seq reads in *scruff*

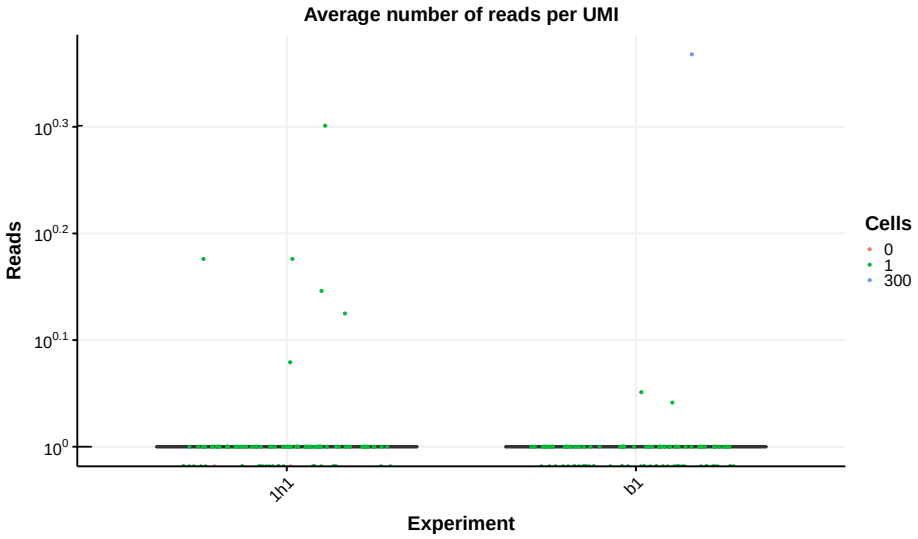


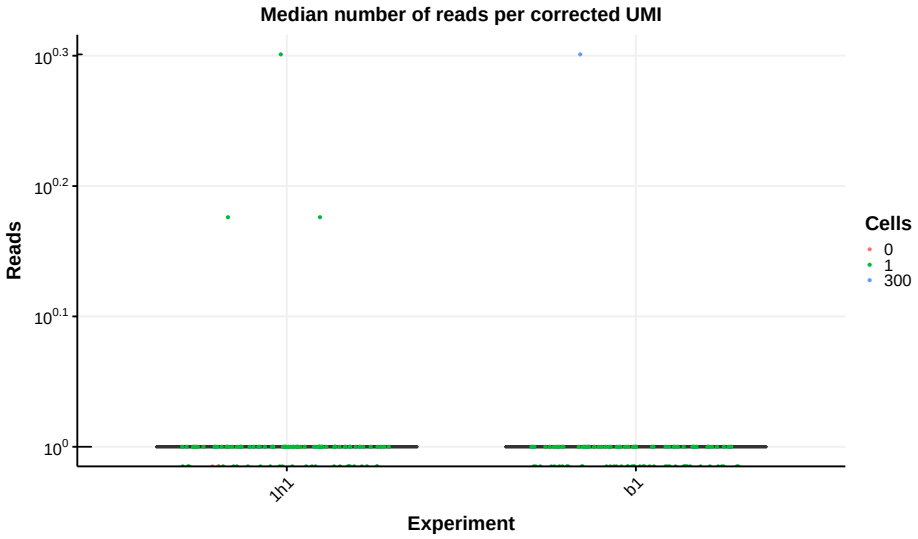

Mitochondrial transcripts
Counts
10^2
$10^{1.5}$
10^1
$10^{0.5}$
10^0
1h1
b1
Experiment
Cells
0
1
300
Fraction of mitochondrial transcripts
1.00
0.75
0.50
0.25
0.00
1h1
b1
Experiment
Cells
0
1
300
Transcribed genes
Genes
$10^{1.25}$
10^1
$10^{0.75}$
$10^{0.5}$
$10^{0.25}$
10^0
1h1
b1
Experiment
Cells
0
1
300

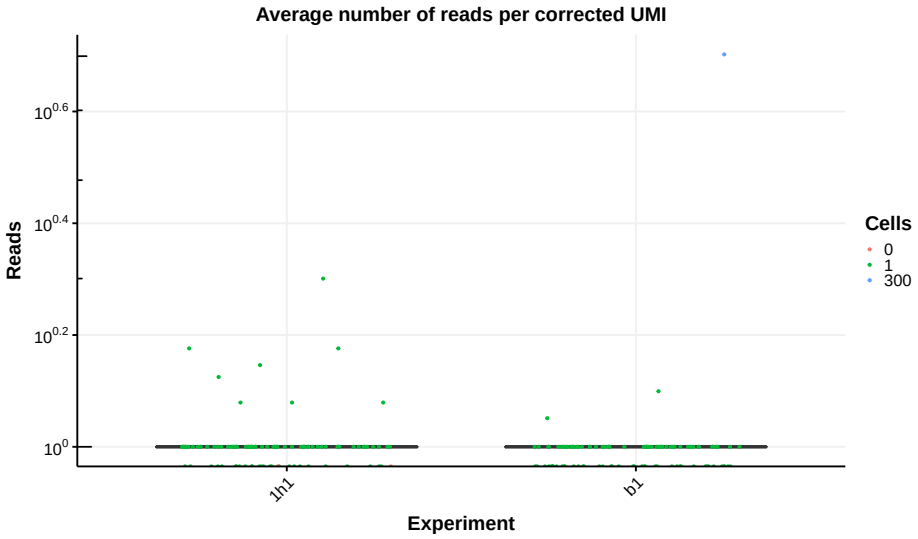
Process scRNA-Seq reads in *scruff*

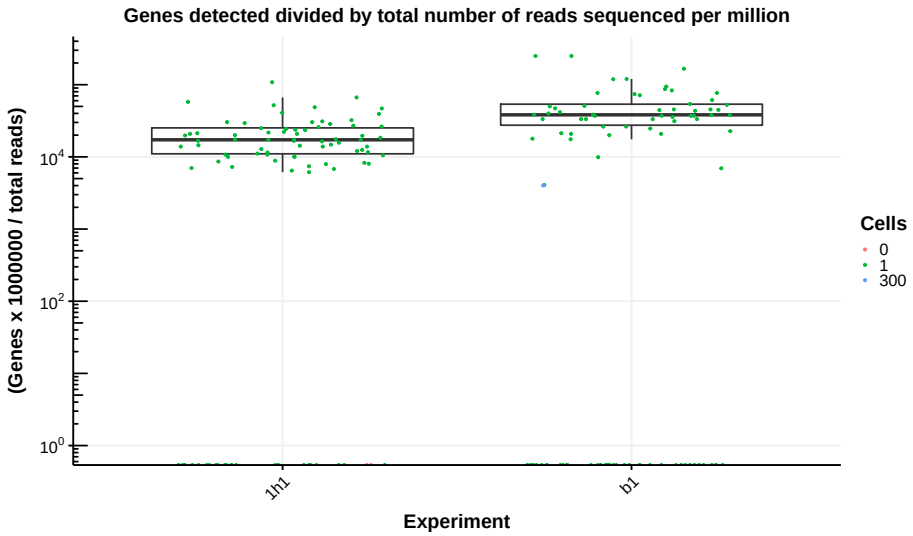





Average number of reads per UMI
Reads
$10^{0.3}$
$10^{0.2}$
$10^{0.1}$
10^{0}
1h1
b1
Experiment
Cells
0
1
300


Median number of reads per corrected UMI
Reads
$10^{0.3}$
$10^{0.2}$
$10^{0.1}$
10^{0}
1h1
b1
Experiment
Cells
0
1
300


Average number of reads per corrected UMI
Reads
$10^{0.6}$
$10^{0.4}$
$10^{0.2}$
10^{0}
1h1
b1
Experiment
Cells
0
1
300



Each of these boxplots shows the distribution of a quality metric within and across experiments. Y axis shows the number or fraction of these metrics. Each point represents a unique cell barcode associated with the cells or bulk samples in the wells of the plate. These points are colored by the `cellPerWell` parameter which is the number of sorted cells in each well. For example, in experiment 1h1, cell barcodes 95 and 96 are empty well controls. In experiment b1, cell barcode 95 is a bulk sample containing 300 cells.

These quality metrics include total reads, aligned reads, reads mapped to genes, fraciont of aligned reads, fraction of gene reads out of aligned reads, fraction of gene reads out of total reads, total transcripts, mitochondrial transcripts, fraction of mitochondrial transcripts, transcribed genes, fraction of protein coding genes and transcripts, median and average number of reads per corrected or uncorrectd UMI, and gene detection rate.

3.6 Visualization of Read mapping locations

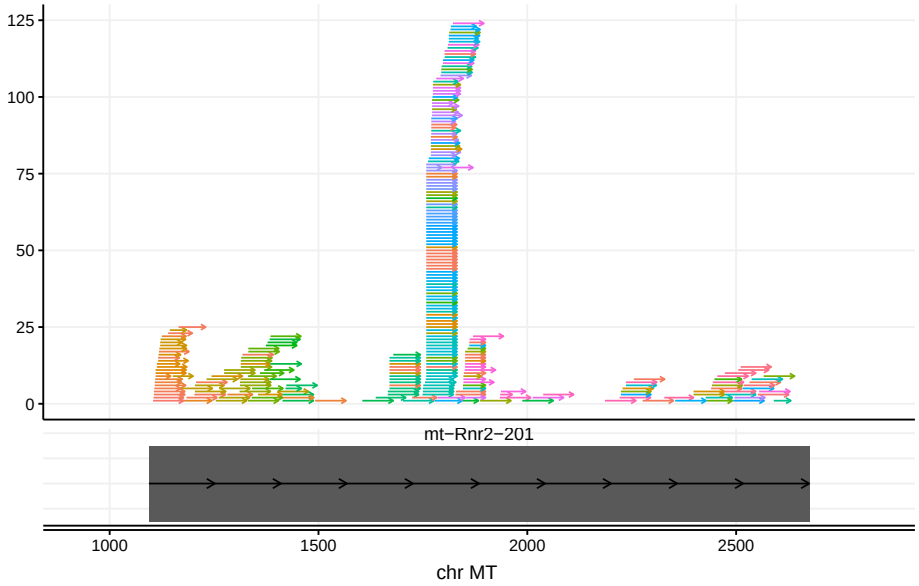
scruff package provides function to visualize read alignment locations at specified genomic coordinates on the reference genome. The following command visualize the reads mapped to gene mt-Rnr2 for the bulk sample `vandenBrink_b1_cell_0095`. Reads are colored by their uncorrected UMI tags. `bamExample` in the following example is a `GAlignments` object generated by the `readGAlignments` function in the *GenomicAlignments* package. Please refer to the documentation of `readGAlignments` and *GenomicAlignments* for details about how to read BAM files into R.

```
# Visualize the reads mapped to gene "mt-Rnr2" in
# cell "vandenBrink_b1_cell_0095".
# bamExample is generated by GenomicAlignments::GAlignments function
data(bamExample, package = "scruff")
```

```
# 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(gtf, chr = "MT", start = start, end = end)
```

```
g <- ggbio::tracks(g1, g2, heights = c(4,1), xlab = "chr MT")
g
```



This plot shows the read alignment information for gene mt-RNA2. The top panel shows the alignment and orientation of each read aligned to this gene. Y axis indicates the number of aligned reads at specific locations. Each arrow represents a read. The length of the arrow indicates the length of the read. The reads are colored by their UMI tags. In the bottom panel, each arrow represents an isoform of the gene. The isoforms are named by their transcript names. The grey boxes indicate the exonic regions of the transcript. X axis is the genomic coordinates on the corresponding chromosome (e.g. chr MT). It is consistent between the top and bottom panels if `legend` is set to FALSE for `rview` function.

4 10X BAM File Quality Assessment

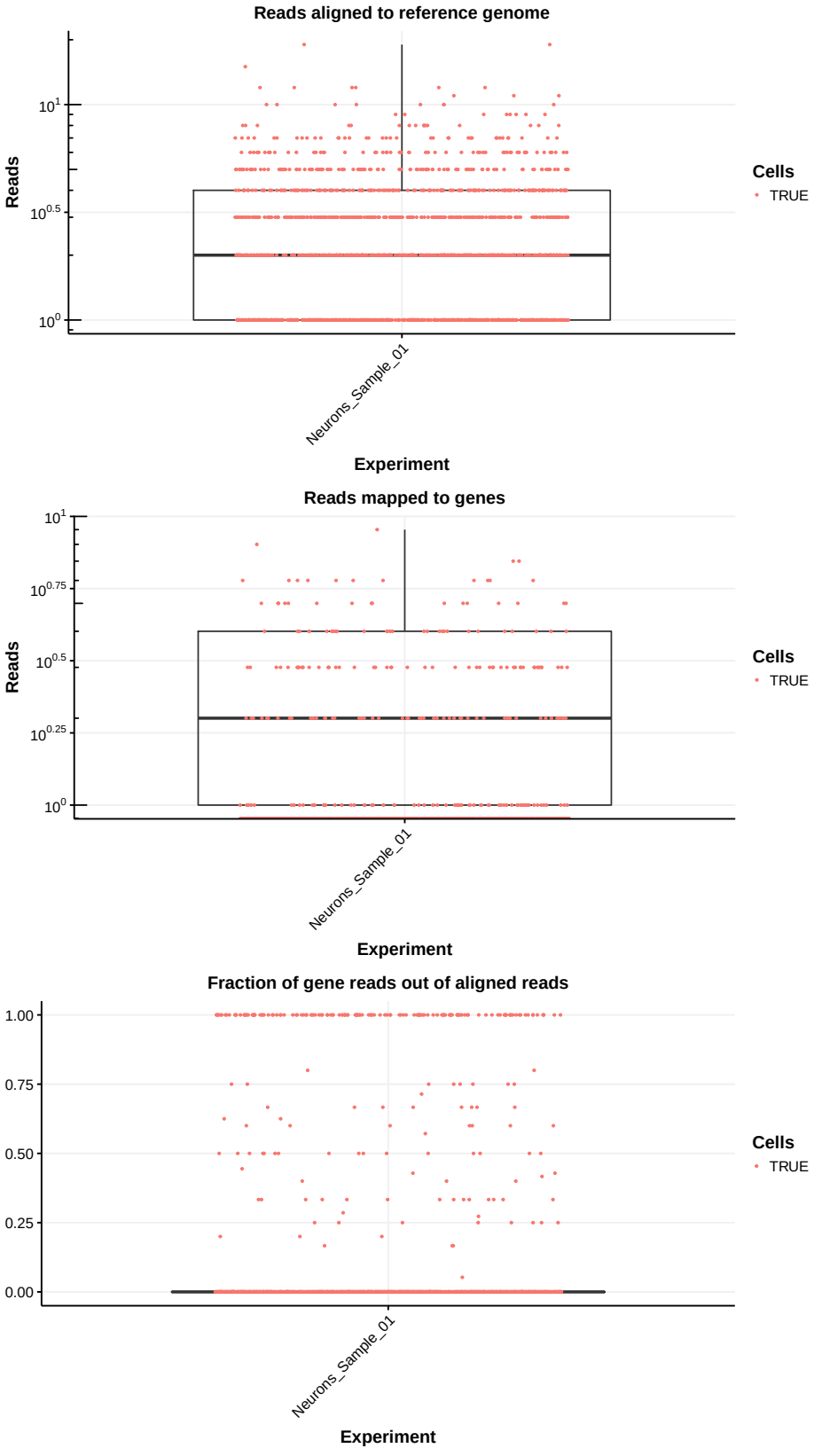
The function `tenxBamqc` collects QC metrics from BAM files generated by 10X Cell Ranger pipeline. The collected QC metrics can be visualized by the function `qcplots`. The collected measurements include number of aligned reads and number of reads aligned to an gene for each valid cell barcode. Users can see the alignment quality for the filtered cells after plugging in the filtered barcode file from the Cell Ranger results. Here we show the visualization of an example BAM file from Cell Ranger output.

```
# The following example BAM file is the first 5000 BAM file records extracted
# from sample 01 of the 1.3 Million Brain Cells dataset from E18 Mice.
# (https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.3.0/
# 1M_neurons)
# The BAM file for sample 01 is downloaded from here:
# http://sra-download.ncbi.nlm.nih.gov/srapub_files/
# SRR5167880_E18_20160930_Neurons_Sample_01.bam
# see details here:
# https://trace.ncbi.nlm.nih.gov/Traces/sra/?study=SRP096558
```

```
# and here:
# https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE93421
bamfile10x <- system.file("extdata",
    "SRR5167880_E18_20160930_Neurons_Sample_01_5000.bam",
    package = "scruff")

# The filtered cell barcodes are generated using the following script:
# library(TENxBrainData)
# library(data.table)
# tenx <- TENxBrainData()
# # get filtered barcodes for sample 01
# filteredBcIndex <- tstrsplit(colData(tenx)[, "Barcode"], "-")[[2]] == 1
# filteredBc <- colData(tenx)[filteredBcIndex, ]["Barcode"]]
filteredBc <- system.file("extdata",
    "SRR5167880_E18_20160930_Neurons_Sample_01_filtered_barcode.tsv",
    package = "scruff")
# QC results are saved to current working directory
qcDt <- tenxBamqc(bam = bamfile10x,
    experiment = "Neurons_Sample_01",
    filter = filteredBc)
## tenxBamqc(bam = bamfile10x, experiment = "Neurons_Sample_01",
##     filter = filteredBc)
qcDt
## class: SingleCellExperiment
## dim: 1 1576
## metadata(0):
## assays(1): ''
## rownames: NULL
## rowData names(0):
## colnames(1576): AAACCTGAGGCCCGTT-Neurons_Sample_01
##   AAACCTGAGTCCGGTC-Neurons_Sample_01 ...
##   TTTGTCATCGGTGTTA-Neurons_Sample_01
##   TTTGTCATCTTGTATC-Neurons_Sample_01
## colData names(5): cell_barcode reads_mapped_to_genome
##   reads_mapped_to_genes experiment number_of_cells
## reducedDimNames(0):
## spikeNames(0):
```

```
g <- qcplots(qcDt)
g
```


Reads aligned to reference genome
Reads
10^1
$10^{0.5}$
10^0
Neurons_Sample_01
Experiment
Cells
TRUE
Reads mapped to genes
Reads
10^1
$10^{0.75}$
$10^{0.5}$
$10^{0.25}$
10^0
Neurons_Sample_01
Experiment
Cells
TRUE
Fraction of gene reads out of aligned reads
1.00
0.75
0.50
0.25
0.00
Neurons_Sample_01
Experiment
Cells
TRUE

5 Session Information

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.2 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.9-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.9-bioc/R/lib/libRlapack.so
##
## 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=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] GenomicFeatures_1.36.4 AnnotationDbi_1.46.0   Biobase_2.44.0
##  [4] GenomicRanges_1.36.0   GenomeInfoDb_1.20.0    IRanges_2.18.1
##  [7] S4Vectors_0.22.0       BiocGenerics_0.30.0    scruff_1.2.3
## [10] BiocStyle_2.12.0
##
## loaded via a namespace (and not attached):
##   [1] ProtGenerics_1.16.0         bitops_1.0-6
##   [3] matrixStats_0.54.0          bit64_0.9-7
##   [5] RColorBrewer_1.1-2          progress_1.2.2
##   [7] httr_1.4.0                  tools_3.6.1
##   [9] backports_1.1.4             R6_2.4.0
##  [11] rpart_4.1-15                Hmisc_4.2-0
##  [13] DBI_1.0.0                   lazyeval_0.2.2
##  [15] colorspace_1.4-1            nnet_7.3-12
##  [17] withr_2.1.2                 tidyselect_0.2.5
##  [19] gridExtra_2.3               prettyunits_1.0.2
##  [21] GGally_1.4.0                curl_4.0
##  [23] bit_1.1-14                  compiler_3.6.1
##  [25] graph_1.62.0                htmlTable_1.13.1
##  [27] DelayedArray_0.10.0         labeling_0.3
##  [29] rtracklayer_1.44.2          bookdown_0.12
##  [31] ggbio_1.32.0                checkmate_1.9.4
##  [33] scales_1.0.0                RBGL_1.60.0
##  [35] stringr_1.4.0               digest_0.6.20
##  [37] Rsamtools_2.0.0             foreign_0.8-71
##  [39] rmarkdown_1.14              XVector_0.24.0
##  [41] Rsubread_1.34.6             dichromat_2.0-0
```

```
##  [43] base64enc_0.1-3             pkgconfig_2.0.2
##  [45] htmltools_0.3.6             ensembldb_2.8.0
##  [47] BSgenome_1.52.0             ggthemes_4.2.0
##  [49] htmlwidgets_1.3             rlang_0.4.0
##  [51] rstudioapi_0.10             RSQLite_2.1.2
##  [53] hwriter_1.3.2               BiocParallel_1.18.0
##  [55] acepack_1.4.1               dplyr_0.8.3
##  [57] VariantAnnotation_1.30.1    RCurl_1.95-4.12
##  [59] magrittr_1.5                GenomeInfoDbData_1.2.1
##  [61] Formula_1.2-3               Matrix_1.2-17
##  [63] Rcpp_1.0.2                  munsell_0.5.0
##  [65] stringi_1.4.3               yaml_2.2.0
##  [67] SummarizedExperiment_1.14.0 zlibbioc_1.30.0
##  [69] plyr_1.8.4                  grid_3.6.1
##  [71] blob_1.2.0                  crayon_1.3.4
##  [73] lattice_0.20-38             Biostrings_2.52.0
##  [75] splines_3.6.1               hms_0.5.0
##  [77] zeallot_0.1.0               knitr_1.23
##  [79] pillar_1.4.2                reshape2_1.4.3
##  [81] biomaRt_2.40.3              XML_3.98-1.20
##  [83] glue_1.3.1                  evaluate_0.14
##  [85] ShortRead_1.42.0            biovizBase_1.32.0
##  [87] latticeExtra_0.6-28         data.table_1.12.2
##  [89] BiocManager_1.30.4          vctrs_0.2.0
##  [91] gtable_0.3.0                purrr_0.3.2
##  [93] reshape_0.8.8               assertthat_0.2.1
##  [95] ggplot2_3.2.0               xfun_0.8
##  [97] AnnotationFilter_1.8.0      survival_2.44-1.1
##  [99] SingleCellExperiment_1.6.0  OrganismDbi_1.26.0
## [101] tibble_2.1.3                GenomicAlignments_1.20.1
## [103] memoise_1.1.0               cluster_2.1.0
```