The QoRTs Analysis Pipeline Example Walkthrough

Stephen Hartley
National Human Genome Research Institute
National Institutes of Health

May 1, 2018

QoRTs v1.0.1
JunctionSeq v1.10.0

Contents

1 Overview

This package contains data produced by the QoRTs [4] software package, which is a fast, efficient, and portable multifunction toolkit designed to assist in the analysis, quality control, and data management of RNA-Seq datasets. Its primary function is to aid in the detection and identification of errors, biases, and artifacts produced by paired-end high-throughput RNA-Seq technology. In addition, it can produce count data designed for use with differential expression [1] and differential exon usage tools [2], as well as individual-sample and/or group-summary genome track files suitable for use with the UCSC genome browser (or any compatible browser).

[1] Such as `DESeq`, `DESeq2` [1] or `edgeR` [5]
[2] Such as `DEXSeq` [2] or `JunctionSeq`

The QoRTs package is composed of two parts: a java jar-file (for data processing) and a companion R package (for generating tables, figures, and plots). The java utility is written in the Scala programming language (v2.11.1), however, it has been compiled to java byte-code and does not require an installation of Scala (or any other external libraries) in order to function. The entire QoRTs toolkit can be used in almost any operating system that supports java and R.

The most recent release of QoRTs is available on the QoRTs github page.

A complete and comprehensive walkthrough demontrating a full set of analyses using DESeq2, edgeR, DEXSeq, and JunctionSeq, is available online, along with a full example dataset (file is 200mb) with example bam files (file is 1.1gb).

2 Data Contained in this package

The example dataset is derived from a set of rat pineal gland samples, which were multiplexed and sequenced across six sequencer lanes. All samples are paired-end, 2x101 base-pair, strand-specific RNA-Seq. They were ribosome-depleted using the "Ribo-zero Gold" protocol and aligned via RNA-STAR.

For the sake of simplicity, the example dataset was limited to only six samples and three lanes. However, the bam files alone would still occupy 18 gigabytes of disk space, which would make it unsuitable for distribution as an example dataset. To further reduce the example bamfile sizes, only reads that mapped to chromosomes chr14, chr15, chrX, and chrM were included. Additionally, all the selected chromosomes EXCEPT for chromosome 14 were randomly downsampled to 30 percent of their original read counts. A few genes had additional fictional transcripts added, to test and demonstrate various tools' handling of certain edge cases. The original dataset from which these samples were derived is described elsewhere [3]. The original complete dataset is available on the NCBI Gene Expression Omnibus, series accession number GSE63309.

THIS DATASET IS INTENDED FOR DEMONSTRATION AND TESTING PURPOSES ONLY. Due to the various alterations that have been made to reduce file sizes and improve portability, it is really not suitable for any actual analyses.

2.1 Example Dataset

For simplicity, we renamed the samples SAMP1 through SAMP6, and renamed the conditions "CASE" and "CTRL" for night and day, respectively.

Thus: there are 6 samples: 3 "cases" and 3 "controls":

```
#Read the decoder:
decoder.file <- system.file("extdata/annoFiles/decoder.bySample.txt",
                            package="JctSeqData");
decoder <- read.table(decoder.file,
                      header=TRUE,
                      stringsAsFactors=FALSE);
print(decoder);

##   sample.ID group.ID
## 1     SAMP1     CASE
## 2     SAMP2     CASE
## 3     SAMP3     CASE
## 4     SAMP4     CTRL
```

```
## 5     SAMP5     CTRL
## 6     SAMP6     CTRL
```

2.2 Annotation Files

There are several gtf or gff annotation files included in the data package:

```
#The original gtf file, from Ensembl
#  (all but a few genes were removed, to save space):
anno.original.gtf.file <- system.file("extdata/annoFiles/anno-original.gtf.gz",
                        package="JctSeqData");
head(read.table(anno.original.gtf.file,sep='\t'));
```

```
##      V1      V2         V3     V4     V5 V6 V7 V8
## 1 chr14 ensembl       gene 182678 194072  .  -  .
## 2 chr14 ensembl transcript 182678 194072  .  -  .
## 3 chr14 ensembl       exon 194056 194072  .  -  .
## 4 chr14 ensembl        CDS 194056 194072  .  -  0
## 5 chr14 ensembl       exon 192862 193068  .  -  .
## 6 chr14 ensembl        CDS 192862 193068  .  -  1
##
## 1
## 2                                                                 gene_id ENSRNOG00000C
## 3       gene_id ENSRNOG00000050954; gene_version 2; transcript_id ENSRNOT00000073973; tr
## 4 gene_id ENSRNOG00000050954; gene_version 2; transcript_id ENSRNOT00000073973; transcri
## 5       gene_id ENSRNOG00000050954; gene_version 2; transcript_id ENSRNOT00000073973; tr
## 6 gene_id ENSRNOG00000050954; gene_version 2; transcript_id ENSRNOT00000073973; transcri
```

```
#Modified gtf file, with a few genes
#  changed to make them into a better test dataset:
anno.gtf.file <- system.file("extdata/annoFiles/anno.gtf.gz",
                        package="JctSeqData");
head(read.table(anno.gtf.file,sep='\t'));
```

```
##      V1      V2         V3     V4     V5 V6 V7 V8
## 1 chr14 ensembl       gene 182678 194072  .  -  .
## 2 chr14 ensembl transcript 182678 194072  .  -  .
## 3 chr14 ensembl       exon 194056 194072  .  -  .
## 4 chr14 ensembl        CDS 194056 194072  .  -  0
## 5 chr14 ensembl       exon 192862 193068  .  -  .
## 6 chr14 ensembl        CDS 192862 193068  .  -  1
##
## 1
## 2                                                                 gene_id ENSRNOG00000C
## 3       gene_id ENSRNOG00000050954; gene_version 2; transcript_id ENSRNOT00000073973; tr
## 4 gene_id ENSRNOG00000050954; gene_version 2; transcript_id ENSRNOT00000073973; transcri
## 5       gene_id ENSRNOG00000050954; gene_version 2; transcript_id ENSRNOT00000073973; tr
## 6 gene_id ENSRNOG00000050954; gene_version 2; transcript_id ENSRNOT00000073973; transcri
```

```
#"Flattened" gff file, for DEXSeq:
DEX.gff.file <- system.file("extdata/annoFiles/DEXSeq.flat.gff.gz",
                        package="JctSeqData");
head(read.table(DEX.gff.file,sep='\t'));

##      V1             V2             V3     V4     V5 V6 V7 V8
## 1 chr14 QoRT_forDEXSeq aggregate_gene 182678 194072  .  -  .
## 2 chr14 QoRT_forDEXSeq    exonic_part 182678 183555  .  -  .
## 3 chr14 QoRT_forDEXSeq    exonic_part 184707 184830  .  -  .
## 4 chr14 QoRT_forDEXSeq    exonic_part 186229 186453  .  -  .
## 5 chr14 QoRT_forDEXSeq    exonic_part 189146 189973  .  -  .
## 6 chr14 QoRT_forDEXSeq    exonic_part 190013 190189  .  -  .
##                                                                                                  V9
## 1                                                                                 gene_id ENSRNOG00000050954
## 2 gene_id ENSRNOG00000050954; transcripts ENSRNOT00000073973; exonic_part_number 001
## 3 gene_id ENSRNOG00000050954; transcripts ENSRNOT00000073973; exonic_part_number 002
## 4 gene_id ENSRNOG00000050954; transcripts ENSRNOT00000073973; exonic_part_number 003
## 5 gene_id ENSRNOG00000050954; transcripts ENSRNOT00000073973; exonic_part_number 004
## 6 gene_id ENSRNOG00000050954; transcripts ENSRNOT00000073973; exonic_part_number 005

#"Flattened" gff file, for JunctionSeq
#  (w/o novel splice junctions):
JS.gff.file <- system.file("extdata/annoFiles/JunctionSeq.flat.gff.gz",
                        package="JctSeqData");
head(read.table(JS.gff.file,sep='\t'));

##      V1         V2             V3     V4     V5 V6 V7 V8
## 1 chr14 ScalaUtils aggregate_gene 182678 194072  .  -  .
## 2 chr14 ScalaUtils    exonic_part 182678 183555  .  -  .
## 3 chr14 ScalaUtils    exonic_part 184707 184830  .  -  .
## 4 chr14 ScalaUtils    exonic_part 186229 186453  .  -  .
## 5 chr14 ScalaUtils    exonic_part 189146 189973  .  -  .
## 6 chr14 ScalaUtils    exonic_part 190013 190189  .  -  .
##
## 1 gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num 000; aggregateGeneStrand -;
## 2                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num
## 3                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num
## 4                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num
## 5                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num
## 6                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num

#"Flattened" gff file, for JunctionSeq
#  (w/ novel splice junctions):
JS.novel.gff.file <-
    system.file("extdata/annoFiles/withNovel.forJunctionSeq.gff.gz",
                        package="JctSeqData");
head(read.table(JS.novel.gff.file,sep='\t'));
```

```
##      V1         V2             V3     V4     V5 V6 V7 V8
## 1 chr14 ScalaUtils aggregate_gene 182678 194072  .  -  .
## 2 chr14 ScalaUtils    exonic_part 182678 183555  .  -  .
## 3 chr14 ScalaUtils    exonic_part 184707 184830  .  -  .
## 4 chr14 ScalaUtils    exonic_part 186229 186453  .  -  .
## 5 chr14 ScalaUtils    exonic_part 189146 189973  .  -  .
## 6 chr14 ScalaUtils    exonic_part 190013 190189  .  -  .
##
## 1 gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num 000; aggregateGeneStrand -;
## 2                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num
## 3                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num
## 4                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num
## 5                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num
## 6                           gene_id ENSRNOG00000050954; tx_set ENSRNOT00000073973; num
```

There are several other annotation files as well:

```
#rn6 chrom.sizes file, from UCSC:
rn6.chrom.sizes <- system.file("extdata/annoFiles/rn6.chrom.sizes",
                        package="JctSeqData");
head(read.table(rn6.chrom.sizes,sep='\t'));

##     V1        V2
## 1 chr1 282763074
## 2 chr2 266435125
## 3 chr4 184226339
## 4 chr3 177699992
## 5 chr5 173707219
## 6 chrX 159970021

#stripped-down chrom.sizes file
# only includes chr14, chrX, chrM, and loose chr14 contigs:
chrom.sizes <- system.file("extdata/annoFiles/chrom.sizes",
                        package="JctSeqData");
head(read.table(chrom.sizes,sep='\t'));

##                         V1        V2
## 1                    chr14 115493446
## 2 chr14_KL568057v1_random     35355
## 3 chr14_KL568053v1_random     33884
## 4 chr14_KL568054v1_random     31451
## 5 chr14_KL568059v1_random     27955
## 6 chr14_KL568052v1_random     23128

#mapping of ensembl id to gene symbol:
ensid.2.symbol.file <- system.file("extdata/annoFiles/ensid.2.symbol.txt",
                        package="JctSeqData");
head(read.table(ensid.2.symbol.file,sep='\t',header=TRUE));
```

```
##           ENS_GENEID geneSymbol
## 1 ENSRNOG00000002227        Kit
## 2 ENSRNOG00000002224      Yipf7
## 3 ENSRNOG00000002225     Scarb2
## 4 ENSRNOG00000030149     Adgrl3
## 5 ENSRNOG00000038572      Ncapg
## 6 ENSRNOG00000054570         U4
```

2.3 Count Files

There are numerous count files included in this data package, intended for use with various external analysis packages:

For use with DESeq, DESeq2, edgeR, limma-voom, or similar gene-level, count-based differential expression tools, we can use the gene-level counts:

```
#Recall the decoder:
print(decoder);

##   sample.ID group.ID
## 1     SAMP1     CASE
## 2     SAMP2     CASE
## 3     SAMP3     CASE
## 4     SAMP4     CTRL
## 5     SAMP5     CTRL
## 6     SAMP6     CTRL

#Gene level counts:
gene.count.files <- system.file(paste0("extdata/cts/",
                                       decoder$sample.ID,
                                       "/QC.geneCounts.formatted.for.DESeq.txt.gz"
                                       ),
                        package="JctSeqData");

#One of the count files:
read.table(gene.count.files[1])[1:10,]

##                     V1   V2
## 1  ENSRNOG00000000035    0
## 2  ENSRNOG00000000041    1
## 3  ENSRNOG00000000043  103
## 4  ENSRNOG00000000044  417
## 5  ENSRNOG00000000048 2391
## 6  ENSRNOG00000000060    7
## 7  ENSRNOG00000000062  205
## 8  ENSRNOG00000000064 1224
## 9  ENSRNOG00000000065   31
## 10 ENSRNOG00000000070    1
```

For use with DEXSeq or similar exon-level, count-based differential exon usage tools, we can use the exon-level counts:

```
#Exon level counts:
exon.count.files <- system.file(paste0("extdata/cts/",
                                       decoder$sample.ID,
                                       "/QC.exonCounts.formatted.for.DEXSeq.txt.gz"
                                       ),
                          package="JctSeqData");

#Part of One of the count files:
read.table(exon.count.files[1])[110:130,]

##                           V1 V2
## 110 ENSRNOG00000033073:003  0
## 111 ENSRNOG00000033073:004  0
## 112 ENSRNOG00000033073:005  0
## 113 ENSRNOG00000033073:006  0
## 114 ENSRNOG00000031928:001  0
## 115 ENSRNOG00000031928:002  0
## 116 ENSRNOG00000031928:003  0
## 117 ENSRNOG00000031928:004  0
## 118 ENSRNOG00000040213:001 20
## 119 ENSRNOG00000040213:002 11
## 120 ENSRNOG00000058018:001  1
## 121 ENSRNOG00000058018:002  0
## 122 ENSRNOG00000052352:001  5
## 123 ENSRNOG00000052352:002  1
## 124 ENSRNOG00000049828:001 30
## 125 ENSRNOG00000049828:002 10
## 126 ENSRNOG00000049828:003  6
## 127 ENSRNOG00000049828:004 10
## 128 ENSRNOG00000049828:005 18
## 129 ENSRNOG00000049828:006 17
## 130 ENSRNOG00000049828:007  8
```

For use with JunctionSeq, we can use a combined gene/exon/junction count file:

```
#JunctionSeq counts:
JS.count.files <- system.file(
    paste0("extdata/cts/",
           decoder$sample.ID,
           "/QC.spliceJunctionAndExonCounts.forJunctionSeq.txt.gz"
           ),
     package="JctSeqData");

#Part of one of the count files:
read.table(JS.count.files[1])[526:552,]
```

```
##                            V1   V2
## 526 ENSRNOG00000024112:J043    8
## 527 ENSRNOG00000024112:J044    3
## 528 ENSRNOG00000024112:J045    1
## 529 ENSRNOG00000000044:A000  417
## 530 ENSRNOG00000000044:E001  188
## 531 ENSRNOG00000000044:E002   43
## 532 ENSRNOG00000000044:E003   52
## 533 ENSRNOG00000000044:E004   59
## 534 ENSRNOG00000000044:E005   39
## 535 ENSRNOG00000000044:E006   46
## 536 ENSRNOG00000000044:E007   37
## 537 ENSRNOG00000000044:E008   53
## 538 ENSRNOG00000000044:E009   35
## 539 ENSRNOG00000000044:E010   44
## 540 ENSRNOG00000000044:E011   30
## 541 ENSRNOG00000000044:J012   15
## 542 ENSRNOG00000000044:J013   23
## 543 ENSRNOG00000000044:J014   21
## 544 ENSRNOG00000000044:J015   20
## 545 ENSRNOG00000000044:J016   18
## 546 ENSRNOG00000000044:J017   30
## 547 ENSRNOG00000000044:J018   27
## 548 ENSRNOG00000000044:J019   19
## 549 ENSRNOG00000000044:J020   14
## 550 ENSRNOG00000000044:J021    0
## 551 ENSRNOG00000000048:A000 2391
## 552 ENSRNOG00000000048:E001   38
```

A similar file is available with novel splice junctions included:

```
#JunctionSeq counts:
JS.novel.count.files <- system.file(
    paste0("extdata/cts/",
           decoder$sample.ID,
           "/QC.spliceJunctionAndExonCounts.withNovel.forJunctionSeq.txt.gz"
           ),
     package="JctSeqData");

#Part of one of the count files:
read.table(JS.novel.count.files[1])[526:553,]

##                            V1   V2
## 526 ENSRNOG00000024112:J043    8
## 527 ENSRNOG00000024112:J044    3
## 528 ENSRNOG00000024112:J045    1
## 529 ENSRNOG00000000044:A000  417
```

```
## 530 ENSRNOG00000000044:E001  188
## 531 ENSRNOG00000000044:E002   43
## 532 ENSRNOG00000000044:E003   52
## 533 ENSRNOG00000000044:E004   59
## 534 ENSRNOG00000000044:E005   39
## 535 ENSRNOG00000000044:E006   46
## 536 ENSRNOG00000000044:E007   37
## 537 ENSRNOG00000000044:E008   53
## 538 ENSRNOG00000000044:E009   35
## 539 ENSRNOG00000000044:E010   44
## 540 ENSRNOG00000000044:E011   30
## 541 ENSRNOG00000000044:J012   15
## 542 ENSRNOG00000000044:J013   23
## 543 ENSRNOG00000000044:J014   21
## 544 ENSRNOG00000000044:J015   20
## 545 ENSRNOG00000000044:J016   18
## 546 ENSRNOG00000000044:J017   30
## 547 ENSRNOG00000000044:J018   27
## 548 ENSRNOG00000000044:J019   19
## 549 ENSRNOG00000000044:J020   14
## 550 ENSRNOG00000000044:J021    0
## 551 ENSRNOG00000000044:N022    9
## 552 ENSRNOG00000000048:A000 2391
## 553 ENSRNOG00000000048:E001   38
```

2.4 Even smaller dataset

A similar set of count files are available for an even smaller, cut-down dataset. This dataset may be useful for running quick and easy tests.

All the count files are available in the "extdata/tiny/" directory instead of the "extdata/cts/" directory. The annotation files are also available in the "extdata/tiny" directory.

2.5 R Data Objects

This package comes with a few R data objects as well, generated by JunctionSeq for running the JunctionSeq examples. You can load these using the commands:

To load the full dataset:

```
data(fullExampleDataSet,package="JctSeqData");
```

To load the "tiny" dataset:

```
data(exampleDataSet,package="JctSeqData");
```

3 Recreating this data package

Only a small selection of the data generated and used in the pipeline walkthrough has been packaged and distributed in the `JctSeqData` R package. The data had to be reorganized in order to fit with the R package format. This section describes exactly how the JctSeqData package was generated.

First you must download the full example output (file is 200mb). Optionally, you can also download the example bam files (file is 1.1gb).

Now we copy over the template and add in the data generated in this walkthrough:

```
#make sure JctSeqData doesn't already exist:
rm -rf outputData/JctSeqData
#Copy over the template
cp -R inputData/JctSeqData-template outputData/JctSeqData
#Copy original annotation files:
cp inputData/annoFiles/*.* outputData/JctSeqData/inst/extdata/annoFiles/
#Copy additional generated annotation files:
cp outputData/forJunctionSeq.gff.gz                          \
   outputData/JctSeqData/inst/extdata/annoFiles/JunctionSeq.flat.gff.gz
cp outputData/forDEXSeq.gff.gz                               \
   outputData/JctSeqData/inst/extdata/annoFiles/DEXSeq.flat.gff.gz
cp outputData/countTables/orphanSplices.gff.gz               \
   outputData/JctSeqData/inst/extdata/annoFiles/
cp outputData/countTables/withNovel.forJunctionSeq.gff.gz \
   outputData/JctSeqData/inst/extdata/annoFiles/
#Copy count tables:
cp -R outputData/countTables/* outputData/JctSeqData/inst/extdata/cts/
```

Next we generate the "tiny" dataset used in the JunctionSeq examples and for rapid testing. This is done simply by using "egrep" to extract a subset of the genes from the various files:

```
cd outputData/JctSeqData/inst/extdata/
#Make a "egrep" regex string to extract the desired genes:
FILTER="ENSRNOG00000048600|ENSRNOG00000045591|etc. etc.";
#Note: this is only the start of the full regex string.
#      see file inputData/JctSeqData-template/inst/extdata/tinyGeneList.txt
#      for a full list of the extracted genes.

#Subsample annotation files:
zcat annoFiles/anno.gtf.gz | \
      egrep $FILTER - | \
      gzip -c - > tiny/anno.gtf.gz
zcat annoFiles/JunctionSeq.flat.gff.gz | \
      egrep $FILTER - | \
      gzip -c - > tiny/JunctionSeq.flat.gff.gz
zcat cts/withNovel.forJunctionSeq.gff.gz | \
      egrep $FILTER - | \
      gzip -c - > tiny/withNovel.forJunctionSeq.gff.gz
zcat cts/withNovel.forJunctionSeq.gff.gz | \
      egrep $FILTER - | \
      gzip -c - > tiny/withNovel.forJunctionSeq.gff.gz
#Subsample count files:
while read line
do
  mkdir ./tiny/$line
  echo $line
  zcat cts/$line/QC.spliceJunctionAndExonCounts.withNovel.forJunctionSeq.txt.gz | \
     egrep $FILTER - | \
     gzip -c - > tiny/$line/QC.spliceJunctionAndExonCounts.withNovel.forJunctionSeq.txt.gz
  zcat cts/$line/QC.exonCounts.formatted.for.DEXSeq.txt.gz | \
     egrep $FILTER - | \
     gzip -c - > tiny/$line/QC.exonCounts.formatted.for.DEXSeq.txt.gz
  zcat cts/$line/QC.geneCounts.formatted.for.DESeq.txt.gz | \
     egrep $FILTER - | \
     gzip -c - > tiny/$line/QC.geneCounts.formatted.for.DESeq.txt.gz
  zcat cts/$line/QC.spliceJunctionAndExonCounts.forJunctionSeq.txt.gz | \
     egrep $FILTER - | \
     gzip -c - > tiny/$line/QC.spliceJunctionAndExonCounts.forJunctionSeq.txt.gz
  zcat cts/$line/QC.spliceJunctionCounts.knownSplices.txt.gz | \
     egrep $FILTER - | \
     gzip -c - > tiny/$line/QC.spliceJunctionCounts.knownSplices.txt.gz
done < annoFiles/sampleID.list.txt
cd ../../../../
```

We can install this almost-finished version of the package using the command:

```
R CMD INSTALL outputData/JctSeqData
```

Next, we build the serialized "Rdata" files in R. To do this, first we load the datasets:

```
#Read the decoder:
decoder.file <- system.file("extdata/annoFiles/decoder.bySample.txt",
                            package="JctSeqData");
decoder <- read.table(decoder.file,
                      header=TRUE,
                      stringsAsFactors=FALSE);
#Here are the full-size count and gff files:
gff.file.FULL <- system.file("extdata/cts/withNovel.forJunctionSeq.gff.gz",
                         package="JctSeqData");
countFiles.FULL <- system.file(paste0("extdata/cts/",
      decoder$sample.ID,
      "/QC.spliceJunctionAndExonCounts.withNovel.forJunctionSeq.txt.gz"),
      package="JctSeqData");
#Here are the "tiny" subset count and gff files:
gff.file.TINY <- system.file("extdata/tiny/withNovel.forJunctionSeq.gff.gz",
                         package="JctSeqData");
countFiles.TINY <- system.file(paste0("extdata/tiny/",
      decoder$sample.ID,
      "/QC.spliceJunctionAndExonCounts.withNovel.forJunctionSeq.txt.gz"),
      package="JctSeqData");
```

Next, we generate the full dataset:

```
jscs2 <- runJunctionSeqAnalyses(sample.files = countFiles,
           sample.names = decoder$sample.ID,
           condition=factor(decoder$group.ID),
           flat.gff.file = gff.file);
```

And save it:

```
save(jscs2, file = "outputData/JctSeqData/data/fullExampleDataSet.RData");
```

And then generate the "tiny" dataset:

```
jscs <- runJunctionSeqAnalyses(sample.files = countFiles.TINY,
           sample.names = decoder$sample.ID,
           condition=factor(decoder$group.ID),
           flat.gff.file = gff.file.TINY);
```

And save it:

```
save(jscs, file = "outputData/JctSeqData/data/tinyExampleDataSet.RData");
```

4 References

References

[1] S. Anders and W. Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11:R106, 2010.

[2] S. Anders, A. Reyes, and W. Huber. Detecting differential usage of exons from RNA-seq data. *Genome Research*, 22:2008, 2012.

[3] S. W. Hartley, S. L. Coon, L. E. Savastano, J. C. Mullikin, C. Fu, D. C. Klein, and N. C. S. Program. Neurotranscriptomics: The effects of neonatal stimulus deprivation on the rat pineal transcriptome. *PloS ONE*, 10(9), 2015.

[4] S. W. Hartley and J. C. Mullikin. Qorts: a comprehensive toolset for quality control and data processing of rna-seq experiments. *BMC bioinformatics*, 16(1):224, 2015.

[5] M. D. Robinson and G. K. Smyth. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*, 23:2881, 2007.

5 Legal

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
```

The MIT license and copyright information can also be accessed using the command:

```
java -jar /path/to/jarfile/QoRTs.jar "?" samjdkinfo
```

JunctionSeq is based on the DEXSeq and DESeq2 packages, and is licensed with the GPL v3 license:

```
JunctionSeq is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

JunctionSeq is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with JunctionSeq.  If not, see <http://www.gnu.org/licenses/>.
```

Other software mentioned in this document are subject to their own respective licenses.