QoRTs Package Advanced User Manual

Stephen Hartley
National Human Genome Research Institute
National Institutes of Health

August 8 2014
December 23, 2014
v0.0.29

Contents

1 Overview

The QoRTs software package is a fast, efficient, and portable toolkit designed primarily to aid in the detection and identification of errors, biases, and artifacts produced by paired-end high-throughput RNA-Seq technology. It can produce a wide variety of graphs, plots, and tables that allow the data to be visualized in various ways. Data can be compiled and contrasted in multiple ways to allow systematic errors or artifacts to reveal themselves more easily. While it will not directly assign pass/fail status, it is a powerful tool for bioinformaticians to detect and identify features in the data.

This is the advanced user manual, intended to provide an overview of some of the secondary capabilities of the QoRTs package. For general-purpose use, see the QoRTs vignette, QoRTs-vignette.pdf, which can be found at `https://github.com/hartleys/QoRTs`.

In addition to its primary functionality, QoRTs includes a variety of data processing and visualization tools. QoRTs can:

Additional Functions:

- Generate the gene-level counts necessary for analysis with the *DESeq*/*DESeq2* [**?**] or *edgeR* [**?**] packages.
- Generate the exon-level counts necessary for analysis with the *DEXSeq* [**?**] differential-exon-usage analysis package.
- Generate "flat" annotation files, needed for use with the *DEXSeq* [?] differential-exon-usage analysis package.
- Generate the splice-junction counts necessary for analysis with the `JunctionSeq` differential splice junction usage package.

- Create "wiggle" files for use with the UCSC genome browser.
- Merge and average wiggle files, optionally with user-specified size factors to adjust for library size.
- Create bed files displaying splice junctions (novel and annotated) with coverage counts, for use with the UCSC genome browser.
- Merge multiple splice junction count bed files, optionally with user-specified size factors to adjust for library size.

1.1 Generating a flattened annotation file

Before counting exons and splice junctions, QoRTs generates a set of non-overlapping exonic fragments out of all the exons in the genome annotation gtf file. It then assigns each exonic fragment a unique identifier. Similarly, it assigns every splice junction its own unique identifier. A gtf file listing all these genomic features and their unique identifiers can be created using the following command:

```
java -jar /path/to/jarfile/QoRTs.jar makeFlatGtf
                                     input.gtf
                                     flattened.gff
```

strandedness: You must use the `--stranded` option to create the flattened gff for use with stranded datasets. DO NOT mix stranded flattened gff with unstranded data, or vice versa.

DEXSeq: DEXSeq also requires a flattened annotation file, which is formatted similarly. In order to produce a flattened gff file that DEXSeq can read, include the `--DEXSeqFmt` option.

This gtf file conforms to the UCSC gff file definition, (found here: http://genome.ucsc.edu/FAQ/FAQformat.html). It will contain 4 different feature types (column 3): "aggregate_gene", "exonic_part", "splice_site", and "novel_splice_site".

2 Genome browser tracks

In addition to the standard QC plots, which examine the data as a whole, it is sometimes desirable to be able to query and examine coverage information at specific genetic loci. In particular, when identifying candidate genes via genome-wide analyses, it is often vital to examine the locus for artifacts before carrying out costly and time-consuming validation experiments.

2.1 Generating wiggle tracks

QoRTs includes a utility to generate ".wig", or "wiggle plot" files. These wiggle plot files include counts for the mean coverage for each equal-sized window across the whole genome. These files are designed to be used with the UCSC browser or similar interfaces, and allow easy and intuitive visualization of your data.

```
java -jar /path/to/jarfile/QoRTs.jar bamToWiggle
```

```
                                    infile.bam
                                    trackName
                                    chromLengthFile
                                    outfilePrefix
```

The chromLengthFile is a simple tab-delimited text file that includes each chromosome in the first column and the chromosome's length (in base-pairs) in the second column. If the wiggle file is intended for use with a standard genome on the UCSC genome browser, then the UCSC utility `fetchChromSizes` should be used to generate this file. (see http://genome.ucsc.edu/goldenPath/help/bigWig.html for more information on `fetchChromSizes`, as well as information on how to compress your wig files into smaller and more efficient bigWig files)

Common options and flags for this function include:

- `--sizefactor 1.0`: A float value. All the coverage values will be divided by this factor. Useful for comparing two samples that may have different normalization factors.
- `--stranded`: Flag to indicate that data should be treated as stranded.
- `--stranded_fr_secondstrand`: Flag to indicate that the data is of the fr_secondstrand stranded library type.
- `--negativeReverseStrand`: If this flag is set, then the negative strand will be counted in negative numbers. This can be useful for plotting both strands in a single multiwig track, via a trackhub. (see http://genome.ucsc.edu/goldenPath/help/trackDb/trackDbDoc.html)

2.2 Merging wiggle tracks

QoRTs includes a utility for summing or averaging multiple wiggle files.

```
java -jar /path/to/jarfile/QoRTs.jar mergeWig
                                     namelist
                                     outfile.wig.gz
```

The `namelist` parameter can be either a comma-delimited list of input files, the path to a text file (which must end with '.txt') containing a list of input filenames (one per line), or "-" to read file names from standard input (one per line). Each input file should be a ".wig" file, preferably created via the bamToWiggle tool. All the wiggle files MUST have the same window size, chromosome lengths, and chromosome ordering. By default, each window in the output wiggle file will be the sum of that same window across all the input wiggle files.

Common options and flags for this function include:

- `--calcMean`: If this flag is raised, the utility will calculate the average rather than the sum-total coverage for each window.
- `--makeNegative`: If this flag is raised, the output will be multiplied by -1.
- `--infilePrefix`: A prefix to preppend to each input name in the `namelist` parameter.
- `--infileSuffix`: A suffix to append to each input name in the `namelist` parameter.

Optionally, each input wiggle file can be adjusted/normalized via a user-specified "size-factor". This size factor is usually used when attempting to normalize and average biological replicates, where the

total read count and library size may not be constant. A simple normalization factor can be calculated by simply using the number of reads. A better normalization factor would be to use the estimateSizeFactorsForMatrix() function contained in the *DESeq2* [?] package. For most purposes, the normalization factors across all samples in the comparison should add up to 1. Each input wiggle file's wiggle counts will be divided by the size factor before the samples are added (or averaged, if the `--calcMean` option is used).

Size factors can be added in a number of ways. the `--sizeFactors` option can be used with a list of size factors, delimited with commas (with no whitespace in between). Alternately, if the `namelist` parameter is a filename or "-", then the size factor can be provided as a second column (delimited with tabs). Finally, even if size factors are included, the utility will ignore them if the

`--ignoreSizeFactors` option is used.

For more information and a full accounting of all parameters and options, use the command:

```
java -jar /path/to/jarfile/QoRTs.jar mergeWig --man
```

2.3 Generating splice-junction tracks

Splice junction counts can be made into a separate bed track using the command:

```
java -jar /path/to/jarfile/QoRTs.jar makeSpliceBed
                                          filelist.txt
                                          outfile.bed
```

Common options and flags for this function include:

- *–rgb*: The color to use for each bed entry.

2.4 Merging splice-junction tracks

For more information and a full accounting of all parameters and options, use the command:

```
java -jar /path/to/jarfile/QoRTs.jar makeSpliceBed --man
```

2.5 Merging Count Data

For the purposes of quality control it is generally preferable to run QoRTs on each sample-run individually, so that potential technical artifacts related to sequencing run or lane can be identified. However, for most downstream purposes these "technical replicates" will be combined and treated as a single sample. Differential expression tools like *DESeq*, *DESeq2* [?], *DEXSeq* [?], and *EdgeR* [**?**] assume that each set of gene counts (or exon counts, for *DEXSeq*) is derived from a different biological sample.

Thus, the java utility includes a function for quickly and easily calculating merged sample-wise counts.

```
java -jar /path/to/jarfile/QoRTs.jar mergeAllCounts
                                     decoder.txt
                                     /path/to/qc/results/dir/
                                     ./merged/
```

Alternatively, the merger can be performed for a single sample directly, via the command:

```
java -jar /path/to/jarfile/QoRTs.jar mergeCounts
          ./CtrlDay1_RG1/,./CtrlDay1_RG2/,./CtrlDay1_RG3/
          ./merged/CD1/
```

The list of QC data directories must be separated by commas and contain no whitespace.

For more information and a full accounting of all parameters and options, use the command:

```
java -jar /path/to/jarfile/QoRTs.jar mergeAllCounts --man
and
java -jar /path/to/jarfile/QoRTs.jar mergeCounts --man
```

3 Importing data into other tools

In addition to providing quality control information, QoRTs also provides the requisite input files needed for the DESeq/DESeq2 [?], *DEXSeq* [?], and *EdgeR* [?, ?, **?**] analysis tools. These files will be identical to those that would be generated by HTSeq (using the default "union rule" option).

All the data files can be found in the `qc.data.dir` directory. The files for use with *DESeq*, *DESeq2*, and *EdgeR* will be named `QC.geneCounts.formatted.for.DESeq.txt.gz` and the files for use with *DEXSeq* will be named `QC.exonCounts.formatted.for.DEXSeq.txt.gz`

4 Documentation of the raw QC data

The QCQORTS_COMPLETED_OK simply indicates that the QoRTs scala utility completed without error. If the file exists, then the scala utility completed.

All QoRTs output QC files are tab-delimited gzip-compressed text files (except the summary file, which is left as uncompressed text).

4.0.1 Chrom Counts

The file QC.chromCount.txt.gz contains read-pair counts tabulating the number of read-pairs that appear on each chromosome.

Columns:

- CHROM: chromosome name
- FWD_CT: forward-strand read-pair count (stranded data only).
- REV_CT: reverse-strand read-pair count (stranded data only).
- CT: total read-pair count.

4.0.2 Cigar Op distribution by read cycle

The files QC.cigarOpDistribution.byReadCycle.R1.txt.gz and QC.cigarOpDistribution.byReadCycle.R
contains information about the rates at which each possible "cigar operation" occurs at each read cycle. "Cigar Operations" are defined by the specification of the SAM file format, and are used to describe how a read-pair aligns to the reference sequence. This includes splice junctions, deletions, insertions, and hard/soft clipping.

Columns:

- TODO!

4.0.3 Cigar Op length distribution

The files QC.cigarOpLengths.byOp.R1.txt.gz and QC.cigarOpLengths.byOp.R2.txt.gz contains the occurance counts of cigar operation lengths, for each cigar operation.

Columns:

- OP: The cigar operation character code.
- LEN: The length of the cigar operation.
- CT: The number of times the given operation and length is observed in the input reads.

4.0.4 GC Count

The files QC.gc.R1.txt.gz, QC.gc.R2.txt.gz, QC.gc.RB.txt.gz, and QC.gc.txt.gz contains information about the G/C distribution.

Columns:

- NUM_BASES_GC: The number of bases that are either G or C.
- CT: The number of times the given NUM$_BASES_GCisfoundintheinputreads$.

QC.gc.R1.txt.gz contains the G/C distribution for read 1. QC.gc.R1.txt.gz contains the G/C distribution for read 2. QC.gc.RB.txt.gz contains the G/C distribution for both reads counted together as a pair. Finally, QC.gc.txt.gz contains the G/C distribution for all reads.

4.0.5 Gene Body Coverage

The files QC.geneBodyCoverage.by.expression.level.txt.gz and QC.geneBodyCoverage.genewise.txt.g describe the 5' to 3' gene body coverage.

TODO!

4.0.6 Gene Counts

TODO!

4.0.7 Insert Size

The file QC.insert.size.debug.txt.gz describes the insert size distribution.

TODO!

4.0.8 NVC

A number of files describe the nucleotide-count by cycle (aka read-position).

NVC files:

- QC.NVC.raw.R1.txt.gz: This
- QC.NVC.raw.R2.txt.gz: see above, but for read 2.
- QC.NVC.minus.clipping.R1.txt.gz:
- QC.NVC.minus.clipping.R2.txt.gz: see above, but for read 2.
- QC.NVC.tail.clip.R1.txt.gz:
- QC.NVC.tail.clip.R2.txt.gz: see above, but for read 2.
- QC.NVC.lead.clip.R1.txt.gz:
- QC.NVC.lead.clip.R2.txt.gz: see above, but for read 2.

Each file is formatted with the columns:

Columns:

- `readPos`: The position in the read.
- `base`: The nucleotide base
- `CT_Aligned_to_Fwd`: The number of reads with the given base at the given position, on the forward strand.
- `CT_Aligned_to_Rev`: The number of reads with the given base at the given position, on the reverse strand.
- `CT`: The number of reads with the given base at the given position.

4.0.9 Phred Quality Scores

The files `QC.quals.r1.txt.gz` and `QC.quals.r2.txt.gz` describe the Phred quality control distribution for read 1 and read 2, respectively.

Columns:

- `readLen`: The read position (aka cycle)
- `min`: The minimum Phred score at the given position, across all reads.
- `lowerQuartile`: The lower quartile bound of the Phred score at the given position, across all reads.
- `median`: The median Phred score at the given position across all reads.
- `upperQuartile`: The upper quartile bound of the Phred score at the given position, across all reads.
- `max`: The maximum Phred score at the given position, across all reads.

4.0.10 Summary Data

TODO!

5 References

6 Session Information

The session information records the versions of all the packages used in the generation of the present document.

```
sessionInfo()
```

```
## R version 3.1.1 (2014-07-10)
## Platform: x86_64-unknown-linux-gnu (64-bit)
```

```
##
## locale:
## [1] C
##
## attached base packages:
## [1] parallel  stats4    methods   stats     graphics  grDevices utils
## [8] datasets  base
##
## other attached packages:
##  [1] QoRTsExampleData_0.0.23 BiocStyle_1.4.1         edgeR_3.8.2
##  [4] limma_3.22.1            DESeq2_1.6.1            RcppArmadillo_0.4.500.0
##  [7] Rcpp_0.11.3             GenomicRanges_1.18.1    GenomeInfoDb_1.2.2
## [10] IRanges_2.0.0           S4Vectors_0.4.0         BiocGenerics_0.12.0
## [13] MASS_7.3-35             QoRTs_0.0.29            png_0.1-7
## [16] Cairo_1.5-6             knitr_1.7
##
## loaded via a namespace (and not attached):
##  [1] AnnotationDbi_1.28.1 BBmisc_1.8           BatchJobs_1.5
##  [4] Biobase_2.26.0       BiocParallel_1.0.0   DBI_0.3.1
##  [7] Formula_1.1-2        Hmisc_3.14-5         RColorBrewer_1.0-5
## [10] RSQLite_1.0.0        XML_3.98-1.1         XVector_0.6.0
## [13] acepack_1.3-3.3      annotate_1.44.0      base64enc_0.1-2
## [16] brew_1.0-6           checkmate_1.5.0      cluster_1.15.3
## [19] codetools_0.2-9      colorspace_1.2-4     digest_0.6.4
## [22] evaluate_0.5.5       fail_1.2             foreach_1.4.2
## [25] foreign_0.8-61       formatR_1.0          genefilter_1.48.1
## [28] geneplotter_1.44.0   ggplot2_1.0.0        grid_3.1.1
## [31] gtable_0.1.2         iterators_1.0.7      lattice_0.20-29
## [34] latticeExtra_0.6-26  locfit_1.5-9.1       munsell_0.4.2
## [37] nnet_7.3-8           plyr_1.8.1           proto_0.3-10
## [40] reshape2_1.4         rpart_4.1-8          scales_0.2.4
## [43] sendmailR_1.2-1      splines_3.1.1        stringr_0.6.2
## [46] survival_2.37-7      tools_3.1.1          xtable_1.7-4
```

7 Legal

This software is "United States Government Work" under the terms of the United States Copyright Act. It was written as part of the authors' official duties for the United States Government and thus cannot be copyrighted. This software is freely available to the public for use without a copyright notice. Restrictions cannot be placed on its present or future use.

Although all reasonable efforts have been taken to ensure the accuracy and reliability of the software and data, the National Human Genome Research Institute (NHGRI) and the U.S. Government does not

and cannot warrant the performance or results that may be obtained by using this software or data. NHGRI and the U.S. Government disclaims all warranties as to performance, merchantability or fitness for any particular purpose.

In any work or product derived from this material, proper attribution of the authors as the source of the software or data should be made, using "NHGRI Genome Technology Branch" as the citation.

NOTE: The Scala package includes (internally) the sam-JDK library (sam-1.113.jar), from picard tools. The MIT license and copyright information can be accessed using the command:

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