RSeQC - Comprehensive Reference Guide

Overview

RSeQC (RNA-seq Quality Control) is a comprehensive package providing quality control metrics specifically designed for aligned RNA-seq data. Unlike FastQC which operates on raw sequencing reads, RSeQC analyzes BAM alignment files to detect biology-specific issues that only become apparent after mapping to a reference genome.

Website: http://rseqc.sourceforge.net/
Publication: Wang et al. Bioinformatics 2012
Current Version: 5.0.1+

Applicable to: RNA-seq, small RNA-seq, long-read RNA-seq, metatranscriptomics, any BAM-based sequencing data

---

What RSeQC Does

Core Functions

1. **Read Distribution** - Analyzes where reads map across genomic features
2. **Junction Analysis** - Classifies splice junctions and tests saturation
3. **Gene Body Coverage** - Detects 5' to 3' bias in coverage
4. **Insert Size Distribution** - Measures fragment sizes (PE only)
5. **Artifact Profiling** - Detects systematic sequencing errors

RSeQC vs Other QC Tools

Feature	RSeQC	FastQC	Qualimap
Input	BAM (aligned)	FASTQ (raw)	BAM (aligned)
Splice junctions	Yes	No	Limited
Gene body coverage	Yes	No	Yes
Read distribution	Detailed	No	Basic
Strand detection	Implicit	No	Yes
Speed	Moderate	Very fast	Slow
RNA-seq specific	Excellent	Limited	Good

Installation

Via conda (recommended)

```
conda install -c bioconda rseqc
```

Via pip

```
pip install RSeQC
```

Check installation

```
# List all RSeQC scripts
ls $(dirname $(which read_distribution.py))/*.py

# Check version
read_distribution.py --version
```

Input Requirements

BAM File Requirements

Required features:

- Sorted by coordinate
- Indexed (.bai file present)
- Contains splice junction information (CIGAR 'N' operations)
- Proper mate pairing (for PE data)

Verify BAM:

```
# Check if sorted
samtools view -H sample.bam | grep "SO:coordinate"

# Check for index
ls sample.bam.bai

# Check splice junctions present
samtools view sample.bam | head -100 | grep "N" | wc -l
```

BED12 Annotation Requirements

RSeQC requires **BED12 format** (12-column BED) containing gene models:

Required columns:

1. Chromosome
2. Start position (0-based)
3. End position
4. Name (gene/transcript ID)
5. Score
6. Strand (+/-)
7. Thick start
8. Thick end

9. RGB color
10. Block count (exon count)
11. Block sizes (exon sizes)
12. Block starts (exon starts)

Convert GTF to BED12:

```
# Using UCSC tools
gtfToGenePred annotation.gtf annotation.genePred
genePredToBed annotation.genePred annotation.bed

# Using custom script (if available)
gtf2bed.py annotation.gtf > annotation.bed
```

Verify BED12:

```
# Check column count (must be 12)
head -n 1 annotation.bed | awk '{print NF}'

# Check exon structure
head -n 5 annotation.bed | cut -f10-12
```

Critical: Chromosome names must match between BAM and BED

```
# Check BAM chromosomes
samtools view -H sample.bam | grep "^@SQ" | cut -f2 | sed 's/SN://'

# Check BED chromosomes
cut -f1 annotation.bed | sort -u

# If mismatch (chr1 vs 1), add/remove "chr" prefix
```

RSeQC Scripts Overview

1. read_distribution.py

Purpose: Determines where reads map across genomic features

Command:

```
read_distribution.py \
    --input-file sample.bam \
    --refgene annotation.bed \
    > sample.read_distribution.txt
```

Output categories:

Feature	Expected % (mRNA)	Interpretation
CDS__Exons	50-70%	Coding sequence - main target
5'UTR__Exons	5-10%	5' untranslated region
3'UTR__Exons	10-20%	3' untranslated region
Introns	<10%	Between exons - should be low
TSS__up__1kb	1-3%	Upstream of transcription start
TES__down__1kb	1-3%	Downstream of transcription end
Intergenic	<5%	Outside annotated genes

Example output:

```
Total Reads                    50000000
Total Tags                     52000000
Total Assigned Tags            48000000

Group                  Total_bases         Tag_count        Tags/Kb
CDS_Exons              120000000           30000000         250.00
5'UTR_Exons            15000000            3000000          200.00
3'UTR_Exons            30000000            7500000          250.00
Introns                80000000            2400000          30.00
TSS_up_1kb             5000000             500000           100.00
TES_down_1kb           5000000             500000           100.00
Intergenic             100000000           2000000          20.00
```

Interpretation guide:

High introns (>20%):

- Genomic DNA contamination
- Degraded RNA (incomplete splicing)
- Pre-mRNA contamination
- Wrong sample type (should be RNA, not DNA)

High intergenic (>10%):

- Wrong genome annotation version
- Contamination with different organism
- Unannotated transcripts (could be novel - not necessarily bad)

Low CDS (<40%):

- Heavily degraded RNA
- Strand-specificity issues
- Wrong library type assumption

---

2. inner_distance.py (PE only)

Purpose: Measures insert size distribution between paired-end mates

Command:

```
inner_distance.py \
    --input-file sample.bam \
    --refgene annotation.bed \
    --mapq 30 \
    --out-prefix sample
```

Parameters:

- `--mapq 30`: Only use high-quality alignments (99.9% confidence)
- Lower MAPQ: More reads, less reliable
- Higher MAPQ: Fewer reads, more reliable

Formula:

```
Inner Distance = Insert Size - (Read1_length + Read2_length)

Example:
  Insert size: 300bp
  Read lengths: 100bp each
  Inner distance: 300 - (100 + 100) = 100bp
```

Output files:

- `sample.inner_distance.txt`: Statistics (mean, median, std)
- `sample.inner_distance_plot.pdf`: Histogram
- `sample.inner_distance_freq.txt`: Frequency table

Expected ranges:

Library Type	Expected Inner Distance
mRNA (standard)	50-300bp
Small fragments	20-100bp
Long fragments	300-1000bp
Small RNA	Negative (reads overlap)

Example output:

```
Mean: 150bp
Median: 148bp
Std: 45bp
25th percentile: 110bp
75th percentile: 190bp
```

Interpretation guide:

Very large distances (>1000bp):

- Genomic DNA contamination (spans large introns)
- Chimeric reads / wrong mate pairing
- Trans-splicing events

Very small distances (<20bp):

- Adapter dimers
- Over-digestion during fragmentation
- Size selection failure

Bimodal distribution (two peaks):

- Mixed libraries (different fragment sizes)
- Size selection issues
- Could indicate two distinct RNA populations (mRNA + small RNA)

---

3. junction_annotation.py

Purpose: Classifies splice junctions as known (annotated) vs novel

Command:

```
junction_annotation.py \
    --input-file sample.bam \
    --refgene annotation.bed \
    --mapq 30 \
    --min-intron 50 \
    --out-prefix sample
```

Parameters explained:

--min-intron 50: Minimum intron size

```
Gap <50bp:  Deletion (sequencing error or genetic variant)
Gap  50bp:  Intron (splice junction)

Biological rationale:
  - Smallest human introns: ~50-70bp
  - Typical intron: 100-10,000bp
  - Largest introns: >100kb
  - Setting too low: False positive junctions
  - Setting too high: Miss tiny rare introns
```

--mapq 30: Mapping quality threshold

```
MAPQ 30 = 99.9% confidence (1 in 1000 chance of error)
MAPQ 20 = 99% confidence
MAPQ 10 = 90% confidence
MAPQ 0 = Multi-mapper or very uncertain

Higher MAPQ:
   Excludes multi-mappers (MAPQ 0-3)
   Prevents false junctions from spurious alignments
   May miss some true junctions in repetitive regions
```

Output files:

- `sample.junction.txt`: Junction statistics table
- `sample.splice_junction.pdf`: Pie chart of categories
- `sample.splice_events.pdf`: Splice event types
- `sample.junction.bed`: BED file of all detected junctions

Junction categories:

Category	Both Sites Annotated?	Interpretation
Complete novel	No	Could be: novel isoform, error, or wrong annotation
Partial novel (donor)	Donor yes, Acceptor no	Alternative acceptor site
Partial novel (acceptor)	Acceptor yes, Donor no	Alternative donor site
Known	Yes	Both splice sites in annotation

Example output:

```
Total junctions:           50000
Known junctions:           42000 (84%)
Partial novel (donor):     3000 (6%)
Partial novel (acceptor):  3000 (6%)
Complete novel:            2000 (4%)

Splice site motifs:
  GT/AG (canonical):       47500 (95%)
  GC/AG:                   2000 (4%)
  AT/AC:                   300 (0.6%)
  Non-canonical:           200 (0.4%)
```

Interpretation guide:

>80% known junctions:

- Good quality alignment
- Correct annotation version
- Accurate junction detection

High novel junctions (>30%):

- Wrong annotation version (major update missed)
- Contamination with different organism

- Poor alignment quality (false junctions)
- Novel isoform discovery (good for research!)

Very low junction count (<10,000 for human):

- Genomic DNA contamination (no splicing)
- Wrong sample type
- Very low depth

High non-canonical junctions (>5%):

- Sequencing errors
- Alignment artifacts
- U12-type introns (rare but real)

---

4. junction_saturation.py

Purpose: Tests if sequencing depth is sufficient to detect all junctions

Command:

```
junction_saturation.py \
    --input-file sample.bam \
    --refgene annotation.bed \
    --mapq 30 \
    --min-intron 50 \
    --out-prefix sample
```

How it works:

```
1. Subsample reads at 5%, 10%, 15%, ..., 95%, 100%
2. Count junctions detected at each depth
3. Plot curve of junctions vs depth
4. Assess saturation

Saturated (plateau):






    More sequencing won't find many new junctions

Not saturated (still rising):
    More sequencing will find more junctions
```

Output files:

- `sample.junctionSaturation_plot.pdf`: Saturation curve
- `sample.junctionSaturation_plot.r`: R script to recreate plot

Interpretation:

Plateau reached:

- Sufficient depth for junction detection
- Additional sequencing won't find many new junctions
- OK to proceed with analysis

Still rising steeply:

- Insufficient depth
- More sequencing recommended
- Proceed with caution - may miss rare isoforms

Never rises:

- Genomic DNA contamination (no junctions)
- Very low quality data

---

5. geneBody_coverage.py

Purpose: Analyzes coverage uniformity across gene bodies (5' to 3')

Command:

```
geneBody_coverage.py \
    --input-file sample.bam \
    --refgene annotation.bed \
    --out-prefix sample
```

How it works:

```
For each gene:
  1. Divide into 100 bins (percentiles)
  2. Calculate coverage in each bin
  3. Normalize to mean coverage = 1.0
  4. Average across all genes

Plot shows:
  X-axis: Gene position (5' → 3')
  Y-axis: Relative coverage
```

```
Perfect uniform coverage:
  1.0

  0.5

  0.0
      0%    50%    100%
          Gene body

3' bias (RNA degradation):
  1.0

  0.5

  0.0
      0%    50%    100%

5' bias (some library preps):
  1.0

  0.5

  0.0
      0%    50%    100%
```

Output files:

- `sample.geneBodyCoverage.txt`: Coverage data (100 columns)
- `sample.geneBodyCoverage.pdf`: Coverage plot

Interpretation guide:

Relatively flat profile (coverage variation <30%):

- Good RNA quality
- Minimal degradation
- Unbiased library prep

Strong 3' bias (3' coverage >2x higher than 5'):

- RNA degradation
- Poly-A selection issues
- Long storage time
- Harsh RNA extraction

Strong 5' bias (5' coverage >2x higher than 3'):

- Some specific library prep methods
- Chemical RNA fragmentation artifacts

- Less common than 3' bias

High variability:

- Low gene count in BED file
- Annotation mismatch with BAM
- Very low sequencing depth
- Mixed sample quality

Why use housekeeping genes?

```
Housekeeping genes (GAPDH, ACTB, etc.):
   Highly expressed → better coverage
   Constitutively expressed → consistent across samples
   Low variability → cleaner signal
   Less affected by experimental conditions

All genes:
   Includes low-expressed genes (noisy)
   Includes differentially expressed genes (variable)
   More variability in results
```

---

6. insertion_profile.py

Purpose: Analyzes where insertions occur along reads

Command:

```
insertion_profile.py \
    --input-file sample.bam \
    --sequencing PE \           # or SE
    --out-prefix sample
```

What it detects:

- Systematic insertion artifacts (not random errors)
- Position-specific insertion bias
- Library prep issues causing insertions

Expected pattern: Random insertions evenly distributed

Red flags:

- Insertions clustered at read ends → Adapter contamination
- Insertions at specific positions → Systematic errors
- Very high insertion rate (>2%) → Quality issues

---

7. deletion_profile.py

Purpose: Analyzes deletion patterns along reads

Command:

```
deletion_profile.py \
    --input-file sample.bam \
    --out-prefix sample \
    --read-align-length 100
```

Expected pattern: Random deletions evenly distributed

Red flags:

- Deletions clustered at positions → Systematic errors
- Very high deletion rate (>2%) → Quality issues

---

8. clipping_profile.py

Purpose: Analyzes where reads are soft/hard clipped

Command:

```
clipping_profile.py \
    --input-file sample.bam \
    --sequencing PE \
    --out-prefix sample
```

Clipping types:

Type	CIGAR	Meaning
Soft clip	S	Bases present in read but not aligned
Hard clip	H	Bases removed from read entirely

Why clipping occurs:

1. **Adapter contamination** - Clipped from ends
2. **Low quality bases** - Clipped from ends
3. **Splice junctions** - Internal clipping (RNA-seq specific)
4. **Structural variants** - Large insertions/deletions

Expected pattern:

- Low clipping at read ends (<5%)
- Some internal clipping (splice junctions in RNA-seq)

Red flags:

- High end-clipping (>20%) → Adapter issues

- Very high internal clipping → Alignment problems

---

9. mismatch_profile.py

Purpose: Analyzes mismatch patterns to detect systematic errors

Command:

```
mismatch_profile.py \
    --input-file sample.bam \
    --out-prefix sample \
    --read-align-length 100
```

Expected pattern: Random mismatches, low rate (<1%)

Red flags:

- Position-specific mismatches → Systematic sequencing errors
- High mismatch rate (>2%) → Quality issues or wrong reference
- Strand-specific bias → Oxidation damage (G>T on one strand)

Workflow-Specific Considerations

RNA-seq

Essential analyses:

- Read distribution (detect gDNA contamination)
- Junction annotation (verify splicing)
- Gene body coverage (detect 3' bias)
- Junction saturation (assess depth)

Expected values:

- CDS exons: 50-70%
- Introns: <10%
- Known junctions: >80%
- Uniform gene body coverage

Small RNA-seq

Essential analyses:

- Read distribution (should map to ncRNA features)
- Clipping profile (adapter trimming verification)

Expected values:

- High mapping to miRNA/snoRNA annotations
- Negative inner distances (reads overlap)
- Heavy adapter clipping

Long-read RNA-seq

Essential analyses:

- Junction annotation (long reads span multiple junctions)
- Gene body coverage (full-length transcripts)

Adjusted thresholds:

- Higher mismatch rate acceptable (~3-5%)
- Different insert size expectations
- More novel junctions (better isoform detection)

Whole Exome Sequencing (WES)

Essential analyses:

- Read distribution (verify on-target)
- Inner distance (check fragment sizes)

Expected values:

- Very high exon mapping (>90%)
- Very low intron/intergenic (<5%)
- No splice junctions (DNA, not RNA)

Note: Many RSeQC tools designed for RNA-seq; use with caution for DNA-seq

---

Whole Genome Sequencing (WGS)

Limited applicability:

- Read distribution less meaningful (no exon enrichment)
- Junction analysis not applicable (no splicing in DNA)
- Insert size distribution useful

Better alternatives for WGS QC:

- Picard CollectAlignmentSummaryMetrics
- Qualimap
- Mosdepth

Resource Requirements

Memory (RAM)

Analysis	Full BAM (30GB)	Subsampled (1M reads)
read_distribution	4-8GB	2GB
junction_annotation	8-12GB	2GB
junction_saturation	12-16GB	4GB
geneBody_coverage	8-12GB	4GB
Other profiles	4-8GB	2GB

Recommendation: 12-16GB for full analysis

CPU Cores

RSeQC scripts are **single-threaded** (no multi-core support)

Parallelization strategy:

- Run multiple samples in parallel
- Run different analyses in parallel for same sample

Disk Space

Per sample:

- Input BAM: 5-10GB
- Output files: 100-500MB
- Temporary files: Minimal

Time

Analysis	Full BAM	Subsampled
read_distribution	5-10 min	1-2 min
junction_annotation	10-20 min	2-5 min
junction_saturation	30-60 min	5-10 min
geneBody_coverage	20-40 min	5-10 min
All profiles	10-20 min	5-10 min

Total per sample: ~1-2 hours (full BAM), 15-30 min (subsampled)

For 100 samples (parallelized): 1-3 hours

Subsampling Strategy

Why Subsample?

Problem: Some RSeQC analyses (junction_saturation, geneBody_coverage) are very slow on full BAMs

Solution: Subsample to 1 million reads for speed

Benefits:

- 5-10x faster runtime
- 90% accuracy maintained
- Still detects major issues
- Enables analysis of large cohorts

How to Subsample

Using samtools:

```
# Calculate fraction for 1M reads
TOTAL_READS=$(samtools view -c sample.bam)
FRACTION=$(echo "scale=10; 1000000 / $TOTAL_READS" | bc)

# Subsample
samtools view -bs ${FRACTION} sample.bam > sample.1M.bam

# Index
samtools index sample.1M.bam
```

Using sambamba (faster):

```
sambamba view -f bam -t 4 --subsampling-seed=42 -s 0.05 sample.bam > sample.subsample.bam
# 0.05 = 5% = ~1M reads for 20M read dataset
```

When to Use Full BAM

Use full BAM for:

- read_distribution (fast anyway)
- junction_annotation (fast anyway)
- All artifact profiles (need full data)

Use subsampled BAM for:

- junction_saturation (very slow on full BAM)
- geneBody_coverage (very slow on full BAM)

MultiQC Integration

RSeQC outputs are automatically parsed by MultiQC for aggregated reporting.

Files MultiQC recognizes:

- `*.read_distribution.txt`
- `*.junction.txt`
- `*.inner_distance_freq.txt`
- `*.geneBodyCoverage.txt`

MultiQC sections created:

- Read Distribution: Stacked bar chart (all samples)
- Gene Body Coverage: Line plot (all samples overlaid)
- Junction Annotation: Pie charts (per sample)
- Inner Distance: Histograms (per sample, PE only)

Benefits:

- Cross-sample comparison at a glance
- Identify outliers easily
- Interactive plots
- Single HTML report

Troubleshooting

”BED file format error”

Causes:

- Chromosome name mismatch (BAM has ”chr1”, BED has ”1”)
- BED file not in BED12 format (wrong column count)
- Corrupted BED file

Solutions:

```bash
# Check chromosome naming
samtools view -H sample.bam | grep "^@SQ" | cut -f2 | sed 's/SN://'
cut -f1 annotation.bed | sort -u

# Verify BED12 format
head -n 1 annotation.bed | awk '{print NF}'  # Should be 12

# Regenerate BED from GTF
gtfToGenePred annotation.gtf annotation.genePred
genePredToBed annotation.genePred annotation.bed
```

"No reads found in region"

Causes:

- BAM file empty or corrupted
- Wrong chromosome naming
- Very low mapping rate

Solutions:

```
# Check BAM has reads
samtools view sample.bam | head -n 10

# Check mapping rate
samtools flagstat sample.bam

# Verify chromosome names match
samtools view -H sample.bam | grep "^@SQ"
```

Script fails with "Killed" or "Out of memory"

Causes:

- Very large BAM file (>50GB)
- Insufficient memory allocation
- Too many reads for analysis

Solutions:

- Increase memory to 16-32GB
- Subsample BAM to 1M reads
- Process fewer samples in parallel

"Could not determine read length"

Causes:

- BAM has no aligned reads
- Alignment completely failed

Solutions:

```
# Check for aligned reads
samtools view sample.bam | head

# Check alignment stats
samtools flagstat sample.bam
```

Gene body coverage produces weird plot

Causes:

- Too few genes in BED file
- Annotation mismatch with BAM
- Subsampling gave too few reads

Solutions:

- Check gene count: `wc -l annotation.bed`
- Verify annotation version matches genome
- Use more reads in subsample

Best Practices

Standard RNA-seq QC

```
# 1. Read distribution (use full BAM)
read_distribution.py \
    --input-file sample.bam \
    --refgene annotation.bed \
    > sample.read_distribution.txt

# 2. Junction annotation (use full BAM)
junction_annotation.py \
    --input-file sample.bam \
    --refgene annotation.bed \
    --mapq 30 \
    --min-intron 50 \
    --out-prefix sample

# 3. Gene body coverage (use subsampled BAM)
geneBody_coverage.py \
    --input-file sample.1M.bam \
    --refgene housekeeping.bed \
    --out-prefix sample

# 4. Junction saturation (use subsampled BAM)
junction_saturation.py \
    --input-file sample.1M.bam \
    --refgene annotation.bed \
    --mapq 30 \
    --min-intron 50 \
    --out-prefix sample

# 5. Inner distance (PE only, use full BAM)
```

```
inner_distance.py \
    --input-file sample.bam \
    --refgene annotation.bed \
    --mapq 30 \
    --out-prefix sample
```

Minimal QC (Fast)

For quick assessment:

```
# Just read distribution and junction annotation
read_distribution.py --input-file sample.bam --refgene annotation.bed > sample.txt
junction_annotation.py --input-file sample.bam --refgene annotation.bed --out-prefix sample
```

Takes ~5-10 minutes, catches most major issues.

Comprehensive QC (Full)

For publication-quality analysis, run all 9 analyses as shown in the process.

Related Documentation

- **FastQC**: `docs/fastqc.md` - Raw read QC
- **MultiQC**: `docs/multiqc.md` - Aggregated reporting
- **STAR Alignment**: `docs/star_align.md` - Generating input BAMs
- **RSeQC Manual**: http://rseqc.sourceforge.net/

Document Version: 1.0
Last Updated: January 2026
RSeQC Version: 5.0.1+
Applicable to: All BAM-based sequencing applications, optimized for RNA-seq