Rcount: User Guide

Marc W. Schmid, marcschmid@gmx.ch

May 11, 2016

Contents

1 Installation

This section describes how to obtain and install Rcount. Source code, 64-bit binaries for Linux, Windows, and Mac, and R-Scripts can be downloaded on github.com/MWSchmid/Rcount. The programs can easily use more than 3 Gb of RAM and it is therefore strongly recommended to use a 64-bit system with at least 6 Gb of RAM (see Table 4 for examples concerning the memory usage).

1.1 Using pre-compiled binaries

If you have a 64 bit (Ubuntu-like) Linux, Windows (7) or MacOSX, use the pre-compiled binaries. The binaries were built on Ubuntu 14.04, Windows 7, and MacOS 10.10 (versions below 10.10 were not tested). If you encounter problems with the binaries, try building the programs from source (see section 5) and send a report to marcschmid@gmx.ch.

1.1.1 Linux

Install first the Qt5 libraries with `sudo apt-get install qt5-default`. Download and unpack the archive `linux_64bit.zip`. Start Rcount-multireads, Rcount-format, and Rcount-distribute directly either by double-clicking on them or from the terminal (you may need to make them executable first, right-click on the binaries, open the "properties" dialog and check the box for "is executable" - or in a terminal type `chmod 755 filename`).

1.1.2 Windows

Download and unpack the archive `windows_64bit.zip`. Start the applications directly by double-clicking on them.

1.1.3 Mac

Download and unpack the archive `mac_64bit.zip`. Mount the `*.dmg` files (double-click) and start the applications by double-clicking on them.

2 Step-by-step example

This section provides a step-by-step tutorial on how to get count tables starting from initial read files. The example data are from *O. sativa* and comprises two "sperm cell" samples [1]. Download and unpack the archive `rice_tutorial.zip` from github.com/MWSchmid/Rcount. It contains a folder with the rice reference genome and its annotation in gff format (MSU7 from rice.plantbiology.msu.edu). It additionally contains pre-processed `.bam` and `.bai` files in case you would like to try only Rcount and to skip the download and alignment part of the tutorial (in this case go to section 2.4). The short reads download and alignment part is written for an Ubuntu-like Linux.

2.1 OPTIONAL: Installation of additional programs

Additional programs are required to download and align the short reads. It is later assumed that these programs reside in a folder that is included in your PATH environment variable. This can be done by either moving the programs into one of the by-default included folders (e.g. /usr/local/bin), or by adding the folder containing the programs to the PATH environment variable. Note that the latter is a temporary solution (the commands have to be entered each time you start a new terminal). Code for both options is given for each of the programs (note that the hash-tag `#` stands for comments, which do not have to be typed into the terminal).

- SRA toolkit
  Visit `www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software`, download the archive for "Ubuntu Linux 64 bit architecture", unpack it, open a terminal, and type (adjust the path and version number):

```
# SOLUTION 1
cd /path/to/sratoolkit.x.x.x-x-ubuntu64/bin
sudo cp -r * /usr/local/bin
# SOLUTION 2 (temporary!)
export PATH="$PATH:/path/to/sratoolkit.x.x.x-x-ubuntu64/bin"
```

- Bowtie2 [2]
  Visit `bowtie-bio.sourceforge.net/bowtie2/index.shtml` and obtain the latest version. Follow the link in the box on the right side of the page, download the archive for linux, unpack it, open a terminal, and type (adjust the path and version number):

```
# SOLUTION 1
cd /path/to/bowtie2-x.x.x
sudo cp bowtie2* /usr/local/bin
# SOLUTION 2 (temporary!)
export PATH="$PATH:/path/to/bowtie2-x.x.x"
```

- TopHat2 [3]
  Visit `ccb.jhu.edu/software/tophat/index.shtml`, follow the link in the box on the right side of the page, download the latest version for linux ("Linux x86_64 binary"), unpack it, open a terminal, and type (adjust the path and version number):

```
# SOLUTION 1
cd /path/to/tophat-2.x.x.Linux_x86_64
rm README
rm AUTHORS
rm COPYING
sudo cp * /usr/local/bin
# SOLUTION 2 (temporary!)
export PATH="$PATH:/path/to/tophat-2.x.x.Linux_x86_64"
```

- SAMtools [4]
  Visit `sourceforge.net/projects/samtools/files/samtools/` and obtain the latest version. Download the archive and unpack it. SAMtools needs to be built from source. For this, install zlib (`zlib1g`, `zlib1g-dev`, and `zlib1g-dev` from the package manager), open a terminal, and type (adjust the path and version number):

```
# COMPULSORY - BUILD INSTRUCTIONS
cd /path/to/samtools-x.x.x
make
# SOLUTION 1
cd /path/to/samtools-x.x.x
sudo cp samtools /usr/local/bin
# SOLUTION 2 (temporary!)
export PATH="$PATH:/path/to/samtools-x.x.x"
```

2.2 OPTIONAL: Obtaining the short read data

The data used in this tutorial can be conveniently retrieved from NCBI using the SRA toolkit. Open a terminal to download the example data (takes quite some time) in the working directory (e.g. rice_tutorial, which has been automatically created by unpacking the archive `rice_tutorial.zip`):

```
cd /path/to/rice_tutorial
fastq-dump SRR976339
fastq-dump SRR976340
```

2.3 OPTIONAL: Aligning the short reads to the reference genome

The alignment of the short reads to the reference genome using TopHat2 requires an index of the reference genome. Build this index with (takes quite some time):

```
cd /path/to/rice_tutorial
bowtie2-build -f -q -o 0 all.chrs.con rice_genome_index
```

You can now align the reads with TopHat2. Note that the option `-p 6` tells the computer to use six cores. You may need to change this according to your system. The options `-g 10` and `--no-coverage-search` are given to save memory and run-time (`-g 10` sets 10 as maximal number of alignments per read and `--no-coverage-search` omits searching novel exon-junctions using read coverage).

```
cd /path/to/rice_tutorial
mkdir SRR976339
mkdir SRR976340
tophat -p 6 -g 10 --no-coverage-search -o SRR976339 rice_genome_index SRR976339.fastq
tophat -p 6 -g 10 --no-coverage-search -o SRR976340 rice_genome_index SRR976340.fastq
```

The Rcount-multireads program requires the `.bam` files to have an index (`.bai` file). Build this index with (takes few time):

```
cd /path/to/rice_tutorial/SRR976339
samtools index accepted_hits.bam
cd /path/to/rice_tutorial/SRR976340
samtools index accepted_hits.bam
```

2.4 Weighting reads that have more than one alignment with *Rcount-multireads*

With the TopHat2 option `-g 10` enabled, it is possible that a read aligns to up to 10 locations in the genome. To avoid counting the read 10 times, we can weight these individual hits with Rcount-multireads. Start the program by clicking on it, specify the input file (`accepted_hits.bam`) and a corresponding output file (e.g. `accepted_hits_weighted.bam`). Leave the allocation distance at 100 bp (approximately one read length). Press `OK` to start the weighting. This will take some minutes. Once the program has finished you can either process another file or close the program. Run the program on both samples (SRR976339 and SRR976340).

2.5 Reformating the genome annotation with *Rcount-format*

The genome annotation has to be in a specific `.xml` format. You can create this file using Rcount-format. Start the program by clicking on it. Specify the input genome annotation file (`all.gff`, stored in your working directory), leave the rest unchanged, and press `Next` (this can take up to a minute or longer on older systems). You can now inspect the structure of the genome annotation. Click on "gene" to expand the menu. If you expand the entry "mRNA", you see five sub-entries (CDS, exon, five_prime_UTR, splice, and three_prime_UTR). Note that only exon and splice are important, considering that the other three types (CDS, five_prime_UTR, splice, and three_prime_UTR) are already included in the exons. Remove those three by changing their priority to 0 (double-click on the "1" to edit it) and click `Next`. Specify an output file (e.g. `Rcount_rice_annotation.xml` in your working directory) and click `Next`. Finally, close the program by clicking on `Finish`.

2.6 Counting the number of hits per gene with *Rcount-distribute*

Use Rcount-distribute to count the number of reads per gene. Open the program and click `new` to create a new project. This will open a multi-tabbed dialog. For each sample, specify following files under the "Files" tab:

- Input > Annotation: `Rcount_rice_annotation.xml`
  (stored in `/path/to/rice_tutorial`)
- Input > Alignments: `accepted_hits_weighted.bam`
  (stored in `/path/to/rice_tutorial/SRR976339`)
- Output > Alignments: `accepted_hits_weighted_mapped.bam`
  (to be stored in `/path/to/rice_tutorial/SRR976339`)
- Output > Counts: `SRR976339_counts.txt`
  (to be stored in `/path/to/rice_tutorial`)

Note that the example refers to the sample SRR976339. The `.bam` files can be found and should be stored in the corresponding folder (`/path/to/rice_tutorial/SRR976339`). The count table should be stored in the working directory (`/path/to/rice_tutorial`). Go to the "Parameters" tab and enable "Use multiple alignments". Leave the rest unchanged and click `OK`. A dialog will open and ask you to save the project (save it as `SRR976339.xml` in `/path/to/rice_tutorial`). Once you have created for each sample a project, start the analysis by clicking `run all`. Once a sample is processed, the alignment statistics are automatically added to its project (saving is done automatically as well).

2.7 Merging the count tables in R

Rcount-distribute creates for each sample a separate count table stored in the working directory (e.g. `SRR976339_counts.txt`). The provided R-Script offers a function to read in all count tables from a given directory and merge them into a single table. Open R and use the following code to obtain a merged table and to generate a plot that gives a first impression of the data correlation and expression value distribution (table 1 and figure 1).

```
# set the path of the R-script provided
# see in linux_64bit.zip or windows_64bit.zip or mac_64bit.zip
pathToScript <- "/path/to/Rcount-R-functions.R"

# set the path to the folder where the count tables are located
pathToWorkingDirectory <- "/path/to/rice_tutorial"

# load the functions from the script
source(pathToScript)

# read in the count tables
# note that the expression values are rounded to integer numbers
# required by the downstream analysis programs
counts <- f.read.Rcount(pathToWorkingDirectory)

# save a merged table
outfile <- file.path(pathToWorkingDirectory, "all_counts.csv")
write.csv(counts, outfile)

# draw pairwise scatter-plots (unsorted/sorted) and histograms
f.pair.all(log2(counts+1), pathToWorkingDirectory)
```

Table 1: The count table produced in the step-by-step example (type `head(counts)` in R to see this).

	SRR976339_counts	SRR976340_counts
LOC_Os01g01019	13	10
LOC_Os01g01030	55	35
LOC_Os01g01040	127	111
LOC_Os01g01060	70	47
LOC_Os01g01080	5	10
LOC_Os01g01115	28	10

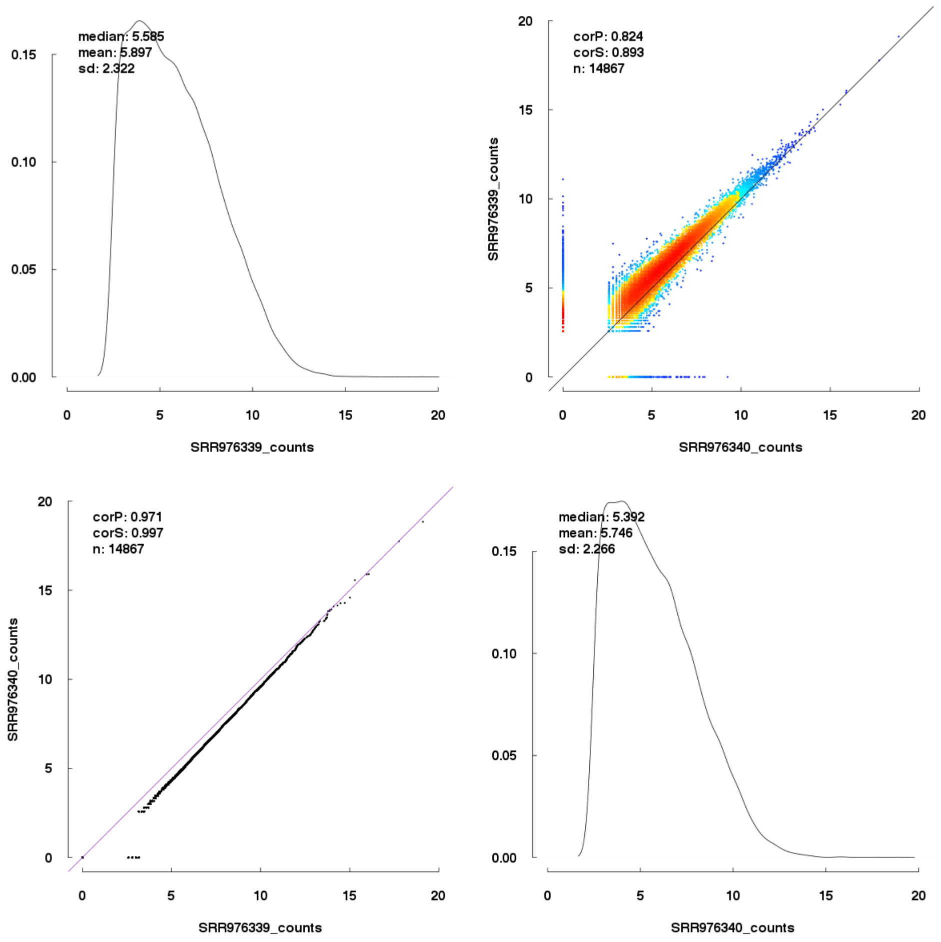

Figure 1: Distribution and pair-wise comparison of gene expression of the two samples processed in the step-by-step tutorial. The plots were generated with the function f.pair.all (included in the R-script). Top-left and bottom-right panels show the distributions of expression values for each sample. The panel on the top-right compares the gene expression values of the two samples (i.e. one dot reflects a gene). Colors indicate the point density: red and blue indicate the highest, respectively lowest, densities. The panel on the lower-left compares the sorted expression values of the two samples (i.e. one dot reflects the x-th highest expression value within one sample). The latter gives an indication of how similar the two distributions of expression values are (e.g. if a certain value means the same in both samples). Expression values correspond to log2(counts+1). CorP: Pearson correlation, CorS: Spearman correlation, n: number of genes plotted.

3 Usage

This section describes the features, requirements and parameters of Rcount in more detail.

3.1 Aligning the short reads

After initial quality checks have been performed (e.g. with the FastQC software), the reads are aligned to a reference genome, preferentially with a splice-aware aligner. In the step-by-step example, we used TopHat2 [3] to align the reads. Other working examples are Subread [5] and STAR [6]. In principle, any aligner works with Rcount, but some may require the `.bam` file to be additionally processed before using it. The following features of the `.bam` file are crucial for Rcount.

- *The file must be sorted and an index must be present.*
  Rcount-multireads needs both, sorting and indexing. Rcount-distribute does not need an index, but sorting is strongly recommended (Rcount-multireads keeps the order intact, resorting is therefore not necessary).
- *The `NH:i:x` tag must be present.*
  Without this tag, multireads are not recognized by Rcount-multireads and treated as unique alignments.
- *There should not be any `XW` or `XM` tag.*
  Rcount uses `XW:f:x` to store the weights of reads with multiple alignments and `XM:i:x` for mapping statistics. The latter is similar to the `FLAG` column of a `.bam` file. The individual bits are explained in table 2.
- *The `CIGAR` string should only contain the operations `M, N, I, D, S, H`.*
  Other operations are not recognized by Rcount-distribute.

Note regarding paired-end reads: Rcount-distribute automatically takes the first read of one pair and ignores the second.

Table 2: Description of the individual bits in the `XM:i:x` tag introduced by Rcount-distribute.

bit	description of the alignment
0x1	belongs to a multiread
0x2	has been skipped by Rcount-distribute
0x4	contains gaps
0x8	has a weight of zero
0x10	maps to a known locus
0x20	maps to a known exon
0x40	maps to a known splice-junction
0x80	maps to ambiguously several loci

3.2 *Rcount-format*: Creating the genome annotation file

To overcome the large variety of genome annotation formats and their sometimes loose definition, Rcount uses a novel format which follows clear structural rules while still offering flexibility to add new features. The conversion of the most common formats (`.gtf, .gff, .bed`) into the `.xml` format is done by Rcount-format. Aside the format conversion, it also enables the user to edit some aspects of the genome annotation.

- *Loci and their transcripts can be extended on both sides.*
  Some samples have an abnormal number of hits in non-genic regions, often caused by hits directly flanking the known loci. Cases like this can be easily detected by comparing the results with and without the extension of the loci.

- *Certain feature types can be removed by setting their priority to zero.*
  The genome annotation sometimes contains ambiguous information. For example, it often contains "coding-sequence" and "UTR" information. However, these are an *in silico* specification of the type "exon" (i.e. exon already includes UTR and CDS), which behave identically during library preparation. Thus, they can be savely removed.

- *Features can be given different priorities during the distribution of ambiguous reads.*
  Depending on the library preparation protocol, it is possible that some of the features in the genome annotation are very unlikely to be sequenced (e.g. rRNA-coding genes with poly(A)-selective library preparation protocols). Instead of removing these features entirely, it is possible to set a lower priority to them. In case a read aligns to a location where two features with different priorities overlap, it is automatically assigned to the one with a higher priority. It is important to note that priorities are only considered on a given level: In case of a gene and a pseudogene (top-level in the genome annotation structure), their priorities are only compared between each other. The pseudogene would not be compared to the gene's sub-feature "mRNA". Likewise, an "mRNA" is not compared to an "exon".

- *Features must have three levels.*
  Rcount-distribute assumes three annotational levels, for example gene-mRNA-exon. Features without three levels are ignored.

- *Specific requirements for BED files.*
  BED files must have at least 12 columns. An optional 13th column holds gene IDs (e.g. AT1G01010). It is assumed that the transcript ID is given as "name" in the fourth column (e.g. AT1G01010.1). The exons are constructed using the fields "blockSizes" and "blockStarts" (column 11 and 12). If there is no 13th column, the gene ID is per default inferred from the transcript ID by removing any `.X` at the end of the transcript ID (e.g. AT1G01010 in case of AT1G01010.1). The feature types are set per default to "gene", "mRNA", and "exon". Non-coding RNAs are identified by identical entries in the fields "thickStart" and "thickEnd" (column 7 and 8) and marked as "non-coding-gene/RNA". Rcount-format further assumes that the coordinates in the BED file are zero-based.

- *Verify that the annotation fits to the reference genome.*
  Fetching the annotation files (BED/GFF/GTF) from an online genome browser can be very convenient. However, it is important to ensure that the reads were aligned to a reference genome for which the annotation file was created (e.g. that the first chromosome is called "Chr1" in the fasta file as well as in the BED/GFF/GTF file). It is therefore recommended to download both, the reference sequence and the genome annotation from the same resource.

For convenience, several pre-processed genome annotations (*A. thaliana*, *B. taurus*, *C. elegans*, *C. familiaris*, *D. melanogaster*, *G. gallus*, *H. sapiens*, *M. musculus*, and *R. norvegicus*) are provided on github.com/MWSchmid/Rcount in the archive `test_data_annotations.zip`. They were build with the data available on tophat.cbcb.umd.edu/igenomes.shtml using the "ENSEMBL data source". The archives on this website also contain pre-built bowtie and bwa indices, which can directly be used in conjunction with the provided genome annotation. To create the individual `.xml` genome annotations, the `genes.gtf` files were first pre-processed with the python script (`convertCufflinksGTFforRcount.py`) supplied in the archives `linux_64bit.zip`, `windows_64bit.zip`, and `mac_64bit.zip` and then processed using Rcount-format (with the ENSEMBL option enabled). Each of the genome annotation has been tested with a random sample from SRA (see section 4 for more details).

3.3 *Rcount-multireads*: Weighting reads with more than one alignment

Some organisms have very large gene families with similar sequence. To avoid underestimation of expression values of transcripts with similar sequence, one can allow multiple alignments of one read to several locations in the genome. However, a read r with $m > 1$ alignments would count m times, resulting in overestimation of expression values. To overcome this, Rcount-multireads calculates for each alignment i of such a read the weight H_i using a "score" S_i divided by the sum of scores from all alignments of the read ($H_i = S_i / \sum_{i=1}^{m} S_i$). If $\sum_{i=1}^{m} S_i$ is zero, all alignments of the respective read are discarded (i.e. all weights set to zero). For ungapped alignments, the score is defined as sum of coverage originating from uniquely aligned reads at the position of the alignment and the surrounding region. The size of the region can be set by choosing the "allocation distance". This value will be added on both sides of the alignment position. For gapped alignments, the score equals to the number of uniquely aligned reads spanning the same gap. Thus, if a read has both types of alignments, the ungapped ones are generally preferred.

Further notes:

- *Rcount-multireads is not required for Rcount-distribute to run.*
  If you do not use reads with multiple alignments, you can safely skip Rcount-multireads. It is important that the number of unique alignments is well above the number of multireads. If not, it is better to use only the uniquely aligned reads (high number of multireads was observed especially in amplified libraries where the cDNA was prepared using random primers, [in-house data]).

- *Memory usage is mainly influenced by the number of multireads.*
  The main memory consumption of Rcount-multireads is caused by storing the multireads. The size of the reference genome does not make a big difference. If RAM is limiting, one could lower the maximum number of alignments per read.

- *Known issues on Mac and Windows.*
  The progress bar is not updated instantly. It instead advances directly to 80 % after finishing reading the files.

- *Rcount-multireads may be substituted by any other program that adds weights to reads with multiple alignments.*
  Rcount-multireads adds the tag `XW:f:x` to specify the weights of reads with multiple alignments. The sole requirement for alternative algorithms is the possibility to add the weights in the form of the `XW:f:x` tag. An example for an alternative software may be CSEM [7] (available on deweylab.biostat.wisc.edu/csem), which was originally developed for ChIP-Seq data analysis using multireads. It calculates the posterior probabilities for all alignments of a multiread and stores them the `ZW:f:x` tag. This tag can be renamed using the following code (example given for an ubuntu-like linux). However, it is important to note that CSEM does not handle gapped alignments in a specific way (treats them as regular alignment). Also, we experienced that summing up the posterior probabilities results in a slightly higher number than the number of reads with multiple alignments (thus, one tends to slightly overweight the multireads).

```
# an example of CSEM usage (instead of Rcount-multireads)
# download, unpack, and compile CSEM (deweylab.biostat.wisc.edu/csem)
cd /path/to/csem
make

# add CSEM to your PATH (temporary!)
export PATH="$PATH:/path/to/csem"

# alignments need to be sorted by read names for CSEM
# however, they are sorted by position in the step-by-step tutorial
# to resort the alignments you need samtools
cd /path/to/rice_tutorial/SRR976339
samtools sort -n accepted_hits.bam byName

# run CSEM to add the posterior probabilities (ZW:f:x tag)
run-csem --bam --no-extending-reads byName.bam 100 withPP

# rename/copy the ZW tag into XW
# (the backslash indicates that all should be written on one single line)
samtools view -h withPP.sorted.bam |\
sed 's/ZW:f:/XW:f:/g' |\
samtools view -bS - > withPP_retagged.bam

# you can now use withPP_retagged.bam directly within Rcount-distribute
```

3.4 *Rcount-distribute*: Mapping aligned reads to genomic features

To count the number of reads per genomic feature, the positions of the aligned reads (termed "hits") have to be mapped to the genomic features. Hits are matched to the genomic features as follows: ungapped hits must be entirely within one exon of a given gene and gapped hits must accurately span exon junctions (corresponds to the intersection_strict mode in HTSeq). Hits can be divided into unambiguous (mapping to transcripts of only one locus) and ambiguous (mapping to transcripts of more than one locus). To avoid counting ambiguous hits multiple times, Rcount-distribute proportionally distributes them based on the number of unambiguous hits. If there are no unambiguous hits, the ambiguous hits are equally distributed. However, we assume a case where two loci A and B overlap such that locus A is entirely located within locus B. Locus A shall be "truly" expressed (i.e. is *in vivo* expressed and therefore has reads mapping to it), locus B not (i.e all reads that map to B are coming from A). Using a single step, all hits of locus A would be declared as ambiguous. In case locus B has no unambiguous hit, the hits from locus A would be equally distributed to locus A and B, leading to an underestimation of the expression value from locus A and an overestimation of the expression value from locus B (a false positive). In another case where locus B has one or two unambiguous hits due to sequencing and/or alignment errors, all the hits from locus A would be wrongly assigned to locus B (one false positive and one false negative). The same error would occur if locus A has a longer transcript *in vivo* than the *in silico* genome annotation would indicate. The hits at the borders of locus A would then be unambiguously assigned to locus B and as a consequence also all the ambiguous hits. To overcome this scenario at least to some extent Rcount-distribute uses a two step approach. In the first step, all hits are mapped to all annotated transcripts. In many cases one can expect that each "truly" expressed transcript should have at least some hits within the first few bps at its 3' end because the library preparation protocols frequently rely on poly(A)-tail priming for cDNA synthesis. In addition, one can set a minimal number of hits. Transcripts not matching these criteria are then discarded. During the second step, the hits are then divided into unambiguous and ambiguous. The unambiguous hits are assigned first and used to distribute the ambiguous hits. The transcripts are then filtered again using the same criteria as before. The final expression value of a locus is then calculated as the sum of hits assigned to any of its transcripts.

Rcount-distribute takes a genome file (`.xml`) and an alignment file (`.bam`) as input. As a result, it writes a new `.bam` file with the mapping tag `XM:i:x` and a count table. All parameters are set while creating a new project and stored in the project file. Additionally, one can enable the use of strand information if (!) the library preparation protocol was strand specific (updated in March 2016: you can specify whether the reads are in sense (same) or antisense (opposite) orientation to the transcripts). This greatly helps to assign the ambiguous reads as overlapping genes are often in opposite orientation to each other. Note that in case of paired-end reads, only the first/forward read is used.

3.5 Reading and merging tables in R

Rcount-distribute creates for each sample a separate count table with multiple columns:

- `sumUnAmb`: number of unambiguously mapped hits.
- `sumAmb`: number of ambiguously mapped hits before distributing them accoring to the number of unambiguously mapped hits.
- `sumAllo`: number of ambiguously mapped hits after distributing them accoring to the number of unambiguously mapped hits.
- `sumDistUnAmb, sumDistAmb, sumDistAllo` are identical to `sumUnAmb, sumAmb, sumAllo`, but only within the first x bp from the 3' of the transcript. The value x can be specified in Rcount-distribute by enabling "consider 3' bias".
- `TH`: total number of hits (equals to `sumUnAmb+sumAllo`).

The provided R-Script offers a function to read in all table files from a given directory and merge them into a single table (see section 2.7). The function `f.read.Rcount` loads per default the column "TH". Other columns in the count tables can be loaded using the argument `toReturn` of the function `f.read.Rcount` (e.g. `f.read.Rcount(pathToWorkingDirectory, toReturn = 'sumUnAmb')`).

4 Test data

Rcount was tested with data from multiple organisms with different genome sizes and RNA-Seq libraries (table 3). Genome annotation data and genome indices were obtained as described before (see section 3.2) from tophat.cbcb.umd.edu/igenomes.shtml. Short reads were downloaded from www.ncbi.nlm.nih.gov/sra. The reformatted `.xml` genome annotation, as well as the resulting project files and count tables, can by found on github.com/MWSchmid/Rcount in the archives `test_data_annotations.zip` and `test_data_results.zip`. Reads were aligned with TopHat2 (version 2.0.11 [3]) allowing up to 10 alignments per read (-g 10). For increasing the speed of TopHat2, the coverage-search was turned off for all data. Allocation distance for weighting multireads was set to 50 for all samples. Run-time and memory usage were obtained on a 64 bit Kubuntu (12.04) running on an Intel® Core$^{\mathrm{TM}}$ i7 930@2.8 GHz with 24 Gb RAM. Input and output files were read and written from the same hard-drive (Samsung HD, 7'200 rpm). Memory usage of Rcount-multireads is mostly influenced by the number of reads with multiple alignments (e.g. high memory requirements for the mouse sample compared to the human sample). However, for Rcount-distribute, the size of the genome annotation has the largest impact on the memory usage (e.g. the xml file holding the genome annotation of the human genome is with 643 Mb almost double the size compared to the xml genome annotation file for the mouse genome, table 4).

Table 3: Data used to test Rcount.

organism	assembly	accession	# reads
A. thaliana	TAIR10	SRX275909[8]	52'181'949
B. taurus	Btau4.0	DRR001892[9]	69'040'753
C. elegans	WS220	SRR1015366[10]	34'542'067
C. familiaris	BROADD2	DRR001151	113'253'542
D. melanogaster	BDGP5.25	SRR629969[11]	31'267'571
G. gallus	WASHUC2	SRR1264638[12]	20'189'799
H. sapiens	GRCh37	DRR000897[13]	22'635'328
M. musculus	NCBIM37	DRR013118[14]	27'028'925
R. norvegicus	RGSC3.4	SRR1041766[15]	28'897'182

Table 4: Alignment statistics, approximate run-time and memory usage of Rcount using the test data given in Table 3. Note that the number of multireads only includes the ones with at least one alignment with a weight above zero.

organism	# reads aligned		Rcount-multireads		Rcount-distribute	
	unique	multiple	time	memory	time	memory
A. thaliana	41'098'964	2'231'636	12 min	1.0 Gb	18 min	1 Gb
B. taurus	34'913'558	3'538'481	15 min	1.1 Gb	18 min	928 Mb
C. elegans	25'492'284	1'889'021	10 min	729 Mb	14 min	1.0 Gb
C. familiaris	66'816'362	4'809'058	18 min	2.8 Gb	22 min	878 Mb
D. melanogaster	25'806'514	969'484	9 min	216 Mb	12 min	981 Mb
G. gallus	10'465'361	158'067	2 min	185 Mb	4 min	803 Mb
H. sapiens	16'056'845	2'169'498	5 min	1.0 Gb	13 min	3.3 Gb
M. musculus	9'630'768	7'898'165	8 min	3.2 Gb	8 min	1.7 Gb
R. norvegicus	21'729'100	1'288'736	10 min	410 Mb	12 min	1.2 Gb

5 Building from source

If the binaries are not working or you would like to implement a new feature, you can build them from the source code.

5.1 Linux

Compiling the programs using QtCreator is quite easy.

- download and unpack the archive `source.zip`
- install QtCreator (on the Ubuntu repository: `qtcreator`) - make sure to use Qt 5.x.x libraries.
- install zlib (on the Ubuntu repository: `zlib1g`, `zlib1g-dev`, `zlib1g-dev`), which is required by Rcount-multireads and Rcount-distribute
- start QtCreator and open each of the *.pro files
  - Rcount-format: `../source/p502_format_wizard/p502_format_wizard.pro`
  - Rcount-multireads: `../source/p502_process_multireads/p502_process_multireads.pro`
  - Rcount-distribute: `../source/p502dataAnalysisRNA/p502dataAnalysisRNA.pro`
- for Rcount-multireads and Rcount-distribute, copy the seqan folder into one of your general include paths (e.g. `sudo cp -r seqan /usr/local/include`) or add a line `INCLUDEPATH += <path containing seqan folder>` into the *.pro files. For Rcount-format, seqan is not required and this step can therefore be skipped.
- finally, in the QtCreater menu select `Build > Build Project ''my_project''`

5.2 Windows

There are no build instructions for Windows anymore. It got quite a bit more comlicated now - partly due to VisualStudio 2015. Aside compiling 64bit versions of the Qt libraries and zlib, one needs to modify some code of the seqan library (`lexicalCast2` and all cases of `std::min` and `std::max`). Furthermore one needs to replace `getopt` with `XGetopt` (www.codeproject.com/Articles/1940/XGetopt-A-Unix-compatible-getopt-for-MFC-and-Win32).

5.3 Mac

Currently, there are no build instructions available for the MacOS.

References

[1] Anderson S, Johnson C, Jones D, Conrad L, Gou X, et al. (2013) Transcriptomes of isolated Oryza sativa gametes characterized by deep sequencing: evidence for distinct sex-dependent chromatin and epigenetic states before fertilization. The Plant Journal 76: 729–741.

[2] Langmead B, Salzberg S (2012) Fast gapped-read alignment with Bowtie 2. Nature Methods 9: 357–359.

[3] Kim D, Pertea G, Trapnell C, Pimentel H, Kelley R, et al. (2013) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. Genome Biology 14: R36.

[4] Li H, Handsaker B, Wysoker A, T F, Ruan J, et al. (2009) The Sequence alignment/map (SAM) format and SAMtools. Bioinformatics 25: 2078-2079.

[5] Liao Y, Smyth G, Shi W (2013) The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Research 41: e108.

[6] Dobin A, Davis C, Schlesinger F, Drenkow J, Zaleski C, et al. (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29: 15-21.

[7] Chung D, Juan P, Li B, Sanalkumar R, Liang K, et al. (2011) Discovering Transcription Factor Binding Sites in Highly Repetitive Regions of Genomes with Multi-Read Analysis of ChIP-Seq Data. PLOS Computational Biology 7: e1002111.

[8] Loraine A, McCormick S, Estrada A, Patel K, Peng Q (2013) RNA-Seq or Arabidopsis Pollen Uncovers Novel Transcription and Alternative Splicing. Plant Physiology 162: 1092–1109.

[9] Takeo S, Kawahara-Miki R, Goto H, Cao F, Kimura K, et al. (2013) Age-associated changes in gene expression and developmental competence of bovine oocytes, and a possible countermeasure against age-associated events. Molecular Reproduction and Development 80: 508–521.

[10] Washburn M, Kakaradov B, Sundararaman B, Wheeler E, Hoon S, et al. (2014) The dsRBP and inactive editor ADR-1 utilizes dsRNA binding to regulate A-to-I RNA editing across the C. elegans transcriptome. Cell Reports 6: 599–607.

[11] Ramaswami G, Zhang R, Piskol R, Keegan L, Deng P, et al. (2013) Identifying RNA editing sites using RNA sequencing data alone. Nature Methods 10: 128–132.

[12] Ku Y, Renaud N, Veile R, Helms C, Voelker C, et al. (2014) The transcriptome of utricle hair cell regeneration in the avian inner ear. The Journal of Neuroscience 34: 3523–3535.

[13] Odawara J, Harada A, Yoshimi T, Maehara K, Tachibana T, et al. (2011) The classification of mRNA expression levels by the phosphorylation state of RNAPII CTD based on a combined genome-wide approach. BMC Genomics 12.

[14] Yukawa M, Akiyama T, Franke V, Mise N, Isagawa T, et al. (2014) Genome-Wide Analysis of the Chromatin Composition of Histone H2A and H3 Variants in Mouse Embryonic Stem Cells. PLOS ONE 9: e92689.

[15] Cortez D, Marin R, Toledo-Flores D, Froidevaux L, Liechti A, et al. (2014) Origins and functional evolution of Y chromosomes across mammals. Nature 508: 488–493.