circtools Documentation

Tobias Jakobi, Dieterich Lab

Oct 01, 2018

Contents

CHAPTER 1

Introduction

Circular RNAs (circRNAs) originate through back-splicing events from linear primary transcripts, are resistant to exonucleases, typically not polyadenylated, and have been shown to be highly specific for cell type and developmental stage. Although few circular RNA molecules have been shown to exhibit miRNA sponge function, for the vast majority of circRNAs however, their function is yet to be determined.

The prediction of circular RNAs is a multi-stage bioinformatics process starting with raw sequencing data and usually ending with a list of potential circRNA candidates which, depending on tissue and condition may contain hundreds to thousands of potential circRNAs. While there already exist a number of tools for the prediction process (e.g. DCC and circTest), publicly available downstream analysis tools are rare.

We developed **circtools**, a modular, Python3-based framework for circRNA-related tools that unifies several functionalities in single command line driven software. The command line follows the *circtools subcommand* standard that is employed in samtools or bedtools. Circtools includes modules for RBP enrichment screenings, circRNA primer design, as well as interfaces to the processing tools FUCHS and DCC; miRNA seed analysis and differential exon usage will we available in the upcoming release.

We intend to add more and more modules in the future in order to provide a comprehensive bioinformatics toolbox and also encourage users to contribute modules to circtools.

circtools is developed in the Dieterich Lab at the University Hospital Heidelberg.

CHAPTER 2

Table of contents

2.1 Installation

circtools is written in Python2 (>= 2.7, `detect` and `reconstruct` module) and Python3 (>=3.4, all other modules). The tool has a number of external dependencies, mostly standard bioinformatics tools and packages. The installation will, by default, try to install all required dependencies.

Installation is performed via *python3 setup.py install*. No sudo access is required if the installation is suffixed with `--user` which will install the package in a user-writeable folder. In this case, the binaries should be installed to `/home/$USER/.local/bin/` (for Debian-based systems).

2.1.1 Supported operating systems

`circtools` was developed and tested on Debian Jessie 8 64 Bit and Debian Stretch 9 64 bit. macOS support is currently (09/2018) being tested and is already available in the `mac-dev` branch of the github repository (however, the macOS functionality cannot be fully guaranteed yet).

2.1.2 Installation from PyPi (preferred)

The default installation will install everything needed to run circtools *except R, STAR, or Stringtie* (see below). If you like you may install circtools locally (first call) or globally (second call, SU required).

```
pip3 install circtools --user # does not require root access, installation to␣
↪local user directory
```

```
pip3 install circtools # will require root access and globally install circtools
```

2.1.3 Installation from GitHub

The GitHub installation will install the most recent version directly from the source repository. Use this method if you want the latest fixes and features.

```
git clone https://github.com/dieterich-lab/circtools.git
cd circtools
pip3 install . --verbose --user
```

2.1.4 Updating circtools

You may want to update the circtools package if new versions are published. Like for the initial installation there are two ways to update circtools:

```
pip3 install circtools --user --upgrade
```

```
cd /path/to/circtools/repo/
git pull
pip3 install . install --verbose --user --upgrade
```

2.1.5 Required dependencies

External tools

- bedtools [>= 2.27.1] required by the enrichment module
- R [>= 3.3] required by visualisation scripts and the primer design module
- STAR [>= 2.6.0] required by the `detect` and `reconstruct` module to map RNA-seq reads against a reference genome and detect back splice junctions
- Stringtie [>= 1.3.3b, optional] required by the `exon` module to carry out exon level analyses.

The installation procedure will automatically install two additional Python-based dependencies: DCC and FUCHS by temporarily cloning the repositories and installing both tools via *setuptools* to `/home/$USER/.local/bin/`. Both tools **require Python 2** in order to run.

The primer design module as well as the exon analysis and circRNA testing module require a working installation of R with BioConductor. All R packages required are automatically installed during the setup.

Important: The setup scripts assumes that the folder for R plugins is writeable (either in the user's home or the system folder).

Python packages

- **For circRNA detection**
  - HTSeq>=0.11.0
  - pysam>=0.13.0
  - numpy>=1.8.2
  - pandas>=0.18.1
- **For circRNA reconstruction**
  - HTSeq>=0.11.0
  - pysam>=0.13.0
  - numpy>=1.8.2
  - pathos>=0.2.1

- **For circRNA enrichment**
  - pybedtools>=0.7.10
  - statsmodels>=0.8.0
- **For circRNA primer design**
  - BioPython>=1.71

2.1.6 Detailed installation

Getting the source code

Step 1: Clone source code from GitHub:

```
git clone https://github.com/dieterich-lab/circtools.git
```

Installation

Step 2: Install circtools using the provided installation script. The `--user` flag installs circtools in your home folder, thus making sure you do not require any administrative rights during the installation:

```
cd circtools
pip3 install . install --verbose --user
```

R environment

Step 3: Setting up R environment. In order for the automatic installation of R packages to work we need to set the package directory to a user-writeable path. The setup automatically sets that path to `/home/$USER/.R/`.

Dependencies

Step 4: The setup script is designed to make sure that the environment is setup correctly to run circtools. The circtools setup will automatically install CircTest, primex, DCC and FUCHS.

Finishing up

Step 5: Adding installation folder to `$PATH`. In order for circtools to find all executables, the setup will add the folder `/home/$USER/.local/bin/` automatically to your `.bashrc` file

This closes the circtools installation. To verify that circtools has been correctly installed, try to call circtools for the first time:

```
$> circtools --help
usage: circtools [-V] <command> [<args>]
```

2.2 Detect module

The circRNA detection module of circtools is based on DCC, a Python 2.7-based tool built to detect and quantify circRNAs with high specificity. DCC works with the STAR (Dobin et al., 2013) `chimeric.out.junction` files which contains chimerically aligned reads including circRNA junction spanning reads.

DCC depends on `pysam`, `pandas`, `numpy`, and `HTSeq`. The installation process of circtools will normally automatically check for the dependencies, install or update missing Python packages and install the latest stable version of DCC.

2.2.1 Manual installation instructions

Cloning the source repository:

```
$ git clone https://github.com/dieterich-lab/DCC.git
$ cd DCC
$ python2 setup.py install --user
```

Verifying the installation:

```
$ DCC --version
```

If the Python installation binary path [e.g. */$HOME/.local/bin* for Debian] is not included the `$PATH`, it is also possible run DCC directly:

```
$ python2 <DCC directory>/scripts/DCC <Options>
# or even
$ python2 <DCC directory>/DCC/main.py <Options>
```

2.2.2 General usage

The detection of circRNAs from RNA-seq data through the detection module can be summarized in a few steps:

- Mapping of RNA-seq data from quality checked Fastq files. For paired-end (PE) data it is recommended to map the reads pairs jointly as well as separately because STAR does not output reads or read pairs that contain more than one chimeric junction.
- Preparation of input files required by the detection module. In summary, only a `samplesheet`, which specifies the locations for the `chimeric.out.junction` files is required (one relative or absolute path per line). [Command line flag: `@samplesheet`]
- A GTF formatted annotation of repetitive regions which is used to filter out circRNA candidates from repetitive regions is recommended. [Command line flag: `-R repeat_file.gtf`]
- For paired-end sequencing two files, e.g. `mate1` and `mate2` which contain the paths to the `chimeric.out.junction` files originating from the separate mate mapping step are required. [Command line flags: `-m1 mate1file` and `-m2 mate2file`]
- The locations of the BAM files can be specified via the -B flag. Otherwise the detection module tries to guess their location based on the supplied `chimeric.out.junction` paths. [Command line flag: `-B @bam_file_list`]
- The detection module requires the `SJ.out.tab` files generated by STAR. Circtools assumes that these are located in the same folder as the BAM files and also retain their `SJ.out.tab` name.
- Circtools can be used to process circular RNA detection and host gene expression detection either in a one-pass strategy or only one part of the analysis:
- Detection of circRNAs and host gene expression: `-D` and `-G` option have to be supplied
- Detection of circRNAs only: `-D`
- Detection of host gene expression only: `-G`

2.2.3 Step by step tutorial

In this tutorial, we use the data set from Jakobi et al. 2016 as an example. The data are paired-end, stranded RiboMinus RNAseq data from *Mus musculus*, consisting of samples of four ages (2, 3, 6, and 12 month) collected from the whole hearts. Data can be downloaded from the NCBI SRA (accession number SRP071584). While the circtools suite does not offer specific module for the initial data processing, this short tutorial should give the user an idea on how to get the sequencing data in shape for the main circtools pipeline.

Throughout this tutorial, we will employ Bash wrapper scripts that automate the analysis for multiple samples. While these scripts have been designed to be used with the SLURM workload manager, it is also possible to use them in conjunction with GNU parallel without SLURM.

Download of raw data files from the NCBI SRA

The raw data of the Jakobi et al. 2016 study was uploaded to the NCBI short read archive (SRA) and converted to the NCBI SRA format. Before we can start processing the data, the sra-toolkit needs to be installed. In order to simplify the download process, the wonderdump script is used.

```
# create main folder and raw reads folder
mkdir -p workflow/reads

cd workflow/reads

# change the default download directory of wonderdump to current directory
sed -i 's/SRA_DIR=~\/ncbi\/public\/sra/SRA_DIR=.\//g' wonderdump.sh

# get list of accession numbers to download
# also get a mapping file from SRA accession to original file name
wget https://data.dieterichlab.org/s/jakobi2016_sra_list/download -O jakobi2016_
↪sra_list.txt
wget https://data.dieterichlab.org/s/sra_mapping/download -O mapping.txt

# downloading and rewriting the files as gzipped .fastq files will take some time
# in the end, the process will generate a set of 16 files (8 samples x 2 pairs)

# start wonderdump with the accession list and download data (~29GB)
cat jakobi2016_sra_list.txt | xargs -n 1 echo ./wonderdump.sh --split-files --gzip␣
↪| bash

# rename files from SRA accessions to file names used throughout this tutorial

# for mate 1:
parallel --link ln -s {2}_1.fastq {1}1.fastq :::: mapping.txt :::: jakobi2016_sra_
↪list.txt

# for mate 2:
parallel --link ln -s {2}_2.fastq {1}2.fastq :::: mapping.txt :::: jakobi2016_sra_
↪list.txt
```

Data structure

Once the data is downloaded, the following data structure is assumed:

```
cd workflow/reads

ls -la

ALL_1654_M__R1.fastq.gz
ALL_1654_M__R2.fastq.gz
ALL_1654_N__R1.fastq.gz
ALL_1654_N__R2.fastq.gz
ALL_1654_O__R1.fastq.gz
ALL_1654_O__R2.fastq.gz
ALL_1654_P__R1.fastq.gz
ALL_1654_P__R2.fastq.gz
ALL_1654_Q__R1.fastq.gz
ALL_1654_Q__R2.fastq.gz
ALL_1654_R__R1.fastq.gz
```

(continues on next page)

(continued from previous page)

```
ALL_1654_R__R2.fastq.gz
ALL_1654_S__R1.fastq.gz
ALL_1654_S__R2.fastq.gz
ALL_1654_T__R1.fastq.gz
ALL_1654_T__R2.fastq.gz
```

Adapter removal and quality trimming

In a first step, remaining adapter sequences originating from the sequencing-process need to be removed together with potential low-quality bases. There are plenty of tools available to perform this task, in this tutorial we go with flexbar. The next steps assume that flexbar has been installed and in visible in the `$PATH` variable.

```
# download the wrapper scrips for flexbar
wget https://raw.githubusercontent.com/dieterich-lab/bioinfo-scripts/master/slurm_
↪flexbar_paired.sh
# add execute permission
chmod 755 slurm_flexbar_paired.sh
mkdir flexbar
cd reads
# we now execute flexbar on all of the sample while keeping all paired end sample␣
↪correctly mapped
parallel -j1 --xapply ../slurm_flexbar_paired.sh {1} {2} ../flexbar/ _R1 ::: *_R1.
↪fastq.gz ::: *_R2.fastq.gz
```

After flexbar finished processing, the folder `flexbar/` contains the trimmed, quality-filtered read sets.

Removal of rRNA with Bowtie2

Next, rRNA reads are removed. Normally, we are not interested in these reads, especially when performing circRNA analysis. Removing those reads will also slightly speed up subsequent steps due to the reduced computational load. In this tutorial, Bowtie2 is used in order to discard reads that map against rRNA loci. This tutorial assumes bowtie2 has already been installed an is callable with `$PATH`. A precompiled bowtie2 index of mouse rRNA loci can has been uploaded for this purpose. In brief, bowtie2 maps the reads against a "reference genome" of rRNA loci and only keeps reads, that do *not* align and therefore are rRNA-free.

```
# download wrapper for bowtie2
wget https://raw.githubusercontent.com/dieterich-lab/bioinfo-scripts/master/slurm_
↪bowtie2_rRNA_filter_paired.sh
chmod 755 slurm_bowtie2_rRNA_filter_paired.sh
mkdir rrna/
cd flexbar/
parallel -j1 --xapply ../slurm_bowtie2_rRNA_filter_paired.sh /path/to/extracted/
↪bowtie2/index/mus-musculus.rRNA {1} {2} ../rrna/ .1 ::: *_1.fastq.gz ::: *_2.
↪fastq.gz
```

After this step the `rrna/` folder contains adapter-free, rRNA-free reads ready for final mapping.

Mapping against the reference genome

In order to map the preprocessed reads against the reference genome and, ultimately detect possible cirRNAs the STAR read mapper is employed. STAR has shown to exhibit a good performance, is highly customizable and, most importantly is able to directly export chimeric reads that are the basis for the circRNA detection process. Again, we are using a wrapper script that simplifies the process of calling STAR for all samples. Like bowtie2, STAR requires an index in order to align reads. Since building this index requires a huge amount of RAM, a precomputed STAR index for the mouse ENSEMBL 90 build has been prepared for direct download (~22GB).

Essentially, the wrapper script for STAR performs the following tasks:

- Map both reads of each pair against the reference genome
- Map the unmapped reads of read 1 or read 2 again against the reference genome without the corresponding paired partner
- Several conversion and cleanup steps of the STAR output

```
# download wrapper for STAR
wget https://raw.githubusercontent.com/dieterich-lab/bioinfo-scripts/master/slurm_
↪circtools_detect_paired_mapping.sh
chmod 755 slurm_circtools_detect_paired_mapping.sh
mkdir star/
# obtain the annotation of the mouse genome for splice junctions
wget ftp://ftp.ensembl.org/pub/release-90/gtf/mus_musculus/Mus_musculus.GRCm38.90.
↪gtf.gz
gzip -d Mus_musculus.GRCm38.90.gtf.gz
cd rrna/
pd -j1 --xapply ../slurm_circtools_detect_paired_mapping.sh ../folder/to/star/
↪index/ {1} {2} ../star/ .1 ../Mus_musculus.GRCm38.90.gtf
```

Manual mapping process

Following a few notes on the manual mapping process:

Note: `--alignSJoverhangMin` and `--chimJunctionOverhangMin` should use the same values to make the circRNA expression and linear gene expression level comparable.

In a first step the paired-end data is mapped by using both mates. If the data is paired-end, an additional separate mate mapping is recommended (while not mandatory, this step will increase the sensitivity due to the the processing of small circular RNAs with only one chimeric junction point at each read mate). If the data is single-end, only this mapping step is required. In case of the Jakobi et al. 2016 data however, paired-end sequencing data is available.

Warning: Starting with version 2.6.0 of STAR, the chimeric output format changed. In order to be compliant with the circtools work flow the old output mode has to be selected via `--chimOutType Junctions SeparateSAMold`

```
$ mkdir Sample1
$ cd Sample1
$ STAR --runThreadN 10 \
       --genomeDir [genome] \
       --outSAMtype BAM SortedByCoordinate \
       --readFilesIn Sample1_1.fastq.gz  Sample1_2.fastq.gz \
       --readFilesCommand zcat \
       --outFileNamePrefix [sample prefix] \
       --outReadsUnmapped Fastx \
       --outSJfilterOverhangMin 15 15 15 15 \
       --alignSJoverhangMin 15 \
       --alignSJDBoverhangMin 15 \
       --outFilterMultimapNmax 20 \
       --outFilterScoreMin 1 \
       --outFilterMatchNmin 1 \
       --outFilterMismatchNmax 2 \
       --chimSegmentMin 15 \
       --chimScoreMin 15 \
       --chimScoreSeparation 10 \
       --chimJunctionOverhangMin 15 \
```

- *This step may be skipped when single-end data is used.* Separate per-mate mapping. The naming of mate1 and mate2 has to be consistent with the order of the reads from the joint mapping performed above. In this case, SamplePairedRead_1.fastq.gz is the first mate since it was referenced at the first position in the joint mapping.

```
# Create a directory for mate1
$ mkdir mate1
$ cd mate1
$ STAR --runThreadN 10 \
       --genomeDir [genome] \
       --outSAMtype None \
       --readFilesIn Sample1_1.fastq.gz \
       --readFilesCommand zcat \
       --outFileNamePrefix [sample prefix] \
       --outReadsUnmapped Fastx \
       --outSJfilterOverhangMin 15 15 15 15 \
       --alignSJoverhangMin 15 \
       --alignSJDBoverhangMin 15 \
       --seedSearchStartLmax 30 \
       --outFilterMultimapNmax 20 \
       --outFilterScoreMin 1 \
       --outFilterMatchNmin 1 \
       --outFilterMismatchNmax 2 \
       --chimSegmentMin 15 \
       --chimScoreMin 15 \
       --chimScoreSeparation 10 \
       --chimJunctionOverhangMin 15 \
```

- The process is repeated for the second mate:

```
# Create a directory for mate2
$ mkdir mate2
$ cd mate2
$ STAR --runThreadN 10 \
       --genomeDir [genome] \
       --outSAMtype None \
       --readFilesIn Sample1_2.fastq.gz \
       --readFilesCommand zcat \
       --outFileNamePrefix [sample prefix] \
       --outReadsUnmapped Fastx \
       --outSJfilterOverhangMin 15 15 15 15 \
       --alignSJoverhangMin 15 \
       --alignSJDBoverhangMin 15 \
       --seedSearchStartLmax 30 \
       --outFilterMultimapNmax 20 \
       --outFilterScoreMin 1 \
       --outFilterMatchNmin 1 \
       --outFilterMismatchNmax 2 \
       --chimSegmentMin 15 \
       --chimScoreMin 15 \
       --chimScoreSeparation 10 \
       --chimJunctionOverhangMin 15 \
```

Detection of circular RNAs from `chimeric.out.junction` files with circtools

Acquiring suitable GTF files for repeat masking

- It is strongly recommended to specify a repetitive region file in GTF format for filtering.
- A suitable file can for example be obtained through the UCSC table browser . After choosing the genome, a group like **Repeats** or **Variation and Repeats** has to be selected. For the track, we recommend to choose

RepeatMasker together with **Simple Repeats** and combine the results afterwards.

- **Note**: the output file needs to comply with the GTF format specification. Additionally it may be the case that the names of chromosomes from different databases differ, e.g. **1** for chromosome 1 from ENSEMBL compared to **chr1** for chromosome 1 from UCSC. Since the chromosome names are important for the correct functionality of circtools a sample command for converting the identifiers may be:
- A sample repeat file for the mouse mm10 genome can also be downloaded

```
# Example to convert UCSC identifiers to to ENSEMBL standard

$ sed -i 's/^chr//g' your_repeat_file.gtf
```

Preparation of input files for circRNA detection step

We first create a new folder for the circtools detection step and populate that folder with links to the required files produced by STAR.

```
# create new folder
mkdir -p circtools/01_detect/
cd star/

# link aligned reads (bam files) and indexing files for the aligned reads (.bai)
parallel --plus ln -s `pwd`/{}/Aligned.noS.bam /scratch/tjakobi/circtools_workflow/
↪workflow/circtools/01_detect/{}.bam ::: ALL_1654_*
parallel --plus ln -s `pwd`/{}/Aligned.noS.bam.bai /scratch/tjakobi/circtools_
↪workflow/workflow/circtools/01_detect/{}.bam.bai ::: ALL_1654_*

# link chimeric junction files of the main mapping
parallel --plus ln -s `pwd`/{}/Chimeric.out.junction /scratch/tjakobi/circtools_
↪workflow/workflow/circtools/01_detect/{}.Chimeric.out.junction ::: ALL_1654_*

# link chimeric junction files of the single mappings
parallel --plus ln -s `pwd`/{}/mate1/Chimeric.out.junction /scratch/tjakobi/
↪circtools_workflow/workflow/circtools/01_detect/{}.mate1.Chimeric.out.junction␣
↪::: ALL_1654_*
parallel --plus ln -s `pwd`/{}/mate2/Chimeric.out.junction /scratch/tjakobi/
↪circtools_workflow/workflow/circtools/01_detect/{}.mate2.Chimeric.out.junction␣
↪::: ALL_1654_*

cd ..
cd circtools/01_detect/

# create input files for detection step
ls | grep bam$ | grep -v mate > bam_files.txt
ls | grep junction$ | grep  mate1 > mate1
ls | grep junction$ | grep  mate2 > mate2
ls | grep junction$ | grep -v mate > samplesheet
```

Additionally the newly created files, a reference genome in Fasta format as well as an appropriate annotation containing repetitive regions should be provided:

```
# step one: obtain reference genome
wget ftp://ftp.ensembl.org/pub/release-90/fasta/mus_musculus/dna/Mus_musculus.
↪GRCm38.dna.primary_assembly.fa.gz
gzip -d Mus_musculus.GRCm38.dna.primary_assembly.fa.gz

# step two: repeat masker file for the genome build:
wget https://data.dieterichlab.org/s/mouse_repeats/download -O GRCm38_90_
↪repeatmasker.gtf.bz2
bunzip GRCm38_90_repeatmasker.gtf.bz2
```

2.2.4 Running circtools circRNA detection

After performing all preparation steps the detection module can now be started:

```
# Run circtools detection to detect circRNAs, using the Jakobi 2016 data set

$ circtools detect @samplesheet \ # @ is generally used to specify a file name
      -mt1 @mate1 \ # mate1 file containing the mate1 independently mapped␣
↪chimeric.junction.out files
      -mt2 @mate2 \ # mate2 file containing the mate1 independently mapped␣
↪chimeric.junction.out files
      -D \ # run in circular RNA detection mode
      -R GRCm38_90_repeatmasker.gtf \ # regions in this GTF file are masked from␣
↪circular RNA detection
      -an Mus_musculus.GRCm38.90.gtf \ # annotation is used to assign gene names␣
↪to known transcripts
      -Pi \ # run in paired independent mode, i.e. use -mt1 and -mt2
      -F \ # filter the circular RNA candidate regions
      -M \ # filter out candidates from mitochondrial chromosomes
      -Nr 5 6 \ minimum count in one replicate [1] and number of replicates the␣
↪candidate has to be detected in [2]
      -fg \ # candidates are not allowed to span more than one gene
      -G \ # also run host gene expression
      -A Mus_musculus.GRCm38.dna.primary_assembly.fa \ # name of the fasta genome␣
↪reference file; must be indexed, i.e. a .fai file must be present
```

Note: By default, circtools assumes that the data is stranded. For non-stranded data the `-N` flag should be used

Note: Although not mandatory, we strongly recommend to the `-F` filtering step

2.2.5 Output files

The output of circtools detect consists of the following four files: CircRNACount, CircCoordinates, LinearCount and CircSkipJunctions.

- **CircRNACount:** a table containing read counts for circRNAs detected. First three columns are chr, circRNA start, circRNA end. From fourth column on are the circRNA read counts, one sample per column, shown in the order given in your samplesheet.
- **CircCoordinates:** circular RNA annotations in BED format. The columns are chr, start, end, genename, junctiontype (based on STAR; 0: non-canonical; 1: GT/AG, 2: CT/AC, 3: GC/AG, 4: CT/GC, 5: AT/AC, 6: GT/AT), strand, circRNA region (startregion-endregion), overall regions (the genomic features circRNA coordinates interval covers).
- **LinearCount:** host gene expression count table, same setup with CircRNACount file.
- **CircSkipJunctions:** circSkip junctions. The first three columns are the same as in LinearCount/CircRNACount, the following columns represent the circSkip junctions found for each sample. circSkip junctions are given as chr:start-end:count, e.g. chr1:1787-6949:10. It is possible that for one circRNA multiple circSkip junctions are found due to the fact the the circular RNA may arise from different isoforms. In this case, multiple circSkip junctions are delimited with semicolon. A 0 implies that no circSkip junctions have been found for this circRNA.

2.2.6 Feedback

- In case of any problems or feature requests please do not hesitate to open an issue on GitHub: Create an issue

- Please make sure to run circtools detect with the `-k` flag when reporting an error to keep temporary files important for debugging purposes
- Please also always paste or include the log file

2.3 Reconstruction module

The circtools reconstruct module is based on FUCHS (FUll circular RNA CHaracterization from RNA-Seq), a Python program designed to fully characterize circular RNAs. It uses a list of circular RNAs and reads spanning the back-splice junction as well as BAM files containing the mappings of all reads (alternatively of all chimeric reads).

The reads from each circle are extracted by the reconstruction module and saved in an individual BAM files. Based on these BAM files, circtools will detect alternative splicing within the same circle boundaries, summarize different circular isoforms from the same host-gene, and generate coverage plots for each circRNA. It will also cluster circles based on their coverage profile. These results can be used to identify potential false positive circRNAs.

2.3.1 Manual installation instructions

Required tools and packages

FUCHS dependes on **bedtools (>= 2.27.0)**, **samtools (>= 1.3.1)**, **Python (> 2.7; pysam>=0.13.0, pybedtools>=0.7.8, numpy>=1.11.2, pathos>=0.2.1)**, and **R(>= 3.2.0; amap, Hmisc, gplots)**. All Python an R dependencies will be installed automatically when installing FUCHS. Please make sure to have the correct versions of bedtools and samtools in your `$PATH`.

Installation of FUCHS

```
$ git clone https://github.com/dieterich-lab/FUCHS.git
$ cd FUCHS

$ python2 setup.py install --user

# This will install a FUCHS binary in $HOME/.local/bin/
# make sure this folder is in your $PATH

# Check the installation:

$ FUCHS --help
```

2.3.2 General usage

In order to characterize circRNAs from RNA-seq data the following steps are necessary:

1. Mapping of RNA-seq data from quality checked FASTQ files with STAR (BWA, TopHat-Fusion in preparation)
2. Detection circRNAs using circtools detect (CIRI, CIRCfinder or CIRCexplorer in preparation)
3. Run circtools reconstruct

Mapping of RNA-Seq data and detection of circRNAs

Please see the documentation of circtools detect for instructions how to pre-process the data.

Use a wrapper script for the circtools reconstruct call

As for other parts of the circtools pipeline, a wrapper Bash script has been developed that does all necessary preprocessing after the initial detection step and directly calls the reconstruction module afterwards. We continue by using the Jakobi et al. 2016 data set that also has been used as an example for the circtools detect module.

```
# download the wrapper scrips for the reconstruct module
wget https://raw.githubusercontent.com/dieterich-lab/bioinfo-scripts/master/slurm_
↪circtools_reconstruct.sh
# add execute permission
chmod 755 slurm_circtools_reconstruct.sh

# create output directory
mkdir 03_reconstruct

# download exon annotations required by the reconstruct module
wget https://data.dieterichlab.org/s/mouse_exons_bed/download -O mm10.ensembl.
↪exons.bed.bz2
bunzip mm10.ensembl.exons.bed.bz2

# circtools reconstruct is independently run on all samples:
parallel -j1  slurm_circtools_reconstruct.sh {} 03_reconstruct/ mm10.ensembl.exons.
↪bed 01_detect/ ::: ALL_1654_M ALL_1654_N ALL_1654_O ALL_1654_P ALL_1654_Q ALL_
↪1654_R ALL_1654_S ALL_1654_T
```

Manually running the reconstruction module

The above wrapper scripts handles all preprocessing and conversion steps. However, advanced users may want to start the module directly. `circtools reconstruct` starts the pipeline which will extract reads, check mate status, detect alternative splicing events, classify different isoforms, coverage profiles, and cluster circRNAs based on coverage profiles. Below a sample call for the reconstruct module in single sample mode using circtools detect input data:

```
# using STAR/circtools detect Input
$ circtools reconstruct -r 2 -q 2 -p ensembl -e 2 -T ~/tmp
        -D CircRNACount
        -J sample/Chimeric.out.junction
        -F sample.1/Chimeric.out.junction
        -R sample.2/Chimeric.out.junction.fixed
        -B merged_sample.sorted.bam
        -A [annotation].bed
        -N sample

# if BWA/CIRI was used, use -C to specify the circIDS list (omit -D, -J, -F and -R)
# For details on the parameters please refer to the help page:
$ circtools reconstruct --help
```

Optional reconstruct module

The additional module `denovo_circle_structure_parallel` can be employed to obtain a more refined circle reconstruction based on intron signals. The circRNA-separated bamfiles (step 2) are the only input required for the module. If an annotation file is supplied, unsupported exons will be reported with a score of 0, if no annotation file is supplied, unsupported exons will not be reported.

```
$ denovo_circle_structure_parallel -c 18 -A [annotatation].bed -I output/folder -N␣
↪sample
```

(continues on next page)

(continued from previous page)

```
# output/folder corresponds to the output directory of the circtools reconstruct␣
↪pipeline
# sample corresponds to your sample name, just as specified for the pipeline
```

2.3.3 Required input data

circRNA IDs

CircRNA data data can be provided via a generic table with the structure found below:

circID	read1,read2,read3
1:3740233|3746181	MISEQ:136:000000000-ACBC6:1:2107:10994:20458,MISEQ:136:000000000-ACBC6:1:1116:13529:8356
1:8495063|8614686	MISEQ:136:000000000-ACBC6:1:2118:9328:9926

The first column contains the circleRNA ID formated as folllowed **chr:start|end**. The second column is a comma separated list of read names spanning the back-splice junction.

BAM input files

Alignment files produced by any suitable read mapping tool. The files *have to* contain all chimerically mapped reads and *may* also contain linearly mapped reads.

BED annotation file

A BED file in BED6 format. The name should contain a gene name or gene ID and the exon_number. You can specify how the name should be processed using -p (platform), -s (character used to separate name and exon number) and -e (exon_index).

Chr	Start	End	Name	Score	Strand
1	67092175	67093604	NR_075077_exon_0_0_chr1_67092176_r	0	-
1	67096251	67096321	NR_075077_exon_1_0_chr1_67096252_r	0	-
1	67103237	67103382	NR_075077_exon_2_0_chr1_67103238_r	0	-

2.3.4 Output produced by circtools reconstruct

*.alternative_splicing.txt

This file summarizes the relationship of different circRNAs derived from the same host-gene. A sample file structure given below:

Transcript	circles	same_start	same_end	overlapping	within
NM_016287	20749723-20773610	.	.	.	.
NM_005097	35358925-35361789,1:35381259-35389082,1:35381259-35390098	1:35381259-35389082|1:35381259-35390098,	.	.	.
NM_001291240	236803428-236838599,1:236806144-236816543	.	.	.	1:236803428-236838599|1:236806144-236816543,

Description of the data columns:

- *Transcript*: Transcript name as defined by the bed-annotation file
- *circles*: Comma-separated list of circRNA ids derived from this transcript
- *same_start*: Comma-separated list of circRNA pairs separated by |. Pairs in this column share the same start coordinates. A "." indicates that there are no circle pairs that share the same start coordinates.
- *same_end*: Same as *same_start*, only now, circle pairs share the same end coordinates.
- *overlapping*: Comma-separated list of circRNA pairs separated by |. Pairs in this column share neither start nor end coordinates, but their relation is such that: start.x < start.y && end.x < end.y && start.y < end.x
- *within*: Same as *overlapping*, but circRNA pairs have the following relation: start.x < start.y && end.x > end.y

*.exon_counts.bed

These files are BED formatted and describe the exon-structure. The files can be loaded into any genome browser. Each line corresponds to a circRNA.

Chr	Circle Start	Circle End	Transcript	Num of Reads	Strand	Start	End	Color	Num of Exon	Exon Lengths	Relative Exon Starts
chr1	35358925	35361789	NM_005095		+	35358925	35361789	0,255,0		521,61,1700	0,2269,2694
chr1	20749723	20773610	NM_016287		-	20749723	20773610	0,255,0		159,90,1430,1	0,1943,21207,23728

Description of the data columns:

- *Chr*: Chromosome of circRNA
- *Circle Start*: The 5' site of the chimeric junction. This is relative to the reference strand, i.e. start < end! The location is 1-index based
- *Cirlce End*: The 3' site of the chimeric junction. This is relative to the reference strand, i.e. start < end! The location is 0-index based
- *Transcript*: Transcript name as defined by the bed-annotation file
- *Num of Reads* : Number of reads supporting this chimeric junction, in other words, reads that are chimerically mapped to this junction
- *Strand*: Strand of the host-gene
- *Start*: Copied *Circle Start* to stay conform with BED12 format
- *End*: Copied *Circle End* to stay conform with BED12 format
- *Color*: pre defined color the exons will show up in the genome viewer (0,255,0 -> green)
- *Num of Exon*: Number of exons in this circRNA consists of
- *Exon Lengths*: Comma-separated list of the length of each exon
- *Relative Exon Starts*: Comma-separated list of the relative starting positions of the exons within the circle boundaries.

*.exon_counts.txt

This file contains similar information as the previous file, just more detailed information on the exons. Each line corresponds to one exon.

sample	circle_id	transcript_id	other_ids	exon_id	chr	start	end	strand	exon_length	unique_reads	fragments	number+	number-
hek293	1:35358925-35361789	NM_005095	NM_005095	2	1	35358925	35359446	+	522	9	9	4	5
hek293	1:35358925-35361789	NM_005095	NM_005095	3	1	35361195	35361256	+	62	3	3	1	2
hek293	1:35358925-35361789	NM_005095	NM_005095	4	1	35361618	35361789	+	171	9	9	4	5
hek293	1:20749723-20773610	NM_016287	NM_016287	2	1	20749720	20749882	-	160	4	4	4	0
hek293	1:20749723-20773610	NM_016287	NM_016287	4	1	20757165	20757256	-	91	1	1	1	0
hek293	1:20749723-20773610	NM_016287	NM_016287	0	0	0	0	0	0	0	0	0	0
hek293	1:20749723-20773610	NM_016287	NM_016287	0	0	0	0	0	0	0	0	0	0
hek293	1:20749723-20773610	NM_016287	NM_016287	1	1	20770929	20771073	-	144	1	1	1	0
hek293	1:20749723-20773610	NM_016287	NM_016287	1	1	20773450	20773610	-	160	4	4	4	0

Description of the data columns:

- *sample*: Sample name as specified by the user. This is useful if the user wants to merge files from different samples
- *circle_id*: circRNA-ID. The circleID is formatted to be copy and pasted to a genome browser for easy access
- *transcript_id*: Transcript name as defined by the bed-annotation file. This is the best fitting transcript. i.e. the splicing variants that contains the most exons that are actually covered
- *other_ids*: Alternative Transcript names that are either just as fitting, or contain more or less exons as supported by reads
- *exon_id*: Exon number relative to the host-gene of the circularized exon. One circle may have more than one exon. These will be listed as consecutive lines
- *chr*: Chromosome the circRNA is located on
- *start*: 5' start of the exon, relative to the reference strand, 0-based
- *end*: 3' end of the exon, relative to the reference start, 0-based
- *strand*: Strand of the host-gene
- *exon_length*: Length of the current exon
- *unique_reads*: Number of unique reads associated with the chimeric junction. When the data is paired end, then both ends are considered as separate reads.
- *fragments*: Number of broken fragments aligning to the circle
- *number+*: Number of reads spanning the chimeric junction on the forward strand
- *number-*: Number of reads spanning the chimeric junction on the reverse strand (if reads are only from one strand, this may indicate that there is a sequencing bias)

*.mate_status.txt

This output file contains the results of analyzing the amount of how often each fragment spans a chimeric junction. A fragment can either span the chimeric junction once (single), only one end spans the junction, twice (double) both ends span the chimeric junction, or more than twice (undefined).

circle_id	transcript_ids	num_reads	min_length	max_length	single	double	undefined
1_20749723_20773610	NM_016287	4	790	790	4	0	0
1_35358925_35361789	NM_005095	9	754	754	9	0	0

Description of the data columns:

- *circle_id*: The circRNA ID in the form *chr_start_stop*
- *transcript_ids*: Names of the corresponding annotated transcript IDs
- *num_reads*: Total number of reads for this circRNA
- *min_length* Minimal length of exons intersecting the circRNA
- *max_length*: Maximal length of exons intersecting the circRNA (if only one exon same as *min_length*)
- *single*: Number of single break points for this circRNA
- *double*: Number of double break points for this circRNA
- *undefined*: Number of undefined break points for this circRNA

*.skipped_exons.bed

Chr	Circle-Start	Circle-End	Transcript	Ratio	Strand	Intron-Start	Intron-End	Color	NumExon	Intron-Length	RelativeStart
chr5	178885614	178931326	NM_030613	60.0	.	178913072	178931236	255,0,0	3	1,146,1	0,30950,45711
chr6	161034259	161049978	NM_001291958	0.8	.	161049332	161049852	255,0,0	3	1,520,1	0,15073,15719

Description of the data columns:

- *Chr*: Chromosome of circRNA
- *Circle-Start*: The 5' site of the chimeric junction. This is relative to the reference strand, i.e. start < end! The location is 1-index based
- *Cirlce-End*: The 3' site of the chimeric junction. This is relative to the reference strand, i.e. start < end! The location is 0-index based
- *Transcript*: Transcript name as defined by the BED annotation file
- *Ratio*: Ratio of reads of this skipped exon
- *Strand*: Strand of the host-gene
- *Intron-Start*: The 5' site of intron. This is relative to the reference strand, i.e. start < end! The location is 1-index based
- *Intron-End*: The 3' site of the intron. This is relative to the reference strand, i.e. start < end! The location is 0-index based
- *Color*: pre defined color the exons will show up in the genome viewer (0,255,0 -> green)
- *Num of Exon*: Number of exons in this circRNA consists of
- *IntronLengths*: Comma-separated list of the length of each intron
- *RelativeStart*: Comma-separated list of the relative starting positions of the introns within the circle boundaries.

*.skipped_exons.txt

circle_id	transcript_id	skipped_exon	intron	read_names	splice_reads	exon_reads
5_178885614	NM_178913328	178916564	set([('5', 178916710, 178913072, 178931236)])	MISEQ:136:000000000-ACBC6:1:2103:10044:24618,MISEQ:136:000000000-ACBC6:1:2115:19571:6931,MISEQ:136:000000000-ACBC6:1:1119:25537:8644	3	5
6_161034259	NM_161049332	161049852	set([('6', 161049852, 161049332, 161049852)])	MISEQ:136:000000000-ACBC6:1:1113:25288:9067,MISEQ:136:000000000-ACBC6:1:2116:11815:3530	2	5

Description of the data columns:

- *Chr*: Chromosome of circRNA
- *Transcript_id*: Transcript name as defined by the BED annotation file
- *Skipped_exon*: Coordinates of the skipped exon
- *Intron*: Set of introns
- *read_names*: Unique read names identifying this skipped exon
- *splice_reads*: Number of reads supporting the splice site
- *exon_reads*: Number of reads supporting the exon

*.sample_name.exon_chain_6.bed

Chr	Exon-Start	Exon-End	ID	Ratio	Strand
11	33286413	33286525	11:33286413-33287511|0|0	5	.
11	33287338	33287511	11:33286413-33287511|1|0	9	.

Description of the data columns:

- *Chr*: Chromosome of circRNA
- *Exon-Start*: The 5' site of the chimeric junction. This is relative to the reference strand, i.e. start < end! The location is 1-index based
- *Exon-End*: The 3' site of the chimeric junction. This is relative to the reference strand, i.e. start < end! The location is 0-index based
- *Name*: CircRNA ID, number of exon, coverage
- *Ratio*: Coverage ratio
- *Strand*: Strand not reported, always "."

*.sample_name.exon_chain_12.bed

Chr	Circle-Start	Circle-End	ID	#reads	Strand	Circle-Start	Circle-End	Color	#Exons	Exon lengths	Exon starts
11	33286413	33287511	11:33286413-33287511|0|0.446265938069	7	.	33286413	33287511	255,0,0	2	112,173	0,925
10	68959806	68960249	10:68959806-68960249|0|0.984198645598	5	.	68959806	68960249	255,0,0	2	146,290	0,153

Description of the data columns:

- *Chr*: Chromosome of circRNA
- *Circle-Start*: The 5' site of the chimeric junction. This is relative to the reference strand, i.e. start < end! The location is 1-index based
- *Cirlce-End*: The 3' site of the chimeric junction. This is relative to the reference strand, i.e. start < end! The location is 0-index based
- *ID*: CircRNA ID, running number, coverage
- *#reads*: Number of reads covering the circRNA
- *Strand*: Strand (always ".")
- *Circle-Start*: See above
- *Circle-End*: See above
- *Color*: pre defined color the exons will show up in the genome viewer (0,255,0 -> green)
- *Num of Exon*: Number of exons in this circRNA consists of
- *Exon lengths*: Comma-separated list of the length of each exon
- *Exon Starts*: Comma-separated list of the relative starting positions of the exon within the circle boundaries.

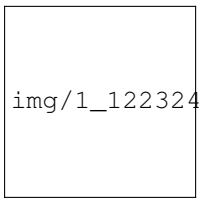
sample [folder]

- 1_35358925_35361789_9reads.sorted.bam
- 1_35358925_35361789_9reads.sorted.bam.bai
- 1_20749723_20773610_4reads.sorted.bam
- 1_20749723_20773610_4reads.sorted.bam.bai

*.coverage_pictures/ [folder]

Using R, circtools will generate a graphical representation of each circle's coverage profile, preserving the exon information as coloured segments. The smoothed profiles are saved as PNGs in a separate folder for easy examination by eye.

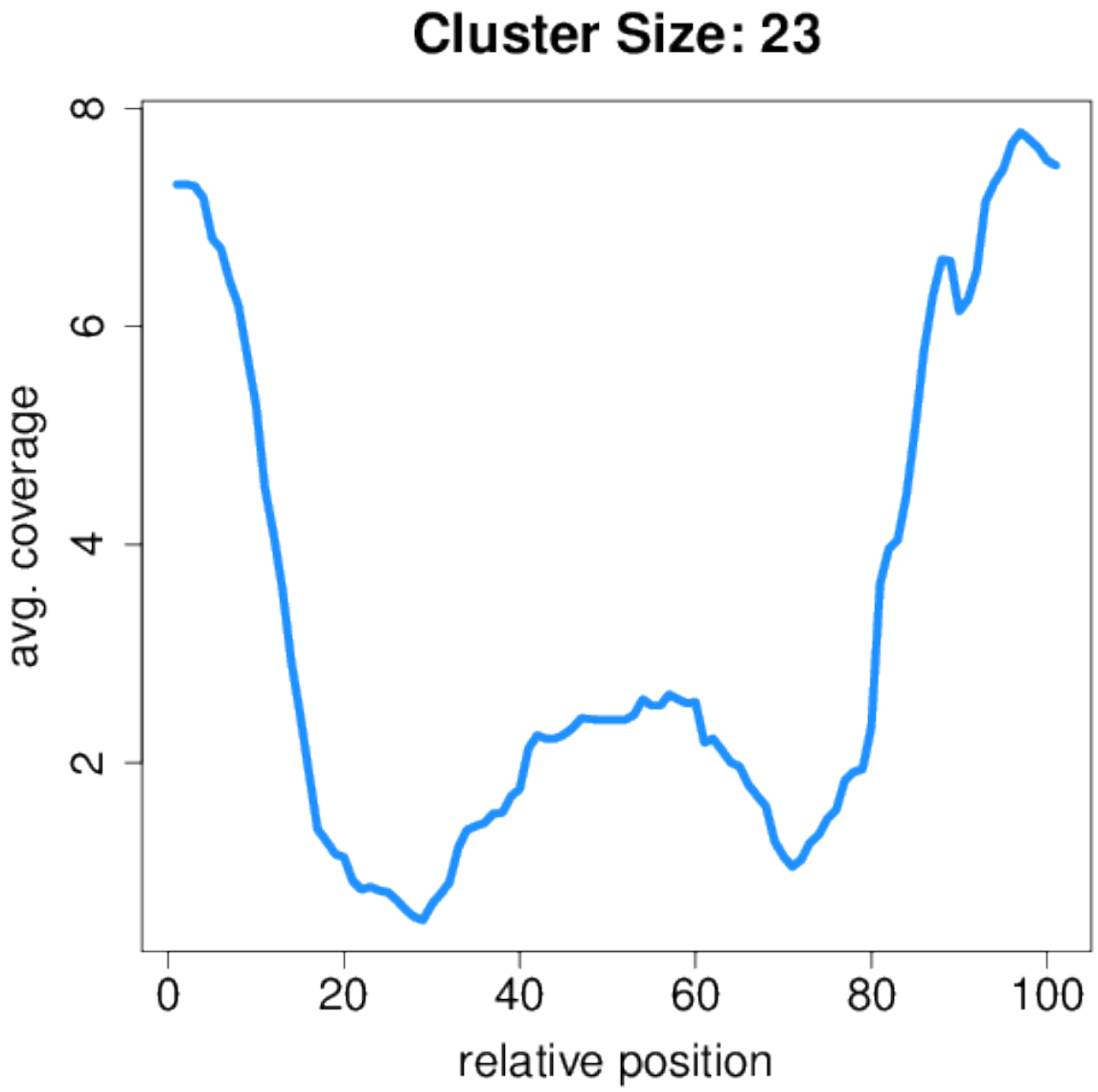
Sample circRNA coverage plot



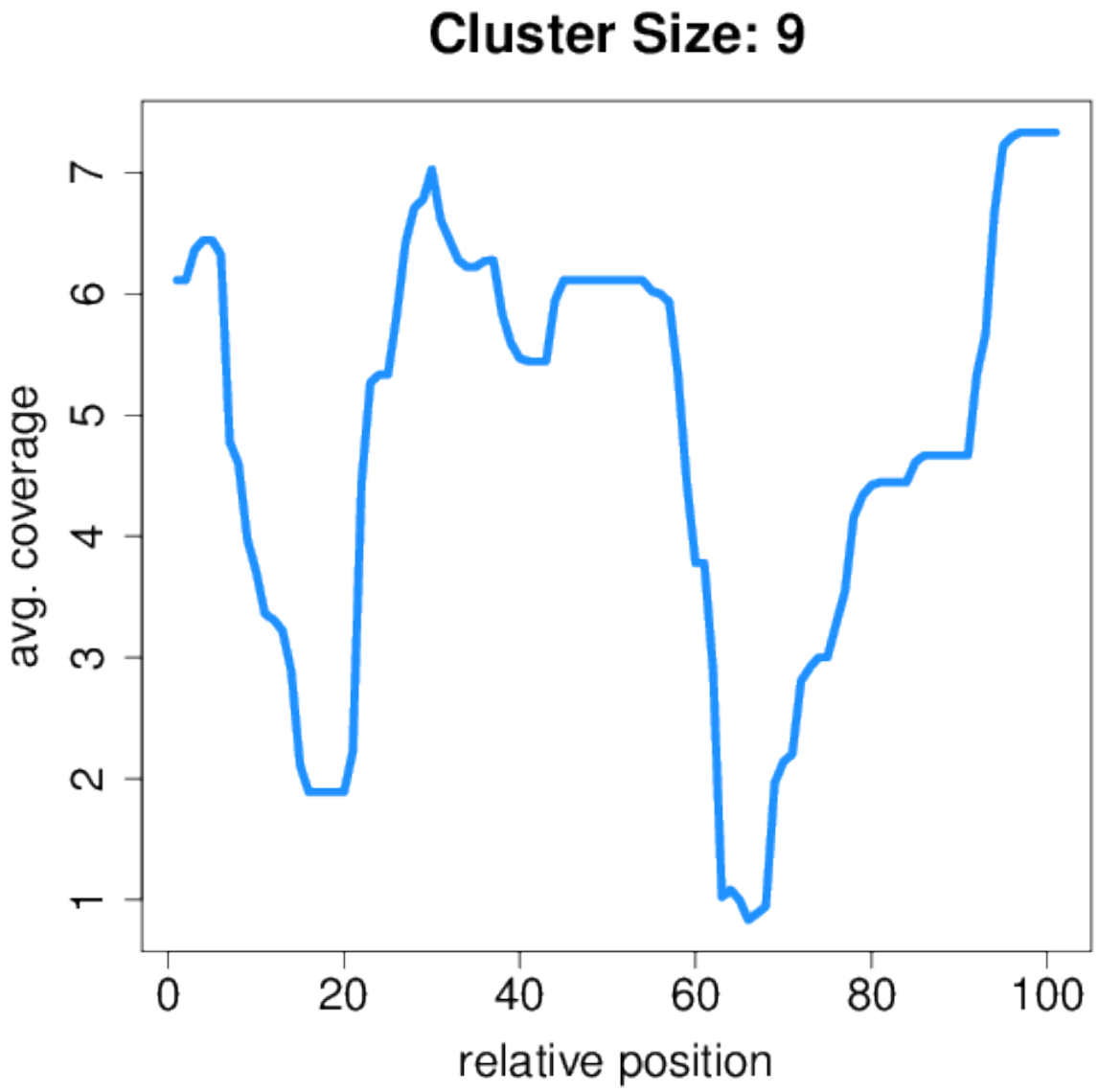
*.coverage_profiles/ [folder]

Circtools will accumulate all coverage profiles, normalize the profiles by circle length and cluster the circles based on their coverage profiles. The clustering is performed on all circles. Additionally, to avoid that the clustering will only group the circles based on their length, a group-wise clustering is performed. Here the circles are separated based on their length into small (<500 BP), medium (500–1,000 BP), and long (1,000 BP) circles. Based on correlation a K-means clustering is performed using the R package amap.

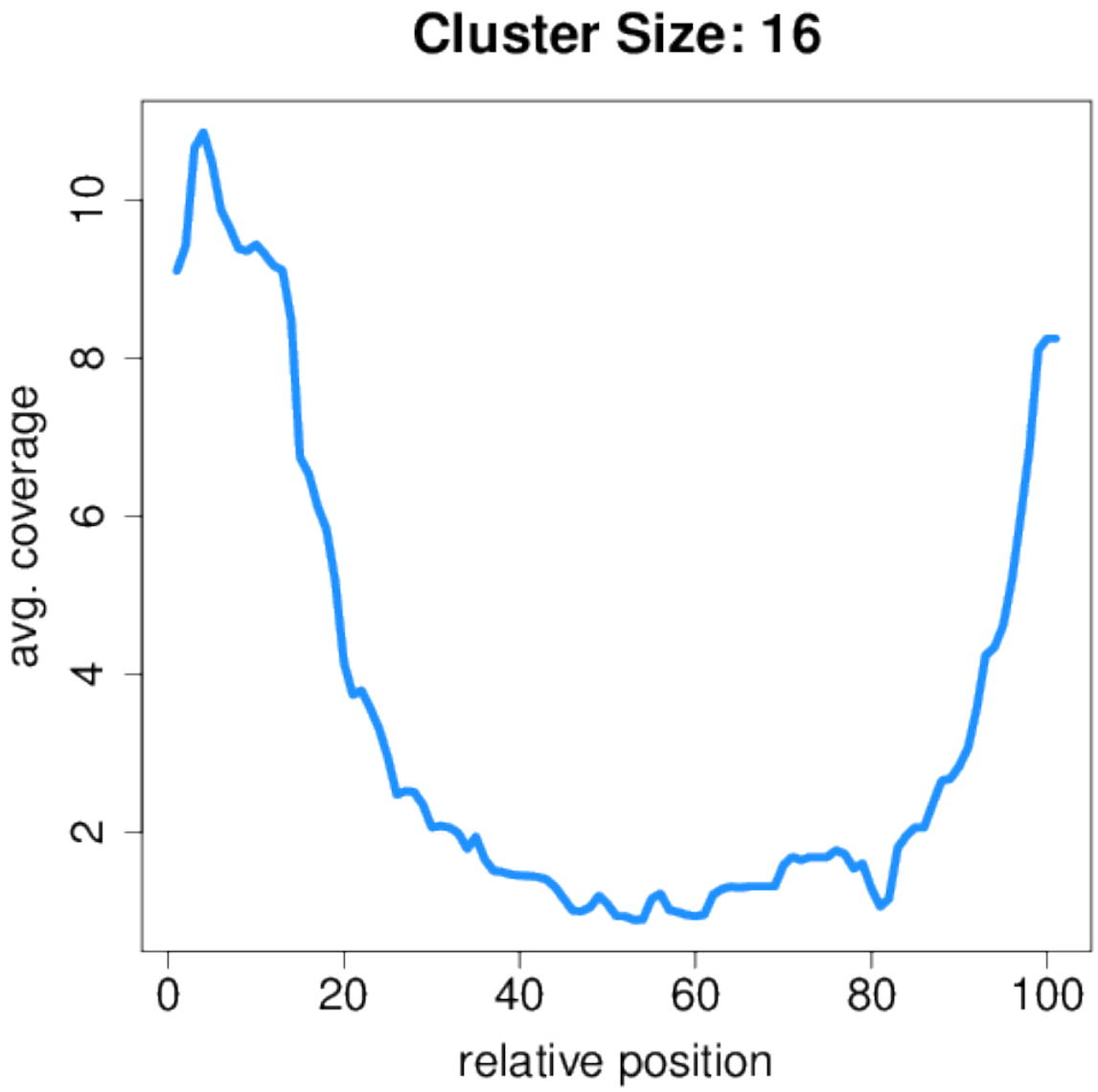
All circles


Cluster Size: 23
avg. coverage
2
4
6
8
0
20
40
60
80
100
relative position

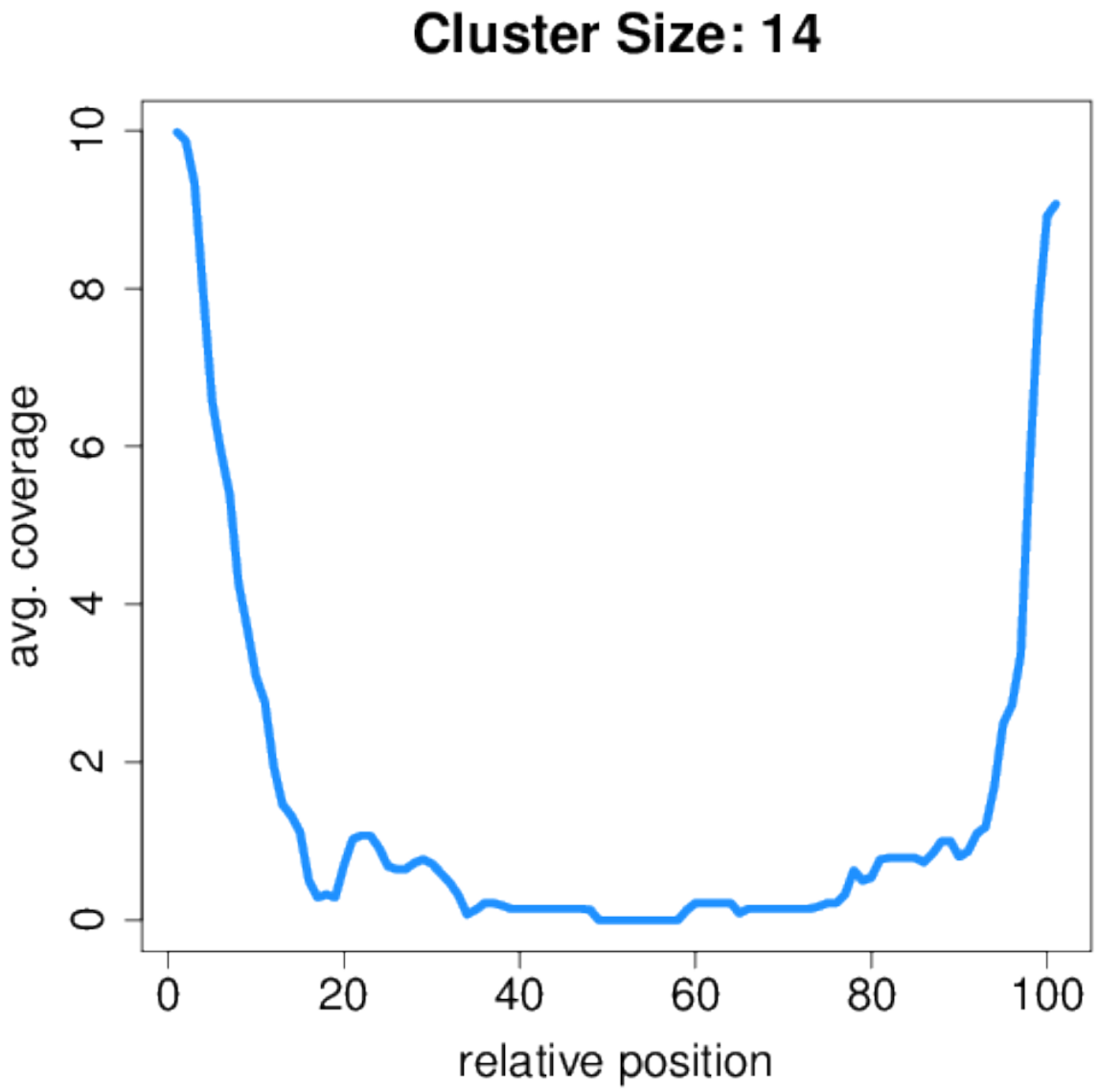
Short circles



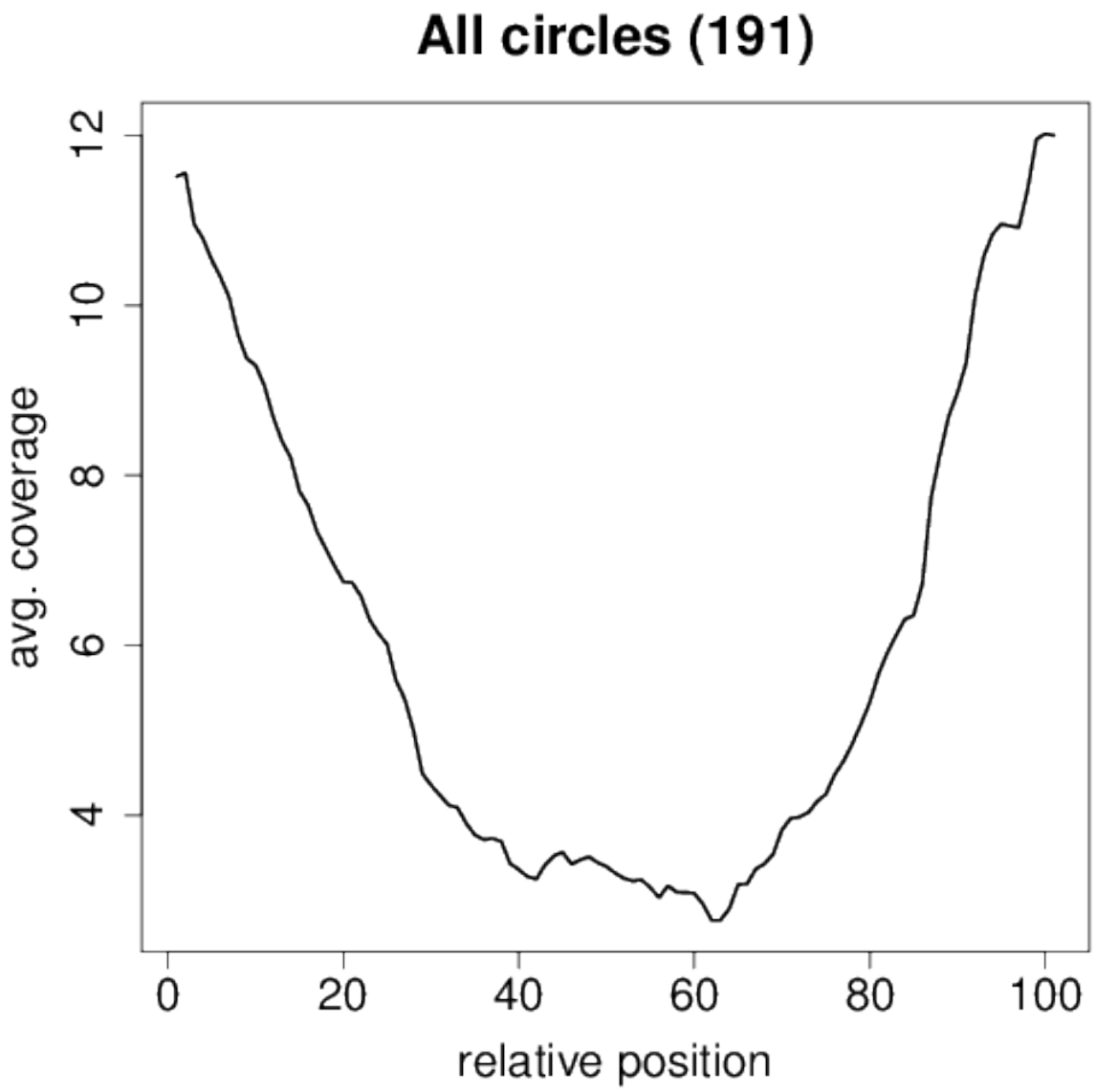
Medium circles



Long circles



Profiles of all circles



2.4 Quick check module

The circtools quickcheck module is designed to equip the user with a fast way of assessing the quality of the circRNA library preparation and the success of the mapping process.

`circtools quickcheck` requires sequencing reads have been mapped with STAR since internally the STAR log files are processed. CircRNA detection metrics are provided via `circtools detect` which has to be run prior to call the quickcheck module.

2.4.1 Required tools and packages

`quickcheck` depends on R and two R packages, namely

- ggplot2: general plotting
- ggrepel: label assignment in plots

2.4.2 General usage

A call to `circtools quickcheck --help` shows all available command line flags:

```
usage: circtools [-h] -d DCC_DIR -s STAR_DIR -l CONDITION_LIST -g GROUPING
                 [-o OUTPUT_DIRECTORY] [-n OUTPUT_NAME] [-c {colour,bw}]
                 [-C CLEANUP] [-S STARFOLDER] [-L REMOVE_SUFFIX_CHARS]
                 [-F REMOVE_PREFIX_CHARS] [-R REMOVE_COLUMNS]

circular RNA sequencing library quality assessment

optional arguments:
  -h, --help            show this help message and exit

Required:
  -d DCC_DIR, --DCC DCC_DIR
                        Path to the detect/DCC data directory
  -s STAR_DIR, --star STAR_DIR
                        Path to the base STAR data directory containing sub-
                        folders with per-sample mappings
  -l CONDITION_LIST, --condition-list CONDITION_LIST
                        Comma-separated list of conditions which should be
                        comparedE.g. "RNaseR +","RNaseR -"
  -g GROUPING, --grouping GROUPING
                        Comma-separated list describing the relation of the
                        columns specified via -c to the sample names specified
                        via -l; e.g. -g 1,2 and -r 3 would assign sample1 to
                        each even column and sample 2 to each odd column

Output options:
  -o OUTPUT_DIRECTORY, --output-directory OUTPUT_DIRECTORY
                        The output directory for files created by circtest
                        [Default: ./]
  -n OUTPUT_NAME, --output-name OUTPUT_NAME
                        The output name for files created by circtest
                        [Default: quickcheck]
  -c {colour,bw}, --colour {colour,bw}
                        Can be set to bw to create grayscale graphs for
                        manuscripts
  -C CLEANUP, --cleanup CLEANUP
                        String to be removed from each sample name [Default:
                        "_STARmapping.*Chimeric.out.junction"]
  -S STARFOLDER, --starfolder STARFOLDER
                        Suffix string of the STAR folders[Default:
                        "_STARmapping"]
  -L REMOVE_SUFFIX_CHARS, --remove-last REMOVE_SUFFIX_CHARS
                        Remove last N characters from each column name of the
                        DCC input data [Default: 0]
  -F REMOVE_PREFIX_CHARS, --remove-first REMOVE_PREFIX_CHARS
                        Remove first N characters from each column name of the
                        DCC input data [Default: 0]
  -R REMOVE_COLUMNS, --remove-columns REMOVE_COLUMNS
                        Comma-separated list of columns in the DCC data files
                        to not includes in the check
```

Sample call

```
circtools quickcheck -d 01_detect/ -s ../star  -l minus,plus -g 1,2,1,2,1,2,1,2  -
↪o 02_quickcheck/  -C .Chimeric.out.junction
```

Here we have the DCC data located in the folder `01_detect/`, the STAR mapping are stored in `star/`, the

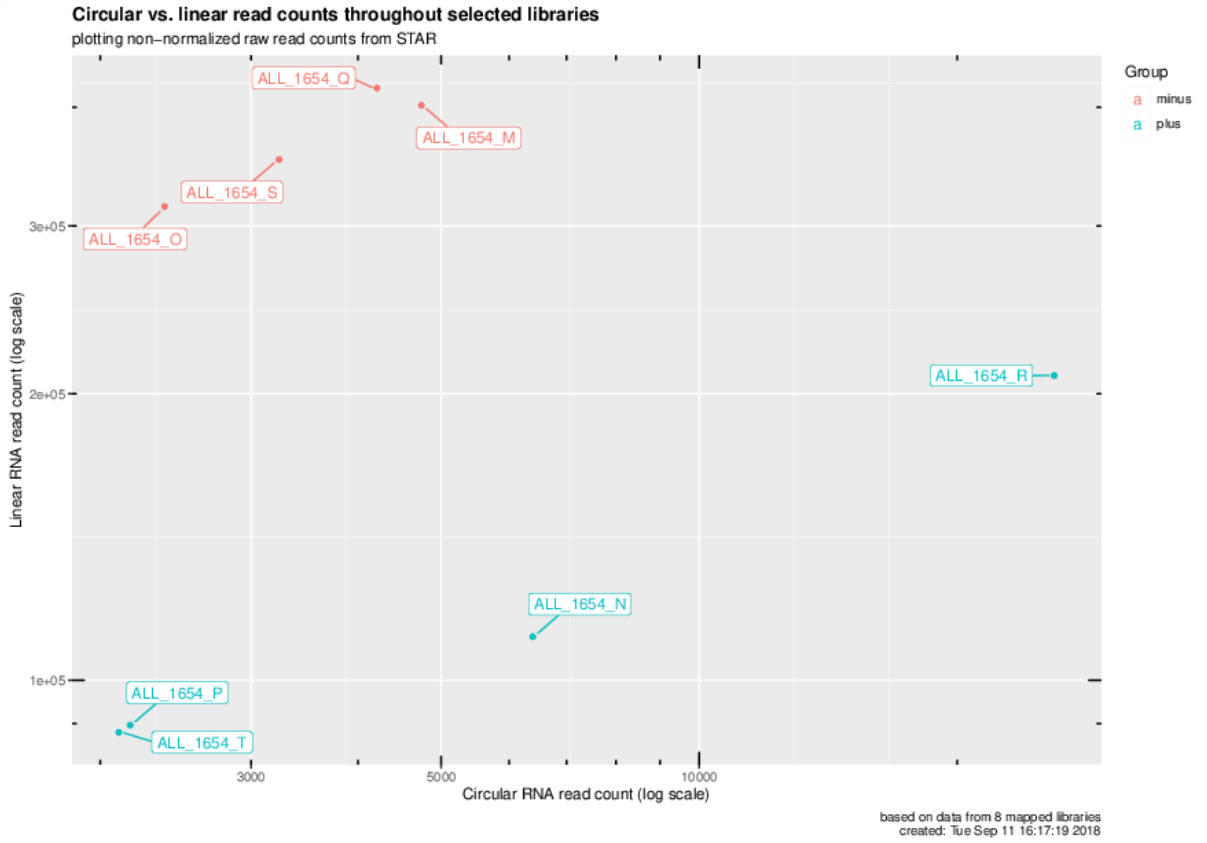
experiment had 4 conditions, listed via `-l RNaseR_minus,RNaseR_plus`, the samples in the detection data file are sorted in the the order specified via `-g 1,2,1,2,1,2,1,2`.

```
Using R version 3.5.0 [/usr/bin/Rscript]
Loading CircRNACount
Loading LinearRNACount
Parsing data
Found 8 data columns in provided DCC data
2 different groups provided
Assuming (1,2),(1,2),(1,2),... sample grouping
plotting data
Done
```

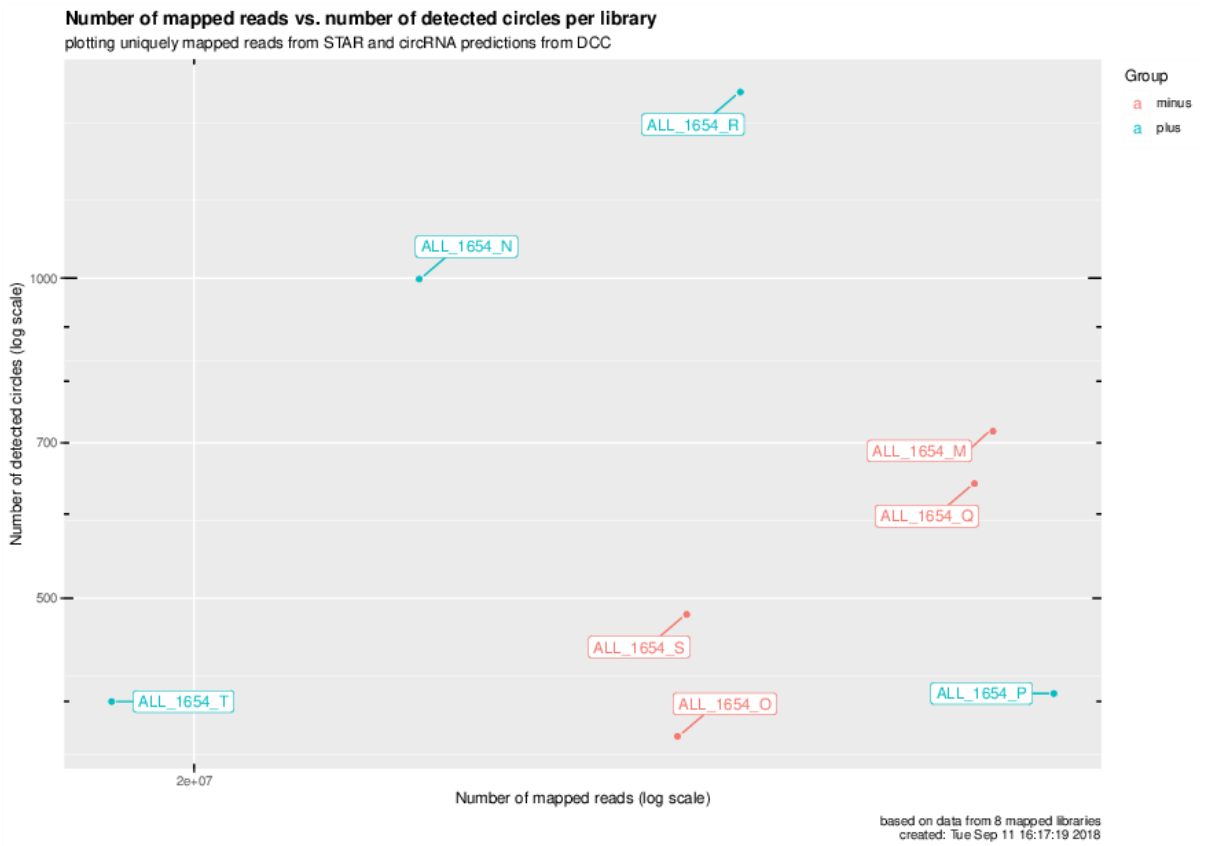
`circtools` takes a few seconds to process the data.

Graphical output

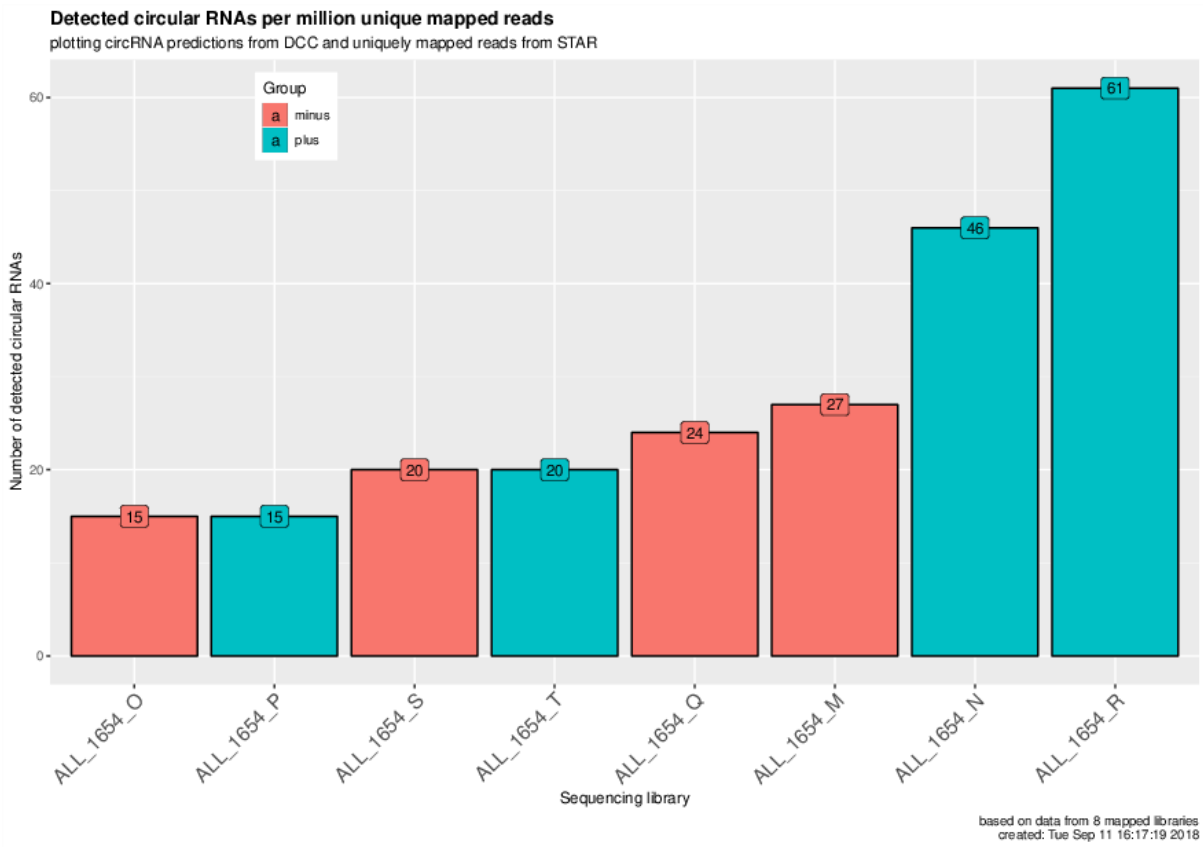
Circular vs. linear read counts for all mapped libraries



Number of mapped reads vs number of detected circRNAs for all mapped libraries



CircRNAs per million uniquely mapped reads



2.5 CircTest module

The CircTest module of circtools allows to test the variation of circRNAs in respect to host genes. It is recommended to work with the output of the `circtools detect` module, but can also run on custom count tables. Required are one table with circular RNA counts and one table containing with host-gene counts. These tables have to have the same order, i.e. `circ[i,j]` and `linear[i,j]` are read-counts for the same circRNA in the same sample.

The `circtools circtest` module is based on the equally named R package CircTest

2.5.1 Required tools and packages

`circtools circtest` depends on R and the following R packages:

- aod
- ggplot2
- plyr

The `CircTest` R package as well as all dependencies are installed during the circtools installation procedure.

2.5.2 Manual installation instructions

The following commands have to be performed within an R shell:

```
> install.packages("devtools")
> require(devtools)
> install_github('dieterich-lab/CircTest')
> library(CircTest)
```

2.5.3 Usage with `circtools detect` data

A call to `circtools circtest --help` shows all available command line flags:

```
usage: circtools [-h] -d DCC_DIR -l CONDITION_LIST -c CONDITION_COLUMNS -g
                 GROUPING [-r NUM_REPLICATES] [-f MAX_FDR] [-p PERCENTAGE]
                 [-s FILTER_SAMPLE] [-C FILTER_COUNT] [-o OUTPUT_DIRECTORY]
                 [-n OUTPUT_NAME] [-m MAX_PLOTS] [-a LABEL] [-L RANGE]
                 [-O ONLY_NEGATIVE] [-H ADD_HEADER] [-M {colour,bw}]

circular RNA statistical testing - Interface to https://github.com/dieterich-lab/
↪CircTest

optional arguments:
  -h, --help            show this help message and exit

Required:
  -d DCC_DIR, --DCC DCC_DIR
                        Path to the detect/DCC data directory
  -l CONDITION_LIST, --condition-list CONDITION_LIST
                        Comma-separated list of conditions which should be
                        comparedE.g. "RNaseR +","RNaseR -"
  -c CONDITION_COLUMNS, --condition-columns CONDITION_COLUMNS
                        Comma-separated list of 1-based column numbers in the
                        detect/DCC output which should be compared; e.g.
                        10,11,12,13,14,15
  -g GROUPING, --grouping GROUPING
                        Comma-separated list describing the relation of the
                        columns specified via -c to the sample names specified
                        via -l; e.g. -g 1,2 and -r 3 would assign sample1 to
                        each even column and sample 2 to each odd column

Processing options:
  -r NUM_REPLICATES, --replicates NUM_REPLICATES
                        Number of replicates used for the circRNA experiment
                        [Default: 3]
  -f MAX_FDR, --max-fdr MAX_FDR
                        Cut-off value for the FDR [Default: 0.05]
  -p PERCENTAGE, --percentage PERCENTAGE
                        The minimum percentage of circRNAs account for the
                        total transcripts in at least one group. [Default:
                        0.01]
  -s FILTER_SAMPLE, --filter-sample FILTER_SAMPLE
                        Number of samples that need to contain the amount of
                        reads specified via -C [Default: 3]
  -C FILTER_COUNT, --filter-count FILTER_COUNT
                        Number of CircRNA reads that each sample specified via
                        -s has to contain [Default: 5]

Output options:
  -o OUTPUT_DIRECTORY, --output-directory OUTPUT_DIRECTORY
                        The output directory for files created by circtest
                        [Default: .]
  -n OUTPUT_NAME, --output-name OUTPUT_NAME
                        The output name for files created by circtest
```

(continues on next page)

(continued from previous page)

```
                        [Default: circtest]
  -m MAX_PLOTS, --max-plots MAX_PLOTS
                        How many of candidates should be plotted as bar chart?
                        [Default: 50]
  -a LABEL, --label LABEL
                        How should the samples be labeled? [Default: Sample]
  -L RANGE, --limit RANGE
                        How should the samples be labeled? [Default: Sample]
  -O ONLY_NEGATIVE, --only-negative-direction ONLY_NEGATIVE
                        Only print entries with negative direction indicator
                        [Default: False]
  -H ADD_HEADER, --add-header ADD_HEADER
                        Add header to CSV output [Default: False]
  -M {colour,bw}, --colour {colour,bw}
                        Can be set to bw to create grayscale graphs for
                        manuscripts
```

Sample call

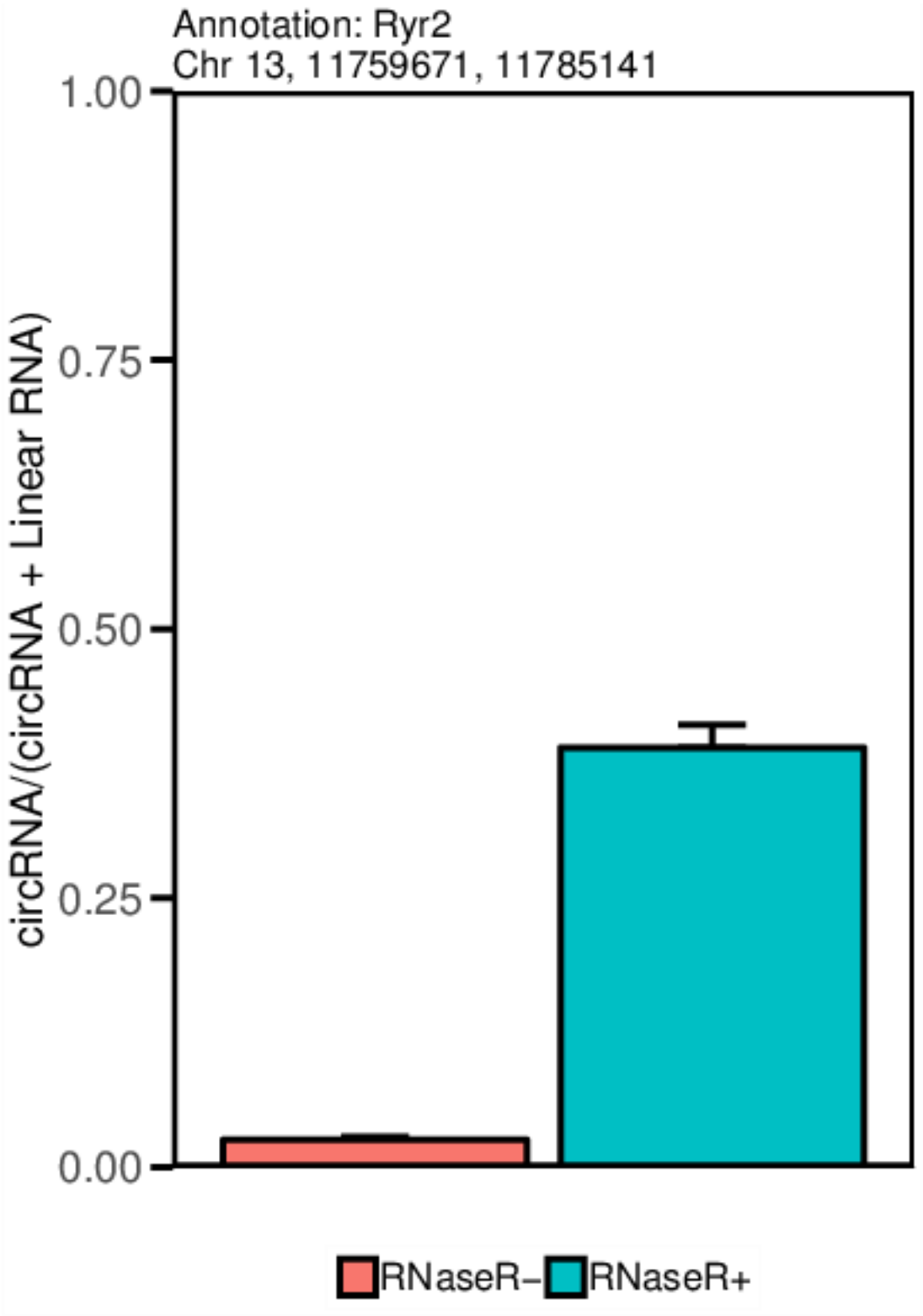
As for the other module tutorials, we use the Jakobi et al. 2016 data set from the detection module in this module. Below is the sample call for the newly generated circtools detect data:

```
circtools circtest -d 01_detect/ -p 0.01 -s 3 -r 4 -C 2 -g 1,2,1,2,1,2,1,2 -l␣
↪RNaseR-,RNaseR+ -c 4,5,6,7,8,9,10,11 -o 04_circtest/
```

Here we have the DCC data located in the folder `01_detect/`, the experiment had 2 conditions, listed via `-l RNaseR-,RNaseR+`, the samples in the circtools detect data file are sorted in the the order specified via `-g 1,2,1,2,1,2,1,2`, i.e. there are 4 `RNaseR-` samples and 4 `RNaseR+` samples. These `4+4=8` columns are found in the circtools detect data file in the columns specified via `-c 4,5,6,7,8,9,10,11`.

Output files

The `circtest` module creates an .xlsx file that contains all circRNA candidates passing the statistical test with the given values, as well as the raw data files. Additionally a .pdf file is generated that contains a graphical representation of the top significant circRNAs (see sample picture).



2.5.4 Usage with external count data

Additional to the built-in functionality to use directly use the data files produced by `circtools detect` it is also possible to use generic count tables. In this case however, the underlying R package `CircTest` has to be used directly. The input tables may have many columns describing the circle or just one column containing the circle ID followed by many columns of read counts.

Example count table for back-spliced reads (`Circular.csv`)

CircID	Control_1	Control_2	Control_3	Treatment_1	Treatment_2	Treatment_3
chr1:100|800	0	2	1	5	4	0
chr1:1050|10080	20	22	21	10	13	0
chr2: 600|1000	0	1	0	10	0	1
chr10:4100|5400	55	54	52	56	53	50
chr11:600|1500	3	0	1	2	2	3

Example table for host-gene reads (`Linear.csv`)

CircID	Control_1	Control_2	Control_3	Treatment_1	Treatment_2	Treatment_3
chr1:100|800	10	11	12	9	10	10
chr1:1050|10080	80	281	83	45	48	46
chr2: 600|1000	5	5	2	12	8	7
chr10:4100|5400	101	110	106	150	160	153
chr11:600|1500	20	21	18	19	20	20

Sample R calls to work with generic data

1. Read in tables

```
Circ <- read.delim('Circ.csv', header = T, as.is = T)
Linear <- read.delim('Linear.csv', header = T, as.is = T)
```

2. Filter tables

To model expression data using the beta binomial distribution and testing for differences in groups, it is beneficial to only test well supported circles. Users may use the package's function `Circ.filter()` to filter the input data. The function has the following parameters:

- `Nreplicates`: specifies the number of replicates in each condition
- `filter.sample`: specifies the number of samples the circle has to have enough circular reads in to be considered.
- `filter.count`: specifies the circular read count threshold.
- `percentage`: specifies the minimum circle to host-gene ratio.
- `circle_description`: tells the function which columns are NOT filled with read counts but the circle's annotation.

```
# filter circles by read counts
Circ_filtered <- Circ.filter(circ = Circ, linear = Linear, Nreplicates = 3, filter.
↪sample = 3, filter.count = 5, percentage = 0.1, circle_description = 1)

#            CircID Control_1 Control_2 Control_3 Treatment_1 Treatment_2␣
↪Treatment_3
# 2 chr1:1050|10080        20        22        21          10          13
↪ 0
# 4 chr10:4100|5400        55        54        52          56          53
↪50


# filter linear table by remaining circles
```

(continues on next page)

(continued from previous page)

```
Linear_filtered <- Linear[rownames(Circ_filtered),]

#            CircID Control_1 Control_2 Control_3 Treatment_1 Treatment_2␣
↪Treatment_3
# 2 chr1:1050|10080        80        81        83          45          48␣
↪46
# 4 chr10:4100|5400       101       110       106         150         160␣
↪153
```

3. Test for changes

Circ.test uses the beta binomial distribution to model the data and performs an ANOVA to identify circles which differ in their relative expression between the groups. It is important that the grouping is correct (**group**) and the non-read-count columns are specified (**circle_description**).

```
test <- Circ.test(Circ_filtered, Linear_filtered, group=c(rep(1,3),rep(2,3)),␣
↪circle_description = 1)
$summary_table
           CircID      sig_p
4 chr10:4100|5400 0.01747407

# $sig.dat
#            CircID Control_1 Control_2 Control_3 Treatment_1 Treatment_2␣
↪Treatment_3
# 4 chr10:4100|5400        55        54        52          56          53␣
↪50

$p.val
[1] 0.153464107 0.008737037

$p.adj
[1] 0.15346411 0.01747407

$sig_p
[1] 0.01747407
```

4. Visualize data

The CircTest library features a built-in plotting functions to view significantly different genes. Sample code for visualizing the ratio as barplot might be something like:

```
for (i in rownames(test$summary_table))  {
  Circ.ratioplot(Circ_filtered, Linear_filtered, plotrow=i, groupindicator1=c(rep(
↪'Control',3),rep('Treatment',3)),
                 lab_legend='Condition', circle_description = 1 )
}
```

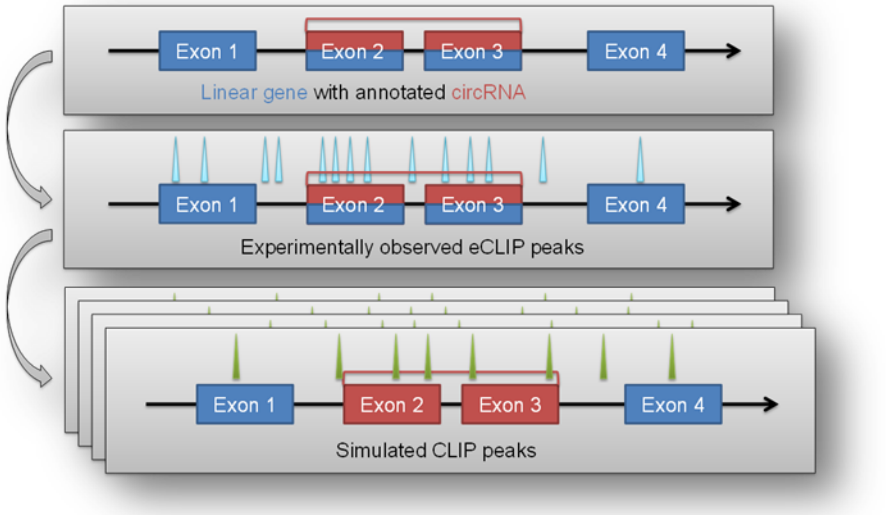
In order to visualize the abundance of host-gene and circle separately in a line plot try

```
 for (i in rownames(test$summary_table))  {
   Circ.lineplot(Circ_filtered, Linear_filtered, plotrow=i, groupindicator1=c(rep(
↪'Control',3),rep('Treatment',3)),
                 circle_description = 1 )
}
```

2.6 Enrichment module

The `circtools enrichment` module was implemented in order to combine circRNA data with positional data from other experiments, for example the eCLIP approach. The setup uses detected circRNA back splice

junctions and the corresponding location of circRNAs within the genome to test if positional features (e.g. eCLIP peaks) are significantly enriched within a circRNA compared to the remaining linear host gene.



This may give hints to potential RBP sponge functions of circRNAs when they show significant enrichment of eCLIP peaks in the circRNA portion of the host gene.

2.6.1 Background

Input

The circtools enrich module requires generally three types of input data:

- circRNA coordinates, e.g. from the circtools package itself or any other BED6-formatted circRNA list.
- a BED6-formatted file with coordinates of features of interest. E.g. RNA binding protein binding sites or any other sequence-based features that can be condensed into genomic coordinates
- A genome in FASTA format as well as a genome annotation in GTF format. Circtools works well and was tested with ENSEMBL-based genomes and annotations

How does it work

In a first step the 'observed' distribution of features throughout the supplied circRNAs is calculated. This observed distribution is used as a baseline in the subsequent 'iteration' step. By employing the `bedtools shuffle` command the features are randomly distributed throughout the genome while keep the number and length of all features constant. After several hundred or thousand randomized iterations circtools counts the number of iterations in which more hits within the defined list of circRNAs are observed than in the initial, actual experimental observation. Circtools than computes the probability that a given number of hits is significantly higher than the simulated random distribution obtained by the random shuffling. The test is carried out for the circRNA and the corresponding host genes, therefore also allowing to distinguish between features enriched in the circRNA and possibly depleted in the circRNA host gene.

2.6.2 Required tools and packages

`circtools enrich` depends on bedtools as well as R and a some R packages for visualization purposes.

R packages:

- ggplot2

- ggrepel
- data.table
- reshape2
- plyr
- gridExtra

Python libraries:

- pybedtools >= 0.7.10
- statsmodels>=0.8.0

Note: The `enrichment` circtools module as well as all R dependencies are automatically installed during the circtools installation procedure.

2.6.3 General usage

A call to `circtools enrich --help` shows all available command line flags:

```
usage: circtools [-h] -c CIRC_RNA_INPUT -b BED_INPUT -a ANNOTATION -g
                 GENOME_FILE [-o OUTPUT_DIRECTORY] [-i NUM_ITERATIONS]
                 [-p NUM_PROCESSES] [-t TMP_DIRECTORY] [-T THRESHOLD]
                 [-P PVAL] [-F OUTPUT_FILENAME] [-I INCLUDE_FEATURES]
                 [-k KEEP_TEMP]

circular RNA RBP enrichment tools

optional arguments:
  -h, --help            show this help message and exit

Required options:
  -c CIRC_RNA_INPUT, --circ-file CIRC_RNA_INPUT
                        Path to the CircRNACount file generated by DCC
  -b BED_INPUT, --bed-input BED_INPUT
                        One or more BED files containing features to overlap
  -a ANNOTATION, --annotation ANNOTATION
                        Genome reference annotation file used to not shuffle
                        into intragenic regions
  -g GENOME_FILE, --genome GENOME_FILE
                        Genome file for use with bedtools shuffle. See
                        bedtools man page for details.

Additional options:
  -o OUTPUT_DIRECTORY, --output OUTPUT_DIRECTORY
                        The output folder for files created by circtest
                        [default: .]
  -i NUM_ITERATIONS, --iterations NUM_ITERATIONS
                        Number of iterations for CLIP shuffling [default:
                        1000]
  -p NUM_PROCESSES, --processes NUM_PROCESSES
                        Number of threads to distribute the work to
  -t TMP_DIRECTORY, --temp TMP_DIRECTORY
                        Temporary directory used by pybedtools
  -T THRESHOLD, --threshold THRESHOLD
                        p-value cutoff
  -P PVAL, --pval PVAL  p-value cutoff
  -F OUTPUT_FILENAME, --output-filename OUTPUT_FILENAME
                        Defines the output file prefix [default: output]
```

(continues on next page)

(continued from previous page)

```
  -I INCLUDE_FEATURES, --include-features INCLUDE_FEATURES
                        Defines the the features which should be used for
                        shuffling. May be specified multiple times. [default:
                        all - shuffle over the whole genome]
  -k KEEP_TEMP, --keep-temp KEEP_TEMP
                        Keep temporary files created by circtools/bedtools
                        [default: no]
```

Generating necessary input data files

In addition to input data produced by the detection and reconstruct module, the enrichment module requires a few processing steps. For our example we employ the circRNAs detected in the murine heart and are interested in possible enrichment of repeat in the flanking intron of those circRNAs. Therefore, as a first step the flanking introns need to be compiled from the circRNA coordinates provided by the detect module.

```
circtools_generate_flanking_introns.py -g /scratch/tjakobi/circtools_workflow/
↪genes_and_introns.gtf -d /scratch/tjakobi/circtools_workflow/workflow/circtools/
↪01_detect/CircCoordinates > /scratch/tjakobi/circtools_workflow/murine_flanking_
↪introns.bed
```

Additionally, the shuffling algorithm of `bedtools` requires knowledge of chromosome sizes. For the mouse genome, a sample file for those length can be easily downloaded:

```
wget https://data.dieterichlab.org/s/mm10_chrom_sizes/download -O mm10.chrom.sizes
```

In order to provide a working example of reasonable size we do not use the full set of repeats as provided by the UCSC genome browser but only the 3 most-common ones, i.e. :

- AT_rich
- B3
- RSINE1

```
wget https://data.dieterichlab.org/s/repeat_selection_mm10/download -O repeat_
↪selection_mm10.tar.bz2
tar -jxvf repeat_selection_mm10.tar.bz2
```

After unzipping the downloaded file, the folder `repeats/` contains BED files with coordinates of the three aforementioned repeat categories. Those files will be used as input in the next step. The circtools enrich module is able to work with arbitrary features of a GTF annotation file. However, our aim is to search in introns for enrichment and introns are not part of normal ENSEMBL GTF annotation files. circtools includes a script that easily converts ENSEMBL GTF files in GTF files enriched with intron information.

```
mkdir 06_enrich/
cd 06_enrich/

# download build 90 annotation
wget ftp://ftp.ensembl.org/pub/release-90/gtf/mus_musculus/Mus_musculus.GRCm38.90.
↪gtf.gz

# unzip and add introns
gzip -d Mus_musculus.GRCm38.90.gtf.gz
circtools_generate_intron_gtf.sh Mus_musculus.GRCm38.90.gtf
```

The resulting files, `genes_and_introns.gtf` will now serve as replacement for the standard ENSEMBL annotation in the module call.

Calling the reconstruct module via wrapper

```
cd 06_enrich/

# download wrapper for STAR
wget https://raw.githubusercontent.com/dieterich-lab/bioinfo-scripts/master/slurm_
↪circtools_enrich_intron.sh
chmod 755 slurm_circtools_enrich_intron.sh

parallel -j1 slurm_circtools_enrich_intron.sh /scratch/tjakobi/circtools_workflow/
↪workflow/circtools/06_enrich/mm10.chrom.sizes /scratch/tjakobi/circtools_
↪workflow/workflow/circtools/06_enrich/genes_and_introns.gtf /scratch/tjakobi/
↪circtools_workflow/workflow/circtools/06_enrich/repeats/{}.bed /scratch/tjakobi/
↪circtools_workflow/workflow/circtools/06_enrich/murine_flanking_introns.bed {} /
↪scratch/tjakobi/circtools_workflow/workflow/circtools/06_enrich/output/ 2000 /
↪scratch/global_tmp/{}/ :::: /scratch/tjakobi/circtools_workflow/workflow/
↪circtools/06_enrich/repeats/repeats_selected.list
```

Manual module call

Below a sample single call of circtools enrich without using the wrapper script:

```
circtools enrich -c /scratch/tjakobi/circtools_workflow/workflow/circtools/06_
↪enrich/murine_flanking_introns.bed -b /scratch/tjakobi/circtools_workflow/
↪workflow/circtools/06_enrich/repeats/AT_rich.bed -a /scratch/tjakobi/circtools_
↪workflow/workflow/circtools/06_enrich/genes_and_introns.gtf -g /scratch/tjakobi/
↪circtools_workflow/workflow/circtools/06_enrich/mm10.chrom.sizes -i 2000 -I␣
↪intron -p 20 -P 1 -T 1 -o /scratch/tjakobi/circtools_workflow/workflow/circtools/
↪06_enrich/output// -F AT_rich -t /scratch/global_tmp/AT_rich/
```

This call to `circtools enrich` will produce output similar to the one shown below. The run time depends on the size of the circRNA dataset as well as the number of peaks used for the analysis.

Command line output

```
2018-09-17 11:13:16,166 circtools 1.1.0.6 started
2018-09-17 11:13:16,166 circtools command line: /home/tjakobi//.local/bin/
↪circtools enrich -c /scratch/tjakobi/circtools_workflow/workflow/circtools/06_
↪enrich/murine_flanking_introns.bed -b
 /scratch/tjakobi/circtools_workflow/workflow/circtools/06_enrich/repeats/AT_rich.
↪bed -a /scratch/tjakobi/circtools_workflow/workflow/circtools/06_enrich/genes_
↪and_introns.gtf -g /scratch/tjakobi/circtools_workflow/workflow/circtools/06_
↪enrich/mm10.chrom.sizes -i 2000 -I intron -p 20 -P 1 -T 1 -o /scratch/tjakobi/
↪circtools_workflow/workflow/circtools/06_enrich/output// -F AT_rich -t /scratch/
↪global_tmp/AT_rich//
2018-09-17 11:13:16,177 bedtools v2.27.1 detected
2018-09-17 11:13:16,177 Parsing annotation...
2018-09-17 11:13:17,864 Found 256488 entries
2018-09-17 11:13:17,865 Done parsing annotation
2018-09-17 11:13:20,126 Parsing BED input file...
2018-09-17 11:13:21,207 Done parsing BED input file:
2018-09-17 11:13:21,207 => 228756 peaks, 33 nt average width
2018-09-17 11:13:21,207 Parsing annotation...
2018-09-17 11:13:21,727 Found 52636 entries
2018-09-17 11:13:21,728 Done parsing annotation
2018-09-17 11:13:22,777 Parsing circular RNA input file...
2018-09-17 11:13:22,787 Done parsing circular RNA input file:
2018-09-17 11:13:22,788 => 2522 circular RNAs, 1801 nt average (theoretical␣
↪unspliced) length
```

(continues on next page)

(continued from previous page)

```
2018-09-17 11:13:23,057 Starting random shuffling of input peaks
2018-09-17 11:13:23,059 Processing shuffling thread 1
2018-09-17 11:13:23,059 Processing shuffling thread 26
[output cut]
2018-09-17 11:35:14,025 Permutation test iteration 1998
2018-09-17 11:35:14,043 Permutation test iteration 1991
2018-09-17 11:35:14,172 Permutation test iteration 2000
2018-09-17 11:35:14,198 Permutation test iteration 1993
2018-09-17 11:35:14,221 Permutation test iteration 1995
2018-09-17 11:35:14,381 Permutation test iteration 1997
2018-09-17 11:35:14,578 Permutation test iteration 1999
2018-09-17 11:35:17,547 Cleaning up... just a second
2018-09-17 11:35:18,740 Cleaning up temporary files
2018-09-17 11:35:20,552 Deleting /scratch/global_tmp/AT_rich/pybedtools.knohds5y.
↪tmp
2018-09-17 11:35:20,553 Deleting /scratch/global_tmp/AT_rich/pybedtools.j0mwk09_.
↪tmp
2018-09-17 11:35:20,553 Deleting /scratch/global_tmp/AT_rich/pybedtools.85vjrbnw.
↪tmp
2018-09-17 11:35:20,553 Deleting /scratch/global_tmp/AT_rich/pybedtools.jli7p0je.
↪tmp
2018-09-17 11:35:20,553 Deleting /scratch/global_tmp/AT_rich/pybedtools.4vz84ujl.
↪tmp
2018-09-17 11:35:20,560 Done
```

2.6.4 Output produced by `circtools enrich`

*.csv

The generated CSV file is the main output of `circtools enrich`. It contains the data generated during the run and has the following fields:

- *circRNA_host_gene*: Name of the circRNA host gene
- *chr*: Chromosome location of the circRNA
- *start*: Absolute circRNA start location
- *stop*: Absolute circRNA end location
- *strand*: Strand of the circRNA
- *p-val_circular*: p-value for the enrichment of peaks within the given circRNA
- *raw_count_circ_rna*: How many simulated peaks have been counted
- *observed_input_peaks_circ_rna*: How many real, experimental peaks have been observed
- *length_circ_rna*: Length of the circRNA
- *length_normalized_count_circ_rna*: Lengt-normalized count of observed peaks
- *number_of_features_intersecting_circ*: How many featured are interesecting the circRNA (only `-i`)
- *circ_rna_confidence_interval_0.05*: 0.05% confidence interval for the circRNA test
- *p-val_linear*: p-value for the enrichment of peaks within the linear host gene *excluding* the circRNA portion
- *raw_count_host_gene*: How many simulated peaks have been counted
- *observed_input_peaks_host_gene*: How many real, experimental peaks have been observed
- *length_host_gene_without_circ_rna*: Length of the host gene minus the circRNA length
- *length_normalized_count_host_gene*: Lengt-normalized count of observed peaks

- *number_of_features_intersecting_linear*: How many featured are interesecting the host gene (only `-i`)
- *host_gene_confidence_interval_0.05*: 0.05% confidence interval for the linear test
- *distance_normalized_counts*: Distance between the length-normalized counts of linear host gene and circRNA

RERE	1	[illegible]	8541213	.	0.0005	1	1	2000	0.5	1	(1.265[illegible]05, 0.00278263983465965 04)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	37	(nan, -0.0018452739 7934069074)	[illegible]

*.bed

The generated BED files are holding the temporary annotation data created by `circtools enrich`.

- *.gtf_features.bed*: Contains one row for each *feature* extract from the supplied genome annotatation (only in if `-i` is used)
- *.gtf_genes.bed*: Conatins one for for each gene entry from the supplied genome annotation
- *.bed_circles.bed*: Contains all supplied CircRNAs in BED format

*.log

The log file generated by `circtools enrich`.

2.6.5 Additional graphical visualization

Circtools is bundled with an additional R-script to post-process the raw data of the enrichment module. In order to be used with the visualization script, the `enrich` data has to be slightly preprocessed. The visualization script is designed to work with multiple sets of peaks, i.e. multiple different eCLIP data sets.

Pre-processing

In case of the example used throughout this tutorial, the experiment has been performed with multiple repeat classes and the following command may be used to merge all runs into one CSV file with a new first column that contains the file name of the originating run (assuming 2000 iterations):

```
awk -F '\t' '{{print FILENAME"\t"$0}}' *.csv | sed 's/_.*2000_.*.csv//g' | grep -v␣
↪circRNA_host_gene > ../results.csv
```

Subsequently, the transformed data file may be used for visualization:

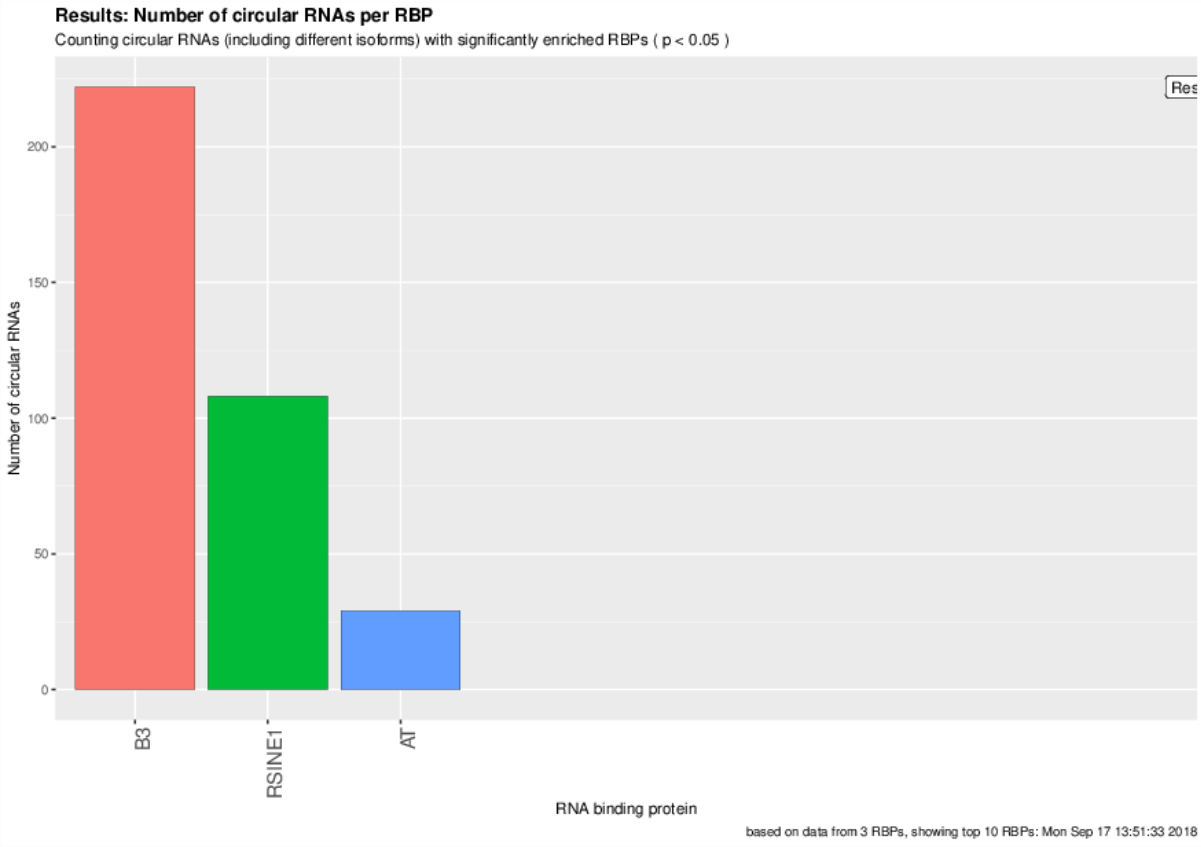
Plotting

```
circtools_enrich_visualization.R results.csv 0.05 10 10 results.pdf "Results"␣
↪colour False
```

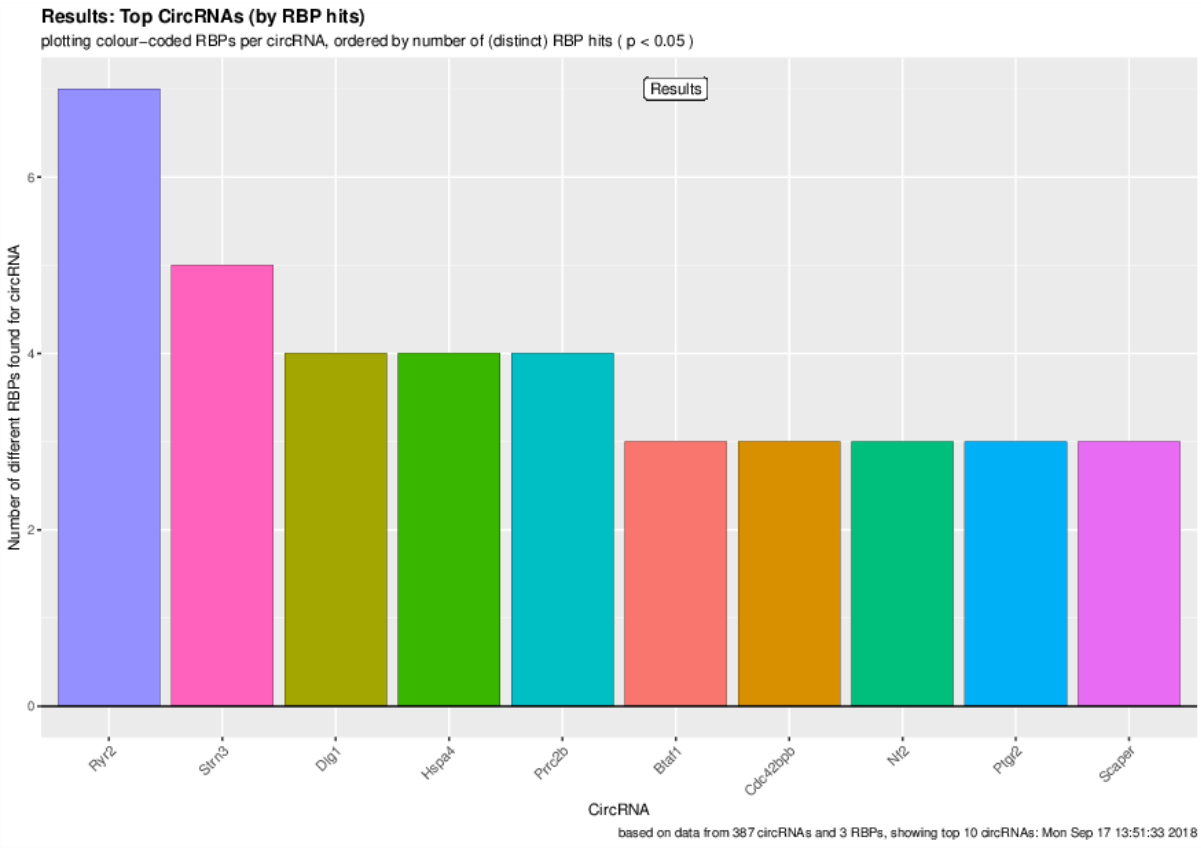
Visualized results

Visualization of the results generated by the enrichment module for the Jakobi 2016 data set. The is sample experiment looks at enriched repeats in the flanking introns of the detect circRNAs.

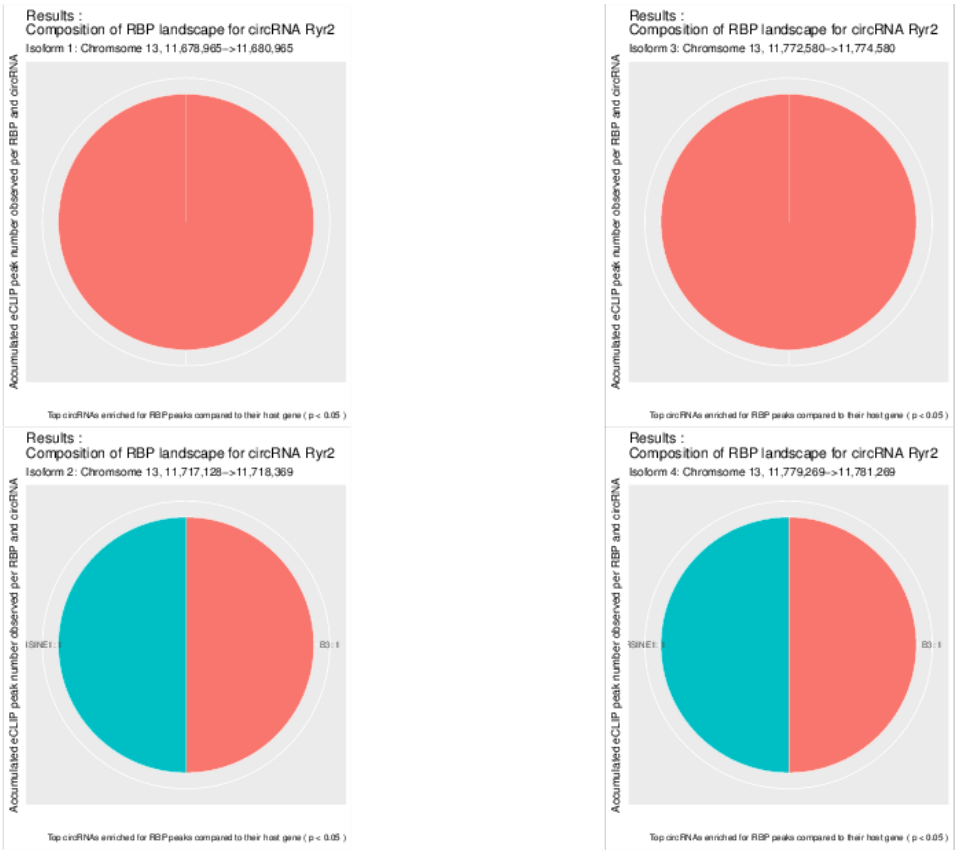
1) Top 3 repeat categories with enrichment within the flanking introns (max. +/- 2kb) of the significantly enriched circRNA candidates



2) Number of circular RNAs per repeat category (p<0.05)



3) Detailed view of the repeat "peaks" enriched in the flanking introns of isoform 1 of circRyr2



2.7 Alternative exon module

The circtools exon usage module was implemented to detect and analyze differential exon usage in circRNA data sets. As an example, the module may list exons as significant differentially spliced in RNaseR treated sample compared to untreated samples, thus pointing out exons that may be part of a circRNA.

`circtools exon` requires mapped sequencing reads that are passed to StringTie in order to generate data readable by the ballgown R package.

2.7.1 Required tools and packages

`exon` depends on R and a few additional R packages, namely

- ballgown
- edgeR
- ggbio
- ggfortify
- openxlsx
- GenomicRanges
- GenomicFeatures

The `exon` circtools module as well as all R dependencies are automatically installed during the circtools installation procedure.

2.7.2 General usage

A call to `circtools exon --help` shows all available command line flags:

```
usage: circtools [-h] -d DCC_DIR -l CONDITION_LIST -c CONDITION_COLUMNS -g
                 GROUPING -r REPLICATES -b BALLGOWN_DATA -G GTF_FILE -C
                 CIRCTEST_FILE [-s {mm,rn,hs}] [-H HAS_HEADER]
                 [-o OUTPUT_DIRECTORY] [-n OUTPUT_PREFIX]

circular RNA exon usage analysis

optional arguments:
  -h, --help            show this help message and exit

Required:
  -d DCC_DIR, --DCC DCC_DIR
                        Path to the detect/DCC data directory
  -l CONDITION_LIST, --condition-list CONDITION_LIST
                        Comma-separated list of conditions which should be
                        comparedE.g. "RNaseR +","RNaseR -"
  -c CONDITION_COLUMNS, --condition-columns CONDITION_COLUMNS
                        Comma-separated list of 1-based column numbers in the
                        detect/DCC output which should be compared; e.g.
                        10,11,12,13,14,15
  -g GROUPING, --grouping GROUPING
                        Comma-separated list describing the relation of the
                        columns specified via -c to the sample names specified
                        via -l; e.g. -g 1,2 and -r 3 would assign sample1 to
                        each even column and sample 2 to each odd column
  -r REPLICATES, --replicates REPLICATES
                        Comma-separated list describing the relation of the
                        samples specified via -g to the sample names specified
```

(continues on next page)

(continued from previous page)

```
                        via -l; e.g. -g 1,2 and -r 3 would assign sample1 to
                        each even column and sample 2 to each odd column
  -b BALLGOWN_DATA, --ballgown-data BALLGOWN_DATA
                        Path to the ballgown data directory
  -G GTF_FILE, --gtf-file GTF_FILE
                        Path to the GTF file containing the employed genome
                        annotation
  -C CIRCTEST_FILE, --circtest-output CIRCTEST_FILE
                        Path to the CircTest CSV file containing the CircTest
                        results
  -s {mm,rn,hs}, --species {mm,rn,hs}
                        Organism of the study (used for primer BLASTing), rn =
                        Rattus norvegicus, mm = Mus musculus, hs = Homo
                        sapiens

Additional options:
  -H HAS_HEADER, --has-header HAS_HEADER
                        Do the CircTest result files have a header? [Default:
                        No]

Output options:
  -o OUTPUT_DIRECTORY, --output-directory OUTPUT_DIRECTORY
                        The output directory for files created by circtools
                        [Default: .]
  -n OUTPUT_PREFIX, --output-prefix OUTPUT_PREFIX
                        The output name (prefix) for files created by
                        circtools [Default: exon_analysis]
```

Generating necessary ballgown data files

In order to perform per-exon analyses, the circtools exon module requires additional data generated with StringTie. The tutorial assumes, that stringtie has been installed and is available via the `$PATH` environment.

```
# download wrapper for Stringtie
wget https://raw.githubusercontent.com/dieterich-lab/bioinfo-scripts/master/slurm_
↪stringtie.sh
chmod 755 slurm_stringtie.sh
mkdir stringtie/

# obtain the annotation of the mouse genome for splice junctions
wget ftp://ftp.ensembl.org/pub/release-90/gtf/mus_musculus/Mus_musculus.GRCm38.90.
↪gtf.gz
gzip -d Mus_musculus.GRCm38.90.gtf.gz

cd star/

# run stringtie for all samples
parallel ../slurm_stringtie.sh {}/Aligned.noS.bam ../Mus_musculus.GRCm38.90.gtf ../
↪stringtie/{}_StringTieBallgown/ ::: ALL*
```

circtools exon module call

After generating all necessary input data, the circtools exon module can now be run via the following command:

```
circtools exon -d 01_detect/ -r1,1,2,2,3,3,4,4 -l minus,plus -c 4,5,6,7,8,9,10,11 -
↪g1,2,1,2,1,2,1,2 -C 04_circtest/circtest.csv -b ../stringtie/ -G ../Mus_musculus.
↪GRCm38.90.gtf -o 05_exon/ -s mm
```

Here we have the DCC data located in the folder `01_detect/`, the stringtie data are stored in `../stringtie/`, the experiment had 2 conditions, listed alternating via `-l minus,plus`, the samples in the circtools detect data file are sorted in the the order specified via '' -g1,2,1,2,1,2,1,2'' and columns 10-15 are used for the analysis, as specified via `-c 4,5,6,7,8,9,10,11`. The genome annotation has to be supplied with the `-G ../Mus_musculus.GRCm38.90.gtf` flag. Significantly enriched circRNAs from the `circtest` module have to be passed via `-C 04_circtest/circtest.csv`, the species for the internal gene ID conversion has been set via `-s mm` to mouse, the output will be stored in `-o 05_exon/`.

```
Using R version 3.5.0 [/usr/bin/Rscript]
Loading required packages
Done loading packages
Loading CircRNACount
Loading CircCoordinates
Starting ballgown processing
Sun Jun 17 21:17:47 2018
Sun Jun 17 21:17:47 2018: Reading linking tables
Sun Jun 17 21:17:48 2018: Reading intron data files
Sun Jun 17 21:17:52 2018: Merging intron data
Sun Jun 17 21:17:54 2018: Reading exon data files
Sun Jun 17 21:18:00 2018: Merging exon data
Sun Jun 17 21:18:02 2018: Reading transcript data files
Sun Jun 17 21:18:05 2018: Merging transcript data
Wrapping up the results
Sun Jun 17 21:18:05 2018
Preparing necessary data structures
Setting treatment and conditions
Found 11031 multi exon genes
Found 1574 single exon genes
Starting dispersion estimation
Fitting model...
Writing bed files...
Writing DCC prediction BED file
Reading and integrating CircTest results
Writing back splice junction enriched BED file
Writing Excel file
Writing additional CSV output
Exon analysis finished
```

`circtools` takes some time to process the data and prints out information on its progress.

2.7.3 Output produced by `circtools exon`

exon_analysis_bsj_enrichment.csv

circRNA-centric view of the exon results in CSV format. Shown are significantly enriched circRNAs merged with the results from the ballgown package.

exon_analysis_exon_enrichment.csv

Exon-centric view of the exon results in CSV format. Shown are differentially spliced exons merged with the circRNA detection and circtest step.

exon_analysis_diff_exon_enrichment.xlsx

An xlsx Excel file containing 4 work sheets:

- Exon FDR 1% (ballgown): differentially spliced exons, 1% FDR
- enriched BSJ FDR 1% (CircTest): enriched circRNAs, 1% FDR

- Other BSJ FDR 1%: non-annotated circRNAs
- Exon events: all exons

exon_analysis_dcc_bsj_enriched_track.bed

A BED file with containing *only* circRNAs predicted by the `circtools detect` module that **also** pass the *circtools circtest'* statistical test. Can be displayed in all common visualization tools like IGV.

exon_analysis_dcc_predictions_track.bed

A BED file with containing *all* circRNAs predicted by the `circtools detect` module. Can be displayed in all common visualization tools like IGV.

exon_analysis_exon_fc_track.bedgraph

A BEDgraph file with fold changes of all differentially spliced exons. Can be displayed in all common visualization tools like IGV.

exon_analysis_exon_pval_track.bedgraph

A BEDgraph file with p-values of all differentially spliced exons. Can be displayed in all common visualization tools like IGV.

2.8 Primer design module

The circtools primex module is a highly specialized primer design tool tailored specifically for circRNA experiments.

`circtools primex` is able to design primer pairs in batches of hundreds of circRNAs based on circRNAs detected with `circtools detect`, but can also work on lists with specific circRNA isoforms or even entirely without any preliminary data purely based on the FASTA sequence of the circRNA.

The `circtools primex` module is based on the equally named R package

primex

2.8.1 Required tools and packages

`circtools primex` depends on R, several R packages, and BioPython:

R packages:

- primex
- formattable
- kableExtra
- dplyr
- RColorBrewer
- colortools

Python libraries:

- BioPython>=1.71

All R package as well as Python dependencies are installed during the circtools installation.

2.8.2 General usage

A call to `circtools primex --help` shows all available command line flags:

```
usage: circtools [-h] -d DCC_FILE -g GTF_FILE -f FASTA_FILE [-O {mm,hs}]
                 [-s SEQUENCE_FILE] [-o OUTPUT_DIR] [-T EXPERIMENT_TITLE]
                 [-t GLOBAL_TEMP_DIR] [-G GENE_LIST [GENE_LIST ...]]
                 [-p PRODUCT_SIZE [PRODUCT_SIZE ...]]
                 [-i ID_LIST [ID_LIST ...]] [-j {r,n,f}] [-b]

circular RNA primer design

optional arguments:
  -h, --help            show this help message and exit

Input:
  -d DCC_FILE, --dcc-file DCC_FILE
                        CircCoordinates file from DCC / detect module
  -g GTF_FILE, --gtf-file GTF_FILE
                        GTF file of genome annotation e.g. ENSEMBL
  -f FASTA_FILE, --fasta FASTA_FILE
                        FASTA file with genome sequence (must match
                        annotation)
  -O {mm,hs}, --organism {mm,hs}
                        Organism of the study (used for primer BLASTing), mm =
                        Mus musculus, hs = Homo sapiens
  -s SEQUENCE_FILE, --sequence SEQUENCE_FILE
                        FASTA file containing the circRNA sequence (exons and
                        introns)

Output options:
  -o OUTPUT_DIR, --output OUTPUT_DIR
                        Output directory (must exist)
  -T EXPERIMENT_TITLE, --title EXPERIMENT_TITLE
                        Title of the experiment for HTML output and file name

Additional options:
  -t GLOBAL_TEMP_DIR, --temp GLOBAL_TEMP_DIR
                        Temporary directory (must exist)
  -G GENE_LIST [GENE_LIST ...], --genes GENE_LIST [GENE_LIST ...]
                        Space-separated list of host gene names. Primers for
                        CircRNAs of those genes will be designed.E.g. -G
                        "CAMSAP1" "RYR2"
  -p PRODUCT_SIZE [PRODUCT_SIZE ...], --product-size PRODUCT_SIZE [PRODUCT_SIZE ...
↪]
                        Space-separated range for the desired PCR product.
                        E.g. -p 80 160 [default]
  -i ID_LIST [ID_LIST ...], --id-list ID_LIST [ID_LIST ...]
                        Space-separated list of circRNA IDs. E.g. -i
                        "CAMSAP1_9_135850137_135850461_-"
                        "CAMSAP1_9_135881633_135883078_-"
  -j {r,n,f}, --junction {r,n,f}
                        Should the forward [f] or reverse [r] primer be
                        located on the BSJ? [Default: n]
  -b, --no-blast        Should primers be BLASTED? Even if selected yes here,
                        not more than 50 primers willbe sent to BLAST in any
                        case.
```

Designing primers with circtools primex

A sample call to primex using the Jakobi et al. 2016 data generated with circtools detect requires as only external parameter the Fasta sequence of the reference genome in order to obtain DNA sequences for the primer design

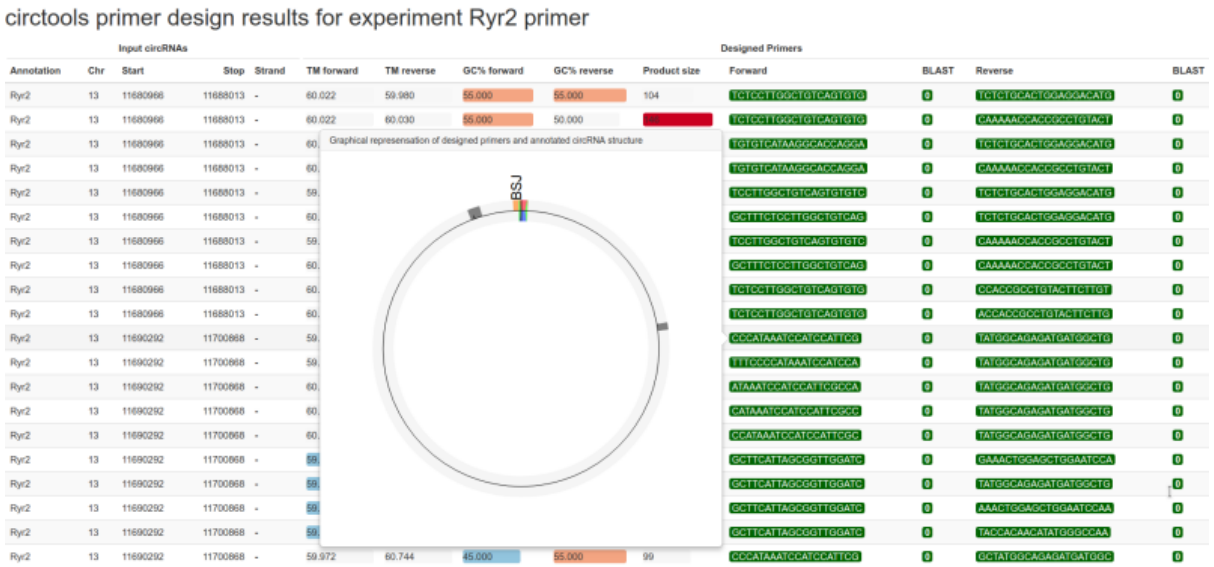
process.

```
# obtain reference genome (if not already downloaded)
wget ftp://ftp.ensembl.org/pub/release-90/fasta/mus_musculus/dna/Mus_musculus.
↪GRCm38.dna.primary_assembly.fa.gz

# obtain annotation (if not already downloaded)
wget ftp://ftp.ensembl.org/pub/release-90/gtf/mus_musculus/Mus_musculus.GRCm38.90.
↪gtf.gz

# unzip
gzip -d Mus_musculus.GRCm38.dna.primary_assembly.fa.gz
gzip -d Mus_musculus.GRCm38.90.gtf.gz

# run circtools primex, design primer for gene Ryr2
circtools primex -d DCC/CircCoordinates -f Mus_musculus.GRCm38.dna.primary_
↪assembly.fa -g Mus_musculus.GRCm38.90.gtf -O mm -G Ryr2 -T "Ryr2 primer"
```

```
Start parsing GTF file
Start merging GTF file
extracting flanking exons for circRNA # 1 Ryr2_13_11680966_11688013_-
extracting flanking exons for circRNA # 2 Ryr2_13_11690292_11700868_-
extracting flanking exons for circRNA # 3 Ryr2_13_11718370_11730486_-
extracting flanking exons for circRNA # 4 Ryr2_13_11737722_11745759_-
extracting flanking exons for circRNA # 5 Ryr2_13_11749436_11785141_-
extracting flanking exons for circRNA # 6 Ryr2_13_11759671_11772579_-
extracting flanking exons for circRNA # 7 Ryr2_13_11759671_11779268_-
extracting flanking exons for circRNA # 8 Ryr2_13_11759671_11785141_-
extracting flanking exons for circRNA # 9 Ryr2_13_11769852_11785141_-
extracting flanking exons for circRNA # 10 Ryr2_13_11779185_11801925_-
extracting flanking exons for circRNA # 11 Ryr2_13_11824274_11853190_-
extracting flanking exons for circRNA # 12 Ryr2_13_11868117_11885538_-
Sending 92 primers to BLAST
This may take a few minutes, please be patient.
Writing results to /tmp/Ryr2_primer.html
```

`circtools primex` takes a few seconds to process the input data and sends the generated primers pairs to the web-based BLAST service of the NCBI in order to give the user hints about potential unwanted off-site targets. The output is written to a HTML file which can be opened with any browser.

Sample of the HTML output generated by `circtools primex`



2.9 Support

We work hard to ensure that **circtools** is a powerful tool empowering your research. However, no software is free of bugs and issues, therefore we would love to get feedback from our users.

2.9.1 Report issues

There are different options file an issues with out circRNA related software. Whenever possible please try to provide as much information as possible - this help to quickly assess the problem and fix it. Especially helpful are

- Software versions used
- Log files
- The command line used to call the software
- Sample data files in order to reproduce the error
- Temporary files that may contain pointers to fix the issues

Via GitHub

Please use the following GitHub projects to file your issues:

- circtools: for issues related to circtools itself
- DCC: for issues related to the circtools detect module / DCC
- CircTest: for issues related to the circtools circtest module / the CircTest library
- FUCHS: for issues related to the circtools circtest reconstruct module / FUCHS
- primex: for issues related to the circtools primex module / the primex library

Via Mail

If you do not have a GitHub account you can also reach use via mail: circtools@dieterichlab.org

2.9.2 Literature

- circRNAs in murine hearts: Jakobi, T., Czaja-Hasse, L.F., Reinhardt, R. and Dieterich, C., 2016. Profiling and validation of the circular RNA repertoire in adult murine hearts. Genomics, proteomics & bioinformatics, 14(4), pp.216-223.
- circRNA analysis workflow: Jakobi, T. and Dieterich, C., 2018. Deep Computational Circular RNA Analytics from RNA-seq Data. In Circular RNAs (pp. 9-25). Humana Press, New York, NY.
- `circtools reconstruct` / FUCHS: Metge, F., Czaja-Hasse, L.F., Reinhardt, R. and Dieterich, C., 2017. FUCHS—towards full circular RNA characterization using RNAseq. PeerJ, 5, p.e2934.
- `circtools detect` / DCC: Cheng, J., Metge, F. and Dieterich, C., 2015. Specific identification and quantification of circular RNAs from sequencing data. Bioinformatics, 32(7), pp.1094-1096.

CHAPTER 3

License

circtools is freely available under the GNU General Public License v3.0.

CHAPTER 4

Issues

Problems or issues should be reported via the GitHub issue system.