










## SOFTWARE TOOL ARTICLE

**REVISED** **bcbioRNASeq: R package for bcbio RNA-seq analysis**
**[version 2; peer review: 1 approved, 1 approved with reservations]**

Michael J. Steinbaugh <sup>1</sup>, Lorena Pantano <sup>1</sup>, Rory D. Kirchner <sup>1</sup>, Victor Barrera <sup>1</sup>, Brad A. Chapman<sup>1</sup>, Mary E. Piper <sup>1</sup>, Meeta Mistry<sup>1</sup>, Radhika S. Khetani <sup>1</sup>, Kayleigh D. Rutherford<sup>1</sup>, Oliver Hofmann<sup>2</sup>, John N. Hutchinson <sup>1</sup>, Shannan Ho Sui<sup>1</sup>

<sup>1</sup>Harvard T.H. Chan School of Public Health, Boston, MA, 02115, USA

<sup>2</sup>University of Melbourne Center for Cancer Research, Melbourne, VIC, 3000, Australia

**V2** **First published:** 08 Nov 2017, **6**:1976  
<https://doi.org/10.12688/f1000research.12093.1>  
**Latest published:** 20 Jun 2018, **6**:1976  
<https://doi.org/10.12688/f1000research.12093.2>

### Abstract

RNA-seq analysis involves multiple steps, from processing raw sequencing data to identifying, organizing, annotating, and reporting differentially expressed genes. bcbio is an open source, community-maintained framework providing automated and scalable RNA-seq methods for identifying gene abundance counts. We have developed bcbioRNASeq, a Bioconductor package that provides ready-to-render templates, objects and wrapper functions to post-process bcbio RNA sequencing output data. bcbioRNASeq helps automate the generation of high-level RNA-seq reports, facilitating the quality control analyses, identification of differentially expressed genes and functional enrichment analyses.

### Keywords











RNA-seq, quality control, differential expression, functional analysis, data import, visualization, report writing, R Markdown




This article is included in the **Bioinformatics** gateway.

### Open Peer Review

**Approval Status**  

	1	2
<b>version 2</b>		
(revision)		
20 Jun 2018		
		
<b>version 1</b>		
08 Nov 2017		

1. **Charlotte Sonesson** , University of Zurich (UZH), Zürich, Switzerland

2. **Davide Riso** , Weill Cornell Medicine, New York, USA

Any reports and responses or comments on the article can be found at the end of the article.



This article is included in the **RPackage** gateway.



This article is included in the **Bioconductor** gateway.

**Corresponding author:** Michael J. Steinbaugh ([mike@steinbaugh.com](mailto:mike@steinbaugh.com))

**Author roles:** **Steinbaugh MJ:** Conceptualization, Methodology, Software, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Pantano L:** Conceptualization, Methodology, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Kirchner RD:** Conceptualization, Software, Writing – Review & Editing; **Barrera V:** Software, Writing – Review & Editing; **Chapman BA:** Writing – Original Draft Preparation, Writing – Review & Editing; **Piper ME:** Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Mistry M:** Writing – Review & Editing; **Khetani RS:** Writing – Original Draft Preparation, Writing – Review & Editing; **Rutherford KD:** Validation, Writing – Review & Editing; **Hofmann O:** Funding Acquisition, Writing – Review & Editing; **Hutchinson JN:** Conceptualization, Funding Acquisition, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Ho Sui S:** Funding Acquisition, Supervision, Writing – Review & Editing

**Competing interests:** No competing interests were disclosed.

**Grant information:** The author(s) declared that no grants were involved in supporting this work.

**Copyright:** © 2018 Steinbaugh MJ *et al.* This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

**How to cite this article:** Steinbaugh MJ, Pantano L, Kirchner RD *et al.* **bcbioRNASeq: R package for bcbio RNA-seq analysis [version 2; peer review: 1 approved, 1 approved with reservations]** F1000Research 2018, **6**:1976 <https://doi.org/10.12688/f1000research.12093.2>

**First published:** 08 Nov 2017, **6**:1976 <https://doi.org/10.12688/f1000research.12093.1>

**REVISED** Amendments from Version 1

The new version of the article contains the following updates:

- We have added a new figure (Figure 1) to describe in more detail the structure of the object the package uses. It has been adapted from the RangedSummarizeExperiment object and shows where the different count data and metadata are stored.
- We have updated the package to use the GRanges structure to store the genomic position of genes. This should help to integrate gene expression with epigenetic information, such as ChIP-Seq data, when trying to find features close to genes using the already available methods for GRanges object.
- We have updated the name of some functions to match the Bioconductor style.
- We have added a new section at the end, "Functional Analysis" that describes the first steps for functional analysis after using the two other templates the package generates (quality control and differential expression). This section focuses on how to work with the data generated from the other templates to complete a functional analysis. We have focused only on the first few steps as the functional analysis follows published workflows.
- All figures and code lines have been updated to match the version of the package.
- We have added a discussion section after the quality control section to give examples of possible interpretations of the figures the package produces. The intention is to provide guidelines to the user on how to act accordingly based on the information generated in the figures.
- We have added a link to the session info from the computer used to run the code shown in the article.

**See referee reports**

## Introduction

RNA sequencing (RNA-seq) analysis seeks to identify differential expression among groups of samples, providing insights into the underlying biology of a system under study<sup>1</sup>. Automating a full analysis from raw sequence data to functionally annotated gene results requires the coordination of multiple steps and tools. From the first data processing steps to quantify gene expression, to the data quality checks necessary for identification of differentially expressed genes<sup>2</sup> and functionally enriched categories, RNA-seq analysis involves the repetition of commands using various tools. This is done on a per-sample basis, and each step can require varying degrees of user intervention. As a bioinformatics core facility that processes a large number of RNA-seq datasets, we have developed a Bioconductor (BioC)<sup>3</sup> package called `bcbioRNASEq` to aggregate the outputs of tools for RNA-seq quality control (QC), differential expression and functional enrichment analysis as much as possible, while still retaining full, flexible control of critical parameters.

This package relies on the output of `bcbio`, a Python framework that implements best-practice pipelines for fully automated high-throughput sequencing analysis (including RNA-seq, variant discovery, and ChIP-seq). `bcbio` is a community driven resource that handles the data processing component of sequencing analysis, providing researchers with more time to focus on the downstream biology. For RNA-seq data, `bcbio` generates QC and gene abundance information compatible with multiple downstream BioC packages. We briefly describe some of the tools included in the `bcbio` RNA-seq pipeline to help our users understand the outputs of `bcbio` that are used in the `bcbioRNASEq` package.

To ensure that the library generation and sequencing quality are suitable for further analysis, tools like `FastQC`<sup>4</sup> examine the raw reads for quality issues. `Cutadapt`<sup>5</sup> can optionally be used to trim reads for adapter sequences, along with other contaminant sequences such as polyA tails and low quality sequences with `PHRED`<sup>6,7</sup> quality scores less than five. `Salmon`<sup>8</sup> generates abundance estimates for known splice isoforms. In parallel, `STAR`<sup>9</sup> aligns the reads to the specified reference genome, and `featureCounts`<sup>10</sup> generates counts associated with known genes. `bcbio` assesses the complexity and quality of the RNA-seq data by quantifying ribosomal RNA (rRNA) content and the genomic context of alignments in known transcripts and introns using a combination of custom tools and `Qualimap`<sup>11</sup>. Finally, `MultiQC`<sup>12</sup> generates an interactive HTML report in which the metrics from all tools used during the analysis are combined into a single dynamic file. `bcbio` handles these first stages of RNA-seq data processing with little user intervention.

The next stages of an RNA-seq analysis include assessing read and alignment qualities, identifying outlier samples, clustering samples, assessing model fit, choosing cutoffs and finally, identifying differentially expressed genes. These steps often occur in multiple iterations, and require more active analyst involvement to integrate multiple tools that accept input data with incompatible formats and properties (see **Use Case** section). For example,

the featureCounts gene counts from STAR-based alignments (a simple matrix) are useful for quality control, providing many more quality metrics than the quasi-alignments from Salmon. However, the quasi-alignments from Salmon (which are imported by tximport into a list of matrices) have been shown to be more accurate when testing for differential gene expression<sup>13,14</sup>. Managing these disparate data types and tools can make analyses unnecessarily time consuming, and increases the risk of inconsistency between analyses. Given the complexity of the analysis, it is essential to report the final parameters and associated results in a cohesive, reproducible manner.

bcbioRNASeq was developed to address these issues and ease the process of documentation and report generation. The package offers multiple R Markdown templates that are ready-to-render after configuration of a few parameters and include example text and code for quality control metrics, differential expression, and functional enrichment analyses. Although other packages have been developed to solve similar issues, bcbioRNASeq allows for tight integration with the bcbio framework, and provides a unified package with objects, functions and pre-made templates for fast and simple RNA-seq analysis and reporting.

## Methods

### Reading data

As noted, bcbio runs a number of tools to generate QC metrics and compute gene counts from RNA-seq data. Additional information about the bcbio RNA-seq pipeline is available on [readthedocs](#)). At the end of a bcbio run, the most important files are stored in a separate directory specified by the user in the bcbio configuration YAML file under the “upload:” parameter; this directory is named “final” by default. Within this directory, there is a dated project directory containing quality metrics, provenance information, and data derived from the analysis that have been aggregated across all samples, e.g. count files. In addition, there is a directory corresponding to each sample that contains the binary alignment map (BAM) files and Salmon count data for that sample.

```
final/
  2018-01-01_illumina_rnaseq/
    annotated_combined.counts
    bcbio-nextgen-commands.log
    bcbio-nextgen.log
    combined.counts
    combined.dexseq
    combined.dexseq.ann
    data_versions.csv
    multiqc/
    programs.txt
    project-summary.yaml
    tx2gene.csv
  sample1/
    qc/
    salmon/
    sample1-ready.bam
    sample1-ready.bam.bai
    sample1-ready.counts
    sample1-transcriptome.bam
```

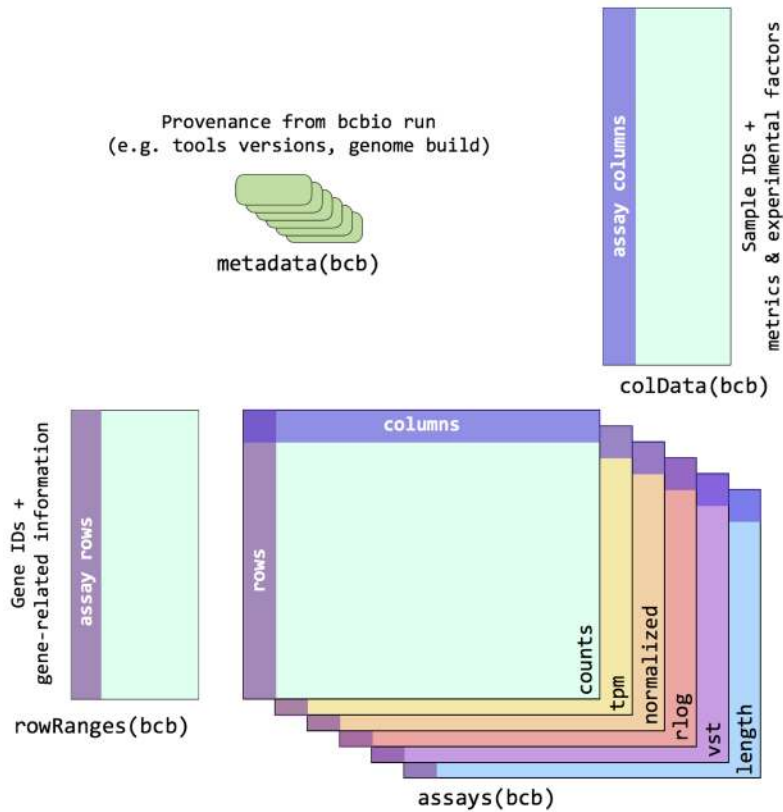
The final upload directory generated by bcbio is used as the input for bcbioRNASeq. Once the bcbio run is complete, you can open an R session and load the bcbioRNASeq package (source code is available at our [GitHub repository](#)). Use the `bcbioRNASeq()` constructor function (see example below) to create a structured S4 object that contains all of the necessary information for downstream analysis. The only required argument when creating this object is `uploadDir`, which specifies the path to the bcbio final upload directory. Note that bcbioRNASeq will transform all sample metadata column names to lowerCamelCase format without spaces, dashes, periods or underscores; therefore the `interestingGroups` argument should be specified in the same format. The values that can be used for `interestingGroups` are the same as the column names found in the CSV file used to `bcbio_nextgen.py`. Additionally, specifying the `organism` of the dataset is strongly recommended, and must use the full Latin name (e.g. *Mus musculus*); this enables automatic downloading of gene annotations from Ensembl. Once the S4 object is assigned, use the `saveData()` function to write the dataset to disk as an R Data file. Note that the following code block does not need to be run to reproduce the figures in this paper; its purpose is to describe how to load the data from a bcbio analysis into R using this package, facilitating use of the downstream functions.

```
# Non-working example demonstrating how to load a bcbio run.
# Load the pre-saved object instead (See use case below).
library(bcbioRNASeq)
bcb <- bcbioRNASeq(
  uploadDir = file.path("gse65267", "final"),
  organism = "Mus musculus",
  interestingGroups = "day"
)
saveData(bcb)
```

This S4 object is unique to the bcbioRNASeq package, as it contains all of the necessary data from the bcbio run required for analysis. From here, you can use various functions in bcbioRNASeq to perform analyses, make figures, and generate data tables and results files as we describe in later sections. This object is also used as the input for the R Markdown templates for report generation. First, we begin by describing the object in more detail.

**Object description**

We have designed a new S4 class named bcbioRNASeq (Figure 1) that is an extension of RangedSummarizedExperiment<sup>15</sup>. The assays() slot contains Salmon quasi-alignment data imported with tximport<sup>13</sup>, and automatically generated DESeq2<sup>16</sup> count transformations that provide support for quality control plots. To see all



Adapted from Bioconductor DOI: 10.18129/B9.bioc.SummarizedExperiment

**Figure 1. bcbioRNASeq S4 object structure.** The bcbioRNASeq object is an S4 class extension of the RangedSummarizedExperiment container. The assays slot contains several matrices with sample IDs as column names and gene IDs as row names, including raw counts and various derivations of the raw counts, and can be accessed via the assays() function. Gene IDs tie these assays to additional information about the genes stored as a GRanges object, and can be accessed via the rowRanges() function. Similarly, sample IDs tie the assays to further sample information such as run metrics from bcbio and experimental factors also stored as a DataFrame, and can be accessed via the colData() function. Non-gene or sample specific metadata about the entire run such as tool versions and genome builds is stored as a list and can be accessed via the metadata() function.

available assays in the `bcbioRNASeq` object listed by name, use `assayNames(bcb)`. These matrices are described in more detail below:

- `@assays`: `ShallowSimpleListAssays` containing count matrices derived from Salmon quasi-aligned counts imported with `tximport` and processed with `DESeq2`. Accessible with `assays()`.
  - `counts`: raw counts, generated by salmon and imported with `tximport`. Also accessible with `assay()`.
  - `tpm`: transcripts per million (TPM), calculated by `tximport`.
  - `length`: gene length matrix, calculated by `tximport`.
  - `normalized`: Normalized counts, with `DESeq2` `sizeFactors` applied.
  - `rlog`: regularized log transformation, calculated by `DESeq2`.
  - `vst`: variance stabilizing transformation, calculated by `DESeq2`.
- `@rowRanges`: `GRanges` describing the rows (genes) of the count matrices slotted in `assays()`. When `organism` is specified in the `bcbioRNASeq()` function call, gene annotations will be downloaded from Ensembl using `AnnotationHub` and `ensemblDb`. Accessible with `rowRanges()` and/or `rowData()`, which returns a `DataFrame()` instead of `GRanges`.
- `@colData`: `DataFrame` describing the columns (samples) of the count matrices slotted in `assays()`. Also contains sample quality metrics from `bcbio` analysis, generated from aligned counts produced by STAR and `featureCounts`. These aligned counts are not saved in the object. Accessible with `colData()`.
- `@metadata`: list containing additional metadata relevant to the dataset and the information pertaining to/generated from previous steps in the workflow. Accessible with `metadata()`.
  - `version`: Version of `bcbioRNASeq` package used to generate the object.
  - `level`: Whether counts are loaded at gene (default) or transcript level.
  - `caller`: Caller used to generate the counts. Defaults to `salmon`.
  - `countsFromAbundance`: Parameter used when reading transcript abundance with `tximport`.
  - `uploadDir`: Path to `bcbio` final upload directory.
  - `sampleDirs`: Paths of sample directories contained in `bcbio` upload directory.
  - `sampleMetadataFile`: Path to custom sample metadata file. Can be used to override the sample metadata saved in the `bcbio` run YAML, but is not normally needed.
  - `projectDir`: Path to project directory in `bcbio` upload.
  - `template`: Name of YAML file used to configure `bcbio` run.
  - `runDate`: Date of `bcbio` run completion.
  - `interestingGroups`: Groups of interest to use by default for quality control plot colors.
  - `organism`: Latin species name (e.g. “*Homo sapiens*”).
  - `genomeBuild`: Genome build (e.g. “GRCm38”).
  - `ensemblRelease`: Ensembl release version (e.g. 90).
  - `rowRangesMetadata`: Metadata describing the `ensemblDb` package used with `AnnotationHub` to define the `rowRanges`.
  - `gffFile`: Transcript annotation file path, if used instead of Ensembl metadata.
  - `tx2gene`: Transcript-to-gene identifier mappings.
  - `lanes`: Number of flow cell lanes used during sequencing.
  - `yaml`: `bcbio` run YAML containing summary statistics and sample metadata saved during configuration.

- `dataVersions`: Genome versions used by bcbio.
- `programVersions`: Program versions used by bcbio.
- `bcbioLog`: bcbio run log.
- `bcbioCommandsLog`: bcbio commands log.
- `allSamples`: Whether the object contains all samples from the run.
- `call`: `bcbioRNASeq()` function call used to create the S4 object.
- `date`: Date the bcbio run was loaded into R with `bcbioRNASeq()`.
- `wd`: Working directory path.
- `utilsSessionInfo`: `utils::sessionInfo()` output.
- `devtoolsSessionInfo`: `devtools::session_info()` output.

### Use case

To demonstrate the functionality and configuration of the package, we have used an experiment from the Gene Expression Omnibus (GEO) public repository of expression data as an example use case. The RNA-seq data is from a study of acute kidney injury in a mouse model (GSE65267)<sup>17</sup>. The study aims to identify differentially expressed genes in progressive kidney fibrosis and contains samples from mouse kidneys at several time points ( $n = 3$ , per time point) after folic acid treatment. From this dataset, we are using a subset of the samples for our use case: before folic acid treatment, and 1, 3, 7 days after treatment.

A pre-computed version of the example `bcbioRNASeq` object used in this workflow (`bcb.rda`) and the code for reproduction are available at the [f1000v2](#) branch of our [package repository on GitHub](#).

First, load the `bcbioRNASeq` object and a few other libraries to demonstrate how to access the different types of information contained in the object.

```
# Load the pre-saved object
library(bcbioRNASeq)
loadRemoteData("https://github.com/hbc/bcbioRNASeq/raw/f1000v2/data/bcb.rda")
```

The `counts()` function returns the abundance estimates generated by Salmon. Read counts for each sample in the dataset are aggregated into a matrix, in which columns correspond to samples and rows represent genes. Multiple normalized counts matrices are saved in the `bcbioRNASeq` object, and are accessible with the `normalized` argument:

1. Raw counts (`normalized = FALSE`).
2. DESeq2 normalized counts, with `sizeFactors` applied (`normalized = TRUE`).
3. Transcripts per million (`normalized = "tpm"`).
4. Trimmed mean of M-values normalization method (`normalized = "tmm"`).
5. Relative log expression (`normalized = "rle"`).
6. Regularized log transformation (`normalized = "rlog"`).
7. Variance stabilization transformation (`normalized = "vst"`).

### Exporting quantified data

The package contains multiple convenience functions to extract the expression abundances described above. Outlined below are the steps to save these counts external to the `bcbioRNASeq` object. These steps utilize functions from the DESeq2, edgeR and tximport packages both directly as well as within wrapper functions. For discussions on RNA-seq data normalization methods and count formats see 18; we typically save at least the DESeq2 normalized counts (library size adjusted) and transcripts per million counts (gene length adjusted) for further analyses.

```
raw <- counts(bcb, normalized = FALSE)
normalized <- counts(bcb, normalized = TRUE)
```

```

tpm <- counts(bcb, normalized = "tpm")
rlog <- counts(bcb, normalized = "rlog")
vst <- counts(bcb, normalized = "vst")
saveData(raw, normalized, tpm, rlog, vst)
writeCounts(raw, normalized, tpm, rlog, vst)

```

### Quality control steps

A typical RNA-seq analysis requires multiple quality control assessments at the read, alignment, sample and model level. Most of the data required to make these assessments is automatically generated by bcbio; the `bcbioRNASeq` package makes it easier for users to access it. For instance, the Qualimap tool runs as part of the bcbio pipeline and generates various metrics that can be used to assess the quality of the data and consistency across samples. The output of Qualimap is stored in the `bcbioRNASeq` object, and the package has several functions to visualize this output in a graphical format. These plots can be used to check data quality. Visual thresholds appear in many plots to help the user to assess quality. For example, a vertical dashed line is used as a cutoff threshold, helping to identify samples that require additional inspection. These default suggested cutoff values are suitable for human/mouse RNA-seq data sets (based on our experience), but should be manually modified for other species. The package does not focus on automatically determining threshold values, but instead provides a mechanism by which to visually communicate the quality of the data. The default cutoffs for these thresholds can be changed using function arguments. The `metrics()` function allows the user to extract a `data.frame` containing all metrics information used by the QC report functions. In this way, custom figures can also be easily created using the same data but with user-preferred packages.

Below, we provide several examples of recommended QC steps for RNA-seq data with a short explanation outlining their utility. By default, normalized counts generated with the DESeq2 `varianceStabilizingTransformation()` function is used to plot gene expression. This can be changed using the option named `normalized` in each function used to plot expression values.

### Read statistics

Total reads per sample and mapping rate are metrics that can help identify imbalances in sequencing depth or failures among the samples in a dataset (Figure 2A–B). Generally, we expect to see a similar sequencing depth across all samples in a dataset and mapping rates greater than 75%. Low genomic mapping rates are indicative of sample contamination, poor sequencing quality or other artifacts. Some figures show lines indicating the optimal minimum number for the quality metric.

```

plotTotalReads(bcb)
plotMappingRate(bcb)

```

### Genomic context

For RNA-seq, the majority of reads should map to exons and not introns. Ideally, at least 60% of total reads should map to exons. High levels of intronic mapping may indicate high proportions of nuclear RNA or DNA contamination. Samples should also have rRNA contamination rates below 10% (not shown) (Figure 2C–D).

```

plotExonicMappingRate(bcb)
plotIntronicMappingRate(bcb)

```

### Number of genes detected

Determining how many genes are detected relative to the number of mapped reads is another good way to assess the sample quality (Figure 2D–E). Ideally, all samples will have similar numbers for genes detected, and samples with higher number of mapped reads will have more genes detected. Large differences in gene detection numbers between samples can introduce biases and should be monitored at later steps for potential influence on sample clustering.

```

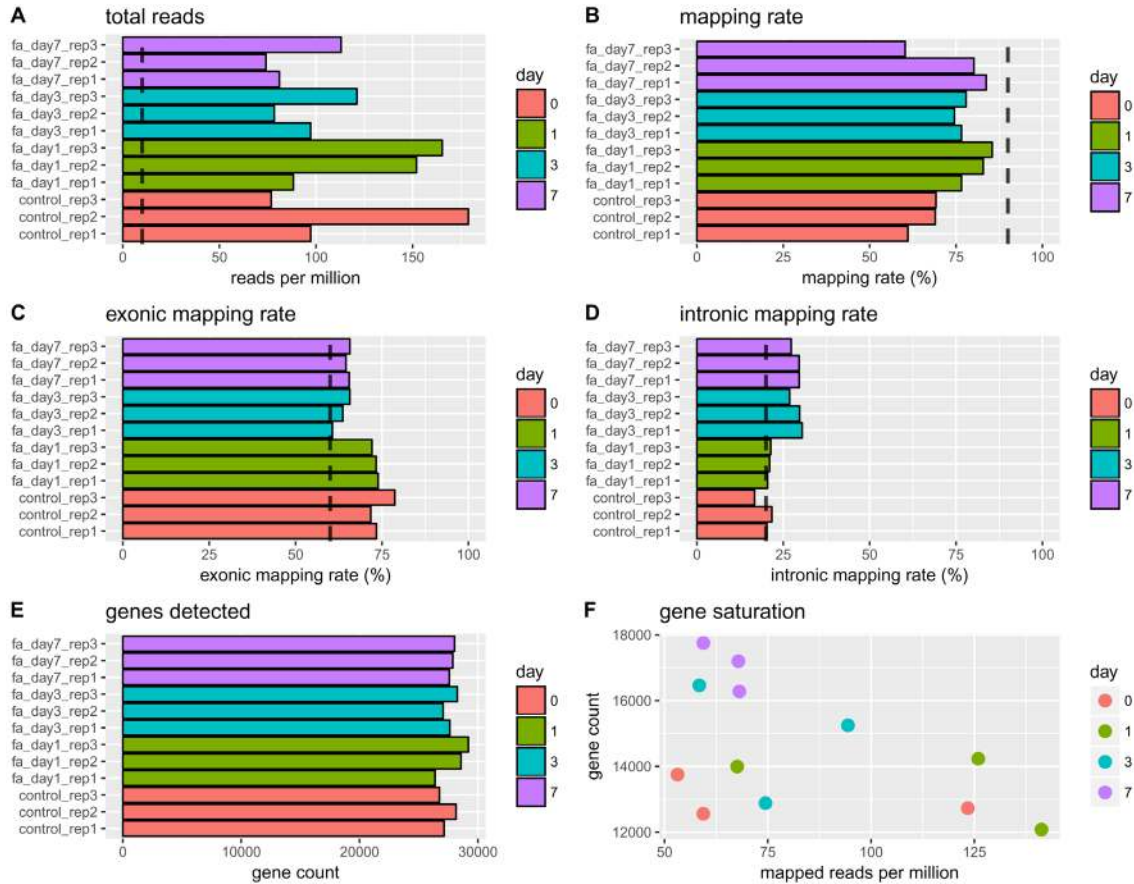
plotGenesDetected(bcb)
plotGeneSaturation(bcb)

```

### Counts per gene

Comparing the distribution of normalized gene counts across samples is one way to assess sample similarity within a dataset. For this figure, normalized counts comes from the trimmed mean of M-values, calculated by edgeR. This is output is from the function `cpm(normalized.lib.sizes = TRUE)` after applying `calcNormFactors` to a `DGEList` object containing the raw counts. We would expect similar count distributions for all genes across the





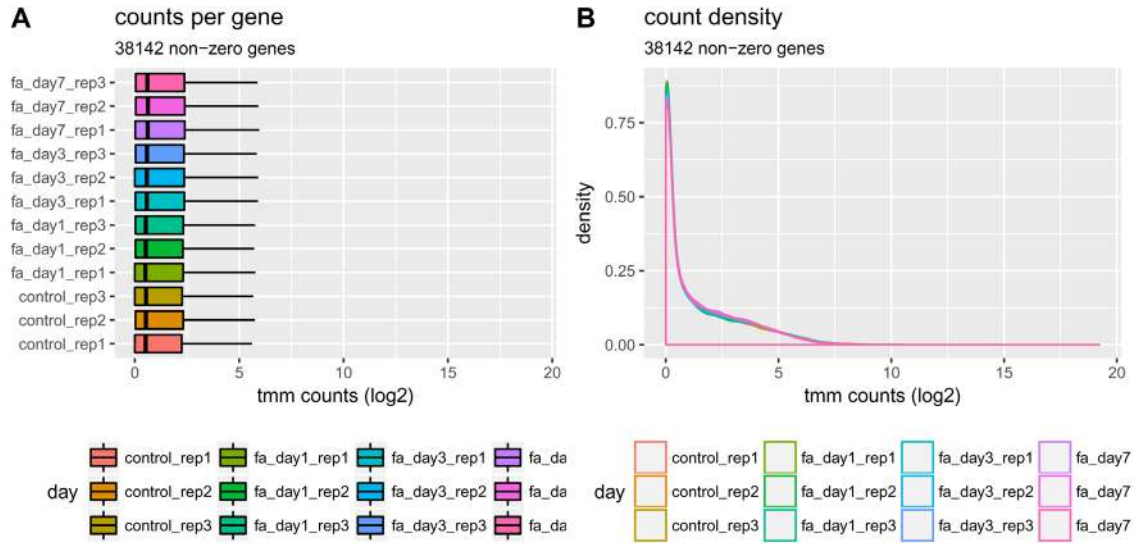
**Figure 2. Read, alignment, genomic context, and gene detection statistics.** Sample classes, as defined by the `interestingGroups` argument, are represented by the different colors defined in the legend of each plot. Vertical dashed black lines indicate suggested cutoff values. The total reads plot (**A**) indicates the total number of reads sequenced per sample. There is some variability, but in all cases the coverage is higher than 20 million reads, which we consider as sufficient to give good quality gene quantification. (**B**) shows the percentage of reads mapping to the reference genome. Here, all samples are well within recommended ranges, having well over 25 million reads and almost 90% of reads mapping. The exonic and intronic mapping rate plots (**C** and **D**) indicate the percentage of reads mapping to exons or introns, respectively. Here, all samples are within recommended ranges, with samples from day 3 and day 7 showing higher proportions of reads mapping to intronic versus exonic regions as compared to the day 1 and normal sample classes. The genes detected plot (**E**) indicates the total number of genes for each sample with at least one mapped read. Optimal minimum gene detection values will vary based on an organism's transcriptome size. The gene detection saturation plot (**F**) shows the relationship between the number of reads mapped and the number of genes detected. If this trend is not linear, it indicates that the sequencing may have been saturated in terms of detecting gene expression. For this specific data, it shows no saturation yet for gene detection, meaning that more genes could be detected if coverage were increased for the samples with lower number of reads.

samples unless the library sizes or total RNA expression are different (Figure 3). The `plotCountsPerGene()` and `plotCountDensity()` functions provide two ways to visualize this comparison. Zero counts are removed in these figures and the values are log2 scale.

```
plotCountsPerGene(bcb)
plotCountDensity(bcb)
```

### Model fitting

It is important to explore the fit of the model for a given dataset before performing differential expression analysis. The normalized and transformed data can be used to assess the variance-expression level relationship in the data, to identify which method is best at stabilizing the variance across the mean for downstream visualization. The `plotMeanSD()` function wraps the output of different variance stabilizing methods (including the `varianceStabilizingTransformation()` and `rlog()` transformations from the `DESeq2`



**Figure 3. Gene count distributions.** Normalized count distributions are displayed as box plots (A) and density plots (B). The log<sub>2</sub> TMM-normalized counts per gene normalization method equates the overall expression levels of genes between samples under the assumption that a majority of them are not differentially expressed<sup>19</sup>. Therefore, by normalizing for total RNA expression by sample, we expect the spread of the log<sub>2</sub> TMM-normalized counts per gene to be similar for every sample. Sample classes (as set with the `interestingGroups` argument) are highlighted in different colors. Here, there is high similarity among the samples. This plot shows how all the samples have very similar gene expression profile what makes them comparable for the differential expression analysis.

package) and plots them with the `vsn` package's `meanSdplot()` function<sup>20</sup> (Figure 4). For the example data, the `vst` and `rlog` transformations work well with respect to reducing the noise of low expression genes, illustrated by a flatter distribution observed in these plots. The user may choose the desired transformation for their figures using the `normalized` argument.

```
plotMeanSD(bcb)
```

### Dispersion

Another plot that is important to evaluate when performing QC on RNA-seq data is the plot of dispersion versus the mean of normalized counts. For a good dataset, we expect the dispersion to decrease as the mean of normalized counts increases for each gene. The `plotDispEsts()` function provides easy access to model information stored in the `bcBioRNASEq` object, using the plotting code provided in the `DESeq2` package (Figure 5).

```
plotDispEsts(bcb)
```

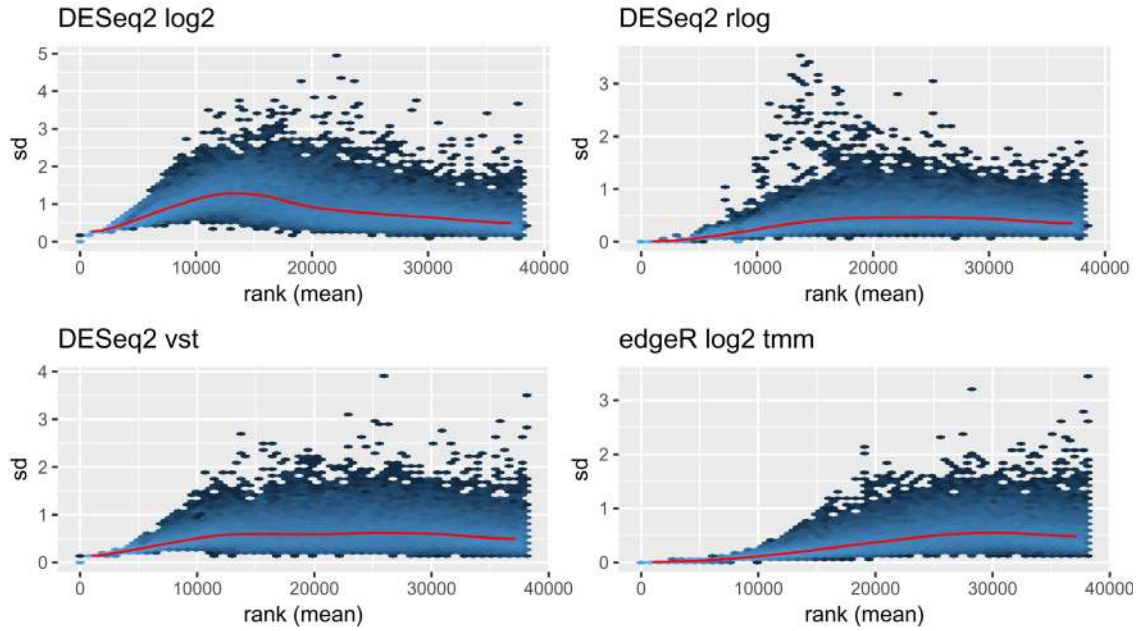
### Sample similarity within groups

The QC metrics assessed up to this point are performed to get a global assessment across the dataset and look for similar trends across all samples. However, often we have a dataset in which samples can be classified into groups and it is common to interrogate how similar replicates are to each other within those groups, and the relationship between groups. To this end, `bcBioRNASEq` provides functions to perform this level of QC with Inter-Correlation Analyses (ICA) and Principal Components Analyses (PCA) between samples. Furthermore, we can use the results of the PCA to identify covariates that correlate with principal components.

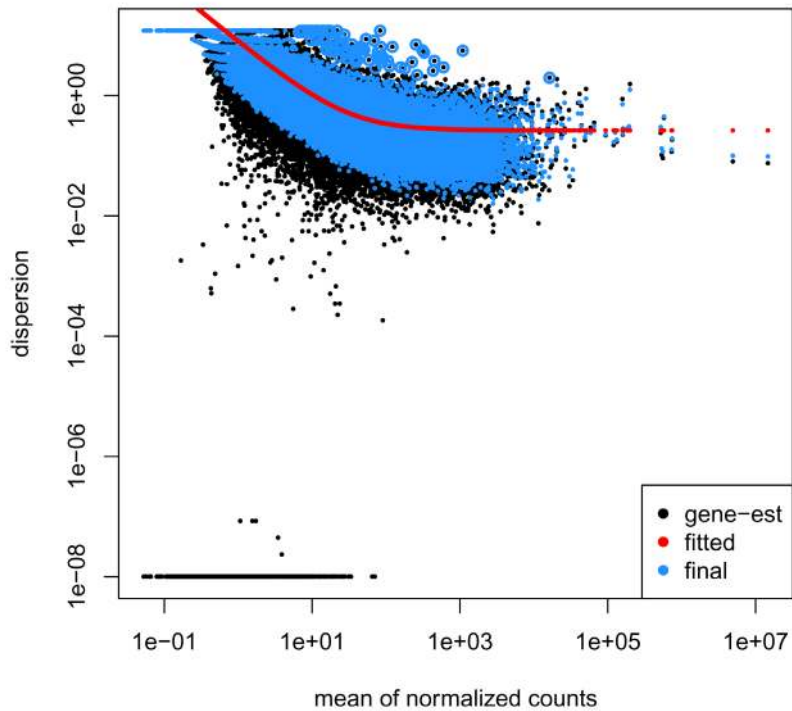
Since these analyses are based on variance measures, it is recommended that the variance stabilized (`vst`) or `rlog` transformed counts be used. Using these transformed counts minimizes large differences in sequencing depth and helps normalize all samples to a similar dynamic range. Simple logarithmic transformations of normalized count values tend to generate a lot of noise for low expressed genes, which can consequently dominate the calculations in the similarity analysis. An `rlog` transformation will shrink the values of low counts towards the genes' average across samples, without affecting the high expression genes.

### Inter-correlation analysis and hierarchical clustering

Inter-correlation analysis allows us to look at how closely samples are related to each other by first computing pair-wise correlations between expression profiles of all samples in the dataset and then clustering based on those



**Figure 4. Variance stabilizing transformations.** Plots showing the standard deviation of normalized counts using `log2()` (top left), `rlog()` (top right), `VST (varianceStabilizingTransformation())` (bottom left), and `TMM` (bottom right) transformations by `rank(mean)`. The red line denotes the running median estimator. As these panels show, the `rlog` and `VST` transformations greatly reduce the standard deviation and variance across the mean.



**Figure 5. DESeq2 dispersion plot.**

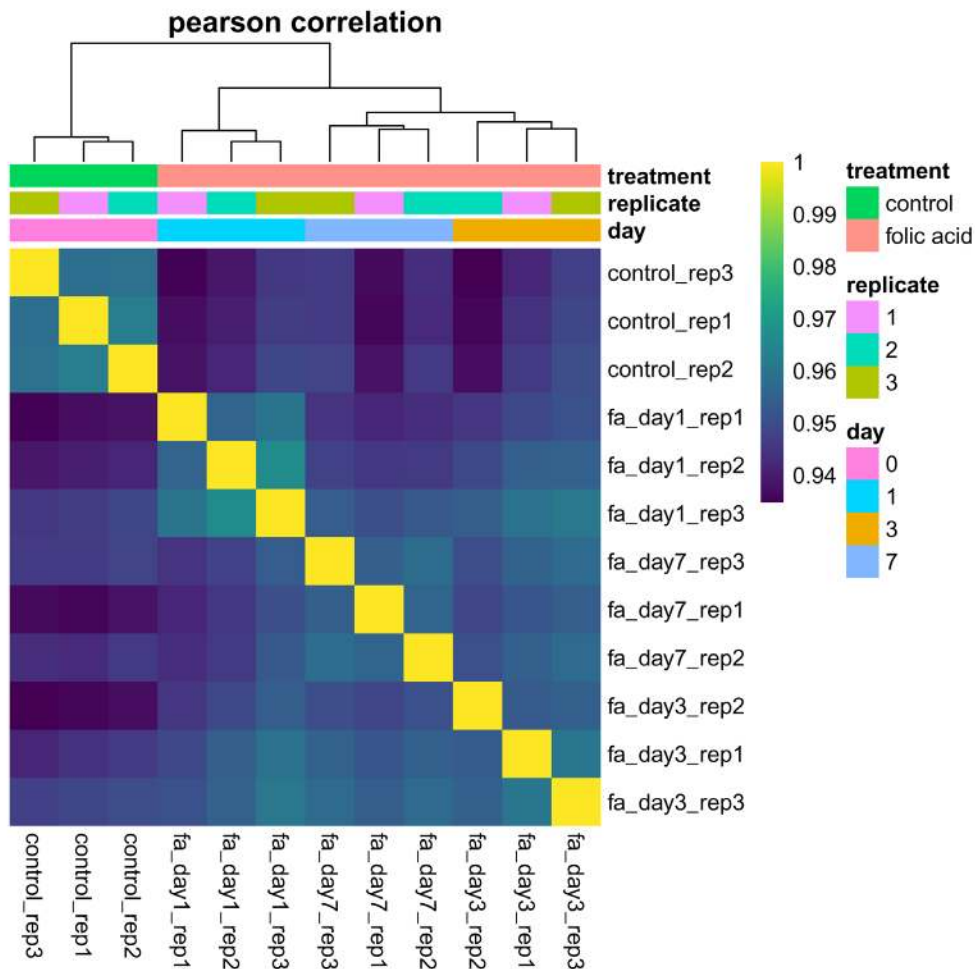
correlation values. Samples that are similar to one another will be highly correlated and will cluster together. We expect samples from the same group to cluster together (Figure 6), although this is not always the case. We can also identify potential outlier samples using a correlation heatmap, if there are samples that show low correlation with all other samples in the dataset. For more control over the graphic parameters of the correlation heatmap, other packages can be used to generate these plots by using the normalized data, accessed with `counts(bcb, normalized = "vst")`, as input. One example is the `pheatmap()` function from the `pheatmap` package<sup>21</sup>, which underlies this plot.

```
plotCorrelationHeatmap(bcb)
```

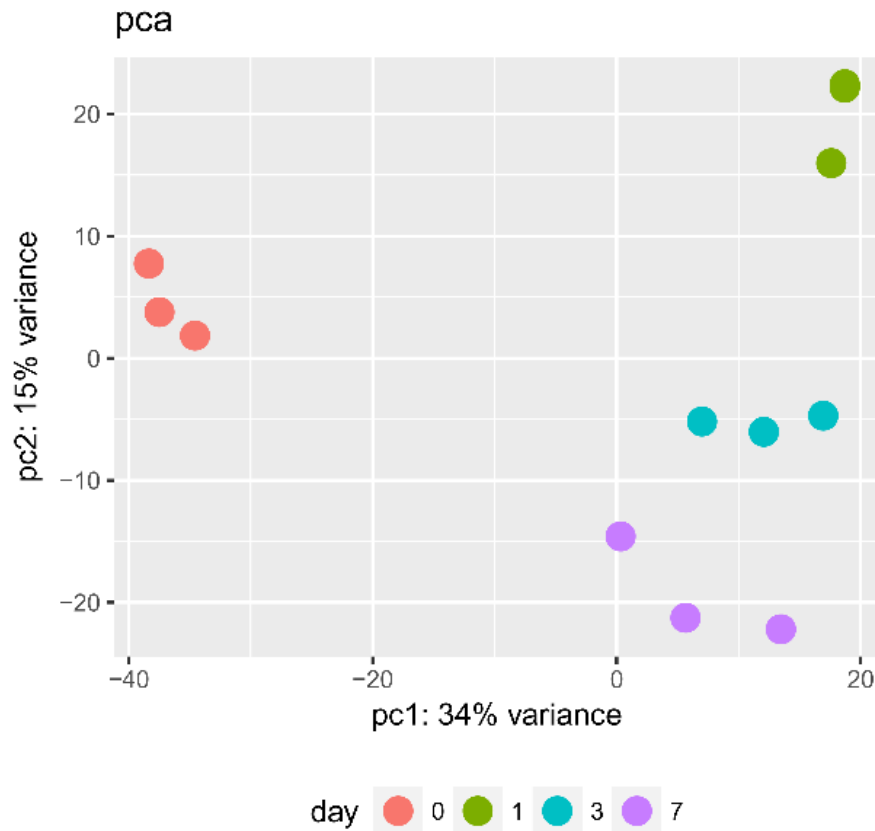
### Principal component analysis (PCA)

PCA is a multivariate technique that allows us to summarize the systematic patterns of variations in the data<sup>22</sup>. PCA takes the expression levels for genes and transforms them in principal component space, reducing each sample to a single point. It allows us to separate samples by expression variation, and identify potential sample outliers. The PCA plot is a valuable way to explore both inter- and intra-group clustering (Figure 7). As with the correlation heatmap plots, other packages can be used to generate these kinds of plots from the normalized data, which can be accessed with `counts(bcb, normalized = "vst")`.

```
plotPCA(bcb, label = FALSE)
```



**Figure 6. Sample correlations.** All pairwise sample Pearson correlations are shown. Correlations are clustered by both row and column, with sample classes (as set with the `interestingGroups` argument) highlighted across the top of the heatmap. Here, the sample classes cluster well, with the normal and day 1 samples showing the highest intra-group correlations.



**Figure 7. Principal component analysis.** The first two principal components of the gene expression dataset are plotted here for each of the samples. Sample classes (as set with the `interestingGroups` argument) are highlighted in different colors. Alternatively, sample labels can be added with the `"label = TRUE"` argument (not shown) to identify individual samples, which is particularly useful for identifying outliers. Here, we see good clustering of the samples by group with no apparent outliers.

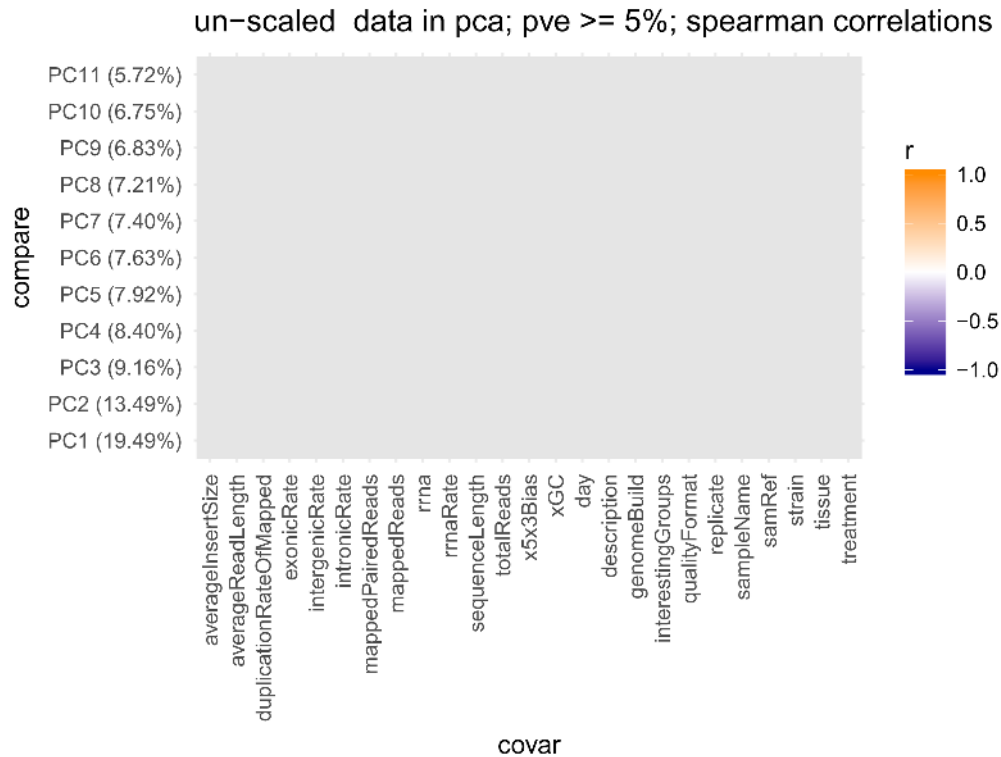
### Correlation of covariates with PCs

When there are multiple factors that can influence the results of a given experiment, it is useful to assess which of them is responsible for the most variance as determined by PCA (Figure 8). The `plotPCACovariates()` function passes transformed count data and metadata from the `bcbioRNASeq` object to the `degCovariates()` function of the `DEGreport` package<sup>23</sup>. This method adapts the method described by Daily *et al.* for which they integrated a method to correlate covariates with principal components values to determine the importance of each factor<sup>24</sup> (Figure 8). For the example data, no covariate shows a strong correlation with the PCs with  $FDR < 0.05$ , indicating that there is no need for correction for any covariates during the differential expression analysis.

```
plotPCACovariates(bcb, fdr = 0.1)
```

### QC discussion

With the QC figures, outlier samples can be identified and may be removed from the set of samples. For instance, in Figure 2B, if one samples shows a very low mapping rate ( $< 50\%$ ) compared to others, this could indicate an issue with the RNA material or the library preparation. The same reasoning applies to Figure 2C, that shows exonic mapping rates. In this case, low mapping rates may indicate DNA contamination if the rate is very low, and should be removed if only a minority of samples are affected. Moreover, Figure 8 can reveal clustering of samples correlated to some covariates, providing guidance on which variables to add to the formula during differential expression analysis.



**Figure 8. Principal component covariate correlations.** PCA is performed on transformed counts and correlations between principal components and the different metadata covariates are computed. Significant correlations (FDR < 0.1) are shaded from blue (anti-correlated) to orange (correlated), with non-significant correlations shaded in gray. Here, non of variables show a significant correlation with the any principal component of the data. Asterisks indicates p-value < 0.05.

### Differential expression analysis using DESeq2

Once the QC is complete and the dataset looks good, the next step is to identify differentially expressed genes (DEG). For this part of the workflow, we follow instructions and guidelines from the [DESeq2 vignette](#), using Salmon-derived abundance estimates imported with tximport. As previously noted, a **Differential Expression R Markdown** template is available with the `bcbioRNASeq` package for these steps.

The first step is to define the factors to include in the statistical analysis as a design formula. We chose to study the difference between the normal group and the day 7 group in our dataset. In our example, we have only one variable of interest; however, DESeq2 is able to model additional covariates. When additional variables are included, the last variable entered in the design formula should generally be the main condition of interest. More detailed instructions and examples are available in the DESeq2 vignette.

```
# DESeqDataSet
dds <- as(bcb, "DESeqDataSet")
design(dds) <- ~day
dds <- DESeq(dds)
vst <- varianceStabilizingTransformation(dds)
saveData(dds, vst)
```

### Alpha level (FDR) cutoffs

The results from DESeq2 include a column for the  $P$  values associated with each gene/test as well as a column containing  $P$  values that have been corrected for multiple testing (i.e. false discovery rate values). The multiple test correction method performed by default is the Benjamini Hochberg (BH) method<sup>25</sup>. Since it can be difficult to arbitrarily select an adjusted  $P$  value cutoff, the `alphaSummary()` function is useful for summarizing results for multiple adjusted  $P$  value or FDR cutoff values (Table 1).

**Table 1. Differentially expressed genes at different adjusted *P* value cutoffs.**

	0.1	0.05
LFC > 0 (up)	1521, 4%	1102, 2.9%
LFC < 0 (down)	1694, 4.5%	1282, 3.4%
outliers	632, 1.7%	632, 1.7%
low_counts	12883, 34%	12883, 34%
cutoff	(mean count < 5)	(mean count < 5)

```
alphaSummary(
  object = dds,
  contrast = c(
    factor = "day",
    numerator = "7",
    denominator = "0"
  ),
  alpha = c(0.1, 0.05)
)
```

### Differential expression analysis

Use the `results()` function to generate a `DESeqResults` object containing the output of the differential expression analysis. The desired BH-adjusted *P* value cutoff value is specified here with the `alpha` argument (< 0.05 shown).

```
res <- results(
  object = dds,
  name = "day_7_vs_0",
  alpha = 0.05
)
saveData(res)
```

### Mean average (MA) plot

The `plotMeanAverage()` function plots the mean of the normalized counts versus the log<sub>2</sub> fold changes for all genes tested (Figure 9).

```
plotMeanAverage(res)
```

### Volcano plot

The `plotVolcano()` function produces a volcano plot comparing significance (here the BH-adjusted *P* value) for each gene against the fold change (here on a log<sub>2</sub> scale) observed in the contrast of interest<sup>26</sup> (Figure 10). This function requires an Internet connection to map Ensembl gene IDs to gene names (symbols).

```
plotVolcano(res)
```

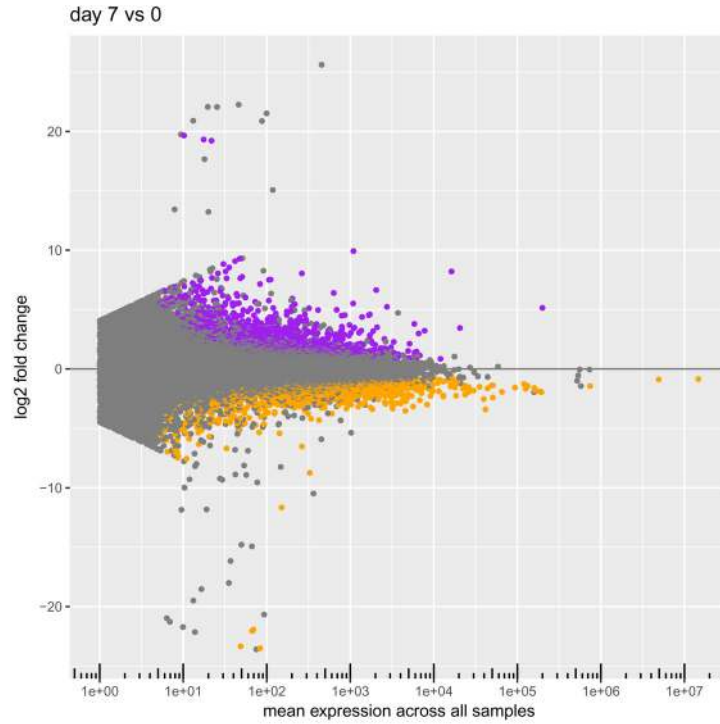
### Gene expression heatmap

The `plotDEGHeatmap()` function produces a gene expression heatmap for visualizing the expression of differentially expressed (DE) genes across samples. The heatmap shows only DE genes on a per-sample basis, using an additional log<sub>2</sub> fold change cutoff. By default, this plot is scaled by row and uses the `ward.D2` method for clustering<sup>27</sup>. The gene expression heatmap is a nice way to explore the consistency in expression across replicates or differences in expression between sample groups for each of the DE genes (Figure 11).

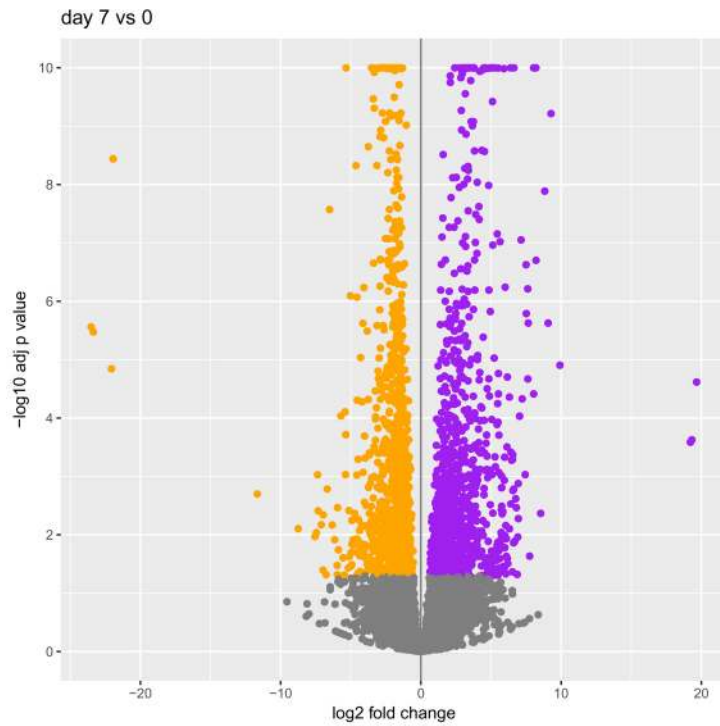
```
plotDEGHeatmap(results = res, counts = vst)
```

### Specific genes plot

In addition to looking at the overall results from the differential expression analysis, it is useful to plot the expression differences for a handful of the top differentially expressed genes. This helps to check the quality of the analysis by validating the expression for genes that are identified as significant. It is also helpful to visualize trends in expression change across the various sample groups (Figure 12). As `bcbioRNASeq` is integrated with the `DEGreport` package<sup>23</sup>, we can use `DEGreport`'s `degPlot()` function to view the expression of individual DE genes.

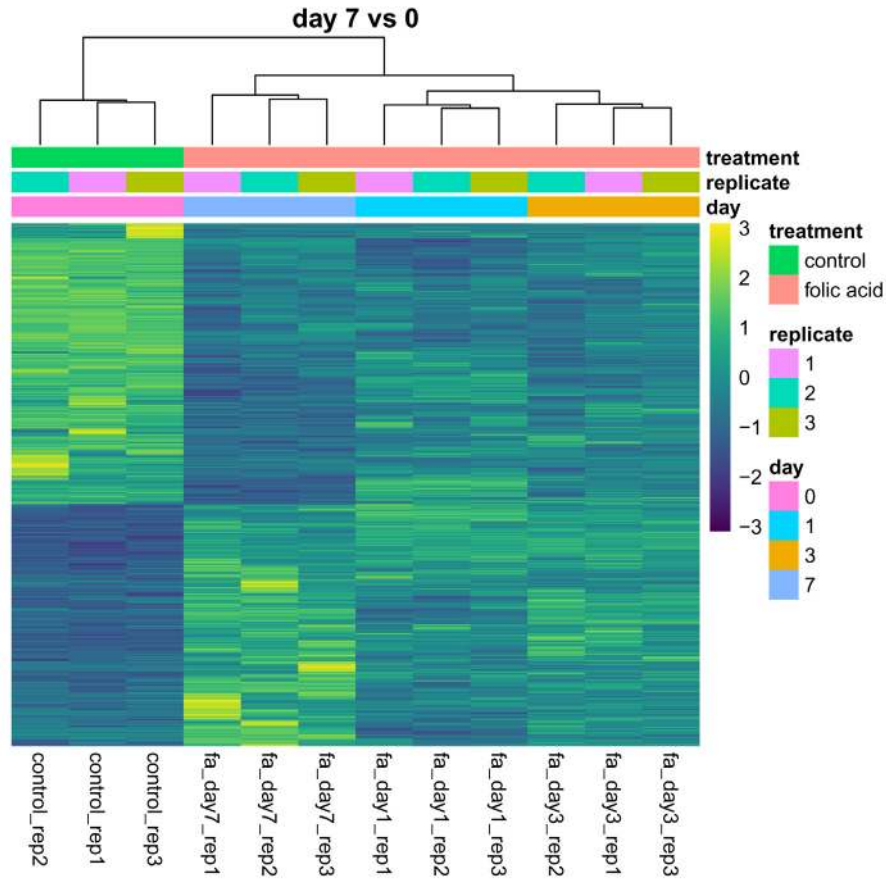


**Figure 9. MA plot.** Each point represents one gene, with mean expression levels across all samples plotted on the x-axis and the log<sub>2</sub> fold change observed in the contrast of interest on the y-axis. Significant differentially expressed genes (with an adjusted *P* value less than the adjusted cutoff *P* value we chose earlier) are colored red<sup>28</sup>.

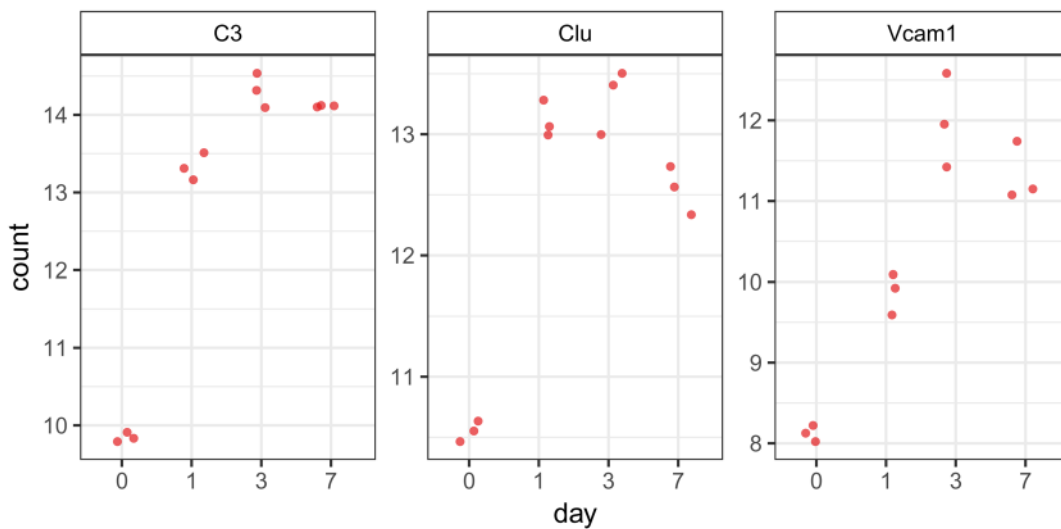


**Figure 10. Volcano plot.** This plot compares the amount of gene expression change (not strictly interpretable here as these fold changes were derived from an LRT test) to the significance of that change (here plotted as the  $-\log_{10}$  transformation of the multiple test adjusted *P* value), with each point representing a single gene. Options are available to highlight top gene candidates by shading (here, genes in the green shaded areas are bounded by a minimal fold change and  $-\log_{10}$  adjusted *P* value cutoffs) and by text labeling. The two (optional) marginal plots showing the distributions of the log<sub>2</sub> fold changes and negative log<sub>10</sub> adjusted *P* values are useful in assessing cutoff choices and trade-offs.





**Figure 11. Differentially Expressed Gene Heatmap.** The heatmap shows only DE genes on a per-sample basis, using an additional log2 fold change cutoff. By default, this plot is scaled by row (centered and scaled) and uses the `ward.D2` method for clustering<sup>27</sup>. Results are clustered by both row and column, with sample classes (as set with the `interestingGroups` parameter) highlighted across the top of the heatmap. Heatmaps are drawn internally with the `heatmap()` function of the `heatmap` package<sup>21</sup>.



**Figure 12. Individual gene expression patterns.** Gene expression patterns are shown for the top 3 (as selected in the function options) differentially expressed genes (by BH-adjusted *P* value). Normalized, transformed counts are shown for each replicate from each sample group; groups are set within the function options. Interestingly, even though our analysis compared day 7 to normal, all 3 genes show their greatest increases in gene expression at day 3, with some leveling off or relative decreases in expression at day 7.

```
degPlot(
  bcb,
  res = res,
  n = 3,
  slot = "vst",
  log = TRUE,
  ann = c("geneID", "geneName"),
  xs = "day"
)
```

### Detecting patterns

The full set of example data is from a time course experiment, as described previously. Up to this point, we have only compared gene expression between two time points (normal and day 7), but we can also analyze the whole dataset to identify genes that show any change in expression across the different time points. As recommended by DESeq2, the best approach for this type of experimental design is to perform a likelihood ratio test (LRT) to test for differences in gene expression between any of the sample groups in the context of the time course. More information about time-course experiments and LRT is available in the [DESeq2 vignette](#). This approach will yield a list of differentially expressed genes, but will not report how the expression is changing. Visualizing patterns of expression change amongst the significant genes is helpful in identifying groups of genes that have similar trends, which in turn can help determine a biological reason for the changes we observe ([Figure 13](#)). The [DEGreport](#) package includes the `degPatterns()` function, which is designed to extract and plot genes that have a similar trend across the various time points. More information about this function can be found in the [DEGreport package](#)<sup>23</sup>. Note that this function works only with significant genes; `significant()` returns the significant genes based on `log2FoldChange` and `padj` values (0 and 0.05 respectively, by default).

```
dds_lrt <- DESeq(dds, test = "LRT", reduced = ~1)
res_lrt <- results(dds_lrt)

ma <- counts(bcb, "vst")[significant(res, fc = 2), ]
res_patterns <- degPatterns(
  ma = ma,
  metadata = colData(bcb),
  time = "day",
  minc = 60
)
saveData(dds_lrt, res_lrt, res_patterns)
res_patterns[["plot"]]
```

The output of the `degPatterns()` function is the plot, as well as a list object that contains a `data.frame` with two columns – the “genes” and the corresponding “cluster” number. To extract genes from a specific cluster for further analysis, the base function `subset()` can be utilized as follows to obtain the list of genes from cluster 3:

```
subset(res_patterns[["df"]], cluster == 3, select = "genes")
```

### Summarize analysis

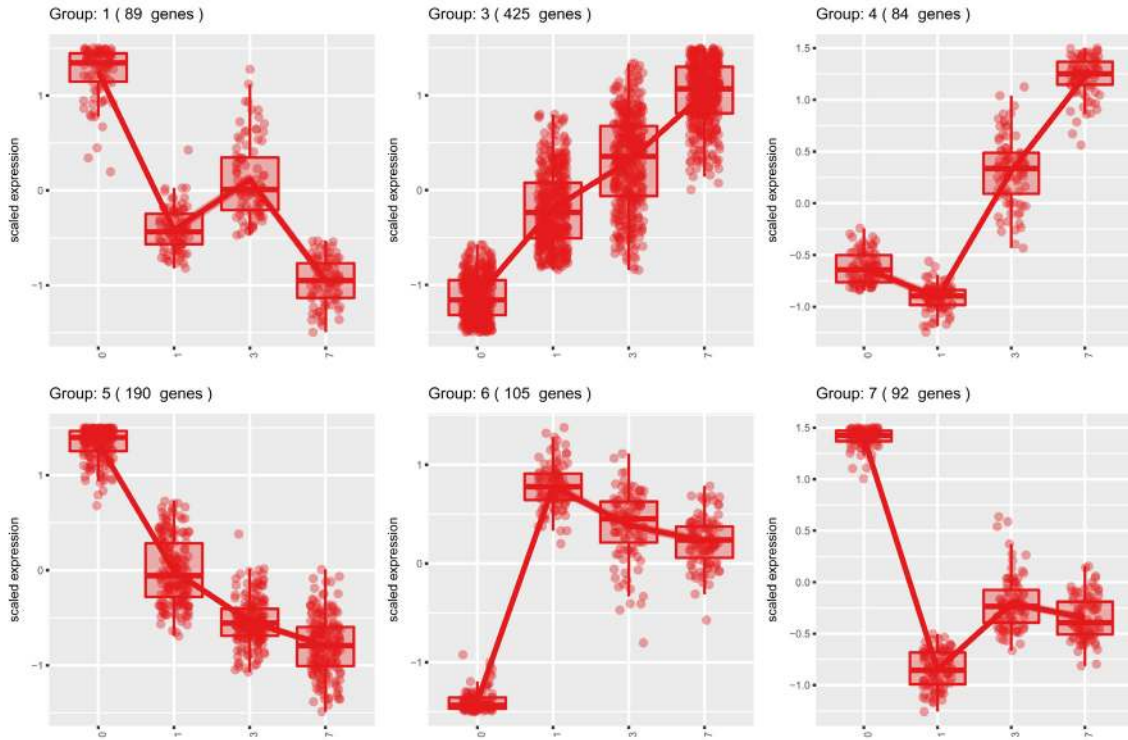
Finally, at the end of the analysis, a results table can be extracted containing the DE genes at a specified log fold change threshold. These results can be written to files in the specified output folder. In addition to the DE genes, a detailed summary and description of results and the output files are generated along with the cutoffs used to identify the significant genes. This function requires an Internet connection to map Ensembl gene IDs to gene names (symbols).

```
res_tbl <- resultsTables(res, lfc = 1)
```

### Top tables

Once the results table object is ready, the top up- and down-regulated genes can now be displayed. Here, we output the top 5 DE genes in each direction ([Table 2](#)).

```
topTables(res_tbl, n = 5)
```



**Figure 13. Clustered expression patterns.** Clustered and scaled expression patterns for the top 500 differentially expressed genes. Each group represents a gene expression pattern shared among different DE genes, with the number of groups determined by the expression correlation patterns of the groups. Box plots are shown for the expression patterns of each gene within the group to give a better idea of how well the groupings fit the expression data (DEGreport\_1.14.1 used for this plot.)

**Table 2. Top differentially expressed genes.**

ensgene	baseMean	log2FoldChange	padj
ENSMUSG00000024164	16284	8.21	1.96e-185
ENSMUSG00000029304	199735	5.14	4.25e-68
ENSMUSG00000027962	2744	5.23	1.92e-30
ENSMUSG00000040405	20543	3.44	4.18e-29
ENSMUSG00000022037	7752	3.21	3.80e-28
ensgene	baseMean	log2FoldChange	padj
ENSMUSG00000049152	13001	-2.30	1.23e-25
ENSMUSG00000011179	3090	-2.74	8.15e-21
ENSMUSG00000060002	23614	-2.14	4.43e-19
ENSMUSG00000062908	2594	-2.09	6.97e-19
ENSMUSG00000064370	183932	-1.87	7.14e-19

**File outputs**

Output files from this use case include the following gene counts files (output at the count normalization step):

- `normalized_counts.csv.gz`: Use to evaluate individual genes and/or generate plots. These counts are normalized for the variation in sequencing depth across samples.
- `tpm.csv.gz`: Transcripts per million, scaled by length and also suitable for plotting.
- `raw_counts.csv.gz`: Only use to perform a new differential expression analysis. These counts will vary across samples due to differences in sequencing depth, and have not been normalized. Do not use this file for plotting genes.

If desired, variance stabilized counts (e.g. `rlog`, `vst`) can also be included for future use in variance based plotting methods such as PCA. DEG tables containing the differential expression results from the analysis summary step are sorted by BH-adjusted  $P$  value, and contain the following columns:

- `geneID`: Ensembl gene identifier.
- `baseMean`: Mean of the normalized counts per gene for all samples.
- `log2FoldChange`: log2 fold change.
- `pvalue`: Wald test  $P$  value.
- `padj`: BH-adjusted Wald test  $P$  value (corrected for multiple comparisons; false discovery rate).

### Functional analysis

To gain greater biological insight into the list of DE genes, it is helpful to perform functional analysis. We provide the **Functional Analysis** R Markdown template, which contains code from the `clusterProfiler` package<sup>29</sup>. The functional analysis includes over-representation analysis to identify significantly enriched biological processes among a list of DE genes, and gene set enrichment analysis (GSEA) to identify pathways significantly enriched for genes with larger fold changes. The functional analysis results suggest genes/pathways that may be involved with the condition of interest; however, the results are not conclusive and all identified processes/pathways require experimental validation.

For over-representation analysis, `clusterProfiler` requires a vector of background genes tested for differential expression, a vector of significant DE genes, and a named vector of fold changes associated with each DE gene.

```
# Generate the gene IDs for the DE list of genes and the background list of
genes
library(clusterProfiler)
all_genes <- rownames(res) %>%
  .[!is.na(res[["padj"]])] %>%
  as.character()
sig_genes <- significant(
  object = res,
  fc = 1,
  padj = 0.05
)

# Generate fold change values for significant results
sig_results <- as.data.frame(res)[sig_genes, ]
fold_changes <- sig_results$log2FoldChange
names(fold_changes) <- rownames(sig_results)
```

The `enrichGO()` function takes the significant gene list, the background gene list, and the ontology to test as input to perform statistical enrichment analysis of gene ontology (GO) terms using hypergeometric testing.

```
# Run GO enrichment analysis
ego <- enrichGO(
  gene = sig_genes,
  OrgDb = "org.Mm.eg.db",
  keyType = "ENSEMBL",
  ont = "BP"
  universe = all_genes,
  qvalueCutoff = 0.05,
  readable = TRUE
)
```

The GO enrichment analysis output is an S4 class object named `enrichResult`. The results can be extracted from this object using the `slot()` accessor function:

```
ego_summary <- slot(ego, "result") %>%
  as_tibble() %>%
  camel()
```

The results table contains the following columns:

- `id`: GO identifier.
- `description`: GO process.
- `geneRatio`: number of DE genes associated with GO process / total number of DE genes.
- `bgRatio`: number of background genes associated with GO process / total number of background genes.
- `pvalue`: hypergeometric test  $P$  value.
- `padj`: BH-adjusted hypergeometric test  $P$  value (corrected for multiple comparisons; false discovery rate).
- `qvalue`: FDR-adjusted hypergeometric test  $P$  value (corrected for multiple comparisons; positive false discovery rate).
- `geneID`: gene symbols of all DE genes associated with GO process.

There are a variety of plots that can be used to explore the enriched processes using the `enrichResult` object, including the dot plot, enrichment GO plot, and category netplot.

The dot plot shows the number of DE genes associated with the top 25 GO terms (size) and the  $P$ -adjusted values for these terms. The order of GO terms is based on the gene ratio (number of significant genes associated with the GO term / total number of significant genes) (Figure 14).

```
# Dotplot of top 25
dotplot(ego, showCategory = 25)
```

The enrichment GO plot shows the relationship between the top 25 most significantly enriched GO terms, by grouping similar terms together (Figure 15).

```
# Enrichment plot of top 25
enrichMap(ego, n = 25, vertex.label.cex = 0.5)
```

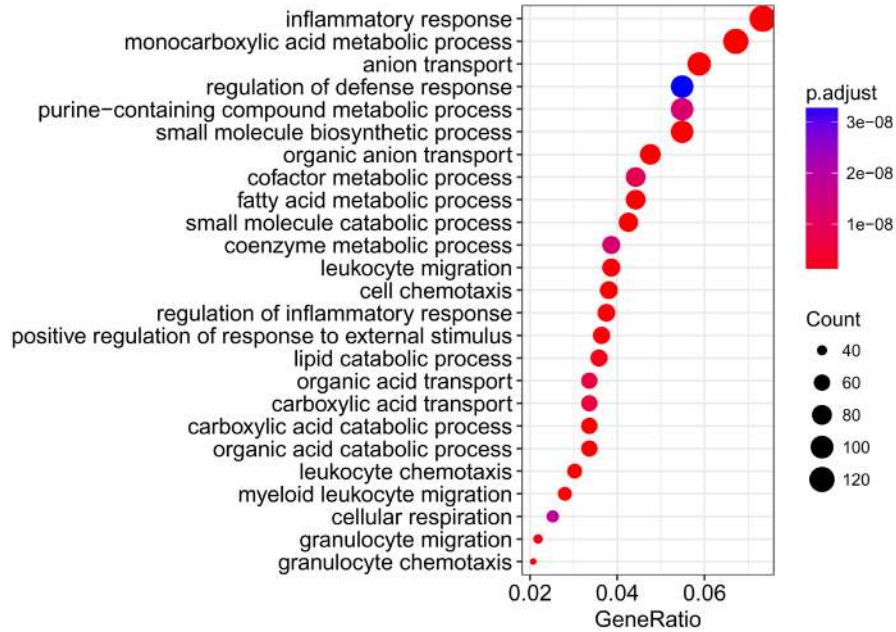
The category netplot shows the relationships between the genes associated with the top five most significant GO terms and the fold changes of the significant genes associated with these terms (color) (Figure 16). The size of the GO terms reflects the  $P$  values of the terms, with the more significant terms being larger.

```
# Cnet plot for top 5 most significant GO processes
cnetplot(
  x = ego,
  showCategory = 5,
  foldChange = fold_changes,
  vertex.label.cex = 0.5
)
```

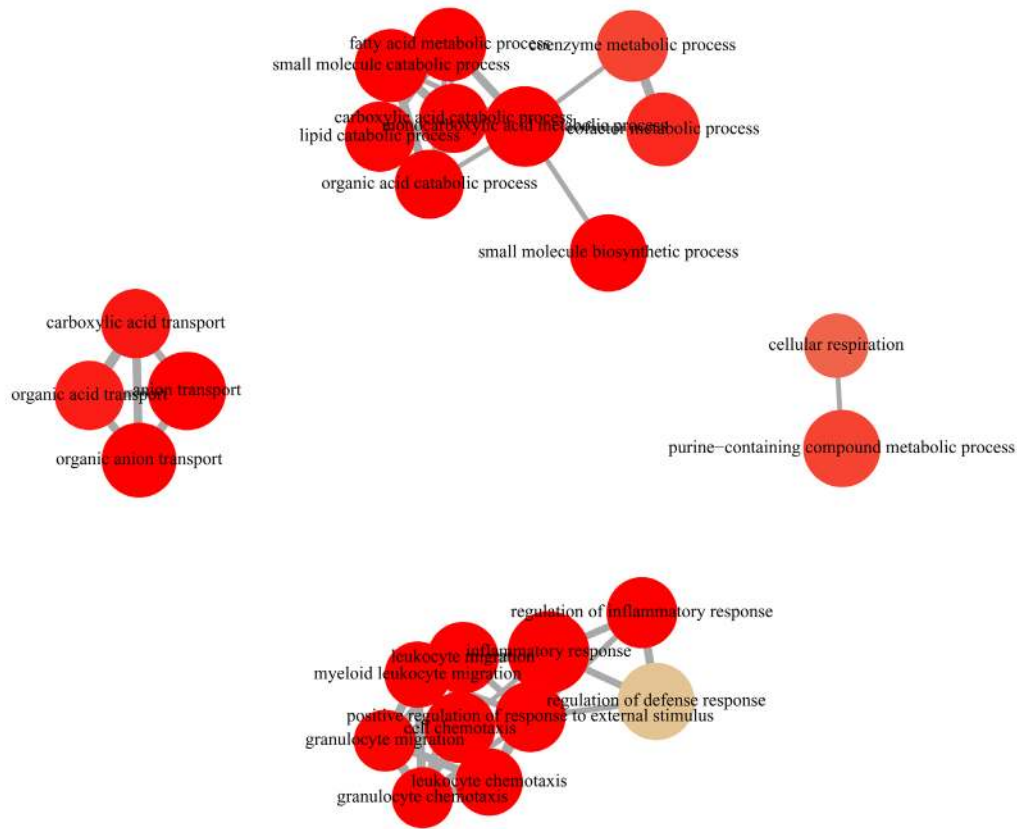
The **Functional Analysis** template also includes code for performing over-representation analysis with GO terms and code to perform GSEA analysis using KEGG and GO terms. The final step in the template is the visualization of the genes and corresponding fold changes associated with the GSEA enriched pathways using the `pathview` package.

## R Markdown Templates

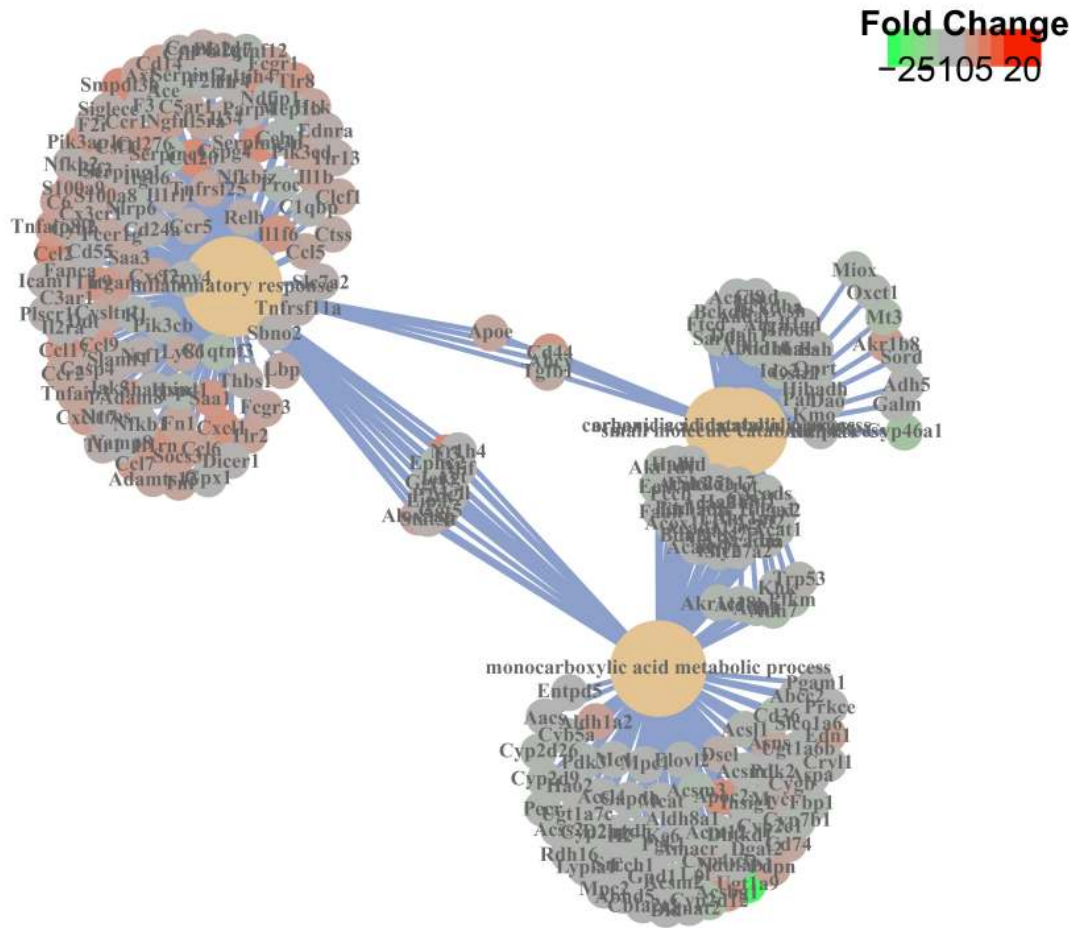
To facilitate analyses and compile results into a report format, we have created easy-to-use R Markdown templates that are accessible in `RStudio`<sup>30,31</sup>. Once the `bcbioRNASeq` package has been installed, you can find these templates under `FILE -> NEW FILE -> R MARKDOWN... -> FROM TEMPLATE`. There are three main templates for RNA-seq: (1) **Quality Control**, (2) **Differential Expression**, and (3) **Functional Analysis**. You may need to restart `RStudio` to see the templates. It is recommended that users run the reports in the order described above, as there may be



**Figure 14. Enriched GO terms: dot plot.** The 25 GO processes with the largest gene ratios are plotted in order of gene ratio. The size of the dots represent the number of genes in the significant DE gene list associated with the GO term and the color of the dots represent the *P*-adjusted values (BH).



**Figure 15. Enriched GO terms: enrichment GO plot.** The top 25 most significantly enriched GO terms (by *P*-adjusted values) are shown with similar terms grouped together. The color represents the *P* values relative to the other displayed terms (brighter red is more significant) and the size of the terms represents the number of genes that are significant from the significant genes list.



**Figure 16. Enriched GO terms: category netplot.** The top five most significant GO terms (*P*-adjusted values) are plotted and connected with lines to associated DE genes. The fold changes of the significant genes associated with these terms are represented by color. The size of the GO terms reflects the *P* values of the terms, with the more significant terms being larger.

functions that depend on data generated from the previous report. If you are not using RStudio, you can create new documents based on the templates using the `rmarkdown::draft()` function:

```
library(rmarkdown)
draft(
  file = "quality_control.Rmd",
  template = "quality_control",
  package = "bcbioRNASeq"
)
draft(
  file = "differential_expression.Rmd",
  template = "differential_expression",
  package = "bcbioRNASeq"
)
draft(
  file = "functional_analysis.Rmd",
  template = "functional_analysis",
  package = "bcbioRNASeq"
)
```

The instructions above will create an R Markdown file from each of the templates. Each file begins with a YAML header, followed by sub-sections containing code chunks and some relevant text and/or a sub-heading to describe that step of the analysis. Each R Markdown file takes as input the `bcbioRNASeq` object, such that various functions from the package can be run on the data stored within the object to output figures, tables and carefully formatted results.

Note you can add more text, headings and code chunks to the body of the R Markdown files to customize the reports as desired. Before rendering the file into a report you will want to run the `prepareRNASeqTemplate()` function in order to obtain the accessory files necessary for a fully working template.

```
library(bcbioRNASeq)
prepareRNASeqTemplate()
```

Finally, the main analysis parameters need to be specified in the YAML section on the top part of each document. For this example we assume R objects are stored in a `data` subdirectory. Edit the following parameters in the **Quality Control** template:

```
params:
  bcb_file: "data/bcb.rda"
  data_dir: "data"
  results_dir: "results/quality_control"
```

`bcb.rda` refers to the `bcbioRNASeq` object.

In the YAML section on the top of the **Differential Expression** template, modify the following parameters:

```
params:
  bcb_file: "data/bcb.rda"
  design: !r formula(~day)
  contrast: !r c("day", "7", "0")
  alpha: 0.05
  lfc_threshold: 0
  data_dir: "data"
  results_dir: "results/differential_expression"
  dropbox_dir: "NULL"
```

In the YAML section on the top of the **Functional Analysis** template, modify the following parameters (note that `clusterProfiler` and `pathview` must be installed):

```
params:
  dds_file: "data/dds.rda"
  res_file: data/res.rda
  organism: "Mus musculus"
  go_class: "BP"
  alpha: 0.05
  lfc_threshold: 0
  data_dir: "data"
  results_dir: "results/functional_analysis"
```

`res.rda` refers to a `DESeqResults` object from the `DESeq2` package.

The downloaded accessory files will be saved to your current working directory and should be kept together with your main analysis R Markdown files that were generated from the templates. These accessory files include: a) `_output.yaml`, to specify the R Markdown render format; b) `_header.Rmd` and c) `_footer.Rmd`, to add header/footer sections to the report; d) `_setup.R`, to fill in some of the parameters related to figures and rendering format; and e) `bibliography.bib`, BibTeX file for citations.



## Conclusions

Here we describe bcbioRNASeq, a Bioconductor package that provides functionality for quality assessment and differential expression analysis of RNA-seq experiments. This package supplements the bcbio community project, as it takes the output from automated bcbio RNA-seq runs as input, allowing for the data to be stored in a special S4 object that can easily be accessed for various steps of the RNA-seq workflow downstream of bcbio. Built as an open source project, bcbio is a well-supported and documented platform for effectively using current state-of-the-art RNA-seq methods. Taken together, bcbio and bcbioRNASeq provide a full framework for rapidly and accurately processing RNA-seq data. The package also provides a set of configurable templates to generate comprehensive HTML reports suitable for biological researchers. With the use of R Markdown, all steps of the analysis are fully configurable and traceable.

We provide a full set of instructions for using bcbioRNASeq, including an example use case that demonstrate all of the main functionality. Quality control (pre- and post-quantification), model fitting, differential expression, and functional analysis provide a comprehensive set of metrics for evaluating the robustness of the RNA-seq results. Methods such as hierarchical clustering, principal components analysis, and time point analysis allow for an interactive examination of the data's structure. Collectively, the workflow we describe can help researchers to identify true biological signal from technical noise and batch effects when analyzing RNA-seq experiments.

## Software and data availability

Current source code:

<https://github.com/hbc/bcbioRNASeq>

Archived source code (v0.2.4), as at time of publication:

<http://doi.org/10.5281/zenodo.1256861><sup>32</sup>

Workflow code:

<https://github.com/hbc/bcbioRNASeq/tree/f1000v2>

License: MIT

The data used in the use case can be accessed from NCBI GEO using accession [GSE65267](#).

## Competing interests

No competing interests were disclosed.

## Grant information

The author(s) declared that no grants were involved in supporting this work.

## Acknowledgments

Thanks to Mike Love of the Department of Biostatistics and Department of Genetics at the University of North Carolina at Chapel Hill for providing advice regarding RNA-seq differential expression and the use of the DESeq2 and tximport packages. We thank Laura Lin of the Harvard Chan Bioinformatics Core for reviewing the code showed here.

## References

1. Wang Z, Gerstein M, Snyder M: **RNA-Seq: a revolutionary tool for transcriptomics**. *Nat Rev Genet*. 2009; **10**(1): 57–63. [PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
2. Love MI, Anders S, Kim V, *et al.*: **RNA-Seq workflow: gene-level exploratory analysis and differential expression [version 2; referees: 2 approved]**. *F1000Res*. 2016; **4**. [PubMed Abstract](#) | [Publisher Full Text](#)
3. Huber W, Carey VJ, Gentleman R, *et al.*: **Orchestrating high-throughput genomic analysis with bioconductor**. *Nat Methods*. 2015; **12**(2): 115–121. [PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
4. Andrews S: **FastQC: a quality control tool for high throughput sequence data**. 2010. [Reference Source](#)
5. Martin M: **Cutadapt removes adapter sequences from high-throughput sequencing reads**. *EMBnet J*. 2011; **17**(1): 10–12. [Publisher Full Text](#)
6. Ewing B, Hillier L, Wendl MC, *et al.*: **Base-calling of automated sequencer traces using phred. I. accuracy assessment**. *Genome Res*. 1998; **8**(3): 175–85. [PubMed Abstract](#) | [Publisher Full Text](#)
7. Ewing B, Green P: **Base-calling of automated sequencer traces using phred. II. error probabilities**. *Genome Res*. 1998; **8**(3): 186–94. [PubMed Abstract](#) | [Publisher Full Text](#)
8. Patro R, Duggal G, Love MI, *et al.*: **Salmon provides fast and bias-aware quantification of transcript expression**. *Nat Methods*. 2017; **14**(4): 417–419. [PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)

9. Dobin A, Davis CA, Schlesinger F, *et al.*: **STAR: ultrafast universal RNA-seq aligner**. *Bioinformatics*. 2013; **29**(1): 15–21.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
10. Liao Y, Smyth GK, Shi W: **featurecounts: an efficient general purpose program for assigning sequence reads to genomic features**. *Bioinformatics*. 2014; **30**(7): 923–30.  
[PubMed Abstract](#) | [Publisher Full Text](#)
11. Okonechnikov K, Conesa A, García-Alcalde F: **Qualimap 2: advanced multi-sample quality control for high-throughput sequencing data**. *Bioinformatics*. 2016; **32**(2): 292–94.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
12. Ewels P, Magnusson M, Lundin S, *et al.*: **MultiQC: summarize analysis results for multiple tools and samples in a single report**. *Bioinformatics*. 2016; **32**(19): 3047–3048.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
13. Soneson C, Love MI, Robinson MD: **Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences [version 2; referees: 2 approved]**. *F1000Res*. 2016; **4**: 1521.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
14. Robert C, Watson M: **Errors in RNA-Seq quantification affect genes of relevance to human disease**. *Genome Biol*. 2015; **16**: 177.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
15. Morgan M, Obenchain V, Hester J, *et al.*: **SummarizedExperiment: SummarizedExperiment container**. 2017.  
[Reference Source](#)
16. Love MI, Huber W, Anders S: **Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2**. *Genome Biol*. 2014; **15**(12): 550.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
17. Craciun FL, Bijol V, Ajay AK, *et al.*: **RNA Sequencing Identifies Novel Translational Biomarkers of Kidney Fibrosis**. *J Am Soc Nephrol*. 2016; **27**(6): 1702–1713.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
18. Li P, Piao Y, Shon HS, *et al.*: **Comparing the normalization methods for the differential analysis of illumina high-throughput RNA-Seq data**. *BMC Bioinformatics*. 2015; **16**: 347.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
19. Robinson MD, Oshlack A: **A scaling normalization method for differential expression analysis of RNA-seq data**. *Genome Biol*. 2010; **11**(3): R25.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
20. Huber W, von Heydebreck A, Sültmann H, *et al.*: **Variance stabilization applied to microarray data calibration and to the quantification of differential expression**. *Bioinformatics*. 2002; **18** Suppl 1: S96–104.  
[PubMed Abstract](#) | [Publisher Full Text](#)
21. Kolde R: **heatmap: Pretty Heatmaps**. 2015.  
[Reference Source](#)
22. Jolliffe I: **Principal component analysis**. Wiley Online Library, 2002.  
[Publisher Full Text](#)
23. Pantano L: **DEGreport: Report of DEG analysis**. 2017.  
[Publisher Full Text](#)
24. Daily K, Ho Sui SJ, Schriml LM, *et al.*: **Molecular, phenotypic, and sample-associated data to describe pluripotent stem cell lines and derivatives**. *Sci Data*. 2017; **4**: 170030.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
25. Benjamini Y, Hochberg Y: **Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing**. *J R Stat Soc Series B Stat Methodol*. 1995; **57**(1): 289–300.  
[Reference Source](#)
26. Cui X, Churchill GA: **Statistical tests for differential expression in cDNA microarray experiments**. *Genome Biol*. 2003; **4**(4): 210.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
27. Ward JH Jr: **Hierarchical grouping to optimize an objective function**. *J Am Stat Assoc*. 1963; **58**(301): 236–244.  
[Publisher Full Text](#)
28. Dudoit S, Yang YH, Callow MJ, *et al.*: **Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments**. *Stat Sin*. 2002; **12**(1): 111–139.  
[Reference Source](#)
29. Yu G, Wang LG, Han Y, *et al.*: **clusterprofiler: an R package for comparing biological themes among gene clusters**. *OMICS*. 2012; **16**(5): 284–287.  
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
30. Allaire JJ, Cheng J, Xie Y, *et al.*: **rmarkdown: Dynamic Documents for R**. 2017.  
[Reference Source](#)
31. RStudio Team: **RStudio: Integrated Development Environment for R**. RStudio, Inc., Boston, MA. 2016.  
[Reference Source](#)
32. Steinbaugh M, Pantano L, Kirchner R, *et al.*: **hbc/bcbioRNASeq: bcbioRNASeq 0.2.4 (Version v0.2.4)**. *Zenodo*. 2018.  
[Data Source](#)

# Open Peer Review

Current Peer Review Status:  

---

## Version 2

Reviewer Report 13 July 2018

<https://doi.org/10.5256/f1000research.16697.r35236>

© 2018 Risso D. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

 **Davide Risso** 

Division of Biostatistics and Epidemiology, Department of Healthcare Policy and Research, Weill Cornell Medicine, New York, NY, USA

The updated manuscript is well written and describe a very useful R package for the quality control, exploration and statistical analysis of RNA-seq data.

The changes in the new version make this a better resource, by adding a useful section on Functional Analysis and by fully leveraging the SummarizedExperiment and GRanges Bioconductor infrastructures.

However, the following (mostly minor) issues are still outstanding and the authors need to address them for me to approve without reservations:

1. As pointed out in my previous review and by the other reviewer, the authors refer to `bcbioRNASeq` as a Bioconductor package, when it is not part of the Bioconductor project. I again suggest that the authors submit their package to Bioconductor. Until then, this should be referred to as a R package.
2. Although the authors addressed my comment on the description of the plots, some figures remain a bit opaque. For instance, what is the interpretation of Figure 2F? Similarly, the range of the x-axis in Figure 3A makes the plot hard to interpret.
3. Figure 8 is not rendered correctly in the PDF and HTML versions of the article.
4. There are some inconsistencies between the code in the article and the code in the Github repository. For instance in Figure 13 the code in the text says "vst", while in the repository it uses "rlog". It is unclear which is shown in the figure. In addition, when describing `plotPCACovariates`, in the text the authors say that the used  $FDR < 0.05$  but in the code they use 0.1. Please double check all the code.

5. The function `enrichMap` is deprecated. The authors should use the function `emaplot`. In addition, it should be made clear that these functions are part of the DOSE package.

**Competing Interests:** No competing interests were disclosed.

**I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.**

Author Response 14 Jul 2018

**Lorena Pantano Rubino**, Harvard T.H. Chan School of Public Health, Boston, USA

Thanks for the review statement. We appreciate the time to review our work and the comments that will make this article more reproducible and accurate. I'll leave the first author to lead this round of reviewing to address all the comments you shared with us.

**Competing Interests:** No competing interests were disclosed.

Reviewer Report 29 June 2018

<https://doi.org/10.5256/f1000research.16697.r35237>

© 2018 Soneson C. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



**Charlotte Soneson** 

Institute of Molecular Life Sciences, University of Zurich (UZH), Zürich, Switzerland

The second version of this manuscript has been extended and now contains additional descriptions of the object class used by the bcBioRNASeq package as well as the functional analysis aspects. The text is well written and easy to follow, and the package is likely to be a welcome addition to the toolbox for users working with the bcBio pipeline to process their RNA-seq data. However, I think there are still aspects that can be improved, in particular with respect to the reproducibility of the provided results and figures.

The package is still referred to as a "Bioconductor package" in the Introduction, but as far as I can see, it is not submitted to Bioconductor. If the authors don't intend to submit it there, I would recommend describing it as an "R package" instead, as was previously pointed out.

Judging from the session info provided in the GitHub repository, the code in the workflow was run with an old version of R (3.4.1) and Bioconductor. One of the used functions (enrichMap, from the DOSE package) is deprecated in the current Bioconductor release, and may be removed in the near future. Moreover, running the commands in the manuscript (or the ones provided in the workflow.R script in the GitHub repository) do not always generate the same figures as are being

shown in the article (and in addition, the code in the manuscript is different from the one in the workflow.R script). The differences are mostly minor, but this still reduces the reproducibility of the workflow.

Minor points:

- The "counts" assay is described as containing "raw counts, generated by salmon and imported with tximport". However, if I understand correctly this slot may also contain other types of "count-like" abundances generated by tximport (depending on the value used for countsFromAbundance).
- The legend of Figure 2B says that all samples "are well within recommended ranges, having [...] almost 90% of reads mapping". However, there are several samples with mapping rate below 70% in the figure.
- The legend of Figure 3 says that sample classes (defined by the interestingGroups argument) are indicated in different colors, but in the figure, each sample has a different color.
- In the "Genomic context" part of the QC section, the text suggests that the results of plotExonicMappingRate() and plotIntronicMappingRate() show the exonic and intronic mapping reads as a fraction of the total number of reads ("Ideally, at least 60% of total reads should map to exons"). However, I believe the plots show the fraction of the mapped reads that map to exons and introns, respectively. This could be clarified.
- In the "Number of genes detected" section, Figure 2D-E should be Figure 2E-F.
- In the "Correlation of covariates with PCs, the text says "PCs with FDR < 0.05" while the code uses `fdr = 0.1`.
- The code in the "Specific genes plot" requires a "library(DEGreport)" to run.
- The third code chunk in the "Functional analysis" section doesn't run (missing a comma after `ont = "BP"`), the fourth code chunk requires a "library(tibble)" to run, and the sixth code chunk similarly requires a "library(DOSE)".
- In the legends of Figures 14-16: P-adjusted values -> adjusted P-values.

**Competing Interests:** No competing interests were disclosed.

**I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.**

Author Response 03 Jul 2018

**Lorena Pantano Rubino**, Harvard T.H. Chan School of Public Health, Boston, USA

Thanks for the excellent work at reviewing our work. We'll work on fix all minus issues and

decide a time for submitting a Bioconductor. We'll report back here with the comments.

Thanks again for all your time.

**Competing Interests:** No competing interests were disclosed.

---

## Version 1

Reviewer Report 12 December 2017

<https://doi.org/10.5256/f1000research.13084.r27759>

© 2017 Risso D. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

? **Davide Risso** 

Division of Biostatistics and Epidemiology, Department of Healthcare Policy and Research, Weill Cornell Medicine, New York, NY, USA

The authors present `bcbioRNASeq`, an R package for the QC and differential expression analysis of RNA-seq data. The package takes as input the output of the `bcbio` software, not presented in this work.

`bcbio` is a community driven resource that handles the data processing of several high-throughput sequencing applications, ranging from variant calling to ChIP-seq and RNA-seq analyses. The `bcbioRNASeq` package focuses on RNA-seq.

Overall, I enjoyed the article and I think it represents a nice resource for practitioners looking to perform standardized RNA-seq analyses of several datasets and for `bcbio` users that want to perform high-level statistical analyses after RNA-seq preprocessing.

The article is well written and provides a reproducible example that walks the reader through the proposed pipelines. I am glad to report that (except for some minor issues reported below) I was able to fully reproduce the analysis. The following points will hopefully help the authors improve the manuscript.

1. Currently, the package does not appear to be available through Bioconductor, but only through the authors' Github. Are the authors planning to submit it to Bioconductor? I definitely encourage them to do so, as a way to manage package versions and dependencies (see next point). If not, I would ask the authors to refer to `bcbioRNASeq` as an R package rather than a Bioconductor package.
2. The authors do not specify the versions of the packages needed for their workflow to work. Although the DESCRIPTION file of the package provides such information, adding it to the

manuscript would allow readers to reproduce the workflow example. This would be automatically taken care of if the package was part of a Bioconductor release. In addition, at the beginning of the paper, the authors load the packages "DESeq2" and "DEGreport". Aren't these packages in the Import: field of the DESCRIPTION file of bcbioRNASeq?

3. S4 object. Is it really needed to store all the normalized data in the S4 object? This could lead to a huge object when the analysis is run on hundreds of samples. Since scaling normalization is very fast wouldn't it be better to compute normalized data on the fly and only store the raw and tpm data computed by tximport and featureCounts? On a related note, wouldn't it be better for the authors to store the featureCounts data in an additional element of the assays() slot and provide coercion methods from their object to the DESeqDataSet and tximport objects? Is it really needed to store both tximport results in the assays slot and in the bcbio slot? Overall, I have the feeling that the object is needlessly big and this could lead to a big memory footprint.
4. Are the plots based on raw or normalized data? If the latter, which normalization / transformation is used by default? How does the user change it?
5. Interpretation of the plots. Although the authors describe the plots in generic terms, it would be useful to explain them more specifically referring to the actual example analysis. For instance, which of the three transformation of Figure 3 is best for the example data? Is Figure 1F a typical pattern or does it uncover something unusual with the data? Same for Figure 7.
6. A better metric to highlight the difference in distribution among samples (Figure 2) is the Relative Log Expression (RLE) plot. The authors might want to include such plot to their already excellent array of QC plots.
7. What to do if the data fail the QC step? The authors present all their QC plots but then move on by simply stating "Once the QC is complete and the dataset looks good [...]". What if the data do not look good? It would be good to advice on what to do (just as a discussion perhaps). E.g., there could be outlying samples to be removed or batch effects to be accounted for in the model.

#### Minor issues:

- The last sentence of the first paragraph of the Introduction seems to indicate that the R package actually runs the QC tools, while these are run in bcbio and the results are loaded in the R package for exploration.
- First chunk of R code (page 4 of the PDF version of the paper): at my first read I was wondering how to get the data to run this command. It should be made clearer that this is only meant to show the syntax and is not part of the runnable example.
- The link to the bcb.rda file is broken.
- ``normalized <- counts(dds, "normalized")`` This line doesn't work. Did the author mean `normalized=TRUE?`
- ``tpm <- tpm(txi)`` This line doesn't work. Did the author mean ``tpm <- counts(dds, normalized="tpm")``?
- Please describe `writeCounts()`.

- Please consider removing the "+" in the R chunks (e.g., at page 13 of the PDF) so that readers could run the code by copying and pasting into an R session.
- In statistics, ICA is often used to refer to Independent Component Analysis, so the authors may want to avoid this acronym for the correlation analysis to avoid confusion.
- In my RStudio session the plotPCACovariates() plot did not work (I couldn't see any points but just a gray background).
- YAML parameters for the Functional Analysis Rmarkdown: I believe that the line "res" should be "resFile".

**Is the rationale for developing the new software tool clearly explained?**

Yes

**Is the description of the software tool technically sound?**

Yes

**Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?**

Yes

**Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?**

Yes

**Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?**

Yes

**Competing Interests:** No competing interests were disclosed.

**I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.**

Reviewer Report 20 November 2017

<https://doi.org/10.5256/f1000research.13084.r27761>

© 2017 Soneson C. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



**Charlotte Soneson** 

Institute of Molecular Life Sciences, University of Zurich (UZH), Zürich, Switzerland

This article describes bcbioRNASeq, an R package for analysis of RNA-seq data for which gene expression estimation and quality assessment have been done via the bcbio pipeline. The package



contains functions for generating plots and performing downstream analysis as well as Rmarkdown templates for generating stand-alone reports of the quality control, differential expression analysis and functional analysis steps.

In general, I find the article well written and easy to follow. The included functionality covers the most important parts of a typical RNA-seq analysis, and the package is likely to be a useful tool for bcbio users. Also, the provided templates are easy to extend with additional analyses if necessary. Following are some suggestions for improvement of specific parts of the article.

1. It would be good to indicate any dependencies on particular versions of R/Bioconductor/specific packages, and preferably also give the session info for the session with which the manuscript was generated. I ran the code with R v3.4.2 (Bioconductor v3.6, DESeq2 v1.18.1, bcbioRNASeq v0.1.2), and while most of the code executes correctly, there are some lines that do not. In particular:

```
> normalized <- counts(dds, "normalized") ## dds doesn't exist
> tpm <- tpm(txi) ## txi doesn't exist
> writeCounts()(raw, normalized, rlog, tpm) ## formatting error
> resTbl <- resultsTables(res, lfc = 1, write = TRUE, dir = deDir) ## deDir doesn't exist
```

Other lines only execute properly with a functional internet connection, which could perhaps be indicated in the text:

```
> plotVolcano(res)
> resTbl <- resultsTables(res, lfc = 1, write = TRUE, dir = ".")
```

Finally, in some places, executing the provided code does not seem to generate the same results as in the article. More precisely:

```
> plotPCACovariates(bcb, fdr = 0.1)
does not generate Figure 7 (the asterisks are missing).
> alphaSummary(dds, contrast = c(factor = "group", numerator = "day7", denominator = "normal"))
does not generate the numbers in Table 1.
> resPatterns <- degPatterns(counts(bcb, "rlog")[significants(res, ), metadata = colData(bcb), time = "group", col = NULL)
does not generate Figure 12. As a consequence, subsetting to cluster 8 also doesn't work.
> topTables(resTbl, n = 5)
does not generate the numbers in Table 2.
```

2. If I read correctly, the "tximport" slot of the bcbio object contains length-scaled TPMs, not aggregated transcript counts from tximport. This should be made clearer in the article. Is there an explicit choice in the bcbio pipeline that determines the type of count-scale abundances that are generated?

3. It would be useful to indicate in the beginning of the article where the metadata is stored in the output from bcbio. I.e., where should one look for the values available to supply to the "interestingGroups" argument of loadRNASeq()?

4. In the object description, it would be worth explaining a bit more clearly how the values contained in the slot "-tmm: trimmed mean of M-values, calculated by edgeR" were calculated.

5. In the object description, devtools::sessionInfo() should be devtools::session\_info()

6. In the Use case, "Also accessible with tpm()" should presumably be under point 3.
7. Regarding the visual thresholds in the plots (warning thresholds and optimal values), how are the default values determined? Are they fixed, or do they depend on some characteristics of the data? And are they particularly suitable for data generated under specific conditions, in specific organisms or with particular protocols? I am also wondering whether the use of the word "optimal" to designate one threshold may cause confusion. For example, if the "optimal" total number of reads is ~20M, and the "optimal" mapping rate is ~90%, it may not be immediately clear how one should interpret values exceeding (and potentially far away from) these "optimal" values. Finally, is there a reason for only having one line in the exonic and intronic mapping rate plots, but both lines in the other plots?
8. In the "Model fitting" section, it is suggested that it is important to evaluate the variance stabilizing performance of different transformations before the differential expression analysis. However, the transformed data are never actually used for the DE analysis (which is performed with DESeq2). Thus, it should be clarified how the results obtained here are used to inform the downstream analysis.
9. For the QC and differential expression analysis, the article outlines the analysis steps in detail. However, the functional analysis is only described through the existence of an Rmarkdown template. It would be nice to have at least part of the functional analysis also explained and written out in the article.
10. In `resPatterns[["plot"]]`, the x-axis labels are not centered under the respective boxplots.
11. The package is referred to as a "Bioconductor package", but as far as I can see it is not (yet) in Bioconductor.
12. It seems that zero counts are excluded from the plots in Figure 2. This could be clarified in the text.
13. In some places, "library" is used in the place of "package".
14. For the `degPatterns()` call, it is indicated that it is "CPU intensive". It might be useful to indicate approximately \*how\* CPU/time intensive, since all other steps in the workflow execute quickly.
15. In the code blocks, it would be easier if non-code characters like `>` and `+` were removed, so that the code could be directly copied into an R session.

**Is the rationale for developing the new software tool clearly explained?**

Yes

**Is the description of the software tool technically sound?**

Yes

**Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?**

Partly

**Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?**

Partly

**Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?**

Yes

**Competing Interests:** No competing interests were disclosed.

**I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.**

---

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact [research@f1000.com](mailto:research@f1000.com)

**F1000Research**