broadSeq Report

Rishi Das Roy, Rintu Kutum   Gautam Ahuja, Santosh Adhikari

Note: All code was compiled using `R-4.3.3`, `Bioconductor version 3.18` and `Rtools43`. The operating system used were Windows 11 Version 23H2 and Ubuntu-22.04

We successfully installed the library on `R-4.3.3`. I worked on a Windows-based system, while Santosh worked on an Ubuntu system. Below is a detailed report of each error encountered during the installation and execution of the `broadSeq` package.

1 Installation - Windows 11

1.1 BiocManager

In windows, we started by installing the `BiocManager` package as follows:

```
if (!require("BiocManager")) {
  install.packages("BiocManager")
}
```

This was installed successfully without any errors.

1.2 devtools

Next, we installed the `devtools` package:

```
install.packages('devtools')
```

This was installed successfully without any errors.

1.3 broadSeq

Before installing `broadSeq`, we first had to install `rtools43` on Windows. We followed the steps mentioned here for Windows. We then attempted to install the development version of `broadSeq` with `vignettes` as follows:

```
devtools::install_github("dasroy/broadSeq", build_vignettes = TRUE)
```

However, this resulted in an error.
To resolve this, we first installed the `BiocStyle` package from `BiocManager`:

```
BiocManager::install('BiocStyle')
```

After this, we were able to install and browse the `vignettes`.

We then installed `broadSeq` from `BiocManager` as follows:

```
BiocManager::install("broadSeq")
```

This was installed successfully without any errors.

1.4 Other Libraries

We installed the `ggplot2` and `ggpubr` packages as mentioned in the `vignettes`. We are now ready to run the `broadSeq` library.

```
install.packages('ggplot2')
install.packages('ggpubr')
```

1.5 Vignettes

We also `knitted` the vignette seperately by downloading the `.Rmd` files and the images. It was compiled without error.

2 Installation - Ubuntu-22.04

2.1 BiocManager

In Linux system with Ubuntu- 22.04, the installation for the `BiocManager` package is as follows:

```
if (!require("BiocManager")) {
  install.packages("BiocManager")
}
```

This was installed successfully without any errors.

2.2 devtools

Next, we proceeded to install the `devtools` package:

```
install.packages('devtools')
```

This was installed successfully without any errors.

2.3 broadSeq

While we attempted to install `broadSeq`, We attempted to install the development version of `broadSeq` with `vignettes` as follows:

```
devtools::install_github("dasroy/broadSeq", build_vignettes = TRUE)
```

However, this resulted in an error stating that BiocStyle library is missing.
To resolve this, we first installed the `BiocStyle` package from `BiocManager`:

```
BiocManager::install('BiocStyle')
```

Again while we attempted to install `broadSeq`, we got the error message stating that `ERROR: dependency ‘clusterProfiler’ is not available for package ‘broadSeq’` along with the following `warning message`.

```
Warning messages:
1: In i.p(...) : installation of package ‘ggtree’ had non-zero exit
↪  status
2: In i.p(...) :
  installation of package ‘enrichplot’ had non-zero exit status
3: In i.p(...) :
  installation of package \\
  ‘clusterProfiler’ had non-zero exit status
4: In i.p(...) :
  installation of package \\
  ‘/tmp/RtmpvfE7g9/file164647e8fbdac/broadSeq_0.99.3.tar.gz’ had
  ↪  non-zero exit status
```

We used the following command to resolve the encountered errors:

```
BiocManager::install("clusterProfiler")
BiocManager::install("ggtree")
```

After this, we were able to install and browse the `vignettes`.

We then installed `broadSeq` from `BiocManager` as follows:

```
BiocManager::install("broadSeq")
```

Before the final installation was over we also got the following warning message:
```
Bioconductor version 3.18 (BiocManager 1.30.23), R 4.3.3 (2024-02-29)
Warning message:
package ‘broadSeq’ is not available for Bioconductor version ’3.18’
```
This was then installed successfully without any errors.

2.4 Other Libraries

We installed the `ggplot2` and `ggpubr` packages as mentioned in the `vignettes`. We are now ready to run the `broadSeq` library.

```
install.packages('ggplot2')
install.packages('ggpubr')
```

3 Testing

The broadSeq package simplifies the process of including many Bioconductor packages for RNA-seq data and evaluating their performance.

We followed the instructions in the vignettes to verify that we could run all the commands correctly.

3.1 Imports

We import both the broadSeq and ggplot2 libraries.

```r
library(broadSeq)
library(ggplot2)
```

On loading the broadSeq package we get the following mesage:

```
Loading required package: dplyr

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag


...

The following objects are masked from ‘package:matrixStats’:

    anyMissing, rowMedians
```

There were no errors encountered.

3.2 Reading the data

The broadSeq takes a SummarizedEnperiment as an input format and gives the output as a data.frame. We load the gene expression data in .rds format as shown:

```r
se <- readRDS(system.file("extdata","rat_vole_mouseSE_salmon.rds",
 ↪  package = "broadSeq"))
SummarizedExperiment::assayNames(se)
```

The SummarizedExperiment output was as follows:

```
[1] "counts"       "abundance"    "avgTxLength"    "vst"
```

3.2.1 Sample Metadata

We then explored the sample metadata:

```r
as.data.frame(colData(se)) %>%
  dplyr::count(stage, species) %>%
  tidyr::spread(stage, n)
se$stage <- factor(se$stage, levels = c("Bud", "Cap", "Late Cap",
 ↪  "Bell"))
```

There were no errors encountered and we got expected results.

3.2.2 Filtering Low Expression Genes

We filtered out low expression genes and visualized the distribution to help in improving the signal-to-noise ratio. The first step involves visualizing the distribution of gene expression counts followed by applying a filter to remove genes with low expression levels.

```r
assays(se)[["counts"]][,5] %>%
  ggpubr::ggdensity(y = "count") +
  ggplot2::geom_vline(xintercept = 10) +
  ggplot2::scale_x_log10()

keep <- (assays(se)[["counts"]] >= 3) %>% rowSums() >= 5
table(keep)
```

There were no errors encountered and we got expected results.

3.3 Normalization

Normalization was performed using `CPM` and `TMM` methods:

```r
# CPM
se <- broadSeq::normalizeEdgerCPM(se, method = "none", cpm.log = TRUE)
SummarizedExperiment::assayNames(se)
# TMM
se <- broadSeq::normalizeEdgerCPM(se, method = "TMM", cpm.log = FALSE)
SummarizedExperiment::assayNames(se)
```

There were no errors encountered and we got expected results. Accessing the normalized data:

```r
assays(se)[["counts"]][1:5, 1:5]
assays(se)[["TMM"]][1:5, 1:5]
assays(se)[["logCPM"]][1:5, 1:5]
```

There were no errors encountered and we got expected results.

3.4 Transformation

We applied various transformations to the data:

```r
# VST
se <- broadSeq::transformDESeq2(se, method = "vst")

# Normalized counts transformation
se <- broadSeq::transformDESeq2(se, method = "normTransform")

# rlog
se <- broadSeq::transformDESeq2(se, method = "rlog")
SummarizedExperiment::assayNames(se)
```

There were no errors encountered and we got expected results.

3.4.1 Comparison of Transformations

We compared the transformations visually:

```r
p <- broadSeq::sampleAssay_plot(se[, se$species=="Mouse" ],
                                assayName = "counts", fill = "stage",
                                yscale = "log2")+ rremove("x.text")
p1 <- broadSeq::sampleAssay_plot(se[, se$species=="Mouse"],
                                 assayName = "vst", fill = "stage")+
                                  ↪  rremove("x.text")
p2 <- broadSeq::sampleAssay_plot(se[, se$species=="Mouse"],
                                 assayName = "TMM", fill = "stage",
                                 yscale = "log10")+ rremove("x.text")
p3 <- broadSeq::sampleAssay_plot(se[, se$species=="Mouse"],
                                 assayName = "logCPM", fill = "stage")+
                                  ↪  rremove("x.text")
ggarrange(p,p1,p2,p3, common.legend = TRUE, labels = c("A","B","C"))
```

There were no errors encountered and we got expected results.
If the `vsn` package is available, we perform mean-variance plots:

```r
if (requireNamespace("vsn", quietly = TRUE)) {
  library("vsn")
  x <- meanSdPlot(log2(assays(se[, se$species == "Rat"])[["counts"]] +
   ↪  1), plot = FALSE)
  print(x$gg + ggtitle(label = "log2(n+1)"))
  x <- meanSdPlot(assays(se[, se$species == "Rat"])[["vst"]], plot =
   ↪  FALSE)
  print(x$gg + ggtitle(label = "Vst"))
  x <- meanSdPlot(assays(se[, se$species == "Rat"])[["logCPM"]], plot =
   ↪  FALSE)
  print(x$gg + ggtitle(label = "logCPM"))
}
```

There were no errors encountered and we got expected results.

3.5 Visualization of Gene Expression

Visualizations of gene expression were generated using various assays:

```r
broadSeq::assay_plot(se, feature = c("Shh"),
                     assayNames = c("counts","logCPM","vst","TMM"),
                     x = "stage", fill="species", add="dotplot", palette = "npg")
broadSeq::genes_plot(se,
                     features = c("Shh","Edar"),
                     facet.by = "symbol",
                     x = "stage", assayName = "vst", fill="species", palette = "jco")
```

Using predefined or custom color palettes based on journals:

```r
jco <- broadSeq::genes_plot(se[,se$species == "Mouse"],
                            features = c("Shh"), facet.by = "symbol", assayName =  "logCPM",
                            x = "stage",  fill="stage", add="dotplot", xlab = "",
                            title = "Journal of Clinical Oncology", palette = "jco")

npg <- broadSeq::genes_plot(se[,se$species == "Mouse"],
                            features = c("Shh"), facet.by = "symbol",assayName =  "logCPM",
                            x = "stage", fill="stage", add="dotplot", xlab = "",
                            title = "Nature Publishing Group", palette = "npg")

aaas <- broadSeq::genes_plot(se[,se$species == "Mouse"],
                             features = c("Shh"), facet.by = "symbol", assayName = "logCPM",
                             x = "stage", fill="stage", add="dotplot", xlab = "",
                             title = "Science", palette = "aaas")

nejm <- broadSeq::genes_plot(se[,se$species == "Mouse"],
                             features = c("Shh"), facet.by = "symbol", assayName = "logCPM",
                             x = "stage", fill="stage", add="dotplot", xlab = "",
                             title = "New England Journal of Medicine",palette = "nejm")

ggarrange(jco+ggpubr::rotate_x_text(), npg+ggpubr::rotate_x_text(),
          aaas+ggpubr::rotate_x_text(),nejm+ggpubr::rotate_x_text(),
          nrow = 1, common.legend = TRUE,legend = "none",
          labels = c("A","B","C","D")) %>%
  annotate_figure( top = text_grob("Color palette"))
```

There were no errors encountered and we got expected results.

3.6 Quality Control with Clustering

Quality control was performed using various clustering methods:

```
# MDS plot
broadSeq::plot_MDS(se, scaledAssay = "vst", ntop=500,
                   color = "species", shape = "stage",
                   ellipse=TRUE, legend = "bottom")
head(rowData(se))
# Hierarchical clustering
p_vst <- broadSeq::plotHeatmapCluster(
  se,
  scaledAssay = "vst",
  annotation_col = c("species", "stage"),
  annotation_row = c("Class","gene_biotype"),
  ntop = 30, show_geneAs = "symbol",
  cluster_cols = TRUE, cluster_rows = FALSE,
  show_rownames = TRUE, show_colnames = FALSE,
  main = "Top 30 variable gene vst"
)
```

3.6.1 PCA plot

```
# prcompTidy
computedPCA_logCPM <- broadSeq::prcompTidy(se, scaledAssay = "logCPM",
 ↪  ntop = 500)
computedPCA_vst <- broadSeq::prcompTidy(se, scaledAssay = "vst", ntop =
 ↪  500)
# Plot
# logCPM
plotAnyPC(computedPCA = computedPCA_logCPM,
          x = 1, y = 2, color = "species", shape = "stage",
          legend = "bottom")
# VST
pca_vst <- plotAnyPC(computedPCA = computedPCA_vst,
                     x = 2, y = 3,  color = "species", shape = "stage",
                     legend = "bottom")
# Other PCs
computedPCA_vst$eigen_values %>%
  dplyr::filter(var_exp >= 2) %>%
  ggbarplot(x="PC",y="var_exp", label = TRUE, label.pos = "out")
pca_vst_2_3 <-plotAnyPC(computedPCA = computedPCA_vst,
```

```r
                        x = 2, y = 3,
                        color = "species", shape = "stage", legend = "bottom")
# Gene loading
computedPCA_vst %>% broadSeq::getFeatureLoadRanking(keep = c("symbol","Class")) %>% head()
computedPCA_vst$loadings %>% top_n(5,abs(PC2)  ) %>% dplyr::select(gene,PC2)
pca_vst_loading <- computedPCA_vst %>%
  broadSeq::getFeatureLoadRanking(keep = c("symbol","Class"), topN = 50, pcs=1:10) %>%
  dplyr::count(Class, PC) %>%
  ggbarplot(
    x = "PC", y = "n", fill = "Class",
    legend = "bottom", palette = c("red","blue","orange","purple","white","grey")
  )
# Biplot
pca_vst_bi <- broadSeq::biplotAnyPC(computedPCA = computedPCA_vst,
                                    x = 1, y = 2, genesLabel = "symbol",
                                    color = "species", shape = "stage",
                                    legend = "bottom")
ggarrange(
  ggarrange(pca_vst_bi+ggtitle(label =  ""),
            pca_vst_2_3+ggtitle(label =  ""), common.legend = TRUE),
  pca_vst_loading, nrow = 2)
# User defined genes
biplotAnyPC(computedPCA = computedPCA_vst,x = 2, y = 3,
            color = "species", shape = "stage",
            genes= computedPCA_vst$loadings %>%
              top_n(5,abs(PC3)) %>% pull(gene),
            genesLabel = "symbol")

## Plot progression gene "Shh"
biplotAnyPC(computedPCA = computedPCA_vst,x = 2, y = 3,
            color = "species", shape = "stage",
            genes=c("Shh"),
            genesLabel = "symbol")
```

There were no errors encountered and we got expected results.

3.7 Compare Differential Expression

We load the data again

```r
se <- readRDS(system.file("extdata","rat_vole_mouseSE_salmon.rds", package = "broadSeq"))
se <- se[,colData(se)$species == "Mouse"]
```

```
# Gene information
head(rownames(se))
head(rowData(se))
# Sample information
head(colData(se))
table(colData(se)$stage)
```

3.7.1 Differential Expression

```
# Function pattern
result_Noiseq <-
  use_NOIseq(se = se,
             colData_id = "stage", control = "Bud", treatment = "Cap",
             rank = TRUE,
             r = 10)
head(result_Noiseq)
pg <- broadSeq::genes_plot(se, x = "stage", assayName =  "counts",
                           features = result_Noiseq %>%
                           ↪  dplyr::filter(rank <5) %>% rownames(),
                           fill="stage", facet.by = "symbol",
                           palette="jco", add =
                           ↪  "dotplot")+rotate_x_text()
pg_sc <- ggscatter(result_Noiseq, x="Bud_mean", y="Cap_mean",color =
↪  "prob")+
  scale_x_log10()+scale_y_log10()
pg+pg_sc
# Available methods
?use_limma_trend(se, colData_id, control, treatment, rank = FALSE, ...)
↪  # limma
?use_limma_voom(se, colData_id, control, treatment, rank = FALSE, ...)
?use_edgeR_exact(se, colData_id, control, treatment, rank = FALSE, ...)
↪  # edgeR
?use_edgeR_GLM(se, colData_id, control, treatment, rank = FALSE, ...)
?use_deseq2(se, colData_id, control, treatment, rank = FALSE, ...) #
↪  deseq2
?use_DELocal(se, colData_id, control, treatment, rank = FALSE, ...) #
↪  DELocal
?use_NOIseq(se, colData_id, control, treatment, rank = FALSE, ...)  #
↪  noiseq
?use_EBSeq(se, colData_id, control, treatment, rank = FALSE, ...) #
↪  EBSeq
?use_SAMseq(se, colData_id, control, treatment, rank = FALSE, ...) #
↪  samseq
```

There were no errors encountered and we got expected results.

3.7.2 Compare DE results

```r
funs <- list(limma_trend = use_limma_trend, limma_voom = use_limma_voom,
             edgeR_exact = use_edgeR_exact, edgeR_glm = use_edgeR_GLM,
             deseq2 = use_deseq2,
             DELocal = use_DELocal, noiseq = use_NOIseq,
             EBSeq = use_EBSeq)
multi_result <- broadSeq::use_multDE(
  se = se,
  deFun_list = funs, return.df = TRUE,
  colData_id = "stage", control = "Bud", treatment = "Cap",
  rank = TRUE)
head(multi_result)
colnames(multi_result)
# Similarity of methods
clusters <- multi_result %>% dplyr::select(ends_with("rank")) %>% t()
↪   %>% dist() %>% hclust()
plot(clusters,main =  "distance: Euclidean")
# Plots
# Volcano
multi_result %>% broadSeq::volcanoPlot(
  pValName = "deseq2_padj",
  lFCName = "deseq2_log2FoldChange",
  labelName = "symbol",
  palette = "lancet" ,
  selectedLabel =
    multi_result %>% dplyr::arrange(deseq2_padj) %>% pull(symbol) %>%
     ↪   head()
)
multi_result %>% broadSeq::volcanoPlot(
  pValName = "deseq2_padj",
  lFCName = "deseq2_log2FoldChange",
  labelName = "symbol",
  palette = c("purple","orange","grey"),
  selectedLabel = list(criteria = "(`x` > 5 | `x` < -2) & (`y` > 10)")
) +xlim(-7.5,7.5)
```

There were no errors encountered and we got expected results.

When ploting, there was a reoccuring warning at multiple steps:

```
Warning message:
In seq_len(computedPCA$pc_scores[, dottedArg$shape]) :
  first element used of 'length.out' argument
```