Analysing data from pooled genetic sequencing screens using edgeR

Matt Ritchie and Oliver Voogd

15 October 2014 (last updated 7 October 2020)

Contents

1 Introduction

This document is intended to provide a how to guide for the analysis of pooled genetic sequencing screen data using the *edgeR* package. Refer to the main article (Dai *et al.* 2014) for a summary of this analysis pipeline, and be sure to cite this article if you make use of the workflow we describe in your own research.

Pooled genetic sequencing screens employ either RNA interference using short hairpin RNAs (shRNAs) or genetic mutation using single guide RNAs (sgRNAs) with the CRISPR-Cas9 system to perturn gene function. This approach has been successfully employed by a number of groups (Zuber *et al.* 2011, Sullivan *et al.* 2012, Bassik *et al.* 2013, Shalem *et al.* 2014 and Wang *et al.* 2014). Depending on the biological question of interest, typically two or more cell populations are compared either in the presence or absence of a selective pressure, or as a time course before and after a selective pressure is applied. Gain of shRNA/sgRNA representation within a pool suggests that perturbing gene function confers some sort of advantage to a cell. Similarly, genes whose loss of function is disadvantageous may be identified through loss of shRNA/sgRNA representation. Screening requires a library of shRNA/sgRNA constructs in a lentiviral or retroviral vector backbone that is used to generate a pool of virus for transducing cells of interest. The relative abundance of these shRNAs/sgRNAs in transduced cells is then quantified by PCR amplification of proviral integrants from genomic DNA using primers designed to amplify all shRNA/sgRNA cassettes equally, followed by second-generation amplicon sequencing. Sample-specific primer indexing allows many different conditions to be analysed in parallel.

In this vignette, a variety of different screens are analysed, ranging in both size (from tens to more than a thousand shRNAs/sgRNAs) and complexity (from the simplest two group comparison through to a time-course design). In every case, loss and/or gain of shRNA/sgRNA representation between different experimental groups is of interest.

The data sets used in this vignette can be downloaded from http://bioinf.wehi.edu.au/shRNAseq/. Users must have the latest version of R and *edgeR* ($\geq$ 3.5.23) installed in order to run the code that follows. The following commands can be run at the R prompt to install edgeR:

```r
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")

BiocManager::install("edgeR")
```

2 Analysis of a small shRNA-seq screen

In our first case study, we begin with raw sequence data available from the fastq file *screen1.fastq*. The structure of each sequence in this fastq file is known in advance, and depends upon the PCR primers used (Figure 1).

In this sequencing run, 4 independent experiments, each with biological replicate samples at Day 2 and Day 14 were available. The hairpins used in each experiment came from 4 different plates (plates 247-0001, 247-0003, 247-0005 and 247-0006 were included in this run). Sequencing was carried out on an Illumina HiSeq 2000 machine. Information about samples and hairpins are available in tab delimited text files named Samples1.txt and Hairpins1.txt respectively. Note that the sample and hairpin specific files must have a particular format with at least two columns (named *'ID'* and *'Sequences'*) containing the sample or hairpin ids (which must be unique) and the sample index or hairpin DNA sequences (these must be of uniform length and also be unique) to be matched. The sample index file may also contain a *'group'* column that indicates which experimental group a sample belongs to. Additional columns in each file will also be retained in the final R object that summarises the data from these files. In this example, the annotation information has been anonymised as this screen is unpublished. These files, along with the fastq file are assumed to be in the current working directory.

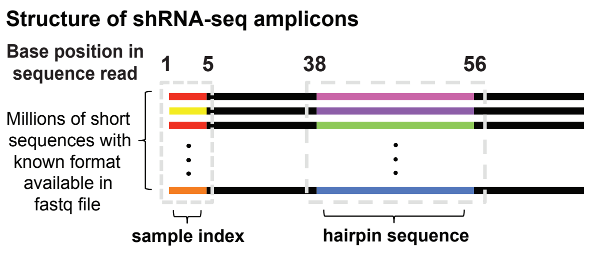

Structure of shRNA-seq amplicons
Base position in sequence read
1
5
38
56
Millions of short sequences with known format available in fastq file
sample index
hairpin sequence

Figure 1: Typical sequence format in a shRNA-seq screen
The base positions of the sample index and hairpin sequence may vary slightly between screens depending upon the PCR strategy. These parameters can be adjusted in the processAmplicons function.

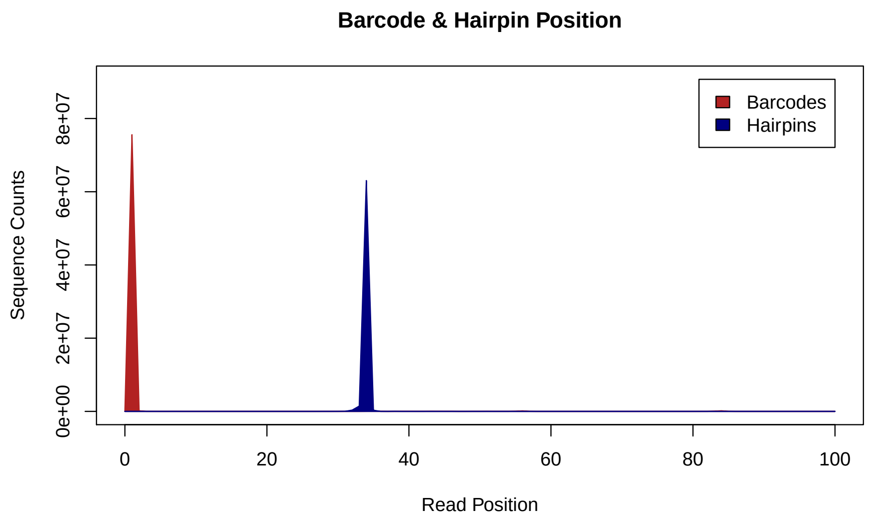
The function processAmplicons can be used to deconvolve the sequences in the fastq file into a matrix of counts summarising the number of times each hairpin was observed in each sample. To obtain more information about this sequence processing function, type the following:

```
?processAmplicons
```

We use this function to process the raw sequence data from this screen in the following commands.

```
library(edgeR)

# Read in sample & hairpin information
sampleanno = read.table("Samples1.txt", sep = "\t", header = TRUE)
sampleanno[1:5, ]

##   ID Sequences     group Experiment Replicate
## 1  1     AAAAA  TF1_Day2        TF1         1
## 2  8     AAACT TF1_Day14        TF1         1
## 3 11     AAAGG  TF1_Day2        TF1         2
## 4 20     AACAT TF1_Day14        TF1         2
## 5 23     AACCG  TF1_Day2        TF1         3

hairpinseqs = read.table("Hairpins1.txt", sep = "\t", header = TRUE)
hairpinseqs[1:5, ]

##         ID           Sequences    Plate
## 1 Hairpin1 CTCAGGACTTTGCAGCCAT 247-0001
## 2 Hairpin2 CAGTGATGCTCAAACAGAA 247-0001
## 3 Hairpin3 GCCTTGAGATACATGCCAA 247-0001
## 4 Hairpin4 CAATTCTCTGCTTAATCAT 247-0001
## 5 Hairpin5 CATGGCTACAGCTATAGGA 247-0001

# Process raw sequences from fastq file
```

```
x = processAmplicons("screen1.fastq", barcodefile = "Samples1.txt", hairpinfile = "Hairpins1.txt",
    verbose = TRUE, plotPositions = TRUE)

## -- Number of Barcodes : 25
## -- Number of Hairpins : 1269
## Processing reads in screen1.fastq.
## -- Processing 10 million reads
## -- Processing 20 million reads
## -- Processing 30 million reads
## -- Processing 40 million reads
## -- Processing 50 million reads
## -- Processing 60 million reads
## -- Processing 70 million reads
## -- Processing 80 million reads
## Number of reads in file screen1.fastq : 76967231
##
## The input run parameters are:
## -- Barcode in forward read: length 5
## -- Hairpin in forward read: length 19
## -- Mismatch in barcode/hairpin sequences not allowed.
##
## Total number of read is 76967231
## There are 76898853 reads (99.9112 percent) with barcode matches
## There are 65271202 reads (84.8039 percent) with hairpin matches
## There are 65243799 reads (84.7683 percent) with both barcode and hairpin matches
```

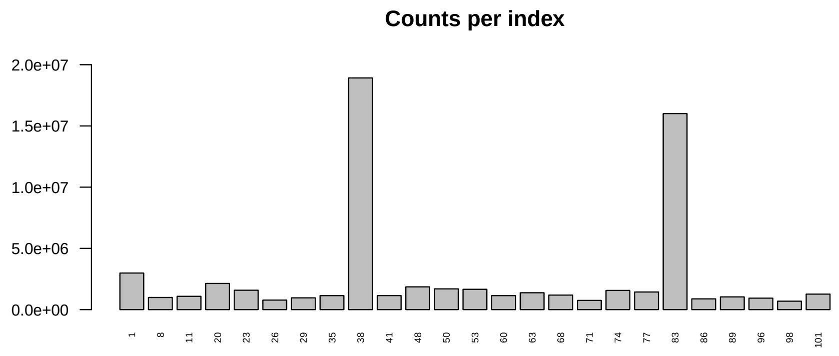


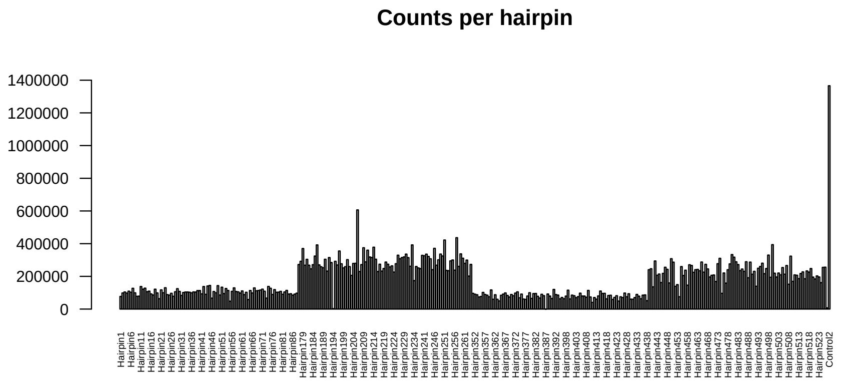
Running the above code takes around 6 minutes and uses 800Mb of RAM. Note that a very high proportion (> 80%) of the reads match to expected combinations from our screen, which is an indication that the sequencing for this screen has gone well. Percentages that are very low, or quite different between the barcode and hairpin values (the hairpin % would generally be lower than the barcode % due to sequencing errors) may indicate problems with the experiment.

The optional plotPositions argument creates a density plot of the read indexes each barcode and hairpin sequence are found in. This plot is useful as a sanity check in order to determine if processAmplicons is finding the expected sequences.

The counts are stored in a DGEList object. We next filter out hairpins with low counts (hairpins with at least 0.5 counts per million in at least 3 samples were retained) and plot the overall number of reads per sample and hairpin in a barplot. Counts per million are used as these values are standardised for systematic differences in the amount of sequencing between different samples, which can be subtantial (see first barplot below).

```
x

## An object of class "DGEList"
## $counts
##              1     8    11    20    23   26 29 35 38 41 48 50  53 60 63 68 71 74 77 83 86
## Hairpin1 25452  5432  9783 12071 17425 6333  1  6  3  0  0  0 970 21 10  4 17  0  0 23  4
## Hairpin2 36705  8329 11954 14240 19047 8269  0 17  3  1  1  0   0 20 10  5 21  0  0 24  2
## Hairpin3 35364 10003 11894 18645 20047 8419  1 12  6  1  0  0 110 34 20  3 26  2  1 21  0
## Hairpin4 29074  9311 12246 20544 16853 9570  1 12  6  1  0  0   0 23 17  2 26  0  0 22  1
## Hairpin5 34998 10562 12071 22317 20447 9099  0 17  4  0  1  1   0 34 21  5 23  1  0 31  2
##          89 96 98 101
## Hairpin1  0  0  0   0
## Hairpin2  0  0  0   0
## Hairpin3  0  1  0   0
## Hairpin4  0  0  0   0
## Hairpin5  0  0  0   0
## 1264 more rows ...
##
## $samples
##   ID lib.size norm.factors     group Experiment Replicate
## 1  1  2987408            1  TF1_Day2        TF1         1
## 2  8   989929            1 TF1_Day14        TF1         1
## 3 11  1085070            1  TF1_Day2        TF1         2
## 4 20  2136955            1 TF1_Day14        TF1         2
## 5 23  1582454            1  TF1_Day2        TF1         3
## 20 more rows ...
##
## $genes
##                ID          Sequences    Plate
## Hairpin1 Hairpin1 CTCAGGACTTTGCAGCCAT 247-0001
## Hairpin2 Hairpin2 CAGTGATGCTCAAACAGAA 247-0001
## Hairpin3 Hairpin3 GCCTTGAGATACATGCCAA 247-0001
## Hairpin4 Hairpin4 CAATTCTCTGCTTAATCAT 247-0001
## Hairpin5 Hairpin5 CATGGCTACAGCTATAGGA 247-0001
## 1264 more rows ...

# Filter hairpins with low counts
sel = rowSums(cpm(x$counts) > 0.5) >= 3
x = x[sel, ]

# Plot number of hairpins that could be matched per sample
par(mfrow = c(2, 1))
barplot(colSums(x$counts), las = 2, main = "Counts per index", cex.names = 0.5, cex.axis = 0.8,
    ylim = c(0, 2e+07))

# Select hairpins from plates run in this screen
plateinfo = x$genes$Plate
selhp = plateinfo == "247-0001" | plateinfo == "247-0003" | plateinfo == "247-
0005" | plateinfo ==
    "247-0006" | plateinfo == "Control"

# Plot per hairpin totals across all samples
barplot(rowSums(x$counts[selhp, ]), las = 2, main = "Counts per hairpin", cex.names = 0.5,
    cex.axis = 0.8, ylim = c(0, 1500000))
```



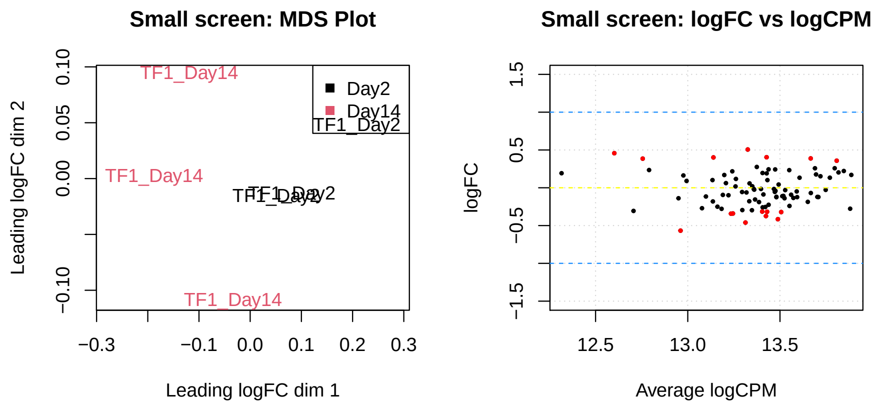


The number of sequences that could be assigned to the different samples and hairpins represented in this set of experiments can be seen to vary substantially. For example, two samples receive many more matches than the others (top barplot). Implicit in any downstream analysis carried out in *edgeR* is an adjustment to account for differences in library size, which is quite important when the overall amount of sequencing can vary considerably between samples. The botttom barplot shows that one particular hairpin appears to be much more abundant than the others. This happens to be a control, which is included in every plate, so is expected to be around 4 times higher than the others.

We next subset the DGEList object to hairpins and samples from the first experiment involving plate 1 (*247-0001*)/experiment *TF1*. A multidimensional scaling plot is generated to assess the consistency between replicate samples. The hairpin-specific variation is then estimated using the replicate samples from each group (Day 2 and Day 14). This simple experimental set-up leads us to use *edgeR*'s classic extact testing methodology (Robinson and Smyth, 2008) via the exactTest function to assess differences between the Day 14 and Day 2 replicate samples. The top ranked hairpins are listed using the topTags function, and those with a false discovery rate (FDR) < 0.05 (Benjamini and Hochberg, 1995) are highlighted on a plot of log-fold-change versus log-counts-per-millions by the plotSmear function.

```
# Select hairpins and samples relevant to plate 1
seltf1r = plateinfo == "247-0001"
seltf1c = x$samples$Experiment == "TF1"

# Subset DGEList
x1 = x[seltf1r, seltf1c]
x1$samples$group = factor(rep(c("TF1_Day2", "TF1_Day14"), times = 3))

# Make an MDS plot to visualise relationships between replicate samples
par(mfrow = c(1, 2))
plotMDS(x1, labels = x1$samples$group, col = rep(1:2, times = 3), main = "Small screen: MDS Plot")
legend("topright", legend = c("Day2", "Day14"), col = 1:2, pch = 15)

# Begin differential representation analysis Estimate dispersions
x1 = estimateDisp(x1)

## Design matrix not provided. Switch to the classic mode.

sqrt(x1$common.dispersion)

## [1] 0.103

# Assess differential representation between Day 14 and Day 2 samples using classic ex-
act
# testing methodology in edgeR
de.14vs2 = exactTest(x1, pair = c("TF1_Day2", "TF1_Day14"))

# Show top ranked hairpins
topTags(de.14vs2)

## Comparison of groups:  TF1_Day14-TF1_Day2
##                  ID           Sequences     Plate  logFC logCPM   PValue     FDR
## Hairpin1    Hairpin1 CTCAGGACTTTGCAGCCAT 247-0001 -0.567    13.0 1.42e-
05 0.00111
## Hairpin88 Hairpin88 CTGTGGTGCTTATTATTTA 247-0001  0.506    13.3 2.52e-
05 0.00111
## Hairpin2    Hairpin2 CAGTGATGCTCAAACAGAA 247-0001 -0.460    13.3 1.30e-
04 0.00383
## Hairpin15 Hairpin15 CCAGCCCAATCACTGTGTA 247-0001 -0.416    13.5 4.72e-
04 0.01038
## Hairpin29 Hairpin29 CTATATTCCTTGTGTAATT 247-0001  0.388    13.7 6.88e-
04 0.01038
## Hairpin37 Hairpin37 CCTTGAAATGTAAATAACT 247-0001  0.404    13.4 7.08e-
04 0.01038
```

```
## Hairpin64 Hairpin64 GCCTTTGTATATATCTGTA 247-0001  0.457    12.6 8.96e-
04 0.01127
## Hairpin86 Hairpin86 CTTAGAAAGGCACCTAGAA 247-0001  0.401    13.1 1.39e-
03 0.01409
## Hairpin11 Hairpin11 CAAAGGAATGTATATACTA 247-0001  0.358    13.8 1.44e-
03 0.01409
## Hairpin28 Hairpin28 GAACTCCAGACAGAACCAA 247-0001 -0.374    13.4 1.72e-
03 0.01516

# Select hairpins with FDR < 0.05 to highlight on plot
thresh = 0.05
top2 = topTags(de.14vs2, n = Inf)
top2ids = top2$table[top2$table$FDR < thresh, 1]

# Plot logFC versus logCPM
ylim = c(-1.5, 1.5)
plotSmear(de.14vs2, de.tags = top2ids, pch = 20, cex = 0.6, ylim = ylim, main = "Small screen: logFC vs logCPM")
abline(h = c(-1, 0, 1), col = c("dodgerblue", "yellow", "dodgerblue"), lty = 2)
```



Looking at the MDS plot we see that the replicate samples cluster reasonably well in dimension 1 (Day 14 samples tend to be on the left and Day 2 samples on the right of the plot).

Summary: In this small screen, the variation between replicates samples is quite small (biological coefficient of variation $\sim$ 10%) which means we are able to detect a number of hairpins with subtle fold-change and a small FDR.

3 Analysis of a second small shRNA-seq screen

In the next screen, there are biological replicates of 4 different experimental groups (Day2, Day10, Day5 GFP- and Day5 GFP+). Below we read in the raw counts from the file *screen2.fastq*. We search for all barcodes and hairpins listed in the files *Samples2.txt* and *Hairpins2.txt* respectively. This unpublished data set has been anonymised.

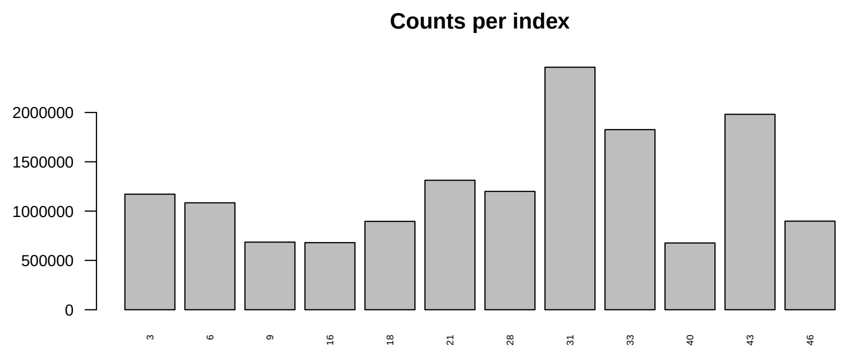
Since we have more than 2 groups, we perform a generalized linear model analysis in edgeR (McCarthy *et al.* 2012) on this data set. We once again use the processAmplicons function to process the raw sequence data from this screen.

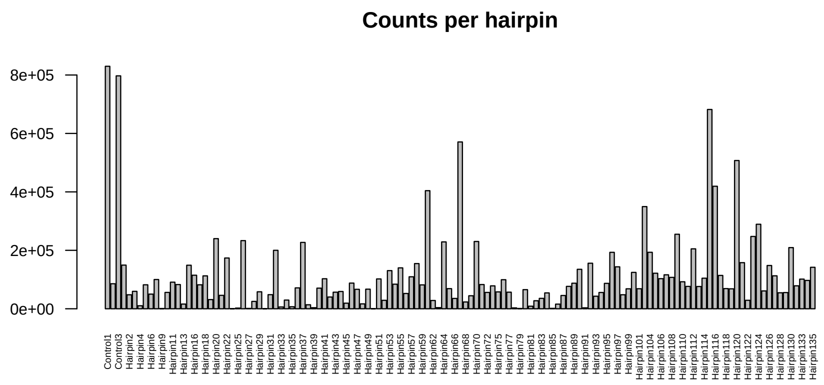
```
# Read in sample & hairpin information
sampleanno = read.table("Samples2.txt", header = TRUE, sep = "\t")
sampleanno

##    ID Sequences      group Replicate
## 1   3     GAAAG       Day2         1
## 2   6     GAACC      Day10         1
## 3   9     GAAGA Day5GFPneg         1
## 4  16     GAATT Day5GFPpos         1
## 5  18     GACAC       Day2         2
## 6  21     GACCA      Day10         2
## 7  28     GACGT Day5GFPneg         2
## 8  31     GACTG Day5GFPpos         2
## 9  33     GAGAA       Day2         3
## 10 40     GAGCT      Day10         3
## 11 43     GAGGG Day5GFPneg         3
## 12 46     GAGTC Day5GFPpos         3

hairpinseqs = read.table("Hairpins2.txt", header = TRUE, sep = "\t")
hairpinseqs[1:5, ]

##         ID             Sequences Gene
## 1 Control1 TCTCGCTTGGGCGAGAGTAAG    2
## 2 Control2 CCGCCTGAAGTCTCTGATTAA    2
## 3 Control3 AGGAATTATAATGCTTATCTA    2
## 4 Hairpin1 AAGGCAGAGACTGACCACCTA    4
## 5 Hairpin2 GAGCGACCTGGTGTTACTCTA    4

# Process raw sequences from fastq file
x2 = processAmplicons("screen2.fastq", barcodefile = "Samples2.txt", hairpinfile = "Hair-
pins2.txt",
    verbose = TRUE)

##  -- Number of Barcodes : 12
##  -- Number of Hairpins : 137
## Processing reads in screen2.fastq.
##  -- Processing 10 million reads
##  -- Processing 20 million reads
##  -- Processing 30 million reads
##  -- Processing 40 million reads
## Number of reads in file screen2.fastq : 38293297
##
## The input run parameters are:
##  -- Barcode: length 5
##  -- Hairpin: length 21
##  -- Mismatch in barcode/hairpin sequences not allowed.
##
## Total number of read is 38293297
## There are 38116328 reads (99.5379 percent) with barcode matches
## There are 14872258 reads (38.8378 percent) with hairpin matches
## There are 14871955 reads (38.8370 percent) with both barcode and hairpin matches
```

Running the above code takes around 2 minutes and uses 600Mb of RAM. In this screen, although we have a high proportion of sample indexes matching ($>$ 99%), only a fairly low proportion of reads ($\sim$ 38%) have a hairpin match, indicating that there is likely to be an issue with contamination in this screen.

In spite of this, we continue our analysis to look for hairpins that are relatively more or less abundant in a comparison of the Day5 GFP+ versus the Day5 GFP- replicate samples. We filter out hairpins with low counts (hairpins with at least 0.5 counts per million in at least 3 samples were retained) and plot the overall number of reads per sample or per hairpin in barplots.

```
x2

## An object of class "DGEList"
## $counts
##              3     6     9     16    18    21     28    31    33    40     43    46
## Control1 22647 26316 36885 290731 35158 49298  10611 99557 51758 36068 103077 67752
## Control2  5664  4623  7381   5010  4937  4163  18821 14113  7578  4952   5541  2883
## Control3 16426 33270 36925  53701 11526 37385 457414 48190 25650 19969  37524 19142
## Hairpin1 22359  7597  6230   3773 14096 10251   7451 20798 26898  3697  16464  9829
## Hairpin2  9593  4515  1563    918  4658  3593   2865  4928  6369   497   7384  1024
## 132 more rows ...
##
## $samples
##   ID lib.size norm.factors      group Replicate
## 1  3  1171539            1       Day2         1
## 2  6  1084243            1      Day10         1
## 3  9   685508            1 Day5GFPneg         1
## 4 16   680275            1 Day5GFPpos         1
## 5 18   895803            1       Day2         2
## 7 more rows ...
##
## $genes
##                ID            Sequences Gene
## Control1 Control1 TCTCGCTTGGGCGAGAGTAAG    2
## Control2 Control2 CCGCCTGAAGTCTCTGATTAA    2
## Control3 Control3 AGGAATTATAATGCTTATCTA    2
## Hairpin1 Hairpin1 AAGGCAGAGACTGACCACCTA    4
## Hairpin2 Hairpin2 GAGCGACCTGGTGTTACTCTA    4
## 132 more rows ...

# Filter hairpins with low counts
sel = rowSums(cpm(x2$counts)>0.5)>=3
x2 = x2[sel,]

# Plot number of hairpins that could be matched per sample
# and total for each hairpin across all samples
par(mfrow=c(2,1))
barplot(colSums(x2$counts), las=2, main="Counts per index", cex.names=0.5, cex.axis=0.8)
barplot(rowSums(x2$counts), las=2, main="Counts per hairpin", cex.names=0.5, cex.axis=0.8)
```



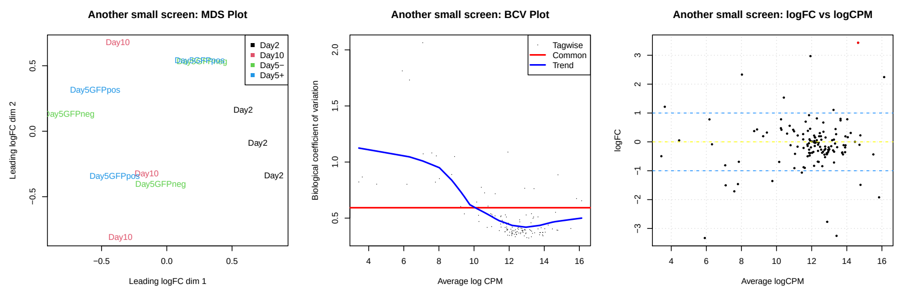


Next we make a multidimensional scaling plot to assess the consistency between replicate samples. A design matrix is set up for the GLM analysis, and the hairpin-specific variation is estimated and plotted (while taking into account the group structure).

We use the function glmFit to fit the hairpin-specific models and glmLRT to do the testing between the Day 5 GFP+ and Day 5 GFP- samples. The top ranked hairpins are listed using the topTags function and hairpins with FDR < 0.05 (Benjamini and Hochberg, 1995) are highlighted on a plot of log-fold-change versus log-counts-per-millions by the plotSmear function.

```
# Make an MDS plot to visualise relationships between replicate samples
par(mfrow = c(1, 3))
plotMDS(x2, labels = x2$samples$group, col = rep(1:4, times = 3), main = "An-
other small screen: MDS Plot")
legend("topright", legend = c("Day2", "Day10", "Day5-", "Day5+"), col = 1:4, pch = 15)

# Begin differential representation analysis We will use GLMs in edgeR in this case since
# there are more than 2 groups Set up design matrix for GLM
des = model.matrix(~x2$samples$group)
des

##   (Intercept) x2$samples$groupDay2 x2$samples$groupDay5GFPneg x2$samples$groupDay5GFPpos
## 1           1                    1                          0                          0
## 2           1               0                  0                   0
## 3           1               0                  1                   0
## 4           1               0                  0                   1
## 5           1               1                  0                   0
## 6           1               0                  0                   0
## 7           1               0                  1                   0
## 8           1               0                  0                   1
## 9           1               1                  0                   0
## 10          1               0                  0                   0
## 11          1               0                  1                   0
## 12          1               0                  0                   1
## attr(,"assign")
## [1] 0 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$`x2$samples$group`
## [1] "contr.treatment"

# Estimate dispersions
xglm = estimateDisp(x2, des)
sqrt(xglm$common.disp)

## [1] 0.593

# Plot BCVs versus abundance
plotBCV(xglm, main = "Another small screen: BCV Plot")

# Fit negative bionomial GLM
fit = glmFit(xglm, des)
# Carry out Likelihood ratio test
lrt = glmLRT(fit, contrast = c(0, 0, -1, 1))

# Show top ranked hairpins
topTags(lrt)

## Coefficient:  -1*x2$samples$groupDay5GFPneg 1*x2$samples$groupDay5GFPpos
##                  ID            Sequences Gene logFC logCPM    LR   PValue      FDR
## Hairpin67 Hairpin67 AAAAGCAGTTCTCAAGATCTA   32  3.43  14.64 23.08 1.55e-
06 0.000204
## Hairpin92 Hairpin92 AAGAGGATGAAGACCTGCTTA   38 -3.26  13.42 11.03 8.97e-
04 0.058749
## Hairpin57 Hairpin57 CTGATTGTTGACAGTGTCAAA   26 -2.77  12.89  8.25 4.07e-
03 0.177527
## Control1   Control1 TCTCGCTTGGGCGAGAGTAAG    2  2.24  16.13  7.73 5.44e-
03 0.178272
## Control3   Control3 AGGAATTATAATGCTTATCTA    2 -1.92  15.84  5.47 1.94e-
02 0.474612
## Hairpin42 Hairpin42 CTGGTATGTCTTGGAGAGATA   20 -1.06  11.44  5.27 2.17e-
02 0.474612
## Hairpin39 Hairpin39 TAGCATGGATATGGAGTTAAA   19  2.33   8.03  4.85 2.77e-
02 0.517038
## Hairpin54 Hairpin54 AGGGTGTCTATTTGTCTTCAA   24  2.97  11.93  4.62 3.16e-
02 0.517038
```

```
## Hairpin97 Hairpin97 CCGCACTTACTCCAAGTTCAA   5  1.11  13.24  4.11 4.28e-02 0.622324
## Hairpin49 Hairpin49 AAGAGGAAGAAGGCAAGTTTA  20 -0.83  12.14  3.63 5.67e-02 0.742544

# Select hairpins with FDR < 0.05 to highlight on plot
thresh = 0.05
top2 = topTags(lrt, n = Inf)
top2ids = top2$table[top2$table$FDR < thresh, 1]

# Plot logFC versus logCPM
plotSmear(lrt, de.tags = top2ids, pch = 20, cex = 0.6, main = "Another small screen: logFC vs logCPM")
abline(h = c(-1, 0, 1), col = c("dodgerblue", "yellow", "dodgerblue"), lty = 2)
```



The biological coefficent of variation (BCV) plot (middle panel) summarises the variability in the screen as a functon of hairpin abundance. These plots tend to have a characteristic shape of decreasing variability as hairpin abundance increases, which is similar to what is observed for other applications such as RNA-seq. The individual black points show hairpin-specific (referred to as *'Tagwise'* variability, while the blue line shows the trend value as hairpin abundance changes (*'Trended'*) and the red line is the common value (calculated by assuming all counts come from the same hairpin).

Summary: In this second small screen, the variation between replicate samples is much higher than in the first one (biological coefficient of variation $\sim$ 62%) which limits our ability to detect any subtle changes. As a result we find only one hairpin with a FDR $<$ 0.05 and a log-fold-change of 3.57.

4 Analysis of a larger shRNA-seq screen

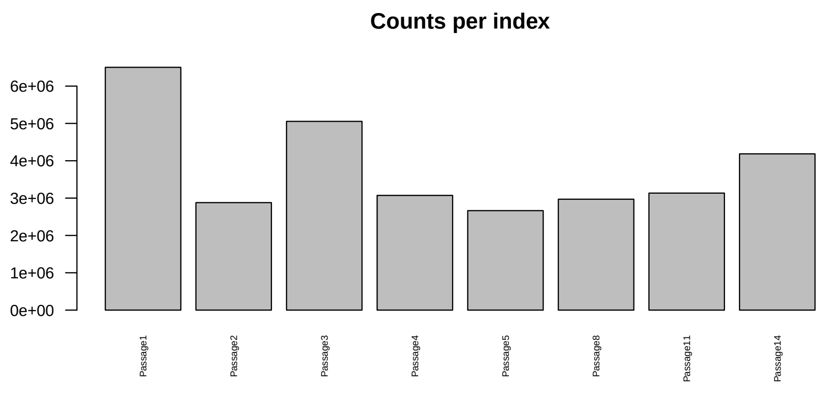
In the third example, a library of around 1,100 hairpins were screened in a time-course experiment, where samples were collected over a period of 8 days. Multiple hairpins per gene (generally between 3-6) were included in this collection. Below we read in the raw sequences from the file *screen3.fastq* and search for matches with sample indexes and hairpins listed in the files *Samples3.txt* and *Hairpins3.txt* respectively using the processAmplicons function to give us a DGEList of counts. This unpublished data set has been anonymised.

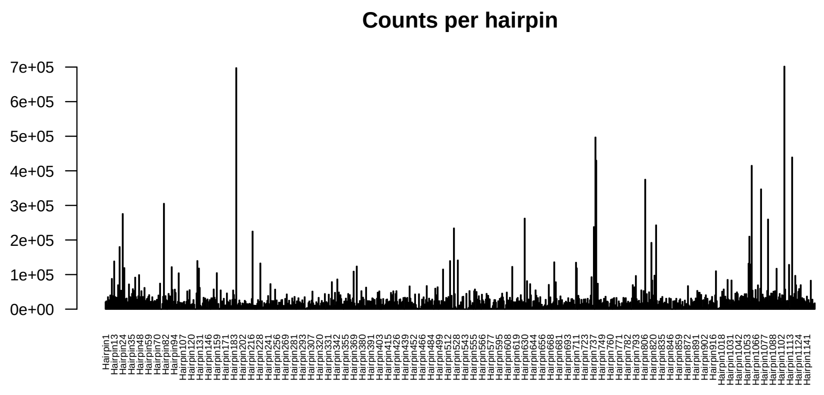
```
# Read in sample & hairpin information
sampleanno = read.table("Samples3.txt", header = TRUE, sep = "\t")
sampleanno

##          ID Sequences
## 1  Passage1     AGCAC
## 2  Passage2     AGCGT
## 3  Passage3     AGGAA
## 4  Passage4     AGGGG
## 5  Passage5     AGTAT
## 6  Passage8     AGTGC
## 7 Passage11     ATACA
## 8 Passage14     ATATG

hairpinseqs = read.table("Hairpins3.txt", header = TRUE, sep = "\t")
hairpinseqs[1:5, ]

##          ID             Sequences Gene
## 1 Hairpin1 CAGGTACAAAGATGGTTGCGA    1
## 2 Hairpin2 CTGGTCTTACCCTGACACCAA    1
## 3 Hairpin3 AAGCCCTGGGTTCCTGTTCTA    1
## 4 Hairpin4 GAGCACAGAGATGACGAGCGA    1
## 5 Hairpin5 TTCCGAGAGTTGGAGCAAGAA    1

# Process raw sequences from fastq file
x3 = processAmplicons("screen3.fastq", barcodefile = "Samples3.txt", hairpinfile = "Hairpins3.txt",
    verbose = TRUE)

##  -- Number of Barcodes : 8
##  -- Number of Hairpins : 1153
## Processing reads in screen3.fastq.
##  -- Processing 10 million reads
##  -- Processing 20 million reads
##  -- Processing 30 million reads
##  -- Processing 40 million reads
##  -- Processing 50 million reads
##  -- Processing 60 million reads
##  -- Processing 70 million reads
##  -- Processing 80 million reads
##  -- Processing 90 million reads
##  -- Processing 100 million reads
##  -- Processing 110 million reads
##  -- Processing 120 million reads
##  -- Processing 130 million reads
##  -- Processing 140 million reads
## Number of reads in file screen3.fastq : 130090268
##
## The input run parameters are:
##  -- Barcode: length 5
##  -- Hairpin: length 21
##  -- Mismatch in barcode/hairpin sequences not allowed.
##
## Total number of read is 130090268
## There are 99766841 reads (76.6905 percent) with barcode matches
## There are 30956029 reads (23.7958 percent) with hairpin matches
## There are 30471462 reads (23.4233 percent) with both barcode and hairpin matches
```

Running the above code takes around 6 minutes and uses 1G of RAM. Although the proportion of sequences that match is low (∼ 23% to the hairpin sequences and ∼23% with both an index and a hairpin match), this was expected, as only around 40% of the sequencing run was dedicated to this screen. The remaining data relates to another project.

As before, we filter out hairpins with low counts (hairpins with at least 0.5 counts per million in at least half of the samples were retained) and plot the overall number of reads per sample or per hairpin in barplots.

```
x3

## An object of class "DGEList"
## $counts
##          Passage1 Passage2 Passage3 Passage4 Passage5 Passage8 Passage11 Passage14
## Hairpin1     9544     3271     1477      547      508     1717      1932      2005
## Hairpin2     8615     3550     1456     1504     1680     1323      2858      2376
## Hairpin3     7306      991     1166      383      607      103       658       177
## Hairpin4     8763     1169     3009     2434     2015     1373      5312     11285
## Hairpin5     7913     3117     4668     1949     1642     2482      2062      2810
## 1148 more rows ...
##
## $samples
##                  ID group lib.size norm.factors
## Passage1   Passage1     1  6506764            1
## Passage2   Passage2     1  2879384            1
## Passage3   Passage3     1  5056008            1
## Passage4   Passage4     1  3073676            1
## Passage5   Passage5     1  2664513            1
## Passage8   Passage8     1  2971301            1
## Passage11 Passage11     1  3134600            1
## Passage14 Passage14     1  4185216            1
##
## $genes
##                ID            Sequences Gene
## Hairpin1 Hairpin1 CAGGTACAAAGATGGTTGCGA    1
## Hairpin2 Hairpin2 CTGGTCTTACCCTGACACCAA    1
## Hairpin3 Hairpin3 AAGCCCTGGGTTCCTGTTCTA    1
## Hairpin4 Hairpin4 GAGCACAGAGATGACGAGCGA    1
## Hairpin5 Hairpin5 TTCCGAGAGTTGGAGCAAGAA    1
## 1148 more rows ...

# Filter hairpins with low counts
sel = rowSums(cpm(x3$counts) > 0.5) >= 4
x3 = x3[sel, ]

# Plot number of hairpins that could be matched per sample and total for each hairpin
# across all samples
par(mfrow = c(2, 1))
barplot(colSums(x3$counts), las = 2, main = "Counts per index", cex.names = 0.5, cex.axis = 0.8)
barplot(rowSums(x3$counts), las = 2, main = "Counts per hairpin", cex.names = 0.5, cex.axis = 0.8)
```





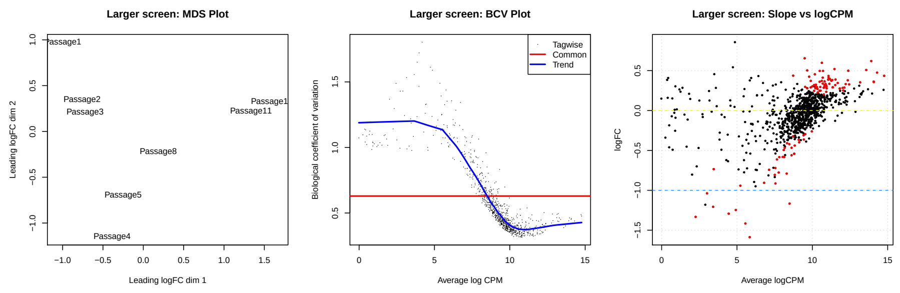
We normalize the counts using the TMM method (Robinson and Oshlack, 2010) and make a multidimensional scaling plot as before. The design matrix for this experiment consists of a model with a slope and intercept. Hairpins with an increasing or decreasing trend over time are of interest. The hairpin-specific dispersion is estimated and plotted. We use the function glmFit to fit hairpin-specific models and glmLRT to test whether the slope is different to zero.

The top ranked hairpins are listed using the topTags function and hairpins with FDR < 0.05 (Benjamini and Hochberg, 1995) are highlighted on a plot of log-fold-change versus log-counts-per-millions by the plotSmear function.

```
# Carry out normalization using TMM
x3 = calcNormFactors(x3, method = "TMM")

# Make an MDS plot to visualise relationships between replicate samples
par(mfrow = c(1, 3))
plotMDS(x3, main = "Larger screen: MDS Plot")

# Begin differential representation analysis We will use GLMs in edgeR in this case since
# the experimental design is a time course with changes expected over time i.e. model is y
# = intercept + slope*time Set up design matrix for GLM
des = model.matrix(~seq(1:8))
des
```

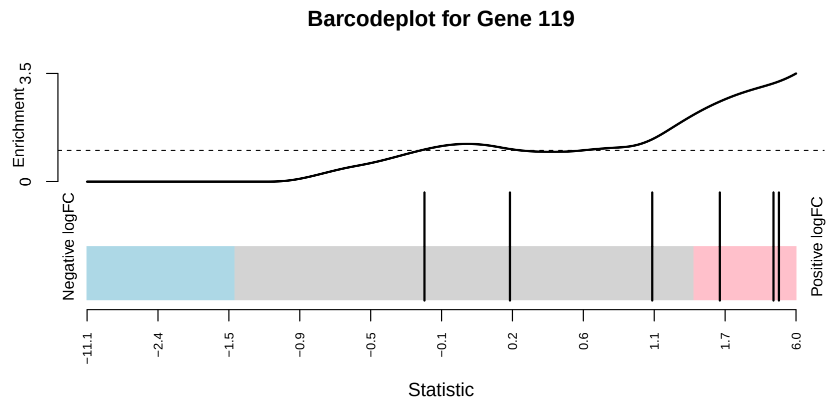
```
##   (Intercept) seq(1:8)
## 1           1        1
## 2           1        2
## 3           1        3
## 4           1        4
## 5           1        5
## 6           1        6
## 7           1        7
## 8           1        8
## attr(,"assign")
## [1] 0 1

colnames(des)[2] = "Slope"

# Estimate dispersions
xglm = estimateDisp(x3, des)
sqrt(xglm$common.disp)

## [1] 0.629

# Plot BCVs versus abundance
plotBCV(xglm, main = "Larger screen: BCV Plot")

# Fit negative bionomial GLM
fit = glmFit(xglm, des)
# Carry out Likelihood ratio test
lrt = glmLRT(fit, coef = 2)

# Show top ranked hairpins
topTags(lrt)

## Coefficient:  Slope
##                    ID            Sequences Gene logFC logCPM   LR   PValue      FDR
## Hairpin648 Hairpin648 AAGAGCTTTGTTAGACAACAA  109 0.598  10.63 62.9 2.22e-
15 2.03e-12
## Hairpin726 Hairpin726 AACATTAACAGTGTTGAGATA  121 0.654   9.49 38.3 6.22e-
10 2.84e-07
## Hairpin807 Hairpin807 CAGAAATTATGTGACTATATA  133 0.620  13.91 36.8 1.28e-
09 3.91e-07
## Hairpin520 Hairpin520 CAGACTATGAGTCTAGTTTAA   86 0.499  12.39 34.4 4.43e-
09 1.01e-06
## Hairpin79   Hairpin79 CTCCAGTGTTCTGTTAATATT   17 0.508  13.58 33.7 6.46e-
09 1.18e-06
## Hairpin248 Hairpin248 CAGAACAGAGGTACATTATAA   44 0.520  11.47 29.2 6.48e-
08 9.86e-06
## Hairpin810 Hairpin810 AAGAAAGTTCTTACAACGAAA  139 0.496  10.71 27.8 1.38e-
07 1.80e-05
## Hairpin241 Hairpin241 CTCCGAGACTATCAGAAGATA   43 0.496  10.49 25.9 3.64e-
07 4.12e-05
## Hairpin336 Hairpin336 ATCCAATGTGTTCCTTTAATA   58 0.389  11.54 25.6 4.10e-
07 4.12e-05
## Hairpin385 Hairpin385 CTCAAGTGTAGATACAGATTA   65 0.396  11.16 25.5 4.51e-
07 4.12e-05
```

```
# Select hairpins with FDR < 0.05 to highlight on plot
thresh = 0.05
top3 = topTags(lrt, n = Inf)
top3ids = top3$table[top3$table$FDR < thresh, 1]

# Plot Slope versus logCPM
plotSmear(lrt, de.tags = top3ids, pch = 20, cex = 0.6, main = "Larger screen: Slope vs logCPM")
abline(h = c(-1, 0, 1), col = c("dodgerblue", "yellow", "dodgerblue"), lty = 2)
```



We finish this analysis by summarising data from multiple hairpins in order to get a gene-by-gene ranking, rather than a hairpin-specific one. The *roast* gene-set test (Wu *et al.* 2010) is used for this purpose. In the screen setting, the collection of individual hairpins that target a specific gene can be regarded as a 'set'. This analysis relies on the availablity of an annotation that indicates which gene each hairpin targets (this has been recorded in the 'Gene' column of the hairpin annotation in this example). In the code below, we restrict our analysis to genes with greater than 3 hairpins. A barcode plot, highlighting the rank of hairpins for a given gene relative to the entire data set is generated for the top-ranked gene (119). The hairpins for this gene tend to increase in abundance over time, with $2/3$ of the hairpins contributing to the test result (FDR=0.0549). Note that a gene-level analysis like this is only possible within the GLM framework.

```
# Carry out roast gene-set analysis
genesymbols = x3$genes[, 3]

genesymbollist = list()
unq = unique(genesymbols)
unq = unq[!is.na(unq)]
for (i in unq) {
    sel = genesymbols == i & !is.na(genesymbols)
    if (sum(sel) > 3)
        genesymbollist[[i]] = which(sel)
}

# Run mroast for all genes
set.seed(6012014)
roast.res = mroast(xglm, index = genesymbollist, des, contrast = 2, nrot = 9999)

# Display results for top ranked genes
roast.res[1:20, 1:6]

##       NGenes PropDown PropUp Direction PValue    FDR
## set096     6   0.500  0.000      Down 0.0002 0.0203
## set119     6   0.000  0.667        Up 0.0006 0.0371
## set151     7   0.000  0.571        Up 0.0012 0.0517
## set024     8   0.000  0.625        Up 0.0021 0.0551
## set122     4   0.750  0.250      Down 0.0023 0.0551
## set039     6   0.333  0.000      Down 0.0025 0.0551
## set141     4   0.000  1.000        Up 0.0032 0.0565
## set121     5   0.200  0.600        Up 0.0034 0.0565
## set023     8   0.500  0.250      Down 0.0042 0.0622
## set009     8   0.250  0.500        Up 0.0055 0.0714
## set007     8   0.000  0.500        Up 0.0063 0.0714
## set143     8   0.375  0.000      Down 0.0064 0.0714
## set110     8   0.625  0.125      Down 0.0071 0.0732
## set003     8   0.000  0.875        Up 0.0085 0.0815
## set068     7   0.429  0.000      Down 0.0096 0.0851
## set127     6   0.000  0.333        Up 0.0111 0.0851
## set036     5   0.400  0.000      Down 0.0113 0.0851
## set019     8   0.125  0.250        Up 0.0114 0.0851
## set086     6   0.000  0.333        Up 0.0135 0.0942
## set104     6   0.167  0.500        Up 0.0147 0.0942

# Make a barcode plot for an example that ranks highly Gene 119 - multiply slopes by 7 to
# convert into logFCs over time-course
par(mfrow = c(1, 1))
barcodeplot(7 * lrt$table$logFC, index = genesymbollist[[119]], main = "Barcodeplot for Gene 119",
    labels = c("Negative logFC", "Positive logFC"))
```



5 Analysis of shRNA-seq screen from Zuber *et al.* (2011)

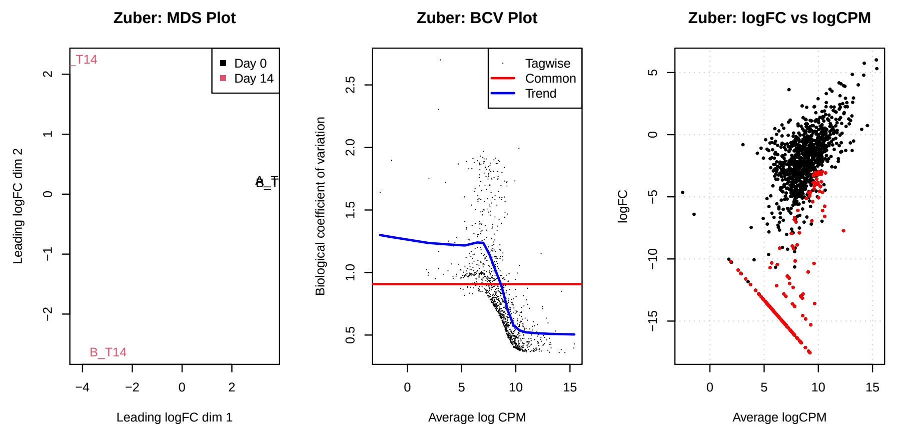
We next look at some published data from Zuber *et al.* (2011). The goal of this screen was to identify new drug targets for acute myeloid leukaemia (AML). A custom library of > 1,000 hairpins targeting 240 genes known to regulate chromatin structure were screened in a mouse model of AML. Between 3 and 6 distinct hairpins per gene were available.

The screen used leukaemia cells from an inducible mouse model and sampled DNA from these cells post infection (Day 0) and at Day 14. Hairpins that consistently decrease in representation across the biological replicate samples were of interest.

Below we take a merged table of counts obtained from the Supplementary Materials of Zuber *et al.* (2011) and analyse it using *edgeR*. We begin with a hairpin-level analysis to rank individual hairpins using GLMs. Diagnostic plots and a list of top hairpins is given.

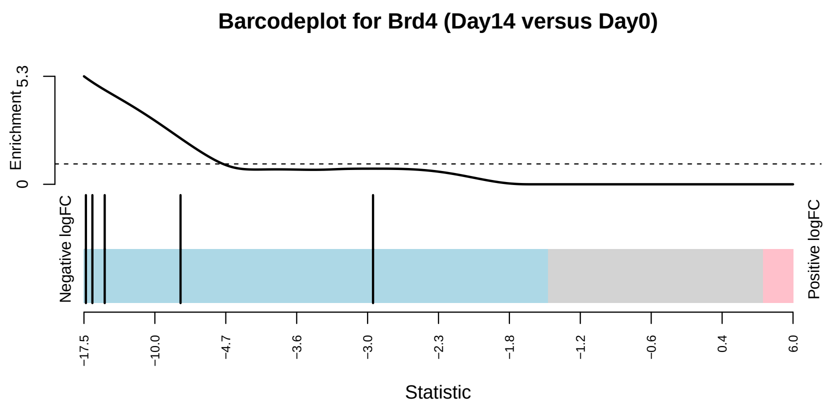
```
# Read in the table of counts
dat = read.table("zuber_screen.txt", sep = "\t", header = TRUE, as.is = TRUE)
dat[1, ]

##       shRN_AID GeneSymbol  EntrezID Pool shRNA_start Mean_T14.T0 T14.T0_A T14.T0_B
## 1 100043305.158  100043305 100043305   LIB         158         0.2    0.269    0.132
##   Reads_A_T0 Reads_A_T14 Reads_B_T0 Reads_B_T14
## 1      34133        9171      31158        4111

# Make DGE list containing hairpin counts
x4 = new("DGEList")
x4$counts = as.matrix(dat[, 9:12])

# Remove hairpins with zero counts in all samples
selnonzero = rowSums(x4$counts) != 0
x4$counts = x4$counts[selnonzero, ]

# Add sample annotation data
x4$samples = data.frame(SampleID = colnames(x4$counts), group = as.factor(rep(c("Day0", "Day14"),
    times = 2)), lib.size = colSums(x4$counts))
x4$samples$norm.factors = 1
x4$genes = dat[selnonzero, 1:5]
rownames(x4$counts) = dat[selnonzero, 1]
dim(x4)

## [1] 1095    4

# Make an MDS plot to visualise relationships between replicate samples
par(mfrow = c(1, 3))
plotMDS(x4, labels = gsub("Reads_", "", colnames(x4)), col = c(1, 2, 1, 2), main = "Zu-
ber: MDS Plot")
legend("topright", legend = c("Day 0", "Day 14"), col = 1:2, pch = 15)

# Assess differential representation between Day 14 and Day 0 samples using GLM in edgeR
# Set up design matrix for GLM
des = model.matrix(~x4$samples$group)
colnames(des)[2] = "Day14"
des

##   (Intercept) Day14
## 1           1     0
## 2           1     1
## 3           1     0
## 4           1     1
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$`x4$samples$group`
## [1] "contr.treatment"

# Estimate dispersions
xglm = estimateDisp(x4, des)

# Plot BCVs versus abundance
plotBCV(xglm, main = "Zuber: BCV Plot")

# Fit negative bionomial GLM
fit = glmFit(xglm, des)
# Carry out Likelihood ratio test
lrt = glmLRT(fit, 2)

# Show top ranked hairpins
topTags(lrt, n = 15)

## Coefficient:  Day14
##                shRN_AID GeneSymbol EntrezID Pool shRNA_start  logFC logCPM    LR
## Rpa3.276         Rpa3.276       Rpa3    68240   PC         278 -13.59   9.66 117.1
## Suz12.1842     Suz12.1842      Suz12    52615  LIB        1842 -17.54   9.22 102.4
## Setd4.1308     Setd4.1308      Setd4   224440  LIB        1308 -15.30   9.30  96.5
## Pcna.1186       Pcna.1186       Pcna    18538   PC        1186 -17.42   9.10  93.0
## Supt16h.1672 Supt16h.1672    Supt16h   114741  LIB        1672 -17.13   8.81  72.4
## Setmar.1589   Setmar.1589     Setmar    74729  LIB        1589   6.02  15.35  71.8
## Rpa3.561         Rpa3.561       Rpa3    68240   PC         561  -7.73  12.32  68.4
## Brd3.187         Brd3.187       Brd3    67382  LIB         187 -14.83   8.83  67.9
## Rpa3.455         Rpa3.455       Rpa3    68240   PC         457  -5.76  10.59  62.0
## Brd4.2097       Brd4.2097       Brd4    57261  LIB        2097 -16.75   8.43  57.6
## Polr2b.2176   Polr2b.2176     Polr2b   231329   PC        2176 -14.56   8.56  57.5
## Wdr5.1765       Wdr5.1765       Wdr5   140858  LIB        1765 -16.72   8.40  56.7
## Aof2.2857       Aof2.2857       Aof2    99982  LIB        2857 -16.67   8.35  55.8
## Pcmt1.840       Pcmt1.840      Pcmt1    18537  LIB         840   4.79  14.19  54.7
## Jmjd1a.371     Jmjd1a.371     Jmjd1a   104263  LIB         371 -16.62   8.29  52.8
##                PValue      FDR
## Rpa3.276     2.75e-27 3.01e-24
## Suz12.1842   4.65e-24 2.54e-21
## Setd4.1308   8.74e-23 3.19e-20
## Pcna.1186    5.11e-22 1.40e-19
## Supt16h.1672 1.77e-17 3.88e-15
## Setmar.1589  2.38e-17 4.34e-15
## Rpa3.561     1.35e-16 2.11e-14
## Brd3.187     1.73e-16 2.37e-14
## Rpa3.455     3.51e-15 4.27e-13
## Brd4.2097    3.15e-14 3.34e-12
## Polr2b.2176  3.35e-14 3.34e-12
## Wdr5.1765    5.19e-14 4.74e-12
## Aof2.2857    8.00e-14 6.74e-12
## Pcmt1.840    1.43e-13 1.12e-11
## Jmjd1a.371   3.71e-13 2.71e-11

# Select hairpins with FDR < 0.0001 and logFC < -1 to highlight on plot
```

```
thresh = 1e-04
lfc = -1
top = topTags(lrt, n = Inf, sort.by = "logFC")

sum(top$table[, 9] < thresh)

## [1] 195

sum(top$table[, 9] < thresh & top$table[, 6] < lfc)

## [1] 183

topids = as.character(top$table[top$table$FDR < thresh & top$table$logFC < lfc, 1])

# Make a plot of logFC versus logCPM
plotSmear(lrt, de.tags = topids, pch = 20, cex = 0.6, main = "Zuber: logFC vs logCPM")
```



We finish this analysis by summarising data from multiple hairpins in order to get a gene-by-gene ranking, rather than a hairpin-specific one using the *roast* gene-set test (Wu *et al.* 2010). The gene *Brd4* is examined first (this was reported as a key finding in the original paper) followed by an analysis for all genes. *Brd4* is also highly ranked in our analysis.

```
# Carry out roast gene-set analysis Begin with hairpins targeting Brd4
genesymbols = x4$genes[, 2]
brd4 = genesymbols == "Brd4"
set.seed(6012014)
roast(xglm, index = brd4, des, contrast = 2, nrot = 9999)

##           Active.Prop P.Value
## Down                1   2e-04
## Up                  0   1e+00
## UpOrDown            1   4e-04
## Mixed               1   4e-04

# Make a barcode plot for Brd4
par(mfrow = c(1, 1))
barcodeplot(lrt$table$logFC, index = brd4, main = "Barcodeplot for Brd4 (Day14 ver-
sus Day0)",
    labels = c("Negative logFC", "Positive logFC"))
```



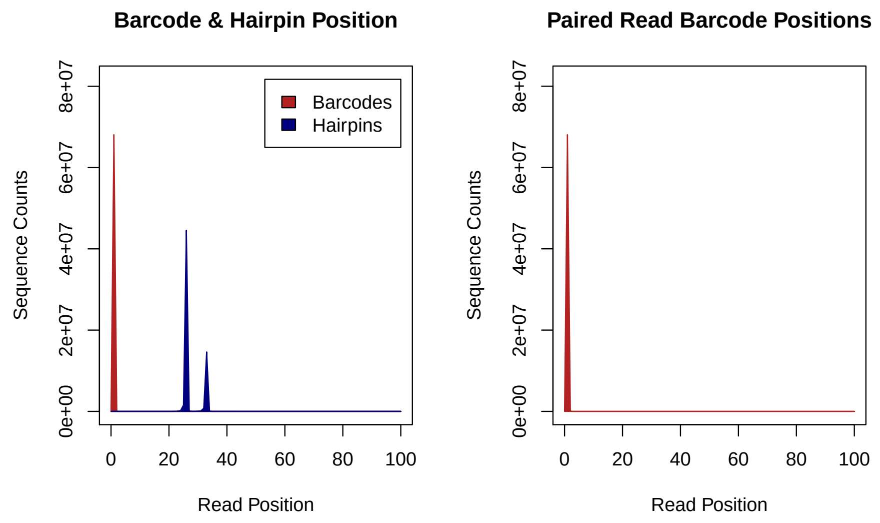
```
# Repeat analysis for all genes using mroast
genesymbollist = list()
for (i in unique(genesymbols)) genesymbollist[[i]] = which(genesymbols == i)

roast.res = mroast(xglm, index = genesymbollist, des, contrast = 2, nrot = 9999)
roast.res[1, ]

##       NGenes PropDown PropUp Direction PValue     FDR PValue.Mixed FDR.Mixed
## Aurkb      6    0.833      0      Down  1e-04 0.00415        1e-04   0.00311

# Display results for top ranked genes
roast.res[1:20, 1:6]

##              NGenes PropDown PropUp Direction PValue     FDR
## Aurkb             6    0.833    0.0      Down 0.0001 0.00415
## Jhdm1d            5    0.800    0.0      Down 0.0001 0.00415
## Cbx2              5    0.600    0.0      Down 0.0001 0.00415
## Srcap             5    0.800    0.0      Down 0.0002 0.00934
## Polr2b            2    1.000    0.0      Down 0.0003 0.01012
## Ing2              5    1.000    0.0      Down 0.0004 0.01012
## Setd2             5    1.000    0.0      Down 0.0004 0.01012
## Hdac11            5    0.800    0.0      Down 0.0004 0.01012
## Brd4              5    1.000    0.0      Down 0.0005 0.01012
## Setd4             3    1.000    0.0      Down 0.0006 0.01012
## LOC100044324      5    0.800    0.2      Down 0.0006 0.01012
## Nap1l1            4    0.750    0.0      Down 0.0006 0.01012
## Sirt5             4    0.500    0.0      Down 0.0006 0.01012
## Prdm11            4    1.000    0.0      Down 0.0007 0.01012
## Prmt2             4    0.500    0.0      Down 0.0007 0.01012
## Hells             4    0.500    0.0      Down 0.0007 0.01012
## Hdac9             5    1.000    0.0      Down 0.0009 0.01176
## Mecp2             4    0.750    0.0      Down 0.0009 0.01176
## Whsc1l1           5    0.800    0.0      Down 0.0011 0.01245
## Smarca4           5    1.000    0.0      Down 0.0012 0.01245
```

6 Analysis of a large CRISPR-Cas9 knockout screen

Next we analyse data from a pooled screen that uses CRISPR-Cas9 (clustered regularly interspaced short palindromic repeats-associated nuclease Cas9) knockout technology. In this example, a library of around 64,000 sgRNAs (as used in Shalem *et al.* 2014) were screened to look for genes that may lead to resistance from a particular drug.

Multiple single guide RNAs (sgRNAs) per gene (generally between 3-6) were included in the screen. Below we read in the raw sequences from the paired end fastq files *screen4_R1.fastq* and *screen4_R2.fastq*. This screen employed a dual indexing strategy where the first 8 bases from each pair of reads contained an index sequence that uniquely identifies which sample a particular sgRNA sequence originated from. Matches between sample indexes and sgRNAs listed in the files *Samples4.txt* and *sgRNAs4.txt* were identified using the processAmplicons function to produce a DGEList of counts. This unpublished data set has been anonymised.

```
# Read in sample & sgRNA information
sampleanno = read.table("Samples4.txt", header = TRUE, sep = "\t")
sampleanno[1:5, ]

##        ID Sequences SequencesReverse   group Infection Replicate IndexF IndexR
## 1 A1_1_1  TAGATCGC         TAAGGCGA    Drug         1         1      1      1
## 2 A2_1_2  TAGATCGC         CGTACTAG Control         1         1      1      2
## 3 A3_1_3  TAGATCGC         AGGCAGAA    Drug         1         1      1      3
## 4 A4_1_4  TAGATCGC         TCCTGAGC Control         1         1      1      4
## 5 A5_1_5  TAGATCGC         GGACTCCT    Drug         1         1      1      5

sgseqs = read.table("sgRNAs4.txt", header = TRUE, sep = "\t")
sgseqs[1:5, ]

##       ID            Sequences Gene
## 1 sgRNA1 TACCCTGGGACTGTACCCCC   99
## 2 sgRNA2 ACCCTTGCTGCACGACCTGC   99
## 3 sgRNA3 TCGCTCGCCCCGCTCTTCCT   99
## 4 sgRNA4 TGACGCCTCGGACGTGTCTG   19
## 5 sgRNA5 CGTCATAGCCAATCTTCTTC   19

# Process raw sequences from fastq files
x4 = processAmplicons("screen4_R1.fastq", readfile2 = "screen4_R2.fastq", barcode-
file = "Samples4.txt",
    hairpinfile = "sgRNAs4.txt", verbose = TRUE, plotPositions = TRUE)

##  -- Number of Barcodes : 72
##  -- Number of Hairpins : 64751
## Processing reads in screen4_R1.fastq and screen4_R2.fastq.
##  -- Processing 10 million reads
##  -- Processing 20 million reads
##  -- Processing 30 million reads
##  -- Processing 40 million reads
##  -- Processing 50 million reads
##  -- Processing 60 million reads
##  -- Processing 70 million reads
##  -- Processing 80 million reads
##  -- Processing 90 million reads
##  -- Processing 100 million reads
## Number of reads in file screen4_R1.fastq and screen4_R2.fastq: 99427748
##
## The input run parameters are:
##  -- Barcode in forward read: length 8
##  -- Barcode in reverse read: length 8
##  -- Hairpin in forward read: length 20
##  -- Mismatch in barcode/hairpin sequences not allowed.
##
## Total number of read is 99427748
## There are 68128813 reads (68.5209 percent) with barcode matches
## There are 62181626 reads (62.5395 percent) with hairpin matches
## There are 46529785 reads (46.7976 percent) with both barcode and hairpin matches
```



The optional plotPositions argument produces a density plot indicating the position sequences were found in each read. For dual indexing reads and paired end reads, two graphs are created side-by-side, to show the sequence locations of both sets of barcodes.

Note that this dual indexing strategy requires an additional column named *'SequencesRev'* in the file that contains the sample annotation information. Also, readFile2 must be specified, along with position information (barcodeStartRev, barcodeEndRev) for the second index in the second read from each pair (in this case the index can be found in the first 8 bases).

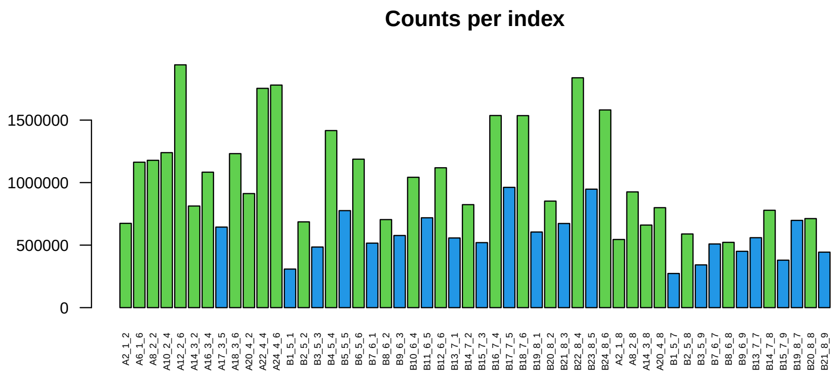
We next filter out sgRNAs and samples with low numbers of reads.

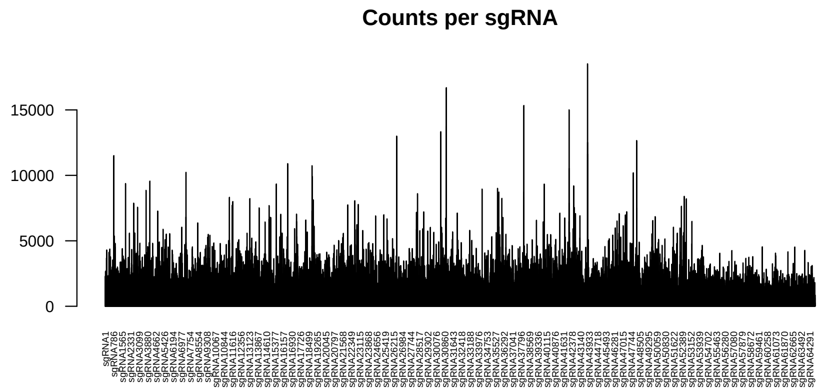
```
x4

## An object of class "DGEList"
## $counts
##        A1_1_1 A2_1_2 A3_1_3 A4_1_4 A5_1_5 A6_1_6 A7_2_1 A8_2_2 A9_2_3 A10_2_4 A11_2_5
## sgRNA1      0     14      0      0      3     37      1     55      0      24       0
## sgRNA2      0     18      0      0      1     23      0     26      0      29       0
## sgRNA3      0     54      0      0      4     52      2    101      0      64       0
## sgRNA4      0     32      0      0      3     56      2     57      0      55       0
## sgRNA5      0      7      0      0      1      3      0      3      0       5       1
##        A12_2_6 A13_3_1 A14_3_2 A15_3_3 A16_3_4 A17_3_5 A18_3_6 A19_4_1 A20_4_2 A21_4_3
## sgRNA1      63       0      23       0      33      22      37       0      39       0
## sgRNA2      43       0      27       0      28      27      31       1      23       0
## sgRNA3     115       0      62       0      65      26      64       0      44       0
## sgRNA4      58       0      48       0      52      20      28       0      44       0
## sgRNA5       5       0       3       0       3       1       7       0       1       0
##        A22_4_4 A23_4_5 A24_4_6 B1_5_1 B2_5_2 B3_5_3 B4_5_4 B5_5_5 B6_5_6 B7_6_1 B8_6_2
## sgRNA1      40       1      66      3      5      5     29     11      5      5      4
## sgRNA2      46       0      35     14     28     19     79     27     44      9     25
## sgRNA3     110       0      73     18     42     30     56     43     47     32     26
## sgRNA4      95       0     110     17     13     13     25     12     23     10     11
## sgRNA5       8       0       3      0      5     13     15      8     27      6      9
##        B9_6_3 B10_6_4 B11_6_5 B12_6_6 B13_7_1 B14_7_2 B15_7_3 B16_7_4 B17_7_5 B18_7_6
## sgRNA1     14       3       9       6       9      11       6       9      12      31
## sgRNA2     13      28      16      46      12      24      14      65      46      70
## sgRNA3     15      36      44      45      39      53      16      60      42      80
## sgRNA4     11      21      15      20      14      13       7      30      16      23
## sgRNA5      7       4      11      19       1       5       0       3       7      13
##        B19_8_1 B20_8_2 B21_8_3 B22_8_4 B23_8_5 B24_8_6 A1_1_7 A2_1_8 A3_1_9 A7_2_7 A8_2_8
## sgRNA1       6      13      10      26      14      19      0     18      0      0     39
## sgRNA2      18      32      12      97      37      73      0     12      0      0     19
## sgRNA3      32      30      31      65      60      76      0     30      0      4     76
## sgRNA4      15      18      20      27      19      36      0     27      0      1     41
## sgRNA5       1       7       3      18      25      12      0      2      0      0      2
##        A9_2_9 A13_3_7 A14_3_8 A15_3_9 A19_4_7 A20_4_8 A21_4_9 B1_5_7 B2_5_8 B3_5_9 B7_6_7
## sgRNA1      0       0      17       0       0      33       0      6      6      5     11
## sgRNA2      0       0      15       0       0      17       0     10     20     10     11
## sgRNA3      0       0      33       0       1      39       0     10     19     25     20
## sgRNA4      0       0      34       0       1      30       0     10      9      9      8
## sgRNA5      0       0       9       0       0       0       0      0      9      4      4
##        B8_6_8 B9_6_9 B13_7_7 B14_7_8 B15_7_9 B19_8_7 B20_8_8 B21_8_9
## sgRNA1      5     17       2      11       3      10      15       9
## sgRNA2     23      7       7      22      17      20      22      11
## sgRNA3     14     31      32      38      25      29      18      17
## sgRNA4     11      5      11       9       8      13      14      12
## sgRNA5     11      3       4       2       0       3      11       8
## 64746 more rows ...
##
## $samples
##       ID lib.size norm.factors SequencesReverse   group Infection Replicate IndexF IndexR
## 1 A1_1_1      223            1         TAAGGCGA    Drug         1         1      1      1
## 2 A2_1_2   687528            1         CGTACTAG Control         1         1      1      2
## 3 A3_1_3     1485            1         AGGCAGAA    Drug         1         1      1      3
## 4 A4_1_4     2550            1         TCCTGAGC Control         1         1      1      4
## 5 A5_1_5    71348            1         GGACTCCT    Drug         1         1      1      5
## 67 more rows ...
##
## $genes
##            ID            Sequences Gene
## sgRNA1 sgRNA1 TACCCTGGGACTGTACCCCC   99
## sgRNA2 sgRNA2 ACCCTTGCTGCACGACCTGC   99
## sgRNA3 sgRNA3 TCGCTCGCCCCGCTCTTCCT   99
## sgRNA4 sgRNA4 TGACGCCTCGGACGTGTCTG   19
## sgRNA5 sgRNA5 CGTCATAGCCAATCTTCTTC   19
## 64746 more rows ...
```

```
table(x4$samples$group)
```

```
##
## Control    Drug
##      32      40
```

```
# Filter sgRNAs and samples with low counts Need a CPM greater than 5 in 15 or more sam-
ples
# to keep sgRNAs
selr = rowSums(cpm(x4$counts) > 5) >= 15
# Need at least 100,000 reads to keep a given sample
selc = colSums(x4$counts) >= 1e+05
x4 = x4[selr, selc]

# Set up drug treatment colours
cols = as.numeric(x4$samples$group) + 2

# Plot number of sgRNAs that could be matched per sample and total for each sgRNA across
# all samples
par(mfrow = c(2, 1))
barplot(colSums(x4$counts), las = 2, main = "Counts per index", col = cols, cex.names = 0.5,
    cex.axis = 0.8)
barplot(rowSums(x4$counts), las = 2, main = "Counts per sgRNA", cex.names = 0.5, cex.axis = 0.8)
```





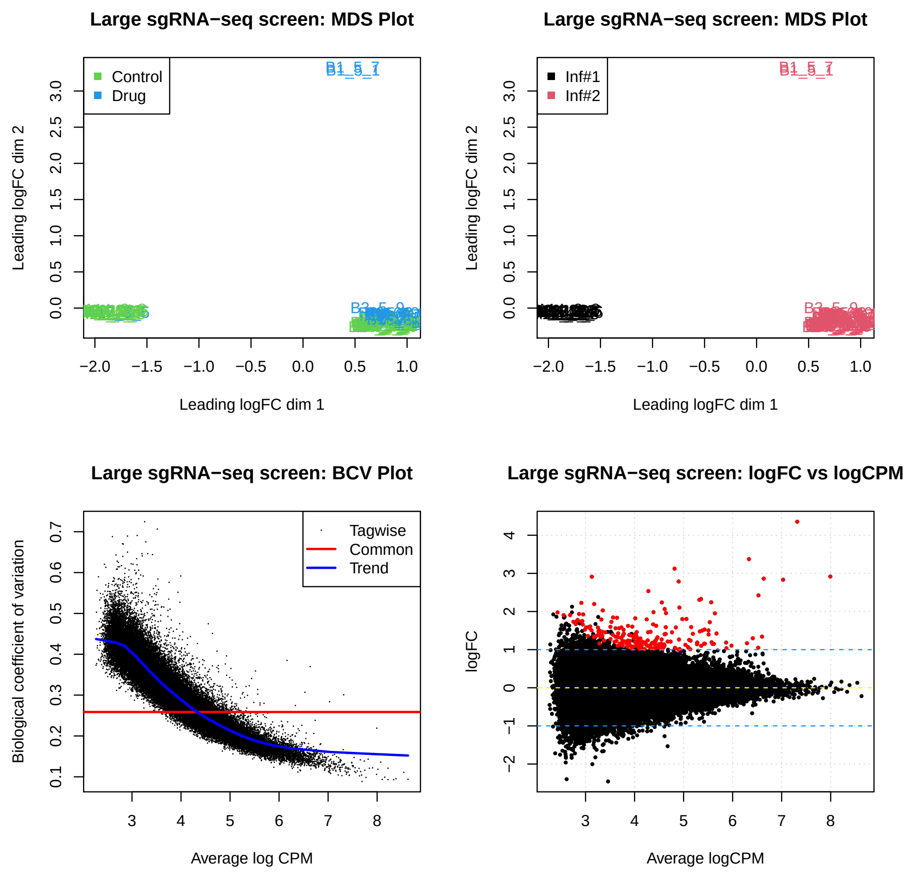
Next we make a multidimensional scaling plot to assess the consistency between replicate samples. There is a clear separation between the two infections, indicating the need to incorporate an effect for this in the GLM. A design matrix is set up for the GLM analysis, and the sgRNA-specific variation is estimated and plotted (while taking into account both drug treatment and infection number).

We use the function glmFit to fit the sgRNA-specific models and glmLRT to do the testing between the drug treated and control samples. The top ranked sgRNAs are listed using the topTags function and sgRNAs with FDR < 0.0001 (Benjamini and Hochberg, 1995) and log-fold-change ≥ 1 are highlighted on a plot of log-fold-change versus log-counts-per-millions by the plotSmear function. Since this is a positive screen, we highlight over-represented sgRNAs (i.e. those with positive log-fold-changes) since the model is parameterized to compare drug treatment versus control (coefficient 2 in the design mtrix).

```
# Make an MDS plot to visualise relationships between replicate samples Set up infec-
tion #
# colours
cols2 = x4$samples$Infection

par(mfrow = c(2, 2))
plotMDS(x4, col = cols, main = "Large sgRNA-seq screen: MDS Plot")
legend("topleft", legend = c("Control", "Drug"), col = c(3, 4), pch = 15)
plotMDS(x4, col = cols2, main = "Large sgRNA-seq screen: MDS Plot")
```

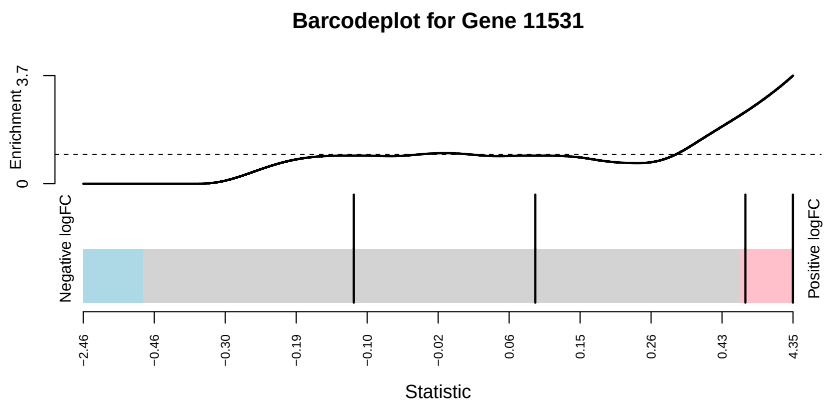
```
legend("topleft", legend = c("Inf#1", "Inf#2"), col = c(1, 2), pch = 15)

# Begin differential representation analysis We will use GLMs in edgeR in this case since
# there are more than 2 groups Set up design matrix for GLM
treatment = as.factor(x4$samples$group)
infection = as.factor(x4$samples$Infection)
des = model.matrix(~treatment + infection)
des[1:5, ]

##   (Intercept) treatmentDrug infection2
## 1           1             0          0
## 2           1             0          0
## 3           1             0          0
## 4           1             0          0
## 5           1             0          0

colnames(des)[2:3] <- c("Drug", "Infection2")

# Estimate dispersions
xglm = estimateDisp(x4, des)
sqrt(xglm$common.disp)

## [1] 0.259

# Plot BCVs versus abundance
plotBCV(xglm, main = "Large sgRNA-seq screen: BCV Plot")

# Fit negative bionomial GLM
fit = glmFit(xglm, des)
# Carry out Likelihood ratio test
lrt = glmLRT(fit, coef = 2)

# Show top ranked sgRNAs
topTags(lrt)

## Coefficient:  Drug
##                    ID           Sequences  Gene logFC logCPM  LR    PValue       FDR
## sgRNA816    sgRNA816 TCCGAACTCCCCCTTCCCGA   269  4.35   7.32 682 2.33e-
150 1.31e-145
## sgRNA4070   sgRNA4070 GTTGTGCTCAGTACTGACTT  1252  2.92   7.99 662 6.06e-
146 1.71e-141
## sgRNA6351   sgRNA6351 AAAAACGTATCTATTTTTAC  1957  3.37   6.33 413  6.62e-
92  1.24e-87
## sgRNA12880 sgRNA12880 CTGCACCGAAGAGAGCTGCT  3979  2.83   7.03 317  7.09e-
71  1.00e-66
## sgRNA23015 sgRNA23015 CAATTTGATCTCTTCTACTG  6714  3.12   4.82 230  5.35e-
52  6.03e-48
## sgRNA62532 sgRNA62532 AAACACGTCCAGTGCAGCCC 19612  2.79   4.90 218  2.51e-
49  2.36e-45
## sgRNA3887   sgRNA3887 AACGCTGGACTCGAATGGCC  1194  2.31   5.32 205  1.36e-
46  1.09e-42
## sgRNA38819 sgRNA38819 TACGTTGTCGGGCGCCGCCA 11531  2.42   6.53 204  2.62e-
46  1.85e-42
```

```
## sgRNA19299 sgRNA19299 GGGGTCTTACCCGAGGCTCC  5732 1.95   5.64 203  4.46e-
46   2.79e-42
## sgRNA52924 sgRNA52924 CCACCGCGTTCCACTTCTTG 16395  2.86   6.64 194  5.47e-
44   3.08e-40

# Select sgRNAs with FDR < 0.0001 and logFC <= -1 to highlight on plot
thresh = 1e-04
lfc = 1
top4 = topTags(lrt, n = Inf)
top4ids = top4$table[top4$table$FDR < thresh & top4$table$logFC >= lfc, 1]

# Plot logFC versus logCPM
plotSmear(lrt, de.tags = top4ids, pch = 20, cex = 0.6, main = "Large sgRNA-seq screen: logFC vs logCPM")
abline(h = c(-1, 0, 1), col = c("dodgerblue", "yellow", "dodgerblue"), lty = 2)
```



We finish this analysis by summarising data from multiple sgRNAs in order to get a gene-by-gene ranking, rather than a sgRNA-specific one. The *camera* gene-set test (Wu and Smyth, 2012) is used for this purpose. As before, the collection of sgRNAs that target a specific gene can be regarded as a 'set'. In the code below, we restrict our analysis to genes with more than 3 sgRNAs. A barcode plot, highlighting the rank of sgRNAs for a given gene relative to the entire data set is generated for the top-ranked gene (11531). Abundance of sgRNAs targeting this gene tends to increase with drug treatment (FDR=0.0003).

```
# Carry out camera gene-set analysis
genesymbols = x4$genes[, 3]

genesymbollist = list()
unq = unique(genesymbols)
unq = unq[!is.na(unq)]
for (i in unq) {
    sel = genesymbols == i & !is.na(genesymbols)
    if (sum(sel) > 3)
        genesymbollist[[i]] = which(sel)
}

# Run camera for all genes
camera.res = camera(xglm, index = genesymbollist, des, contrast = 2)

# Display results for top ranked genes
camera.res[1:10, ]

##       NGenes Direction   PValue      FDR
## 19612      5        Up 1.11e-08 6.14e-05
## 8370       4        Up 3.79e-06 1.05e-02
## 8808       4        Up 1.88e-05 3.14e-02
## 11531      4        Up 2.27e-05 3.14e-02
## 3979       4        Up 2.89e-05 3.19e-02
## 10386      4        Up 1.30e-04 1.19e-01
## 10784      4        Up 1.74e-04 1.38e-01
## 2005       4        Up 2.60e-04 1.76e-01
## 4086       4        Up 2.87e-04 1.76e-01
## 11412      4        Up 3.86e-04 2.13e-01

# Make a barcode plot for an example that ranks highly Gene 11531
par(mfrow = c(1, 1))
barcodeplot(lrt$table$logFC, index = genesymbollist[[11531]], main = "Barcodeplot for Gene 11531",
    labels = c("Negative logFC", "Positive logFC"), quantile = c(-0.5, 0.5))
```



7 Analysis of a CRISPR-Cas9 knockout screen from Shalem *et al.* (2014)

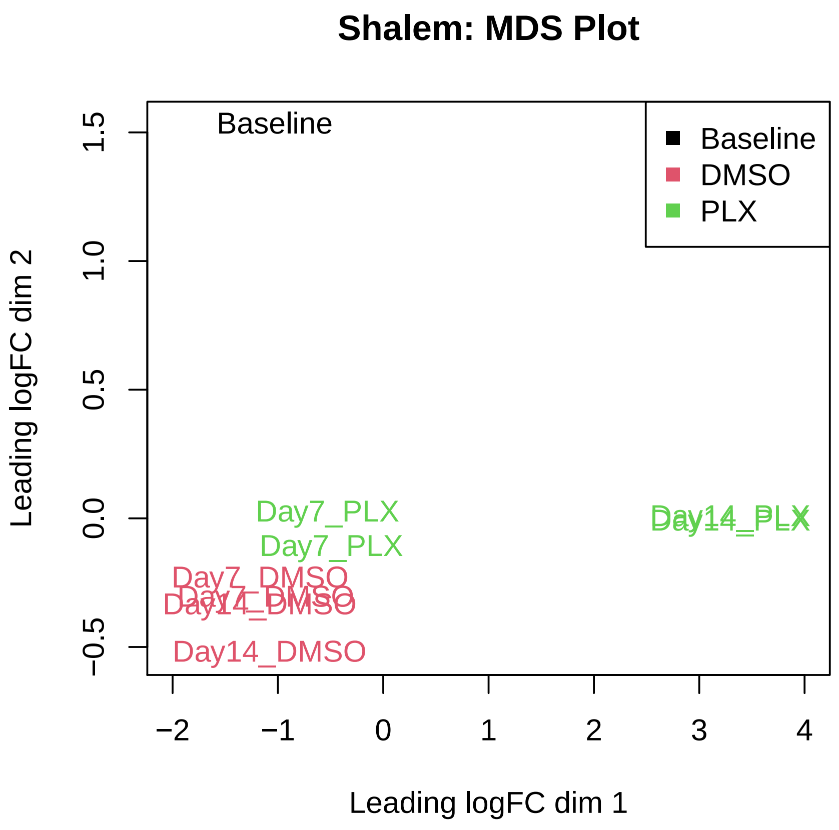
The final analysis is of a recently published CRISPR-Cas9 knockout screen published by Shalem *et al* (2014).

The goal of the screen analysed below was to identify genes whose loss is involved in resistance to vemurafenib (PLX) in a melanoma model. A genome-wide library of sgRNAs (∼64,000) targeting ∼18,000 genes was used in the melanoma cell-line A375. Samples at baseline (Day 0), Day 7 and Day 14 for control (DMSO treated) and vemurafenib (PLX) were available. sgRNAs/genes that consistently increase in representation in the PLX samples compared to the DMSO samples in the biological replicates are of interest.

We thank Ophir Shalem and Feng Zhang for providing access to this data set, which was downloaded from http://genome-engineering.org/gecko/?page_id=114.

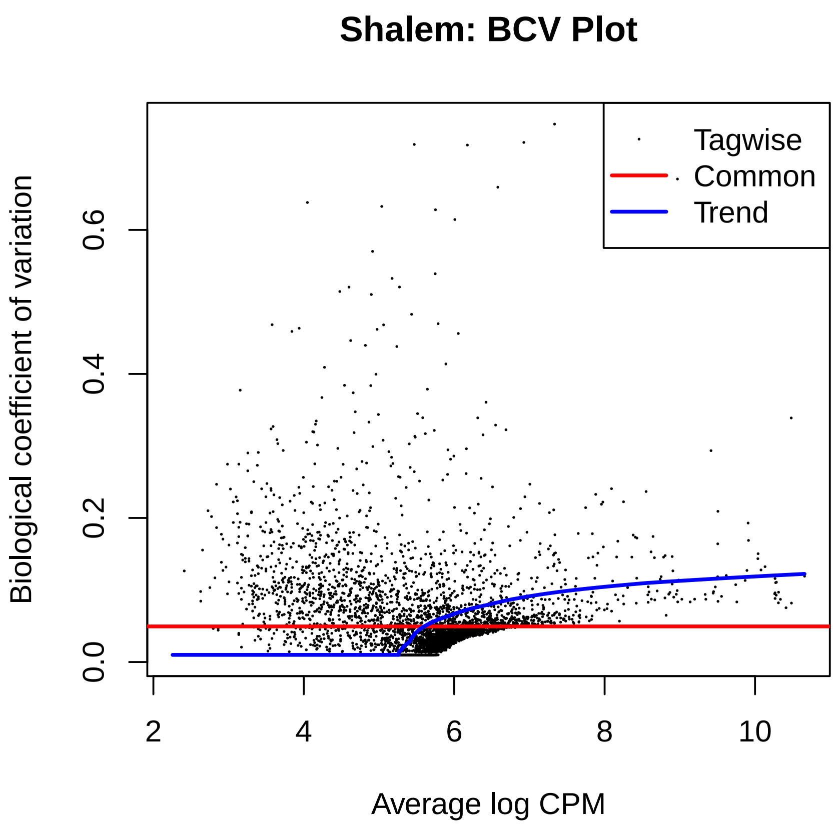
We first read in the data downloaded from the URL above in preparation for an sgRNA-level analysis. The data available has been normalized, and was rounded to ensure we are dealing with integer values. A ceiling of 5000 was put on the counts (a small number sgRNAs had values up to ∼ 82,000). A multidimensional scaling plot was generated to see if the samples cluster by treatment (DMSO/PLX for Day 7/Day 14).

```
## Read in the table of counts
shalem = read.table("norm_read_count_A375", header=TRUE, sep="\t", as.is=TRUE)

counts = matrix(NA, nrow(shalem), 9)
for(i in 1:9)
  counts[,i] = round(shalem[,-(1:3)][,i],0)
## Set max counts to 5000
counts[counts>5000] = 5000
colnames(counts) = colnames(shalem)[-(1:3)]
rownames(counts) = shalem[,2]
dim(counts)

## [1] 64076     9

## Make DGE list containing sgRNA counts
x5 = new("DGEList")
x5$counts = counts

## Add sample annotation data
x5$samples = data.frame("SampleID"=colnames(x5$counts),
               "group"=as.factor(c("Baseline", rep(c("Day7_DMSO", "Day14_DMSO", "Day7_PLX", "Day14_PLX"), e
                      "lib.size"=colSums(x5$counts),
                      "norm.factors" = rep(1,9))
x5$genes = shalem[,1:3]
rownames(x5$genes) = shalem[,2]

# Filter sgRNAs with low counts
sel = rowSums(cpm(x5$counts)>5)>=2
x5 = x5[sel,]

## Plot Multi-dimensional scaling of data to visualise
## relationships between replicate samples
plotMDS(x5, labels=x5$samples$group, xlim=c(-2,4),
        col=c(1,rep(c(2,3),each=4)), main="Shalem: MDS Plot")
legend("topright", legend=c("Baseline", "DMSO", "PLX"), col=1:3, pch=15)
```



A design matrix is set up for the GLM analysis (McCarthy *et al.* 2012), and the sgRNA-specific variation is estimated and plotted (while taking into account the group structure). The baseline sample is used to estimate the intercept term in the model. We use the functions glmFit to fit the sgRNA-specific models and glmLRT functions to do the testing between the PLX and DMSO samples at Day 7 and Day 14 respectively. Single guide RNAs with false discovery rate (FDR) <0.0001 (Benjamini and Hochberg, 1995) and log-fold-change below −1 are listed using the topTags function and highlighted on a plot of log-fold-change versus log-counts-per-millions by the plotSmear function.

```
## Assess differential representation between Day 14 PLX and Day14 DMSO samples
## and Day 7 PLX and Day 7 DMSO samples using GLM in edgeR
## Set up design matrix for GLM
des = model.matrix(~x5$samples$group)
colnames(des)[2:ncol(des)] = c("Day14_DMSO", "Day14_PLX", "Day7_DMSO", "Day7_PLX")
des

##    (Intercept) Day14_DMSO Day14_PLX Day7_DMSO Day7_PLX
## 1          1         0         0         0        0
## 2          1         0         0         1        0
## 3          1         0         0         1        0
## 4          1         1         0         0        0
## 5          1         1         0         0        0
## 6          1         0         0         0        1
## 7          1         0         0         0        1
## 8          1         0         1         0        0
## 9          1         0         1         0        0
## attr(,"assign")
## [1] 0 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$`x5$samples$group`
## [1] "contr.treatment"

## Estimate variability in the screen amongst replicate samples
xglm = estimateDisp(x5, des)

## Plot BCVs versus abundance
plotBCV(xglm, main="Shalem: BCV Plot")
```

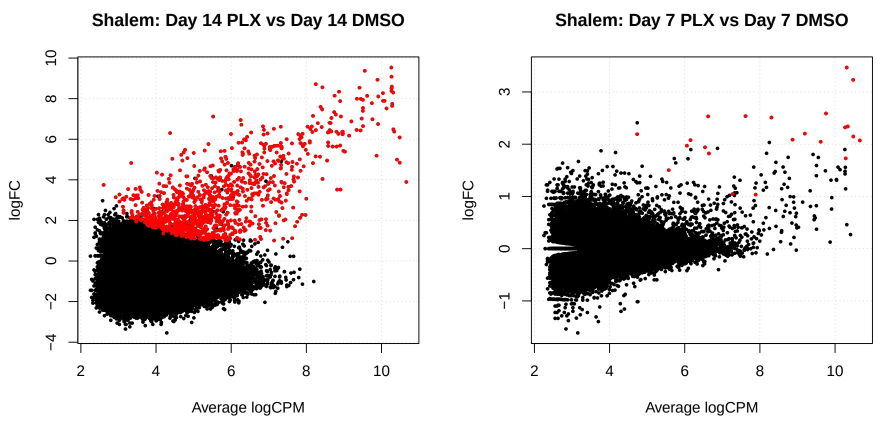


```
## Fit negative bionomial GLM
fit = glmFit(xglm, des)

## Carry out Likelihood ratio test for Day 14 contrast
lrtday14 = glmLRT(fit, des, contrast=c(0,-1,1,0,0))
dt14 = decideTestsDGE(lrtday14)
summary(dt14)

##        -1*Day14_DMSO 1*Day14_PLX
## Down                        23873
## NotSig                      32069
## Up                           2349

## Carry out Likelihood ratio test for Day 7 contrast
lrtday7 = glmLRT(fit, des, contrast=c(0,0,0,-1,1))
dt7 = decideTestsDGE(lrtday7)
summary(dt7)

##        -1*Day7_DMSO 1*Day7_PLX
## Down                          0
## NotSig                    58229
## Up                           62

## Show top ranked sgRNAs for Day 14 contrast
topTags(lrtday14, n=15)

## Coefficient:  -1*Day14_DMSO 1*Day14_PLX
##         gene_name spacer_id          spacer_seq logFC logCPM   LR    PValue       FDR
## s_800      ACTA2    s_800 GGGACAAAAAGACAGCTACG  9.53  10.26 1502  0.00e+00  0.00e+00
## s_37190    NLGN1   s_37190 ATCACAGTCAACTATCGACT  8.50  10.27 1591  0.00e+00  0.00e+00
## s_14313     CUL3   s_14313 GAATCCTGTTGACTATATCC  8.30  10.31 1754  0.00e+00  0.00e+00
## s_14312     CUL3   s_14312 CTTACCTGGATATAGTCAAC  6.99   9.76 1402 7.14e-
307 1.04e-302
## s_35735    MYO1E   s_35735 CAACCTTGTATGAGCCCGAG  9.37   9.55 1400 2.61e-
306 3.04e-302
## s_52770     SNCG   s_52770 GCTCTGTACAACATTCTCCT  8.38  10.27 1381 2.79e-
302 2.71e-298
## s_7274   C1orf27    s_7274 CAAGTTATCCAACTTAGCTT  7.64  10.28 1375 7.06e-
301 5.88e-297
## s_12138   CLDN10   s_12138 ACATGTCCAGGGCGCAGATC  7.99   9.35 1344 2.97e-
294 2.16e-290
## s_36799      NF2   s_36799 GTACTGCAGTCCAAAGAACC  6.37  10.34 1309 1.27e-
286 8.22e-283
## s_47803     RNH1   s_47803 CGGCGTGCATTGCGTGCTCC  6.65   9.51 1286 1.13e-
281 6.56e-278
## s_8730    CACNB2    s_8730 ATCCGATTCCGATGTATCTC  4.99  10.41 1277 1.44e-
279 7.66e-276
## s_33342    MED12   s_33342 CGTCAGCTTCAATCCTGCCA  6.88   9.20 1185 8.91e-
260 4.33e-256
## s_30886   LGALS4   s_30886 GATGGCCTATGTCCCCGCAC  7.47   8.42 1158 7.60e-
254 3.41e-250
## s_33855      MIA   s_33855 GTCTTCACATCGACTTTGCC  7.99   9.45 1156 2.83e-
253 1.18e-249
```

```
## s_29387   KIF13A  s_29387 AGCAGCTGGGCCTTATTCCA  6.17   9.14 1149 6.30e-
252 2.45e-248

## Show top ranked sgRNAs for Day 7 contrast
topTags(lrtday7, n=15)

## Coefficient:  -1*Day7_DMSO 1*Day7_PLX
##         gene_name spacer_id          spacer_seq logFC logCPM    LR   PValue      FDR
## s_14313     CUL3  s_14313 GAATCCTGTTGACTATATCC  3.47  10.31 232.6 1.61e-
52 9.38e-48
## s_36796      NF2  s_36796 AAACATCTCGTACAGTGACA  2.15  10.48 203.6 3.40e-
46 9.92e-42
## s_14312     CUL3  s_14312 CTTACCTGGATATAGTCAAC  2.59   9.76 146.9 8.20e-
34 1.59e-29
## s_36799      NF2  s_36799 GTACTGCAGTCCAAAGAACC  2.34  10.34 144.4 2.89e-
33 4.21e-29
## s_36798      NF2  s_36798 CCTGGCTTCTTACGCCGTCC  2.07  10.66 119.1 9.74e-
28 1.14e-23
## s_55205    TADA1  s_55205 AGCTCATAGACTTCTCACAC  2.54   7.62  85.6 2.18e-
20 2.12e-16
## s_14314     CUL3  s_14314 GACCTAAAATCATTAACATC  2.51   8.31  69.8 6.64e-
17 5.53e-13
## s_33342    MED12  s_33342 CGTCAGCTTCAATCCTGCCA  2.20   9.20  66.3 3.83e-
16 2.79e-12
## s_55215    TADA3  s_55215 TCAGTAACTCCTCAAGTGTG  1.82   6.64  63.3 1.76e-
15 1.14e-11
## s_55204    TADA1  s_55204 ACTGGGCTAACCTAAAGCTG  2.53   6.62  53.2 3.07e-
13 1.79e-09
## s_8980     CAND1   s_8980 TCACCTAAAGTCCTTGTCGC  2.08   6.15  52.7 3.95e-
13 2.09e-09
## s_2661    ANKZF1   s_2661 GGGAACATTATAAGCTTGAC  2.19   4.73  49.7 1.78e-
12 8.65e-09
## s_55276    TAF5L  s_55276 CAGCCCTATTCTGCAGAACG  1.94   6.54  47.3 6.16e-
12 2.76e-08
## s_64856      ZP1  s_64856 ACCAGCTCATCTATGAGAAC  2.32  10.27  46.5 8.99e-
12 3.74e-08
## s_17709    EHMT2  s_17709 TCAGATTCATCCCCAATGAG  2.05   9.62  43.1 5.10e-
11 1.98e-07
```

```
## Select sgRNAs with FDR < 0.0001 and logFC < -1 to highlight on plot
thresh = 0.0001
lfc = 1

top14 = topTags(lrtday14, n=Inf, sort.by="logFC")
top7 = topTags(lrtday7, n=Inf, sort.by="logFC")

sum(top14$table[,8]<thresh)

## [1] 4536

sum(top14$table[,8]<thresh & top14$table[,4]>lfc)

## [1] 1135
```

```
sum(top7$table[,8]<thresh)

## [1] 22

sum(top7$table[,8]<thresh & top7$table[,4]>lfc)

## [1] 22

topids14 = as.character(top14$table[top14$table$FDR<thresh & top14$table$logFC>lfc,2])
topids7 = as.character(top7$table[top7$table$FDR<thresh & top7$table$logFC>lfc,2])

## Make a plot of logFC versus logCPM for Day 14 contrast
par(mfrow=c(1,2))
plotSmear(lrtday14, de.tags=topids14, pch=20, cex=0.6,
          main="Shalem: Day 14 PLX vs Day 14 DMSO")

## Make a plot of logFC versus logCPM for Day 7 contrast
plotSmear(lrtday7, de.tags=topids7, pch=20, cex=0.6,
          main="Shalem: Day 7 PLX vs Day 7 DMSO")
```

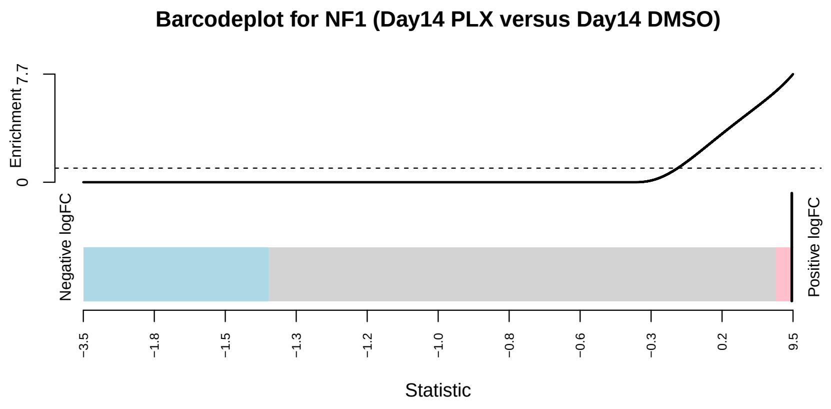


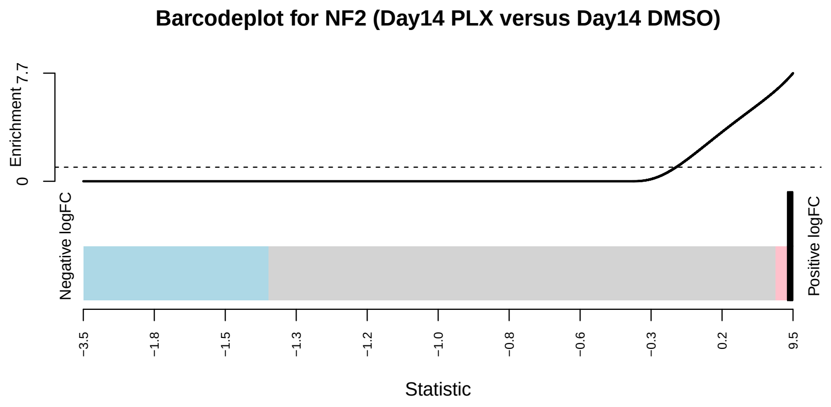
We complete the analysis by summarising the data at the gene-level using the *roast* (Wu *et al.* 2010) gene-set test. The collection of individual sgRNAs that target a specific gene are regarded as a 'set'. Genes with multiple sgRNAs that go down in the Day 14 'PLX versus DMSO' comparison are of primary interest. The genes *NF1*, *MED12*, *NF2*, *CUL3*, *TADA2B*, and *TADA1* are examined first, as they were reported as key genes finding in the original paper, followed by an analysis for all genes.

```
## Carry out roast gene-set analysis
genesymbols = x5$genes[,1]

genesymbollist = list()
unq = unique(genesymbols)
unq = unq[!is.na(unq)]
for(i in unq) {
  sel = genesymbols==i & !is.na(genesymbols)
  if(sum(sel)>=3)
    genesymbollist[[i]] = which(sel)
}
```

```
## Begin with sgRNAs targeting NF1, MED12, NF2, CUL3, TADA2B and TADA1
## that were reported as top hits in the paper
topgenes = c("NF1", "MED12", "NF2", "CUL3", "TADA2B", "TADA1")

set.seed(3042014)
for(i in topgenes) {
  ind = genesymbols==i
  cat("Roast results for Day 14 contrast", i, "\n")
  print(roast(xglm, index=ind, des, contrast=c(0,-1,1,0,0), nrot=9999))
}

## Roast results for Day 14 contrast NF1
##           Active.Prop P.Value
## Down                0   1e+00
## Up                  1   1e-04
## UpOrDown            1   2e-04
## Mixed               1   2e-04
## Roast results for Day 14 contrast MED12
##           Active.Prop P.Value
## Down                0   1e+00
## Up                  1   5e-05
## UpOrDown            1   1e-04
## Mixed               1   1e-04
## Roast results for Day 14 contrast NF2
##           Active.Prop P.Value
## Down                0   1e+00
## Up                  1   5e-05
## UpOrDown            1   1e-04
## Mixed               1   1e-04
## Roast results for Day 14 contrast CUL3
##           Active.Prop P.Value
## Down            0.167   1e+00
## Up              0.500   5e-05
## UpOrDown        0.500   1e-04
## Mixed           0.667   1e-04
## Roast results for Day 14 contrast TADA2B
##           Active.Prop P.Value
## Down                0   1e+00
## Up                  1   5e-05
## UpOrDown            1   1e-04
## Mixed               1   1e-04
## Roast results for Day 14 contrast TADA1
##           Active.Prop P.Value
## Down                0   1e+00
## Up                  1   1e-04
## UpOrDown            1   2e-04
## Mixed               1   2e-04

## Make a barcode plot for NF1
nf1 = genesymbols=="NF1"
par(mfrow=c(2,1))
```

```
barcodeplot(lrtday14$table$logFC, index=nf1,
        main="Barcodeplot for NF1 (Day14 PLX versus Day14 DMSO)",
        labels=c( "Negative logFC", "Positive logFC"))

## Make a barcode plot for NF2
nf2 = genesymbols=="NF2"
barcodeplot(lrtday14$table$logFC, index=nf2,
        main="Barcodeplot for NF2 (Day14 PLX versus Day14 DMSO)",
        labels=c( "Negative logFC", "Positive logFC"))
```





```
## Run mroast for all genes for Day 14 contrast
set.seed(3042014)
roast.res.day14 = mroast(xglm, index=genesymbollist,
                des, contrast=c(0,-1,1,0,0), nrot=9999)

## Display ranked results for top ranked genes that drop out in the screen
roast.res.day14[roast.res.day14$Direction=="Up",][1:10,1:6]

##        NGenes PropDown PropUp Direction PValue      FDR
## MED12       4        0      1        Up  1e-04 0.000576
## MED15       4        0      1        Up  1e-04 0.000576
## NF2         4        0      1        Up  1e-04 0.000576
## CCDC101      3       0      1        Up  1e-04 0.000576
## KCTD10       3       0      1        Up  1e-04 0.000576
## PGD          3       0      1        Up  1e-04 0.000576
## SMARCB1      3       0      1        Up  1e-04 0.000576
## TADA1        3       0      1        Up  1e-04 0.000576
## TADA2B       3       0      1        Up  1e-04 0.000576
## TAF6L        3       0      1        Up  1e-04 0.000576
match(topgenes, rownames(roast.res.day14[roast.res.day14$Direction=="Up",]))
## [1] NA  1  3 41  9  8
sum(roast.res.day14$Direction=="Up" & roast.res.day14$FDR<0.001)
## [1] 94
```

8 Further reading

Studies that have made use of our software in their screen analyses include Sheridan *et al.* (2015), Ziller *et al.* (2015) [both shRNA-seq pooled screens] and Toledo *et al.* (2015) [CRISPR-Cas9 knockout screen].

Since publication of our work, a number of other groups have also advocated for the use of RNA-seq style analysis workflows that assume a negative binomial distribution of the underlying count data in CRISPR-Cas9 screen analyses. These include Li *et al.* (2014) in the MAGeCK software and Winter *et al.* (2015) in the caRpools software.

9 References

Bassik MC *et al.*. *A systematic mammalian genetic interaction map reveals pathways underlying ricin susceptibility*, Cell. 2013, 152:909-22.

Benjamini Y, Hochberg Y. *Controlling the false discovery rate: a practical and powerful approach to multiple testing*. Journal of the Royal Statistical Society Series B. 1995, 57:289-300.

Dai Z *et al. edgeR: a versatile tool for the analysis of shRNA-seq and CRISPR-Cas9 genetic screens*, F1000Research. 2014, 3:95

Li W *et al. MAGeCK enables robust identification of essential genes from genome-scale CRISPR/Cas9 knockout screens*, Genome Biol. 2014;15(12):554.

McCarthy DJ, Chen Y, Smyth GK. *Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation*. Nucleic Acids Research. 2012, 40:4288-97.

Robinson MD, McCarthy DJ, Smyth GK. *edgeR: a Bioconductor package for differential expression analysis of digital gene expression data*, Bioinformatics. 2010, 26:139-140.

Robinson MD, Oshlack A. *A scaling normalization method for differential expression analysis of RNA-seq data*. Genome Biol. 2010, 11(3):R25

Robinson MD, Smyth GK. *Moderated statistical tests for assessing differences in tag abundance*, Bioinformatics. 2007, 23:2881-2887.

Robinson MD and Smyth GK. *Small-sample estimation of negative binomial dispersion, with applications to SAGE data*, Biostatistics. 2008, 9:321-332.

Shalem O *et al. CRISPR-Cas9 Knockout Screening in Human Cells*, Science. 2014, 343:84-7.

Sheridan JM *et al. A pooled shRNA screen for regulators of primary mammary stem and progenitor cells identifies roles for Asap1 and Prox1*, BMC Cancer. 2015, 15:221.

Sullivan KD *et al. ATM and MET kinases are synthetic lethal with nongenotoxic activation of p53*, Nature Chemical Biology. 2012, 8:646-54.

Toledo CM *et al. Genome-wide CRISPR-Cas9 Screens Reveal Loss of Redundancy between PKMYT1 and WEE1 in Glioblastoma Stem-like Cells*. Cell Reports. 13:2425-39.

Wang T *et al. Genetic Screens in Human Cells Using the CRISPR/Cas9 System*, Science. 2014, 343:80-4.

Winter J *et al. caRpools: an R package for exploratory data analysis and documentation of pooled CRISPR/Cas9 screens*, Bioinformatics. 2015 Oct 27. pii: btv617. [Epub ahead of print].

Wu D *et al. ROAST: rotation gene set tests for complex microarray experiments*, Bioinformatics, 2010, 26(17):2176-82.

Wu D and Smyth GK. *Camera: a competitive gene set test accounting for inter-gene correlation*, Nucleic Acids Research, 2012, 40:e133.

Xie Y. *knitr: A Comprehensive Tool for Reproducible Research in R* In Victoria Stodden, Friedrich Leisch and Roger D. Peng, editors, Implementing Reproducible Computational Research. Chapman and Hall/CRC, 2013, ISBN 978-1466561595.

Ziller MJ *et al. Dissecting neural differentiation regulatory networks through epigenetic footprinting*, Nature. 2015, 518:355-9.

Zuber J *et al. RNAi screen identifies Brd4 as a therapeutic target in acute myeloid leukaemia*, Nature. 2011, 478:524-8.

10 Software information

A summary of the packages used to complete this case study is given below. The *knitr* package (Xie, 2013) was used to generate this vignette.

```
sessionInfo()

## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18362)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Australia.1252  LC_CTYPE=English_Australia.1252
## [3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Australia.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.30
##
## loaded via a namespace (and not attached):
##  [1] BiocManager_1.30.10 compiler_4.0.2      BiocStyle_2.16.1    magrittr_1.5
##  [5] formatR_1.7         htmltools_0.5.0     tools_4.0.2         yaml_2.2.1
##  [9] rmarkdown_2.4       stringi_1.5.3       highr_0.8           digest_0.6.25
## [13] stringr_1.4.0       xfun_0.18           rlang_0.4.7         evaluate_0.14
```