Differential Expression Analysis with DEVis

Contents

Installation

DEVis can be installed using the following commands. This section will be updated after CRAN submission.

```r
#Install DESeq2 dependency from bioconductor.
source("https://bioconductor.org/biocLite.R")
biocLite("DESeq2")

#Install devtools to allow installation from GitHub
if (!require("devtools")) install.packages("devtools")

#Install DEvis from GitHub repository.
devtools::install_github("price0416/DEvis/DEvis")

#Load the package.
library(DEVis)
```

- *Note: DEvis requires DESeq2 as a dependency. However, some of DESeq2's dependencies require that compilers such as gcc or gfortran be installed on your computer. To address any issues with DESeq2 install, ensure that you have the latest version of xcode installed (OS X) or rtools (windows).*

Introduction

The DEVis packakge is a comprehensive toolset for data aggregation, visual analytics, exploratory analysis, and project management that builds upon the DESeq2 differential expression package. DEVis provides a framework for the analysis of multi-factor experiments, increasing the power and potential of differential expression analysis while reducing workload and potential for human error. This package implements a wide range of data visualization tools, as well as data aggregation, transformation, and selection methods that smoothly integrate with the DESeq2, making extensive analysis and interpretation a more seamless process. DEVis also implements an automated result management system that automatically organizes and stores results, further simplifying the challenge of complex transcriptomic analysis and the creation of publication-quality figures. The visualizations provided by DEVis incorporate parameters that can be used to provide control over the data being visualized in sensible ways, with each visualization offering several layout and color schemes, unique filtering, sorting, and subset parameters for displaying and retrieving data, making DEVis visualizations powerful functions for navigating and retrieving transcriptomic data within the context of a complex experiment, rather than just simple plots.

Data Aggregation

Data aggregation is one of the core features of the DEVis package that makes it possible to examine and extract meaningful data from complex result sets. For example, analysis of time-series data currently requires the identification of differentially expressed genes for data from each time point with regard to a control sample, which may or may not also consist of time controlled data. This type of experiment will produce a potentially different set of differentially expressed genes for each time point contrast, meaning that direct comparisons between gene sets will not be immediately possible. DEVis makes it possible to combine multiple result sets based on experimental conditions using set-based merging. This aggregated result set can then be used to examine changes in expression across multiple result sets and provides a master result set that can be filtered, subset, sorted, and visualized using other DEVis methods.

Visualization

DEVis provides a set of visualization tools for exploring and displaying transcriptomic data sets. Visualizations are configurable through user defined parameters, which provide options for data merging, subsetting, sorting, and selection, allowing DEVis visualizations to be used as data subset and retrieval tools. All plots can be saved as high-resolution png or pdf files, and include several color and layout themes that can be used to customize resulting plots and produce publication-ready figures.

Project Management

Several data organization functions are built into the DEVis package. A directory structure is created on initialization containing folders to house data files, such as differentially expressed gene lists, and visualized plots. Users can toggle whether data and plots should be automatically saved to the appropriate directory whenever a visualization is employed, standardizing project results and simplifying project management. Additionally, several functions for exporting selected data are implemented in DEVis.

DEVis Analysis

Example Data

For the purposes of this vignette, we will using sample data from a study titled, "Ebolaviruses Associated with Differential Pathogenicity Induce Distinct Host Responses in Human Macrophages" (Olejnik, J. et. al, 2017).

The aim of this experiment was to uncover differences in host response of human monocyte-derived macrophages (MDMs) to infection with the highly pathogenic EBOV and the presumably nonpathogenic RESTV. Ebola virus (EBOV) and Reston virus (RESTV) are members of the Ebolavirus genus which greatly differ in their pathogenicity. However, while EBOV causes a severe disease in humans, there are no reported disease-associated human cases of RESTV infection, suggesting that RESTV is nonpathogenic for humans.

Macrophages derived from 3 different donors were either infected with Ebola virus (EBOV/Kikwit-95), or infected with Reston virus (RESTV). As controls, cells were treated with lipopolysaccharide (LPS) or were mock infected, with total cellular RNA isolated for analysis at 6h, 1d, and 2d post-infection.

Table 1: Summary of Data

Sample Type	Treatment	6 Hours	1 Day	2 Days
Case	Reston Virus (RESTV)	3	3	3
Case	Ebola Virus (EBOV)	3	3	3
Control	Uninfected (Mock)	3	4	3
Control	Lipopolysaccharide (LPS)	3	3	3

Input Data Format

DEVis requires as input data a matrix of counts for sequencing reads/fragments and a metadata matrix that provides information regarding samples. There are several ways to arrive at such a matrix, such as alignment of data using a read aligner such as STAR and extraction of read counts using the rSubread package's featureCounts function.

For the purposes of this vignette, this count matrix will be referred to as count data, and the accompanying metadata will be referred to as target data, as the primary function of the metadata matrix is to allow for selective targeting of count data. For example, the target data may contain a field identifying which sample belongs to a control condition. Using that information, we can target the corresponding count data for the control condition.

Count data and target data must therefore correspond to one another. That is, the columns of count data, each of which represent a single sample, must correspond to the rows in the target data. This is due to the way that DESeq2 structures data as an extention of the summarizedExperiment data format. For more in depth information, see the DESeq2 vignette on preparing count data and the summarizedExperiment data format.

Examples of count and target data are provided below. In this case, 4 columns of count data are shown and 4 rows of target data are shown. Note the correspondance between row and column names.

Sample format for count data:

	Donor10_EBOV_1D_LPS_6H_S27	Donor10_EBOV_1D_S21	Donor10_EBOV_2D_S25	Donor10_EBOV_6H_S17
1060P11.3	0	0	0	0
A1BG	65	101	87	93
A1BG_AS1	132	113	212	93
A1CF	0	0	0	0
A2M	14149	11579	10482	11431
A2M_AS1	5	0	10	0

Sample format for target data:

	donor	treatment	time	sample	targetID
Donor10_EBOV_1D_LPS_6H_S27	m10	ebov	restim	27	Donor10_EBOV_1D_LPS_6H_S27
Donor10_EBOV_1D_S21	m10	ebov	1d	21	Donor10_EBOV_1D_S21
Donor10_EBOV_2D_S25	m10	ebov	2d	25	Donor10_EBOV_2D_S25
Donor10_EBOV_6H_S17	m10	ebov	6h	17	Donor10_EBOV_6H_S17

Note that the targetID field will automatically be generated by the prep_targets() function to correspond to the row IDs provided in the target metadata file.

Initialization

Let's get started on our example analysis by initializing a new project and loading data. For the purposes of this example we will assume that a count file and target file are available and data will be read from these files.

```r
library(DEVis)

#Define the base directory for this analysis.
base_dir <- "/path/to/base/"

#Generate the directory structure for results.
create_dir_struct(base_dir)

#Specify input directories.
cnt_dir        <- "/path/to/base/counts/"
tgt_dir        <- "/path/to/base/targets/"
init_data_paths(cnt_dir, tgt_dir)

#Specify count and target file names.
count_matrix  <- "count_matrix.txt"
target_matrix <- "target_matrix.csv"

#Set significance cutoffs for p-values and log fold-change.
init_cutoffs(p_signif=0.05, lfc_cut=1.5)

#Read count and target data
count_data  <- prep_counts(count_matrix)
target_data <- prep_targets(target_matrix, delim="c")

#Initialize output mode.
set_output_mode("both")
```

This represents a typical project initialization. Here, the base directory, as well as count and target directories are defined. When the create_dir_struct(base_dir) function is run, the directory structure for housing DEVis results is created. Note that if this directory already exists that it will not be overwritten upon subsequent runs of this command at the same base directory.

The following directory structure is created using ~DEVis as a base directory:

```
#' ~/DEVis/
#'   /results/
#'     /DE/
#'       /boxplot/
#'       /counts/
#'       /data/
#'       /density_plots/
#'       /divergence/
#'       /heatmaps/
#'       /profile_plots/
#'       /series_plots/
#'       /volcano/
#'    /dendrograms/
#'    /geneplots/
#'    /group_stats/
#'    /MDS/
#'      /standard/
#'      /hulls/
#'    /sample_distance/
#'      /euclidian/
#'      /poisson/
```

Next, cutoffs are initialized. The init_cutoffs() function initializes the p-value cutoffs and log2foldChange values for most visualizations, filtering, and aggregation steps. Significant differentially expressed genes will be initially determined based on the provided p-value cutoff, and when appropriate, functions that employ a minimum log fold-change cutoff will use the provided log fold-change cutoff initialized here. In this example we are setting a 0.05 minimum p-value for differentially expressed genes, however this could be reduced to identify only more statistically significant genes if desired.

The output mode is then initialized, which determines how DEVis will display and store visualizations. Three output modes are available, "screen", "file", or "both". Screen will plot visualizations to the screen only, whereas "file" will save the visualization to the appropriate directory in the previously generated directory structure. The "both" option will save the visualization to file and also display it on screen. The "screen" option is ideal for exploratory analysis, as not every visualization will require storage. The "file" option can be used when automating analyses with DEVis. The "both" option is ideal for most analyses, however, automatically organizing and storing data as you proceed with analysis.

Finally, the count and target data are read. DEVis provides functions for reading count and target data in tab or comma delimited formats. In addition to assuring the data are properly formatted, these functions provide the visualizations and functions in DEVis access to count and target data so that behind-the-scenes calculations and reformatting can be performed when necessary without the user having to keep track of formats, subsets, and data transformations required for visualization. For this reason, many functions require that these functions be executed prior to analysis as a part of initialization.

Preparing composite fields for experimental design

Once initialization is complete, we can prepare for differential expression calculation. The first step is to determine the contrasts that need to be made based on the target data. In this case, we have fields representing donor, treatment, time, and sampleID available in our target data. For an experiment such as this, however, it is likely that we will want to look at multiple factors. In this case, we are interested in looking at treatment (infection with evoc, lps, restv, or mock samples) and time points, so we will first create a composite column of target data that merges these two fields.

- *Note: This is a good time in analysis to consider any other fields you may want to look at in tandem downstream. By making several composite fields that may be of interest now, additional options for exploratory analysis will be available later.*

Starting target data:

~>head(target_data)

	donor	treatment	time	sample	targetID
Donor10_EBOV_1D_LPS_6H_S27	m10	ebov	restim	27	Donor10_EBOV_1D_LPS_6H_S27
Donor10_EBOV_1D_S21	m10	ebov	1d	21	Donor10_EBOV_1D_S21
Donor10_EBOV_2D_S25	m10	ebov	2d	25	Donor10_EBOV_2D_S25
Donor10_EBOV_6H_S17	m10	ebov	6h	17	Donor10_EBOV_6H_S17
Donor10_LPS_1D_S20	m10	lps	1d	20	Donor10_LPS_1D_S20
Donor10_LPS_2D_S24	m10	lps	2d	24	Donor10_LPS_2D_S24

```
#Create a composite field for treatment and time.
target_data <- make_composite_field(c("time","treatment"))

#Let's also create a composite field for treatment, time, and donor ID.
target_data <- make_composite_field(c("time","treatment", "donor"))
```

Composite fields added to target data:

~>head(target_data)

	donor	treatment	time	sample	targetID	time_treatment	time_treatment_donor
Donor10_EBOV_1D_LPS_6H_S27	m10	ebov	restim	27	Donor10_EBOV_1D_LPS_6H_S27	restim_ebov	restim_ebov_m10
Donor10_EBOV_1D_S21	m10	ebov	1d	21	Donor10_EBOV_1D_S21	1d_ebov	1d_ebov_m10
Donor10_EBOV_2D_S25	m10	ebov	2d	25	Donor10_EBOV_2D_S25	2d_ebov	2d_ebov_m10
Donor10_EBOV_6H_S17	m10	ebov	6h	17	Donor10_EBOV_6H_S17	6h_ebov	6h_ebov_m10
Donor10_LPS_1D_S20	m10	lps	1d	20	Donor10_LPS_1D_S20	1d_lps	1d_lps_m10
Donor10_LPS_2D_S24	m10	lps	2d	24	Donor10_LPS_2D_S24	2d_lps	2d_lps_m10

Filtering data

In this experiment, some samples time point data was unclear at sequencing time, and those samples were marked "restim" in in the target data's "time" column. For this case, we want to remove these samples from both our count and target data so they are not included in the analysis. DEVis provides a function for this kind of filtering.

```
#Keep timepoints 1d, 2d, and 6h, but throw away "restim" samples.
subset  <- keep_data_subset(counts=count_data,
                            targets=target_data,
                            target_count_id_map="targetID",
                            target_keep_col="time",
                            target_keep_val=c("1d","2d","6h"))

#Update count and target data to the filtered set.
count_data  <- subset[[1]]
target_data <- subset[[2]]
```

The data are now filtered to have only the samples we are interested in. For more information on this function, see ?keep_data_subset.

DESeq2 Object Preparation

We can now use the composite column we created previously, "time_treatment", as the experimental design for differential expression. You can generally determine what your design field should be by considering the contrasts you want to make with your data. In this case, we want to ask, "How do data differ for patients infected with different viruses at different time points?" Based on this question, we can know to use a composite field that takes both time and treatment into account for our experimental design. Design choice can be a deep topic, however, for more information on experimental designs, see the DESeq2 vignette.

```r
#Prepare our DESeq2 Object based on our data and experimental design.
dds <- prep_dds_from_data(count_input=count_data,
                          target_input=target_data,
                          experiment_design= ~ time_treatment,
                          stabilization="vst")
```

The prep_dds_from_data() function provided by DEVis combines several steps involved in creating and preparing a DESeq2 object. This function creates the DESeq2 object based on count and target data using the provided experimental design. It also performs TMM normalization, and variance or rlog stabilization. This function can optionally merge replicates as well. For more information see, ?prep_dds_from_data.

Once this step is complete, DEVis should have calculated and stored all of the necessary information to begin visualization of the data.

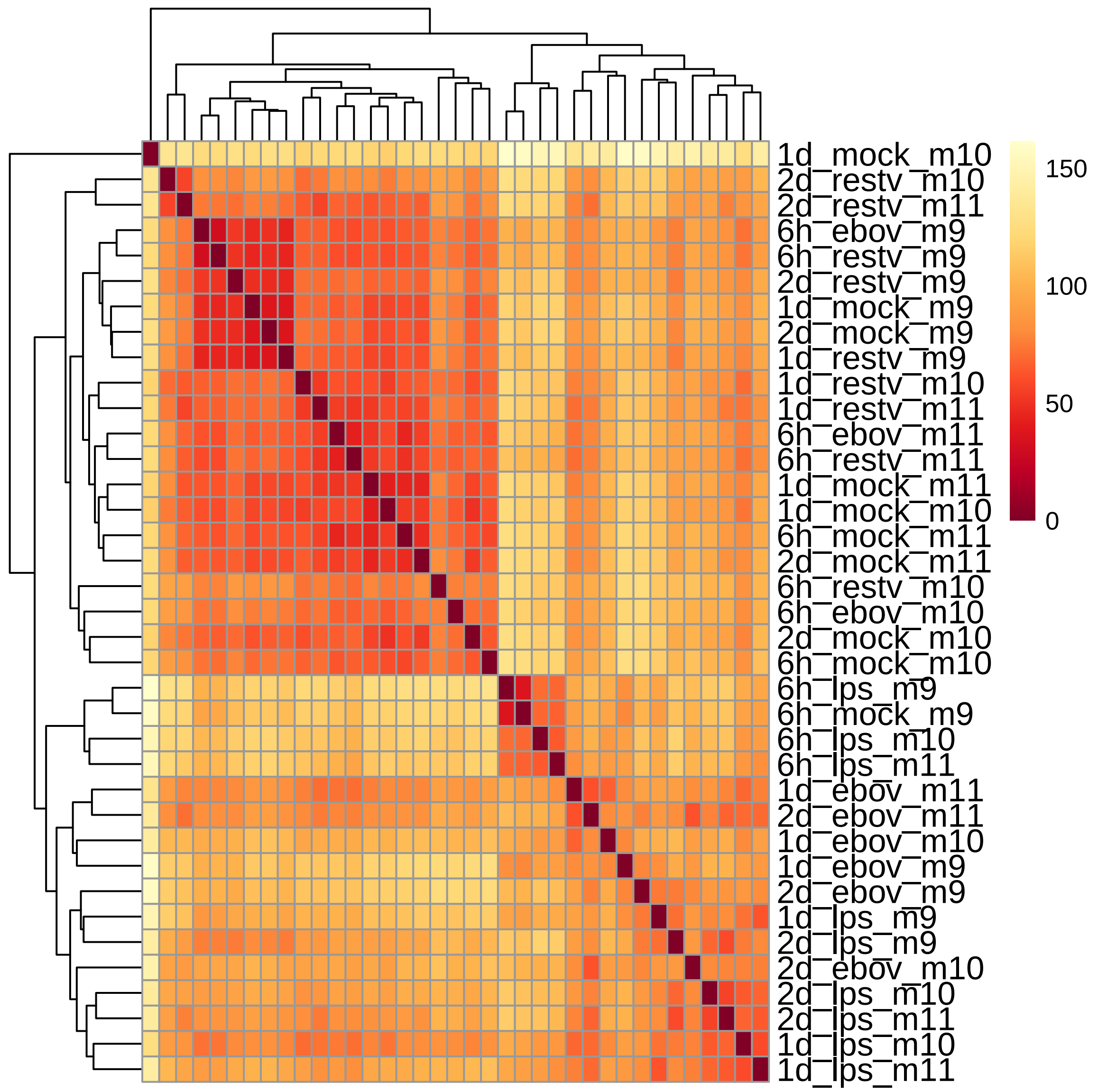
Examination of Data as a Whole

Before proceeding to differential expression analysis, it's best to look at the data set as a whole first to see overall patterns, find outliers, and identify batch effects. DEVis offers several tools to achieve this.

Distance Analysis (Two-way Hierarchical Clustering)

Let's first take a look at the euclidian distances between samples.

```r
plot_euclid_dist(row_labels="time_treatment_donor", filename="euclidian_distance.pdf",
                 theme=2, returnData=FALSE)
```



This plot performs two-way hierarchical clustering of a euclidian distance matrix and displays the results as a heat map. By using the previously created composite field, “time_treatment_donor”, we can see how strongly each factor influences each sample. In this symmetrical plot, each row (or column because of symmetry) represents a single sample and its relative distance to all other samples in the data set. The diagonal line that passes through the center of the plot with 0 distance between samples indicates the sample represented in that row.

In this plot, an outlying sample would be identified by its entire row/column showing very high distance (very light yellow color). In those cases, the sample could be eliminated from the data set with keep_data_subset() function provided with DEVis. The sample, “1d_mock_10”, has some potential to be an outlying sample in this case, and does not cluster similarly to other mock samples. It does however, appear to have some similarity with samples in the upper left clustered block, and is not uniformly distant from all samples. This puts this sample in a grey area, and its removal or inclusion in the data set is up to the researcher in this case. With this being a mock sample, however, it is quite possible that the inclusion or exclusion of this sample will impact the results of differential expression. It is, at this point, something to keep in mind as analysis continues. For the purposes of this vignette, we will include this sample in subsequent analysis.

The euclidian distance plot is also useful in indicating batch effects. For instance, if these data were prepared in two seperate labs or by three different technicians on four different sequence runs, that information could be included in the targets file and used as the labeling parameter here. Strong clustering by a particular batch metric would indicate that the data set may be biased and subject to a batch effect.

In this case, we see that a strong factor appears to be the donor. This makes sense because individuals are likely to be most similar to themselves regardless of the change of a few hours or their infection status. Next we can see that many of the "treatment" factors also tend to cluster together, indicating that after donor id, the type of infection seems to be the next cause of similarity between samples. There appears to be a strong clustering effect between restv and mock samples (seen in the large upper left cluster) and between the ebola and lps samples (lower right). As mocks are essentially serving as the control samples in this experiment, this might indicate that the restv type treatment does not cause as substantial differences in gene expression as do ebov and lps treatments.

This first step begins to tell the story of this data, providing a high-level view of the overall similarity for the data set as a whole, and allowing us to ask if the story the data tells agrees with a the expectations we might have for our experiment. In this case, so far the data appear to make sense, however if this plot behaved differently than what we would expect, we would need to investigate the cause of that discrepancy, or adjust our hypothesis to account for it.

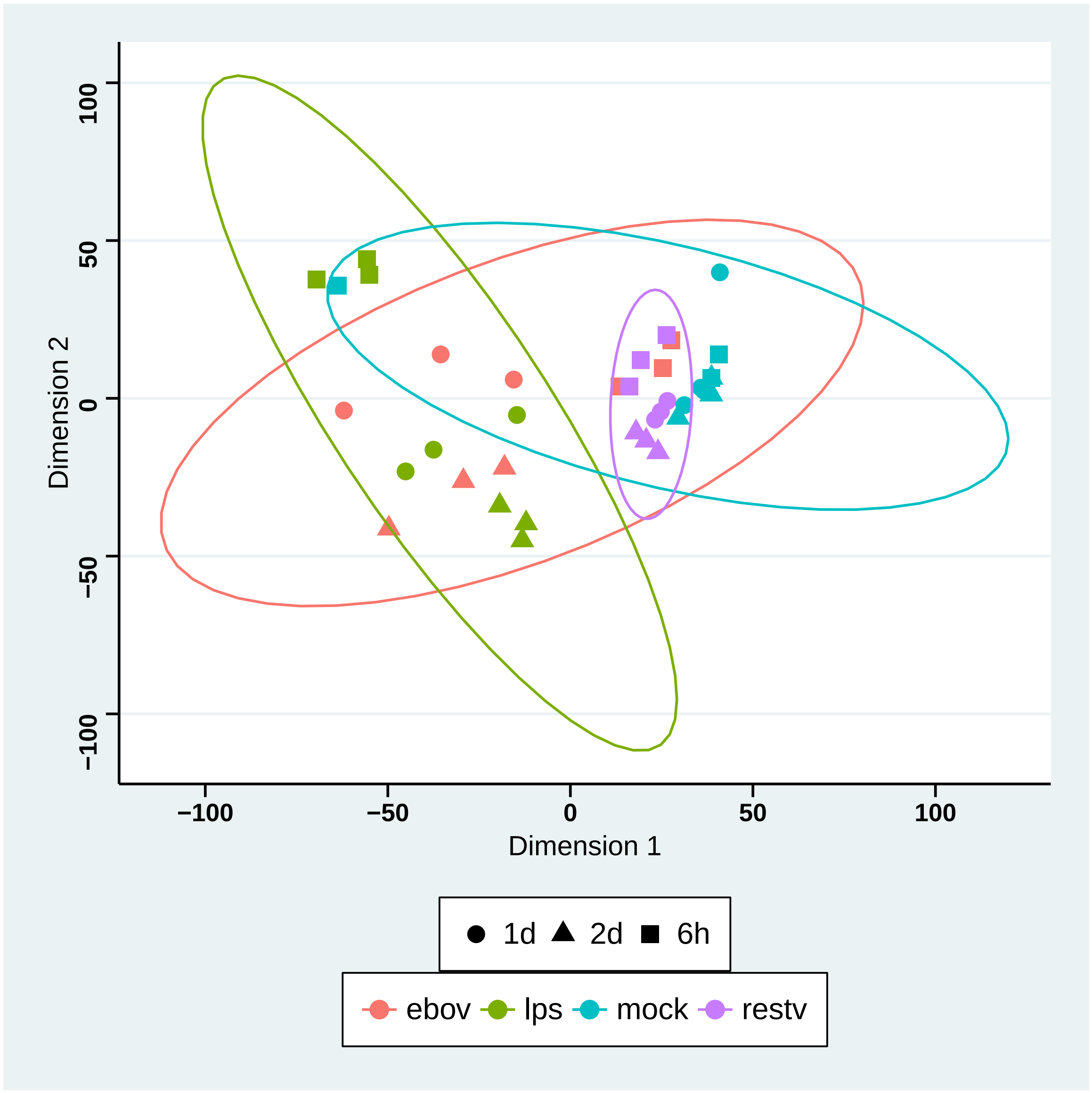
As with all DEVis visualizations, the generated plot will be automatically saved to the appropriate output directory, in this case, to the /sample_distance/ folder as a .pdf file.

- *Note: A similar plot that uses poisson distance measurements instead of euclidian distance can be used as well. See ?plot_poisson_dist() for more information. Furthermore, independent one-way hierarchical clustering dendrograms without heatmaps can also be generated using the plot_dendro function, see ?plot_dendro for more information.*

Multi-dimensional Scaling Analysis (MDS)

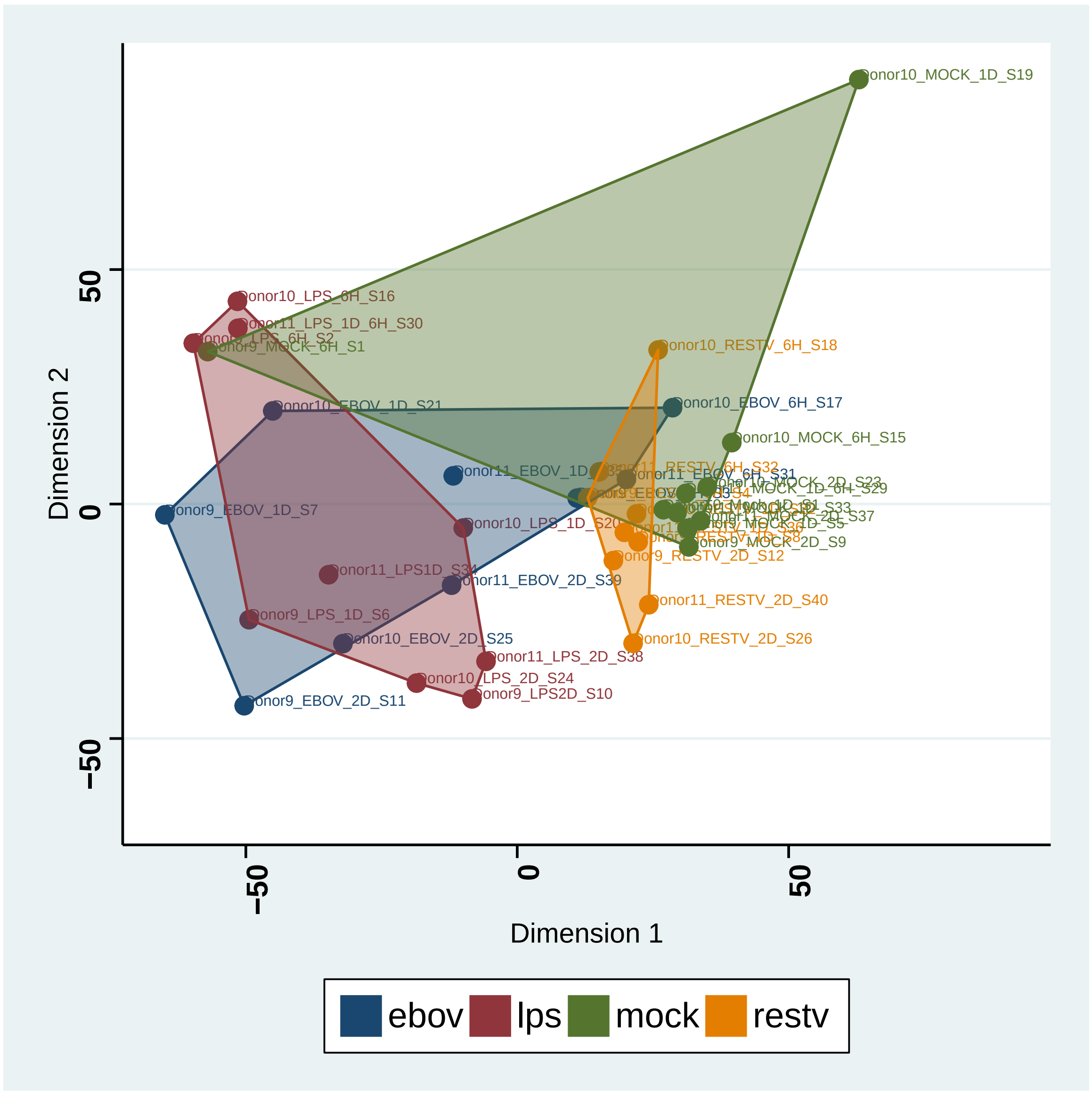
Next, let's look at the data through the lens of multi-dimensional scaling, in which each sample is given a x/y coordinate in an arbitrary space. MDS plots are excellent for finding separation and overlap between sub-groups within data, and by drawing MDS plots with reference to different subgroups, it is possible to identify the factor responsible for large scale changes in a data set.

```r
#Create a MDS plot showing effects of treatment and time.
plot_mds(filename="MDS.pdf",
         color_var="treatment",
         shape_var="time",
         theme=1)
```



This function allows you to make standard MDS plots based on any factors in your target data. In this scenario, colors are indicated by the "treatment" column of the target data, and shape is determined by the "time" column. By using different combinations of available metadata, it is possible to identify patterns in the data. Here again, we see that this data tells a similar story to what we saw in the euclidian distance plot, with restv and mock samples generally clustering tightly together, with ebola and lps type treatments being generally separate. A single mock sample additionally sits at the top left of the plot, very distant from all other mock samples, which clustered very nicely. This could potentially be the outlier we identified in the previous step. Let's investigate further by using a variation of the MDS plot, the MDS Hulls plot. The elipses here indicate the 95% confidence interval range for each cluster, and can optionally be toggled off.

```
#Create a MDS Hull plot showing effects of treatment and time.
plot_mds_hulls(filename="MDS_Hull.pdf",
               color_var="treatment",
               shape_var="none",
               deOnly=FALSE,
               showLabel=TRUE,
               hullType="solid",
               theme=1)
```



The MDS Hull plot offers a wide range of functionality. In this plot, a convex hull is drawn covering the outermost extreme sample from, each group to highlight changes between groups more clearly. In this case, as we are interested in investigating the potential outlying sample, we do not need to specify shape variable, and toggle the "showLabel" parameter to TRUE so we can see the identity of our outlying mock sample. As expected, the sample that strongly differs from the other samples is the suspected sample, "1d_mock_10", with the sample ID of Donor10_MOCK_1D_S19. With that being the case, let's update our data to remove that sample from analysis.

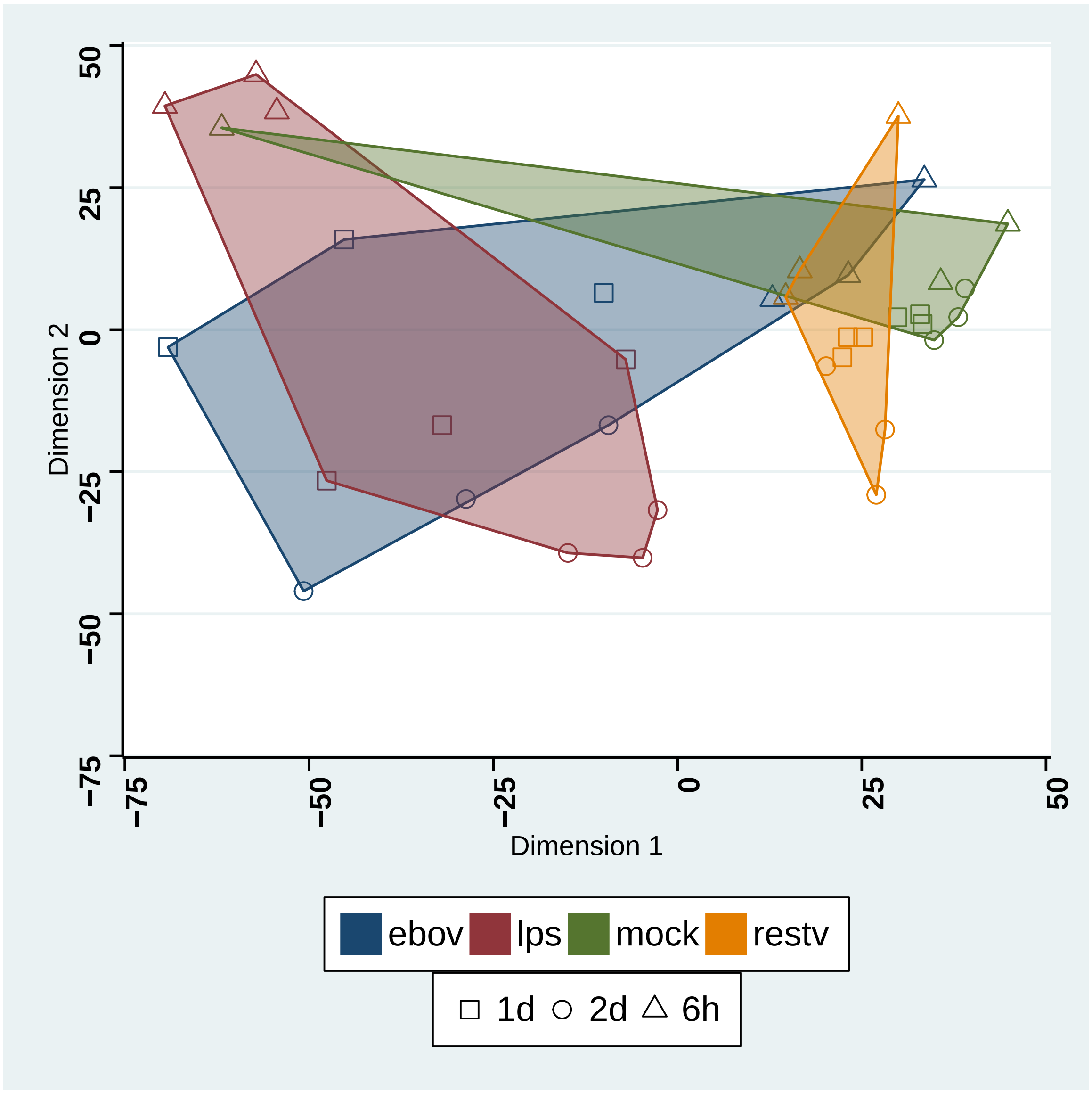
```
#Remove the outlying sample from our data.
subset  <- exclude_data_subset(counts=count_data,
                               targets=target_data,
                               target_count_id_map="targetID",
                               target_exclude_col="targetID",
                               target_exclude_val="Donor10_MOCK_1D_S19")
```

```r
#Update count and target data to the filtered set.
count_data  <- subset[[1]]
target_data <- subset[[2]]

#Update our DESeq2 Object based on the filtered data.
dds <- prep_dds_from_data(count_input=count_data,
                          target_input=target_data,
                          experiment_design= ~ time_treatment,
                          stabilization="vst")
```

Now that our outlying sample has been eleminated, let's look at our data again with MDS Hulls, this time, specifying the treatment and time as shape and color variables, and removing the labels.

```r
#Create a MDS Hull plot showing effects of treatment and time.
plot_mds_hulls(filename="MDS_Hull_filtered.pdf",
               color_var="treatment", shape_var="time",
               deOnly=FALSE, showLabel=FALSE,
               hullType="solid", theme=1)
```

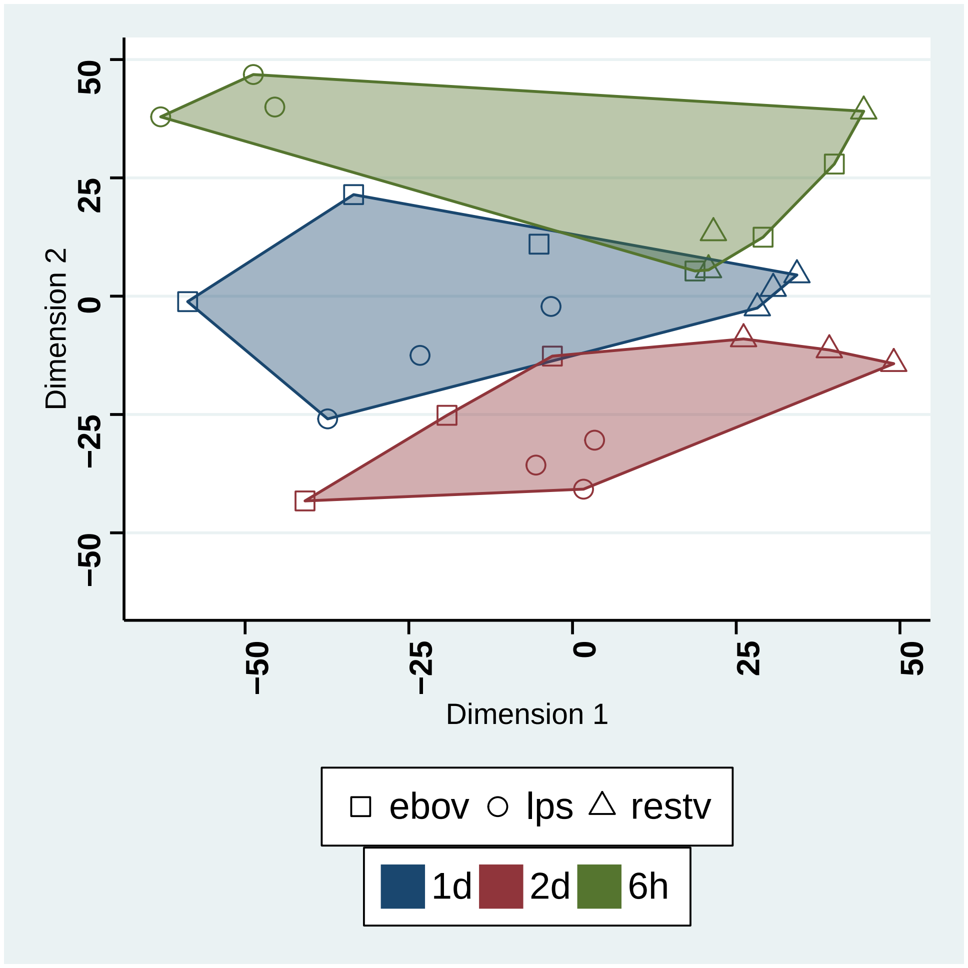


We can now see that the outlying sample has been removed. Additionally, there is one sample among the mocks that seems to not cluster quite so well, however as this sample did not show suspicious behavior in the Euclidian distance analysis, in combination with the fact that this is a sample taken at 6 hours and it is clustering well with other 6h points, we will assume for now that this is a valid sample to include in analysis.

From this perspective, we can see that mock and restv samples tend to cluster fairly strongly together, while the ebov and lps samples are fairly distinct with a wider range of variablility. This echoes what the previous data showed in both the Euclidian distance and standard MDS plots.

Now let's examine the same data from a different perspective by using different groupings for color and shape variables. Let's also hide mock samples from this view to reduce the amount of information displayed in the plot and focus on the differences in the other treatment conditions.

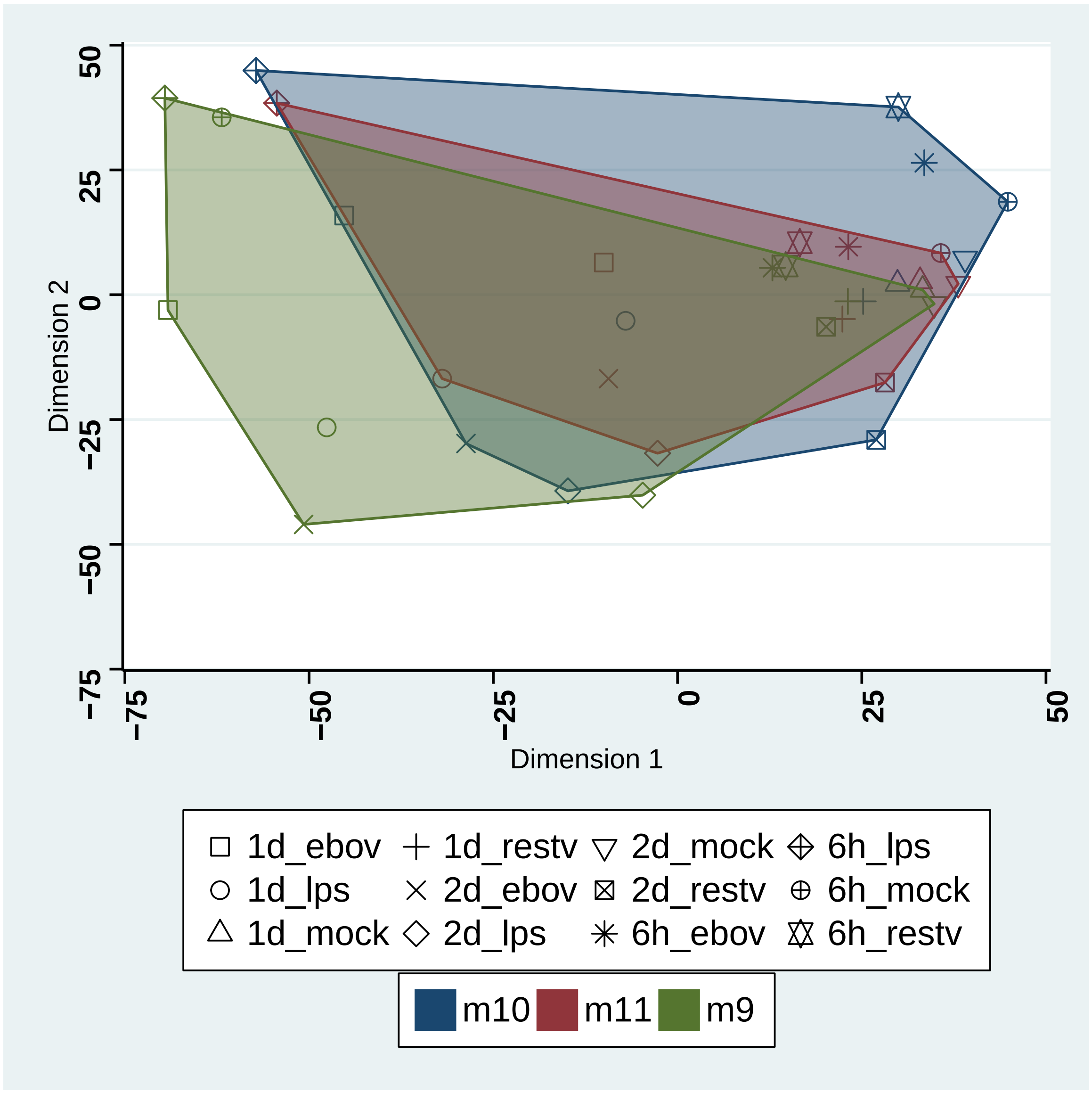
```
plot_mds_hulls(filename="MDS_Hull_byTime.pdf",
               color_var="time",
               shape_var="treatment",
               deOnly=FALSE,
               showLabel=FALSE,
               hullType="solid",
               theme=1,
               exclude_data=TRUE,
               idCol="targetID",
               excludeCol="treatment",
               excludeName="mock")
```



By simply setting the color variable to reflect the time factor rather than the treatment factor, we can see now that there are distinct differences between the groups based on the time point. We can see again the strong clustering of mock and restv based on the triangle and cross shapes, however we now can see more clear trends in the ebov and lps samples that were not apparent before. This suggests that the variablity we see in the ebov and lps samples is likely due to the time factor. This confirms our expectations about our experimental design and suggests that our data is valid and that transcriptomic changes are occuring over time.

Finally, let's use MDS Hulls to confirm there is no bias between our three donors.

```
plot_mds_hulls(filename="MDS_Hull_byDonor.pdf",
               color_var="donor",
               shape_var="time_treatment",
               deOnly=FALSE,
               showLabel=FALSE,
               hullType="solid",
               theme=1)
```



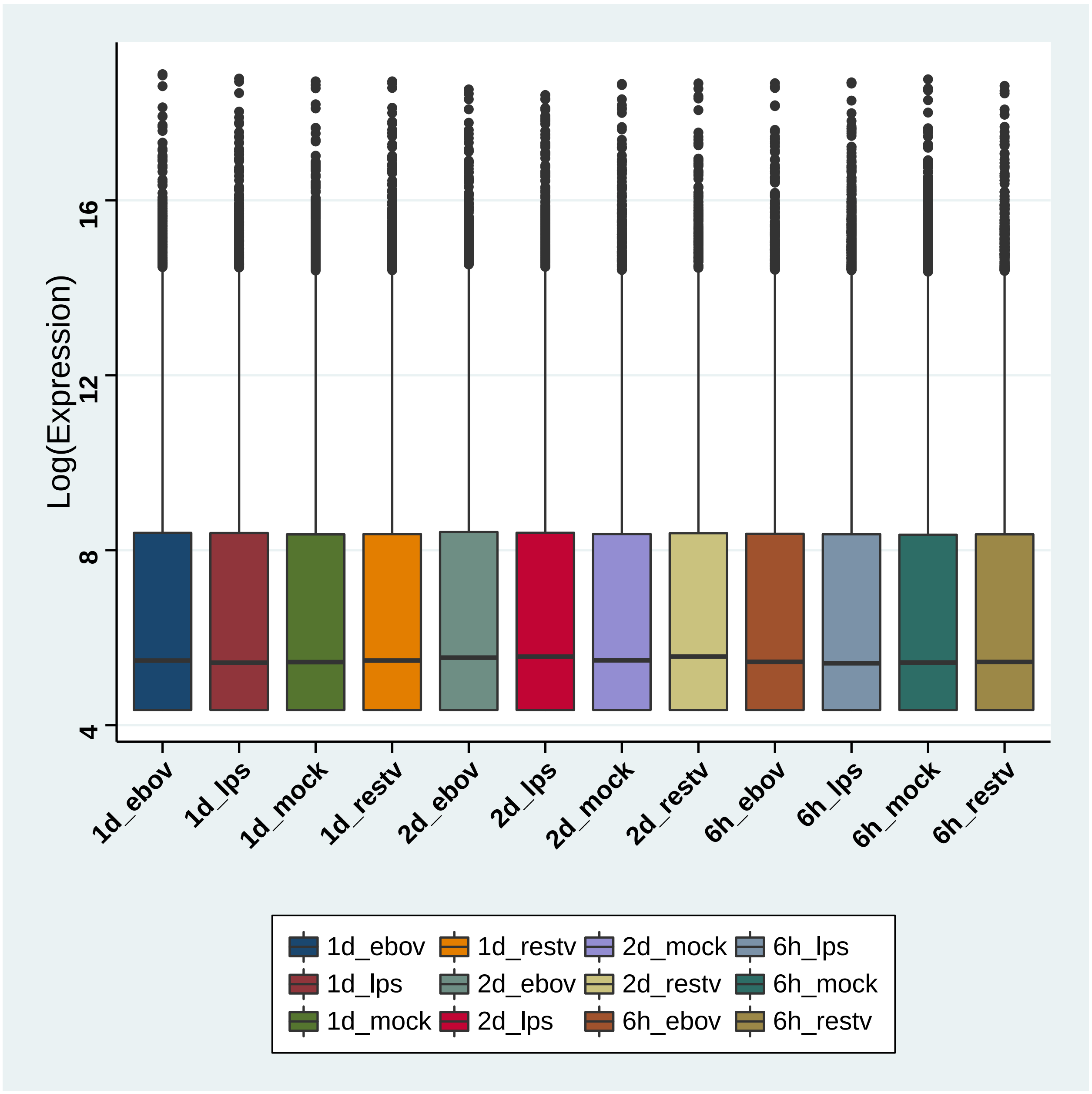
In this case, we have used our composite column, time_treatment, to allow us to look at three factors at once, by specifying the composite column as the shape variable and the donor as the color variable. In this way as many factors as desired can be examined using the MDS Hulls plot.

We can see clearly that there is a strong overlap and similar pattern of simiarity between our three donors, excluding the possibility of a single donor is contributing to bias in the data.

Normalization

Our final step before proceeding to differential expression analysis is to confirm that our counts have normalized and there is no apparent bias in our count data. Normalization is performed using DESeq2 at the prep_dds_from_data() step. For more information on the details of normalization, see the DESeq2 manual. DEVis offers a function for plotting normalized / un-normalized count data based on any factor in the target data. For now, let's just look at the normalized counts to confirm.

```
plot_group_stats(filename="Normalized_Counts.pdf",
                 id_field="targetID",
                 groupBy="time_treatment",
                 normalized=TRUE,
                 theme=1)
```



This plot shows the group-wise counts for any group specified in the groupBy parameter. It is recommended to use a detailed field that covers a wide-range of options when looking at this plot, as splitting the count data into smaller sub-groups would be more likely to indicate abnormality in count data. We can see in our case, using the composite field, time_treatment, that our count data appears to be appropriately normalized and is ready for analysis. To see the un-normalized version, set normalized=FALSE.

Differential Expression

After initial examination and cleaning of our data, we are now ready to start looking at differential expression (DE). First, let's calculate DE using the DESeq2 object we prepared earlier.

```
#Run DESeq on our previously prepared DESeq2 object.
dds <- DESeq(dds)
```

We can now determine the contrasts we are interested in examining by using DESeq2's results() function. This function takes as its contrast parameter the column we used for our design parameter when we created our DESeq2 object, in this case, "time_treatment", and then case and control conditions. We are interested in how the ebov and restv viruses were differed from mock samples at each time point. Since we have two varieties of mock samples (mock and LPS), we might also be interested in how the viruses behave with regard to these different treatments. We could merge the LPS and mock samples into a single control group or treat LPS and mock samples independently. For the purpose of this vignette and for the sake of simplicity and clarity, we will ignore the LPS samples for now and focus on ebola / restv vs mock comparisons.

```
#Time matched ebola vs mock contrasts.
res.ebov.6h.vs.mock <- results(dds, contrast=c("time_treatment", "6h_ebov", "6h_mock"))
res.ebov.d1.vs.mock <- results(dds, contrast=c("time_treatment", "1d_ebov", "1d_mock"))
res.ebov.d2.vs.mock <- results(dds, contrast=c("time_treatment", "2d_ebov", "2d_mock"))

#Time matched reston vs mock contrasts.
res.restv.6h.vs.mock <- results(dds, contrast=c("time_treatment", "6h_restv", "6h_mock"))
res.restv.d1.vs.mock <- results(dds, contrast=c("time_treatment", "1d_restv", "1d_mock"))
res.restv.d2.vs.mock <- results(dds, contrast=c("time_treatment", "2d_restv", "2d_mock"))

#Make a list of all of our contrasts.
result_list <- list(res.ebov.6h.vs.mock, res.ebov.d1.vs.mock, res.ebov.d2.vs.mock,
                    res.restv.6h.vs.mock, res.restv.d1.vs.mock, res.restv.d2.vs.mock)

#Aggregate differentially expressed genes across all contrast results.
master_dataframe <- create_master_res(result_list, filename="master_DE_list.txt", method="union", lfc_fi
```

Result Set Aggregation

We have now determined 6 contrasts of interest and aggregated them into a single result set that can be visualized. The create_master_res() function is key to data aggregation. This function combines data from all contrasts based on the union or intersection of identified DE genes. For example, if two contrasts were performed, and in the first contrast "geneA" was identified as a differentially expressed gene, and in the second "geneA", and "geneB" were identified as differentially expressed genes, the union-based aggregated data set would contain expression and significance values for both "geneA" and "geneB", despite the fact that "geneB" was not significant in the first contrast. Alternatively, the intersection-based aggregation would contain only "geneA", because it was identified as a differentially expressed gene in both contrasts.

Union-based aggregation makes it possible to compare genes across conditions that may be significant in one set, but not necessarily in another. For instance, if a gene was not differentially expressed at the 6 hour time point, but it's expression became significant by the 1 day time point, union-based aggregation would identify the gene as a potential gene of interest and visualizations could then show the change over time. It is important to keep in mind with union-based aggregation that as the number of contrasts and the variation of contrasts increases, the number of DE genes in the aggregated set will also increase, potentially increasing the difficulty of finding the most relevant genes of interest. This can be mitigated somewhat by later visualizations that employ sorting and filtering methods, but should be considered when determining which contrasts to aggregate.

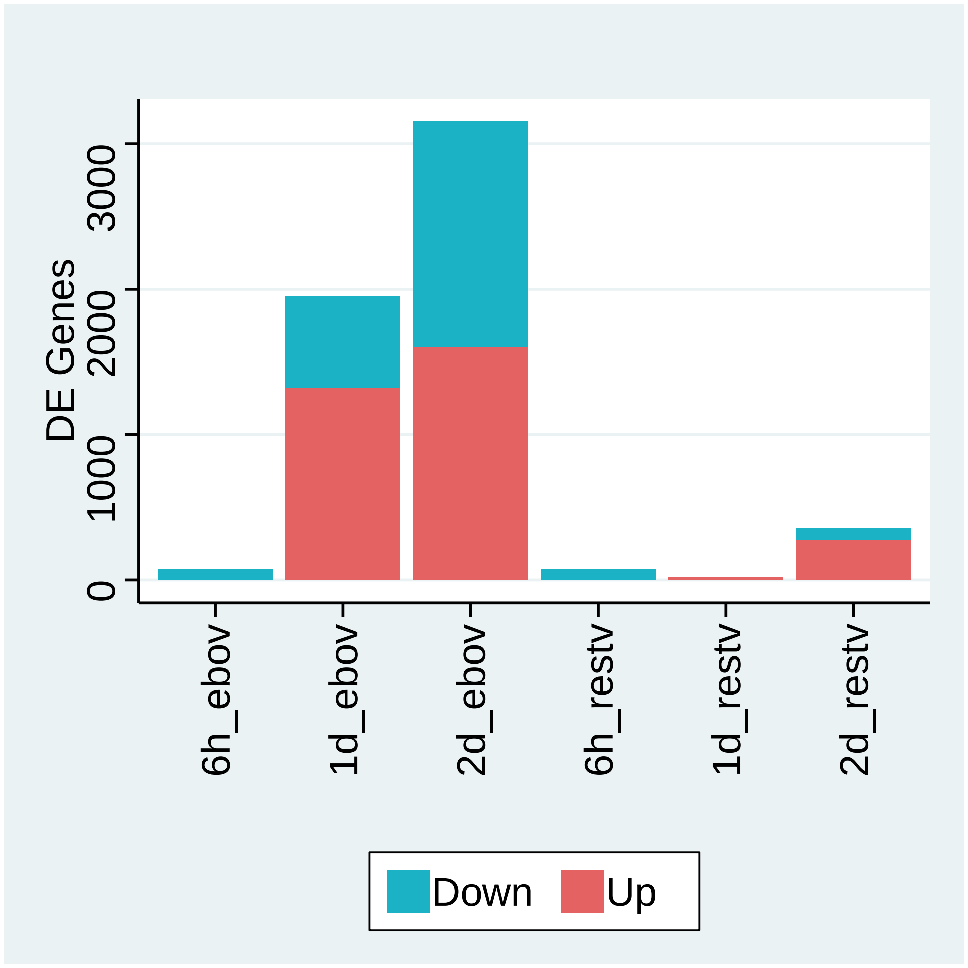
Intersection-based aggregation, on the other hand, eliminates the problem of an increasingly large DE gene set by only including genes that are differentially expressed in all contrasts. This means, however, that genes that are not DE in all samples will be excluded, so a gene that was not significantly differentially expressed even in a single contrast would eliminate that gene from the data set. Furthermore, as the number and variation of contrasts provided to intersection-based aggregation increases, it becomes increasingly likely that genes will be eliminated from the data set. On the other hand, by providing a small set of contrasts to an intersection-based aggregation, it is possible to easily identify the genes that are significant across multiple conditions.

With these caveats in mind, the proper choice of contasts becomes essential for appropriate data aggregation. In our case here, we have 6 contrasts, ebola vs mock for three time points, and reston virus vs mock for three time points. Here we are aggregating all DE genes from the 6 contrasts using union-based aggregation, and applying a minimum log fold-change cutoff (previously set at initialization time to 1.5). This lfc_filter has the effect of eliminating genes that may not have been significant in most cases, but slightly overcame the threshold in at least one contrast. This mitigates somewhat the growth of union-based aggregation and can be used in cases where there are a high number of differentially expressed genes to eliminate potentially less interesting genes. This list could also be reduced by decreasing the minimum p-value cutoff for differential expression detection in the init_cutoffs() function performed at initialization.

Another question that should be considered is the variability of the contrasts. We are essentially merging (ebola vs mock) and (reston vs mock), but based on the previous MDS hulls analysis, we can tell that there are fairly stark differences between the ebov and the restv data. To get a clear picture of how ebola expression changed over time, it might be preferable to consider the ebov contrasts and the restv contrasts separately. However, as the purpose of this experiment was to contrast these viruses, the first aggregation will include DE from both virus contrasts, and will first rely on sorting and filtering methods of subsequent visualizations to find the relevant genes of interest between viruses.

The master result set will be saved to the /DE/data/ directory for future use or for import into other software packages like the IPA functional analysis platform.

```r
de_counts(result_list,
          filename="DE_counts.pdf",
          theme=1)
```



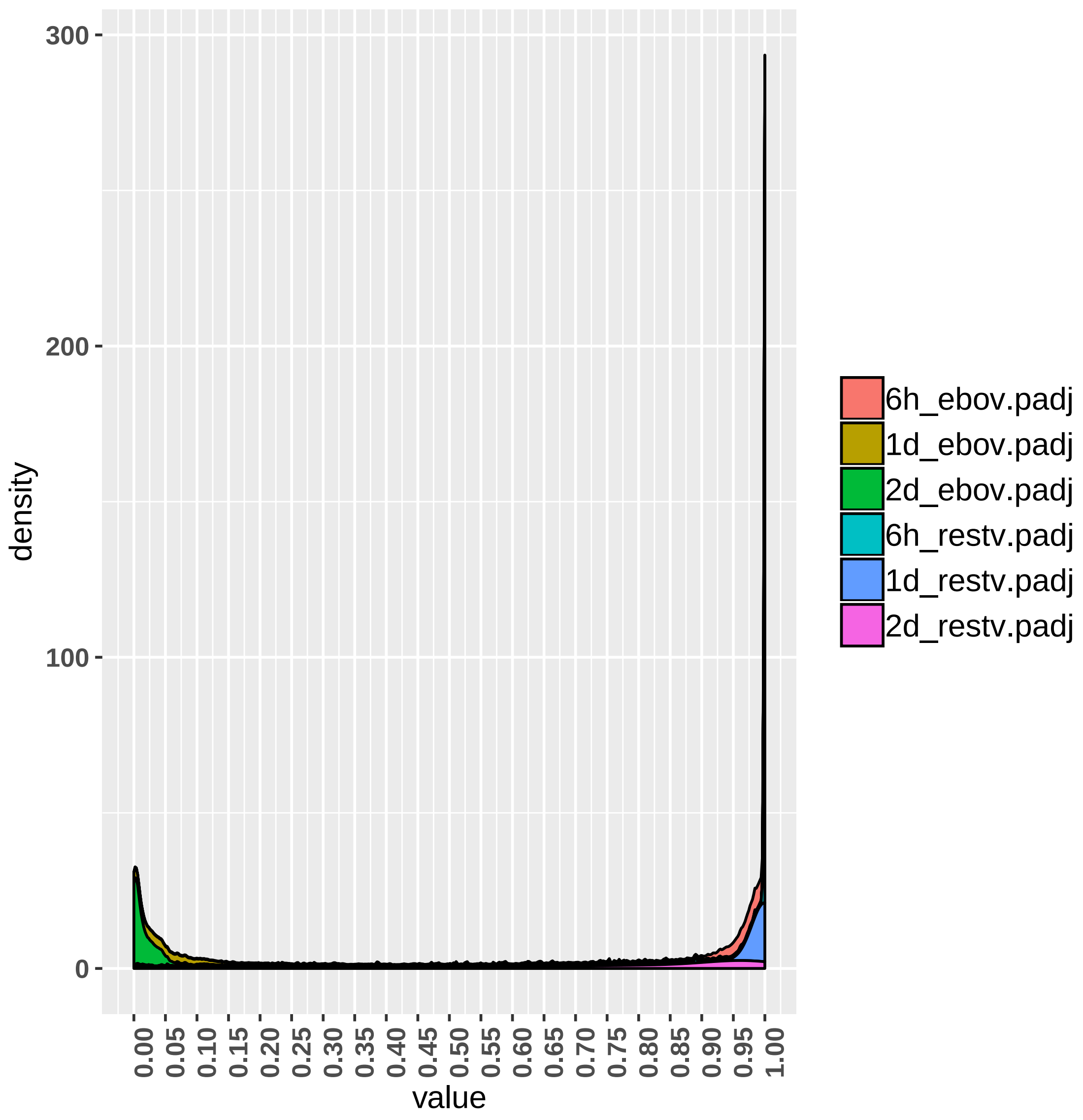
To see how our aggregation performed, we can look at the density plot of adjusted p-values and the number of differentially expressed genes identified in each contrast. The de_counts function shows the up and down regulated differentially expressed genes for each contrast, and additionally will save the plot and accompanying data to the DE/counts/ directory. We can see that the progression of ebola virus over time leads to an increasing number of differentially expressed genes when compared to time matched mock samples, while the reston virus shows a much lower level of differentially expressed genes for the same time points.

Based on these DE count data, we should consider how aggregation will influence subsequent analysis. By union-based aggregation we will have a large number of genes that are differentially expressed only in day 1 and day 2 ebola infection contrasts. Using intersection-based aggregation however would limit the maximum data set size to the number of genes expressed in the reston virus day 1 sample, which showed the lowest number of differentially expressed genes.

The trade-off, therefore, between union-based and intersection-based aggregation is similar to a 'needle in a haystack' type problem. Depending on the experiment being analyzed, union-based aggregation will increase the size of the haystack, but will certainly not throw out the needle. Intersection-based aggregation on the other hand has the potential to drastically reduce the size of the haystack, but also runs the risk of throwing out the needle as well.

In this case, we are using union-based merging and will rely on sorting and filtering methods to reduce the size of the dataset and identify genes that are potentially of interest.

```
de_density_plot(result_list,
                filename="aggregate_density.pdf",
                type="pval",
                method="union")
```



We can further see how our aggregation performed by looking at the density plot of adjusted p-values and the number of differentially expressed genes identified in each contrast. In this plot, genes below the 0.05 threshold are significant for the contrasts indicated by thier respective colors. Genes that were included in the data set because of union-aggregation but were not significant a sample will similarly display at the upper end of the graph. We can see that the day 1 and day 2 ebola genes have strong agreement for many differentially expressed genes between those two contrasts, meaning that many of the genes that were differentially expressed at day 1 were still significant at day 2. On the other hand, we can see that many of the genes included were not significant in the 6 hour and 1 day time point for reston virus. This is a natural consequence of the differences in the number of DE genes identified at each contrast. For instance, the very large green spike for day 2 must exist because there are many DE genes that only occur in the day 2 ebola contrast.

This plot shows us the level of variation between contrasts. We can see there is quite a lot of difference between the ebola and reston samples, but fair similarity within the viral treatment groups. Depending on the experimental design, it may be desirable in some situations to aggregate the ebola samples and reston sample separately due to these differences, however because this particular experimental design is interested in comparing these viruses, it is necessary in this case to combine all contrasts.

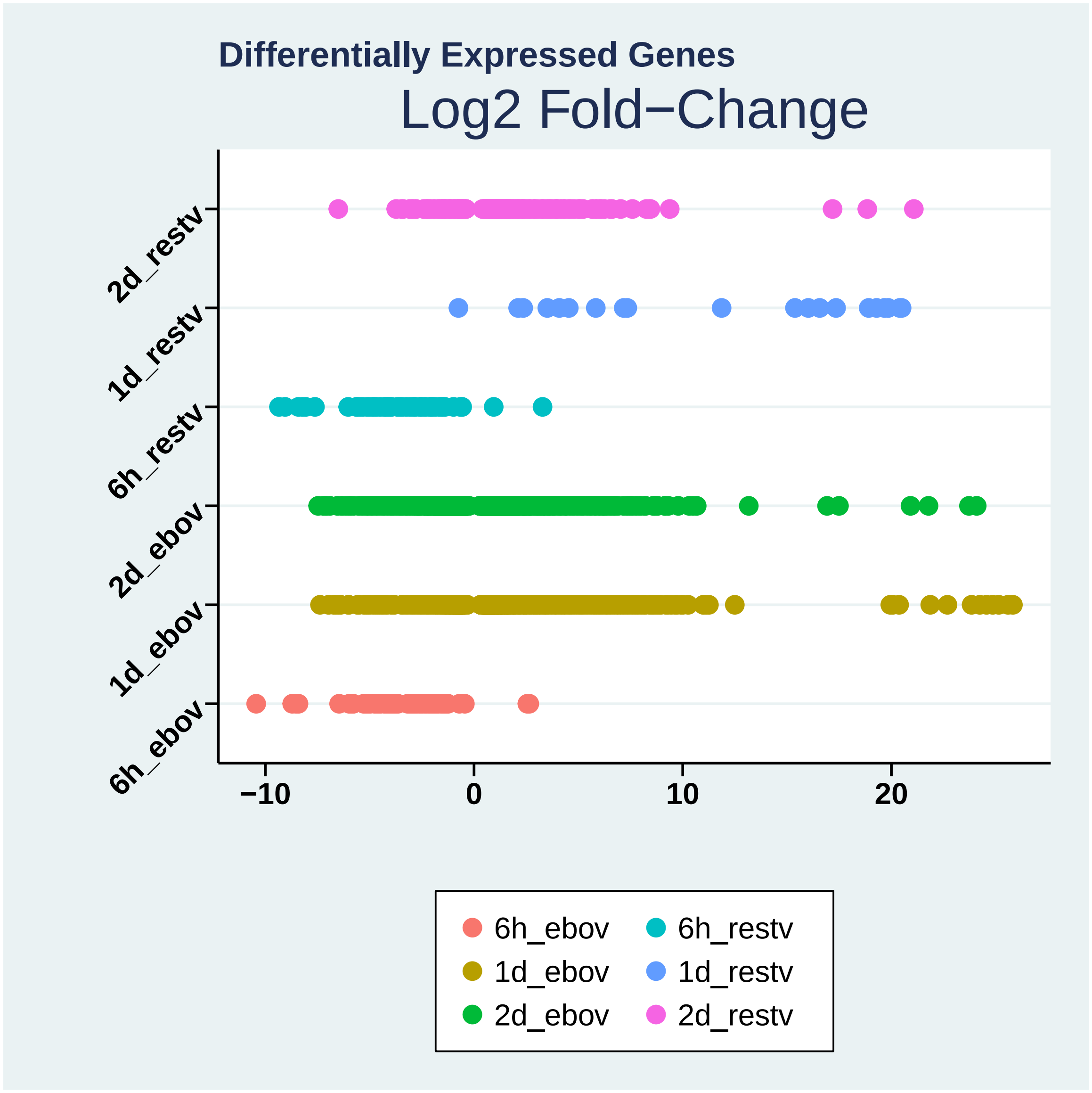
Examination of Differentially Expressed Genes

Now that our DE gene data are prepared, we can start to examine the differences and similarities between contrasts.

Differences Between Conditions

First, let's look at how expression levels for DE genes change over time. To do that, we can use the de_diverge_plot function.

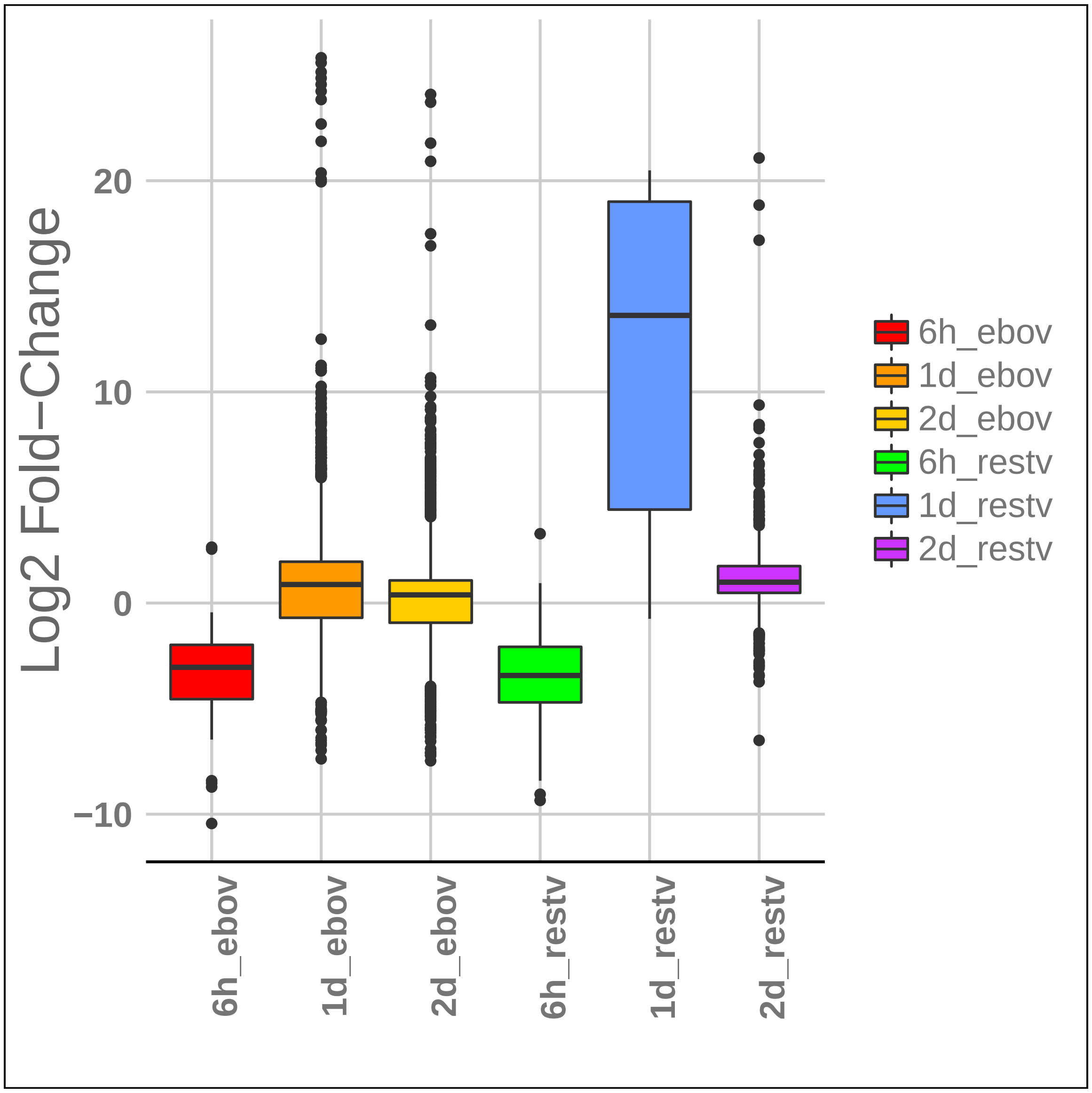
```
de_diverge_plot(result_list, filename="DE_diverge.pdf", theme=1)
```



We can see here that for both ebola and reston virus cases that most differentially expressed genes at the 6 hour time point for both ebola and reston viruses range from downregulation of about -10 fold to about +3 fold upregulation in expression. However at the day 1 time point, expression for a large number of genes increases up to 20-fold differences and downregulated genes are less extremely different. This pattern continues into the day 2 time points for both viruses, with overall expression levels staying relatively the same between days 1 and 2, with slightly less extreme changes at the day 2 time point.

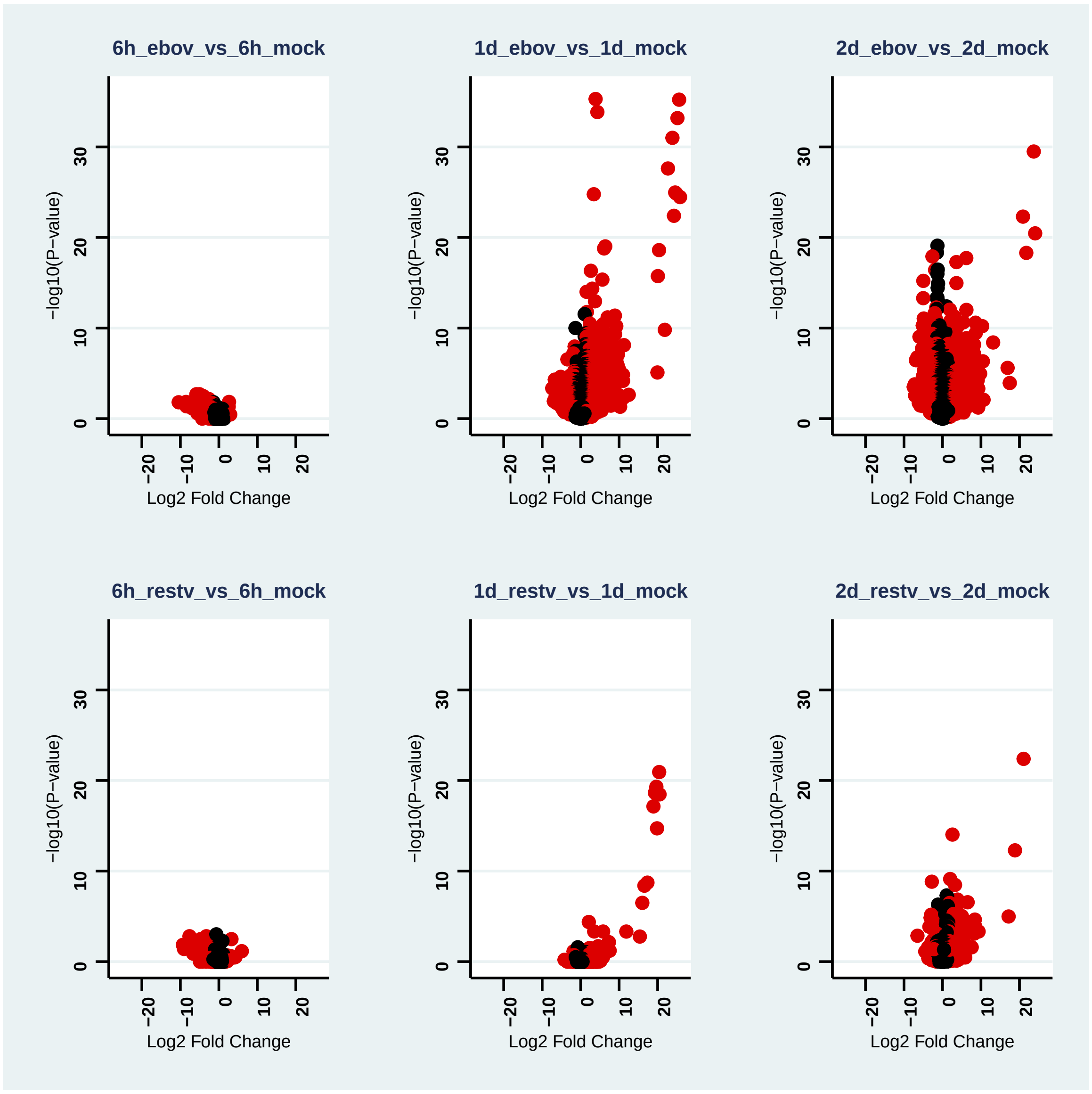
In some cases, examining the most extreme cases for up and down regulated genes may shed light on the biological mechanisms or gene networks involved underlying the infection. In these cases, the de_filter function can be used to select genes using logical expresisons. The de_filter function allows you to filter an aggregated master result to only contain genes with fold-changes less than or greater than a specified threshold for a given metric. Several metrics are provided for flexibility of selection. The use of metrics in this function makes it possible to ask questions such as, "which genes have a mean fold-change of at least 2?", "which genes have a fold change of less than 5?", or "which genes have fold-change with a variance across all conditions of at least 10?" For more information, see ?de_filter.

```
de_boxplot(result_list, filename="DE_boxplot.pdf", theme=4)
```



We can also examine the relationship between the significance of our DE genes and their expression change by examining volcano plots for each sample.

```
de_volcano(result_list, filename="DE_volcano.pdf", theme=1, strict_scale=TRUE)
```



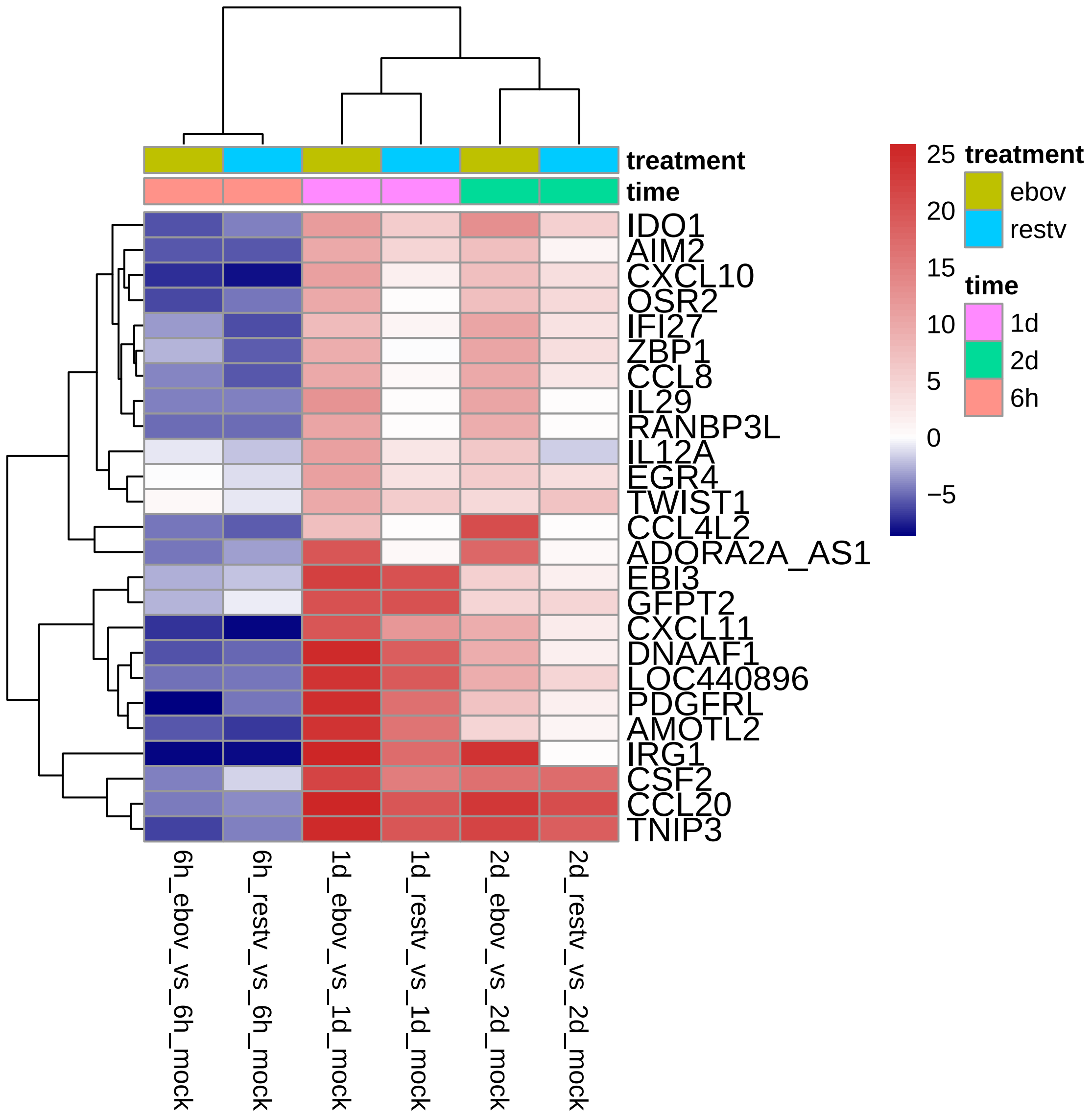
Here we can see that at the early timepoint there are not as significant changes in either the reston or ebola groups, however as the one and two day time points proceed, we can see that both expression and significance of DE genes increases dramatically compared to the reston virus cases at the same time points. This de_volcano() plot can also be used as a visualization for examining the effects of filtration. By using the lfc_thresh parameter, it is possible to see the genes that will be selected when filtering by fold change or p-value.

Relative overall expression levels can also be visualized using the de_boxplot, de_diverge_plot, and de_volcano functions. These plots together can give a picture of expression and allow you to make informed decisions about filtering and data selection. For instance, based on these plots we may decide to further investigate the set of differentially expressed genes with a log fold-change of greater than 15. Such a filtration would substantially reduce the size of the data set to a managable scale for further investigation.

Differences Between Genes

Now that we have a picture of the differences between our different conditions, we can next look at differences between specific genes. Let's begin by applying the de_heat function for creating heat maps. DEVis heat maps offer functionality for plotting genes based on specific features of their differentiation. Specifically, it allow you to look at subsets of differentailly expressed genes based on a sorting criteria. Seven sorting methods are available: “mean”, “max”, “min”, “variance”, “max_mean”,“min_mean”, “sd”. These methods allow you to examine different subsets of genes based on their expression values. For example, let’s look at the top 25 most upregulated genes in at least one contrast by using the “max” sort option.

```
de_heat(result_list, anno_columns=c("time", "treatment"),
        filename="upReg_heatmap.pdf", sort_choice="max",
        numGenes=25, theme=2)
```



The “max” option identifies the genes with the highest log-fold change among any contrast. This sorting option is ideal for identifying genes that may be highly expressed in at least one contrast of the aggregated result set. In this plot we can see many potential genes of interest, such as IDO1 and TNIP3, which are antibody proteins. These genes appear to behave similarly in both the restv and the ebola infections, which makes sense because of the shared background of these viruses. On the other hand, CCL4L2, a chemokine coding gene, is highly expressed in ebola samples at both day 1 and day 2, but is not expressed in the reston virus samples at all, which lends support to the hypothesis that reston virus does not trigger a pathogenic response in humans. Similar patterns of expression can be seen for many genes in this heat map, where the highly expressed genes are present in ebola virus infected samples, with little to no expression in the reston virus condition. This provides a compelling case for the hypothesis of the study.

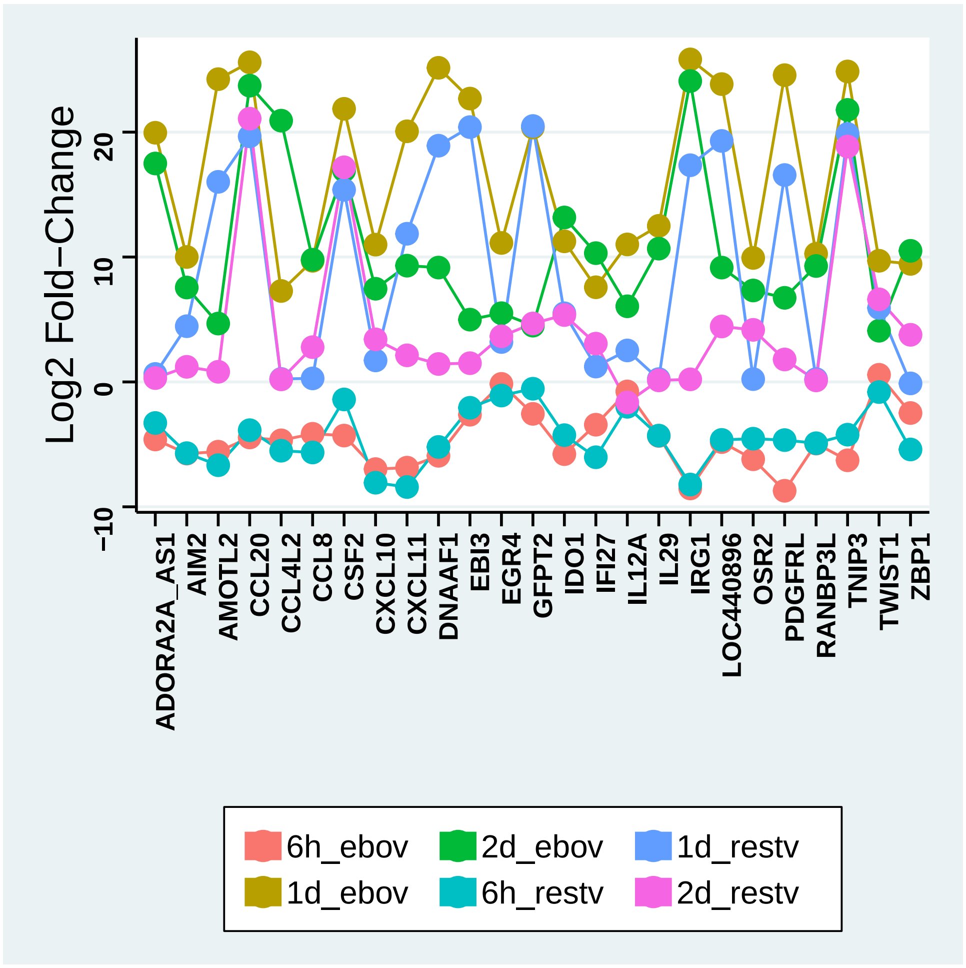
The “max_mean” option adds an additional constraint to the selection criteria. In “max_mean”, the selected subset are the genes that are most highly expressed as the mean of all contrasts, rather than simply the highest expression in any given sample. The “min” and “max_min” functions work similarly, but examine down-regulated genes rather than up-regulated genes.

The “sd” and “variance” sorting options identify the genes with the highest gene-wise standard deviation or variance. This option is ideal for finding genes that have substantial contrasts between groups or between time points.

Additionally, we could have toggled the clustering of contrasts off to preserve the order of samples. In this case the clustering by contrasts (columns) allows us to look at the data with corresponding timepoints adjacent to one another. However, by disabling contrast clustering for heat maps you could force the order of samples to be maintained and reflect the order of your result_list. For more information on the heat map functionality, see the ?de_heat help page.

Next, let’s look at a the expression of some differentially expressed genes in another way using the de_series plot. The de_series() function displays log2 fold-change values for each contrast as a hybrid line/point plot. This plot makes it possible to visualize data that is often viewed in a heat map in a different way that includes absolute expression values to be examined for each gene and each group, rather than the type of relative color-based comparison offered by heat maps.

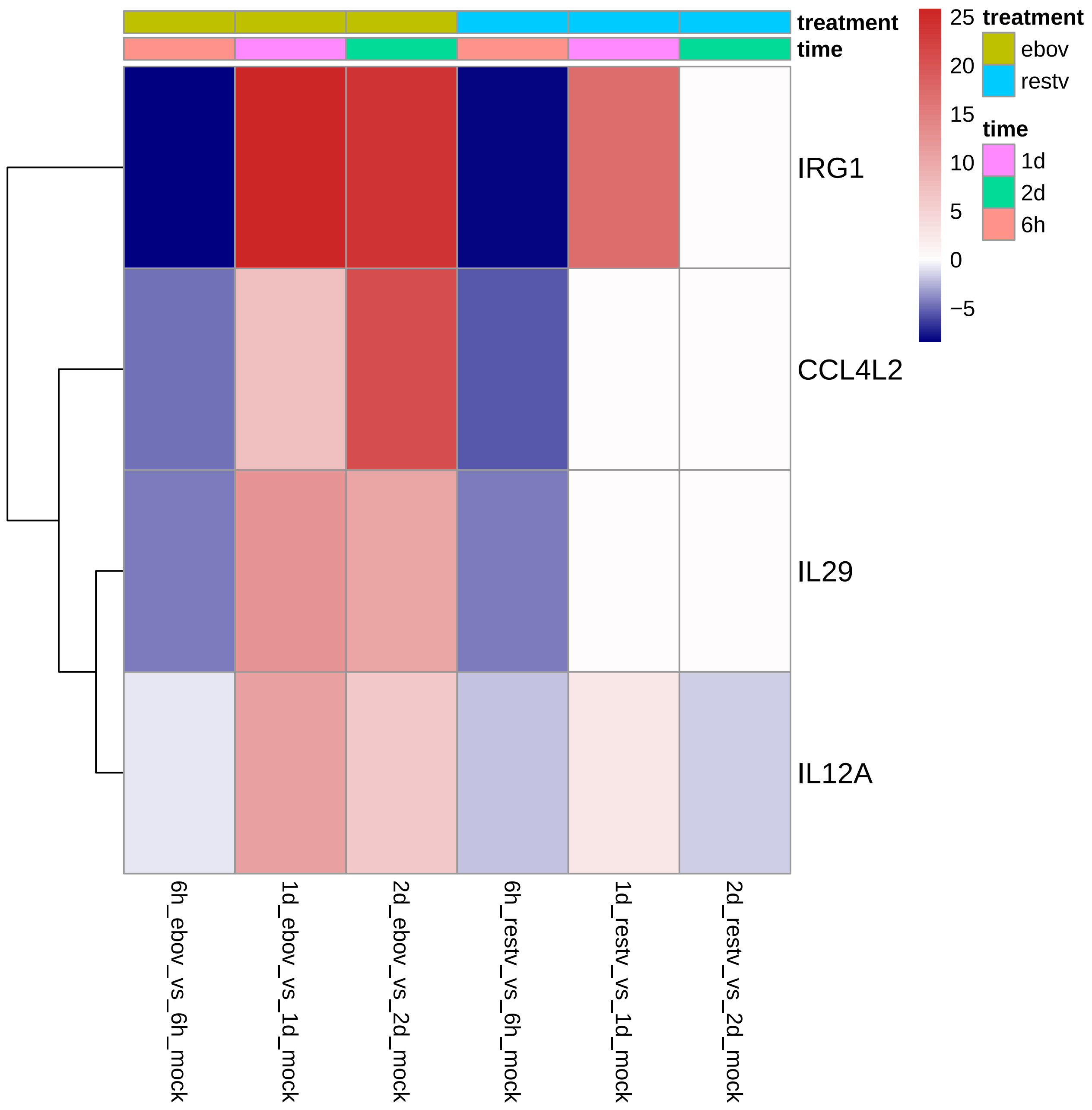
```
de_profile_plot(result_list,filename="DE_profile_upReg_10.pdf",
                sort_choice="max", numGenes=25, theme=1)
```



Here, for instance, we can see that the 6 hour time points for both ebola and reston virus are highly similar. At day 1, while the ebola and reston virus conditions are still following relatively similar patterns of expression in these genes, ebola treatment generally shows much higher expression. At day 2, many genes continue to follow relatively similar patterns of expression with ebola being more strongly expressed, however, some genes at this time point now show changes that are not relatively similar between the two viruses. CCL4L2, IL29, IRG1, and IL12A, for instance show deviations at day 2 that are relatively different from their expression profiles at 6 hours and 1 day time points.

Let's take a closer look at these genes in particular with a heat map.

```r
de_heat(result_list, anno_columns=c("time", "treatment"),
        filename="upReg_heatmap.pdf", sort_choice="max",
        specific_genes=c("CCL4L2", "IL29", "IRG1", "IL12A"),
        cluster_contrasts=FALSE,
        theme=2)
```



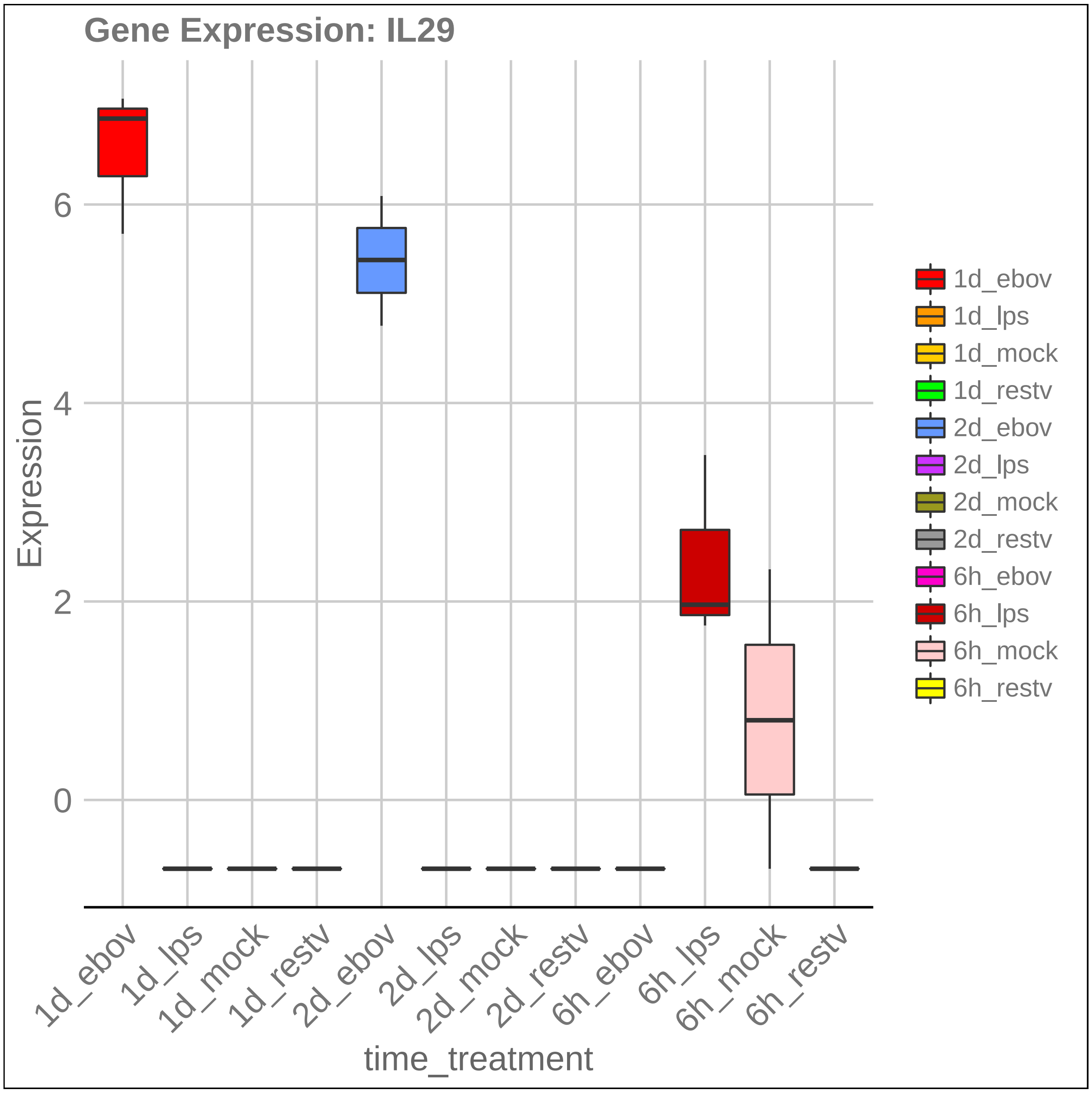
Here, we have provided the specific__genes argument to force de__heat to display the genes we are interested in looking at. Also, because it is more informative to see the columns of the heatmap in order, we have turned off the "cluster__contrasts" parameter. We can see here a clear pattern. At hour six these genes are down regulated relative to the mock. At day 1, in all cases expression increases in the ebola treated samples, but only similarly increases in IRG1 and IL12A in the reston virus samples. At day 2, these genes are no longer being expressed strongly and are back to baseline, but they have continued to be highly expressed in the ebola treatment samples.

Investigation into the function of these genes shows they are of biological significance to this experiment. IRG1 is involved in the inhibition of the inflammatory response. CCL4L2 is a cytokine gene that regulates inflammatory and immunoregulatory processes. IL12A is a Natural Killer Cell Stimulatory Factor. IL29 encodes a cytokine related to type I interferons and the IL-10 family, which has been shown to have expression induced by viral infection.

This supports the hypothesis of this study that reston virus is non-pathogenic in humans. We see early immune and inflamitory responses occuring in the ebola treated samples, which are not expressed in the reston virus samples. Furthermore, the IL29 gene, which has been previously identified as being highly expressed in response to viral infection, and the CCL4L2 cytokine genes remain at baseline across all timepoints.

Let's take a look at the IL29 gene as a function of the different treatment/time groups using the plot_gene function.

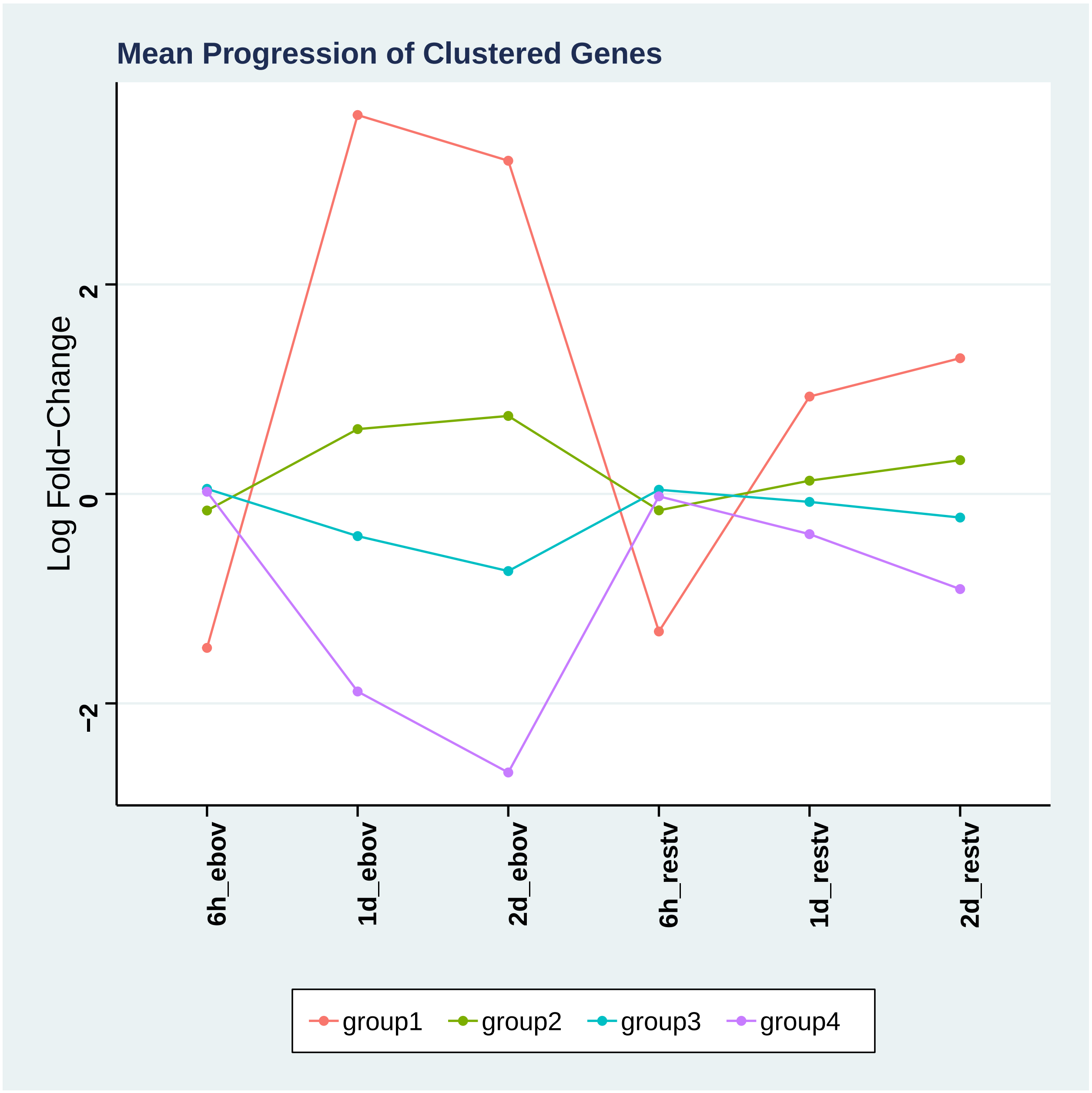
```r
plot_gene(filename="IL29.pdf", gene_name="IL29",
          groupBy="time_treatment", theme=4)
```



Here we can see that the ebola treated samples show strong expression at the day 1 and day 2 time points, whereas all other samples show no expression for this gene. In combination with what we know about the IL29 gene itself, this finding supports the hypothesis of the study that reston virus is not pathogenic in humans.

Finally, let's look into gene co-expression using the de_series function.

```r
de_series(result_list, filename="series_pattern.pdf",
          designVar="time_treatment", groupBy="time_treatment",
          method="mean", numGroups=4, writeData=TRUE, returnData=FALSE,
          theme=1)
```



This functions takes the aggregated DE gene set and clusters genes by similarity into a specified number of groups. This makes it possible to identify patterns of expression for multiple genes and extract those genes for further investigation. Here, we have specified that we should cluster DE genes into 4 groups based on their mean expression. Alternatively, we can specify method="glm" to use a generalized linear model as the criteria for clustering. The "glm" method is ideal for splitting genes into 2 or 3 groups that will likely represent downregulation and upregulation over a time series, whereas the “mean” method is more useful for examining changes between each point in the series. By increasing the number of groups, it is possible to increase the granularity of expression patterns and extract meaningful subsets of genes based on the pattern observed between groups.

Here we can see that our differentially expressed genes have been clustered into 4 groups. We can see that in some ways this image echos the other contrasts we have seen in this data, with similar patterns of expression being identified between reston and ebola virus, but with ebola infection showing significantly stronger changes in expression. For example, here group2 genes increase in expression at day 1 and slightly more at day 2. The reston virus group2 shows the same pattern of increase for those genes, only with less intensity. Similarly group3 and group4 samples become very strongly downregulated over days 1 and 2 in the ebola infected samples, but only decrease moderately for the same genes in reston infection. Group1 shows the strongest log-fold changes in both samples.

Overall what we can see from this is that patterns of expression across the time series are very similar between ebola and reston virus. The same groups of genes are behaving similarly in response to infection, however the ebola infection causes much stronger changes to gene expression. This makes sense as reston virus and ebola virus are from the same class of viruses, and reston virus appears to be non-pathogenic in humans. The fact that the patterns are so similar between differentially expressed genes however might indicate that similar biological mechanisms are at work in reston virus infection, however the infection ilicits a much weaker response. This may suggest that reston virus is indeed pathogenic in humans, contrary to the hypothesis of the example study, but is simply a much weaker infection that doesn’t cause the strong transcriptomic variation observed in ebola virus infections.

We could dig further here also by asking de_series() to return the genes clustered in any given group, and/or to write the data for each group to file. For advanced usage, the genes from a group could be fetched using this function, filtered to meet a given criteria using the de_filter function, and then re-processed into another visualization such as de_heat.

Additional Features

Themes & Labels

DEVis visualizations have a “theme” parameter that can be set from 1-6. Each theme uses a different layout and color scheme, with theme=6 always being reserved for greyscale images.

Additionally, all visualizations have a “customLabels” parameter that allows you to rename your labels. When specifying “customLabels=TRUE” you will be propted to provide new values for each label in your plot.

Data Management

The data management component of DEVis makes it easy to keep track of figures and data files generated by DEVis analysis. Note that the directory initialization will NOT overwrite existing folders.

Some functions, such as de_series, offer optional data output options in addition to their graphs. In the case of de_series, setting “writeData=TRUE” will generate a data file containing genes and log2 fold-changes for each group with filenames generated to correspond to the image file generated for the plot.

Output Formats

DEVis outputs plots in either .pdf or .png format, with the default size being 10x10 inches and a resolution of 600dpi. Some plots, which may grow in size depending on the amount of data displayed, are designed to automatically scale so that data and labels can be read. For example, the heatmap function will automatically resize the output plot to ensure that all rows can be seen and that gene id labels do not overlap and can be clearly read. The R console image will not necessarily reflect this, so when visualizing large amounts of data at once for certain plots it may be more helpful to set the output mode to file rather than screen. All text documents are exported as tab-delimited text.

Once DEVis analysis is complete, it is possible to export selected data of interest into other analysis platforms for additional analysis. Future iterations of DEVis will include additional analysis and visualization methodologies to meet the demand of the research community.

References

- 1. Olejnik, J., Forero, A., Deflubé, L. R., Hume, A. J., Manhart, W. A., Nishida, A., … Mühlberger, E. (2017). "Ebolaviruses Associated with Differential Pathogenicity Induce Distinct Host Responses in Human Macrophages." Journal of Virology, 91(11), e00179-17.
- 2. Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15(12).