1 DEAGO

1.1 Introduction

DEAGO generates user-friendly quality control (QC) and analysis reports for RNA-Seq datasets. These interactive reports can be opened simply in a web browser and provide a simple way of sharing information about your analysis with collaborators.

DEAGO uses a Perl wrapper module (**Bio-Deago**) to generate a HTML report from markdown templates by calling functions from an R package (**deago**).

There are three main steps to each analysis:

- Generating QC plots (**ggplot2**)
- Identifying differentially expressed (DE) genes (**DESeq2**)
- Performing gene ontology (GO) term enrichment analyses (**topGO**)

For more information, please see the user guide.

1.2 Learning outcomes

By the end of this tutorial you can expect to be able to:

- Understand what input files are required for DEAGO and their formats
- Generate a QC report for an RNA-seq dataset
- Generate a DE report that can be used to identify differentially expressed genes
- Generate a GO term enrichment analysis report

1.3 Tutorial sections

This tutorial comprises the following sections:

1. Introducing the tutorial dataset
2. Preparing your input data
3. Running a quality control (QC) analysis
4. Running a differential expression (DE) analysis
5. Running a GO term enrichment analysis
6. Troubleshooting

1.4 Authors

This tutorial was created by Victoria Offord.

1.5 Running the commands from this tutorial

You can run the commands in this tutorial either directly from the Jupyter notebook (if using Jupyter), or by typing the commands in your terminal window.

1.5.1 Running commands on Jupyter

If you are using Jupyter, command cells (like the one below) can be run by selecting the cell and clicking *Cell -> Run* from the menu above or using *ctrl Enter* to run the command. Let's give this a try by printing our working directory using the *pwd* command and listing the files within it. Run the commands in the two cells below.

```
pwd
```

```
ls -l
```

1.5.2 Running commands in the terminal

You can also follow this course by typing all the commands you see into a terminal window. This is similar to the "Command Prompt" window on MS Windows systems, which allows the user to type DOS commands to manage files.

To get started, select the cell below with the mouse and then either press control and enter or choose Cell -> Run in the menu at the top of the page.

```
echo cd $PWD
```

Now open a new terminal on your computer and type the command that was output by the previous cell followed by the enter key. The command will look similar to this:

```
cd /home/manager/pathogen-informatics-training/Notebooks/DEAGO/
```

Now you can follow the instructions in the tutorial from here.

1.6 Prerequisites

This tutorial assumes that you have deago and Bio-Deago installed on your computer. For download and installation instructions, please see:

- Bio-Deago (Perl module)
- deago (R package)

The following software versions were used when preparing this tutorial:

- Bio-Deago 1.2.0
- deago 1.1.2

Note: For Sanger pathogens users, these are already available on pcs5 and farm3. We also have a separate Sanger pathogen users page which will walk you through preparing your input data from the Sanger pathogen pipelines.

To check that you have installed the software correctly, you can run the following command:

```
deago -h
```

1.7 Let's get started!

To get started with the tutorial, head to the first section: Introducing the tutorial dataset.

2 Introducing the tutorial dataset

2.1 Introduction

Working through this tutorial, you will be investigating IL-22RA1 signalling at the intestinal epithelium. The dataset you will be using for this tutorial forms part of the following publication:

> **Epithelial IL-22RA1-Mediated Fucosylation Promotes Intestinal Colonization Resistance to an Opportunistic Pathogen**
> Pham TA, Clare S, Goulding D, Arasteh JM *et al.*
> *Cell Host Microbe. 2014 Oct 8;16(4):504-16. doi: 10.1016/j.chom.2014.08.017.*
> PMID: 25263220

For Sanger pathogen users, you can find this dataset under study id **2319** using the **`pf`** commands. Click here for more information.

2.2 Background

Cytokines are small, secreted proteins which effect the behavior of other cells. Due to their crucial role in cell signalling, they are often targets of RNA-Seq studies. In this study, the authors were interested in interleukin 22 (**IL-22**), an important mediator of host mucosal defence, and its receptor, interleukin 22 receptor subunit alpha 1 (**IL-22RA1**). IL-22 targets receptors on the surface of cells that line the intestines, also known as the intestinal epithelium. It can stimulate these cells to multiply, produce antimicrobial peptides and shed, providing a local defence against colonisation of bacterial and fungal pathogens.

However, the relationship between IL-22 and host defense is complex. While it may be involved in preventing colonisation, in some situations it has been shown to promote colonisation and in others, play no obvious role in susceptibility. So, in this study, the authors generated organoids (small, 3D tissue cultures which mimic the larger organ they represent) from **wild type** mice and organoids from IL-22RA1 **knockout** mice (i.e. mice which don't express/produce IL-22RA1). To investigate IL-22RA1 signalling in the intestinal epithelium, they then compared the gene expression from WT and KO organoids stimulated with IL22.

2.3 Exercise 1

In this tutorial, you will be analysing **32** RNA samples, each of which has been sequenced on an Illumina HiSeq sequencing machine. There are four conditions: wild type cells with no treatment (**WT_Ctrl**), wild type cells stimulated with IL22 (**WT_IL22**), IL-22RA1 knockout cells with no treatment (**KO_Ctrl**) and IL-22RA1 knockout cells stimulated with IL22 (**KO_IL22**). There are **four biological replicates** and **two technical replicates** for each condition.

Condition	Cell type	Treatment	Biological Replicates	Technical Replicates
WT_Ctrl	Wild type	Control	4	2
WT_IL22	Wild type	Control	4	2
KO_Ctrl	IL-22RA1 knockout	IL22	4	2
KO_IL22	IL-22RA1 knockout	IL22	4	2

Note: this is a two factor study, but we must reduce it to a single factor study to run DEAGO

Move into the directory containing the tutorial data files.

```
cd data
```

List the files and folders in the directory.

```
ls -l
```

You should see a directory called **counts**. This contains the files which have our gene counts (number of reads assigned to each gene ~ gene abundance) for each of the samples, one file per sample. Let's count them.

```
ls counts | wc -l
```

You will also see a file called **targets.txt** which tells us the relationship between sample and experimental condition.

```
head targets.txt
```

There are two files with similar names, **ensembl_mm10.tsv** and **ensembl_mm10_deago_formatted.tsv**, which are the gene annotations. The first contains the annotations as downloaded from Ensembl BioMart, one line per annotation.

```
head ensembl_mm10.tsv
```

The second contains those annotations converted for use with DEAGO, one line per gene.

```
head ensembl_mm10_deago_formatted.tsv
```

These are the input files for DEAGO. We'll take a closer look in the next section of this tutorial.

2.4 What's next?

For a quick recap of what the tutorial covers and the software you will need, head back to the Introduction.

Otherwise, let's take a closer look at how to prepare input data.

3 Preparing input data

3.1 Introduction

In this part of the tutorial we will look at preparing input data for use with DEAGO. The objectives of this part of the tutorial are:

- understand the format of the three main input data types used by DEAGO
- be able to prepare counts for use with DEAGO
- be able to prepare a sample/condition mapping file
- be able to prepare an annotation file for use with DEAGO

3.1.1 Input data

We will be using three types of input data in this tutorial. Our gene counts and sample/condition mapping are required:

- **counts**
  a directory containing count data files (one file per sample)
- **targets.txt**
  a sample/condition mapping file

We can also use an annotation file which gives us details about the genes such as their gene names and the gene ontology (GO) terms they are associated with. An annotation file is optional for quality control (QC) and differential expression (DE) analyses, but required for a GO term enrichment analysis.

- **ensembl_mm10_deago_formatted.tsv**
  _ an annotation file containing gene symbols and GO terms_

For more information on the input datatypes DEAGO uses and their formats see the user guide.

The `data` directory contains all of our input files. Let's go there.

```
cd data
```

3.1.2 Count data

DEAGO looks in a single folder for all of your count data, one file per sample. When we run an analysis we will need to tell DEAGO the location or path for this folder.

Let's take a look at our tutorial count data which can be found in the **counts** directory:

```
ls counts
```

You should see a list of 32 count files were generated using featureCounts, one file per sample. Each file contains gene counts i.e. the number or reads assigned to each gene or our gene expression profiles.

Let's take a look inside one of the count files:

```
head counts/8380_3#1.390176.pe.markdup.bam.featurecounts.csv
```

There are several things to notice about the **featureCounts** file format:

- Gene identifiers are in the first column **Geneid** (e.g. ENSMUSG00000090025)
- Gene counts are in the last column (**7**)
- The first line of the file is a comment which gives the details of the program and command used to generate the count file
- The fields are tab-delimited (****)

The reason we're highlighting this is because DEAGO has an option called **`count_type`** which we will be using in our analyses. By default, DEAGO is designed to take the expression counts which are the output from the Sanger pathogens RNA-Seq Expression pipeline. These have a different format from the featurecounts files. So, we'll be adding **`--count_type featurecounts`** to our commands which will tell DEAGO to expect count files which were generated by featureCounts and have the above characteristics.

You can also use raw count data as input as long as you have one file per sample. There are several options you can use which will configure DEAGO to be able to use them:

- **count_column** - number of column containing count values
- **skip_lines** - number of lines to skip in count file
- **count_delim** - count file delimiter
- **gene_ids** - name of column containing gene ids

3.1.3 Sample/condition mappings

DEAGO also requires a **targets** file which tells it which counts file is associated with each sample and the experimental conditions that were applied.

Let's take a look at our tutorial targets file:

```
head targets.txt
```

In our targets file, each row corresponds to a sample. There are **three** columns which DEAGO always expects to find in our **tab-delimited** targets file:

- **filename**
  name of the sample count file in the counts directory
- **condition**
  experimental condition that was applied
- **replicate**
  number or phrase representing a replicate group

The **filename** is the name of the file containing the count data for the sample. We don't need to put the directory name in the filename (e.g. count/). This is because we tell DEAGO the directory the count files are stored in and the filename is the path in relation to that directory.

The **condition** is a short label which will be used to identify the condition. The condition label should be the same for all of the samples which share that condition. Here, our condition is a combination (e.g. WT_Ctrl) of the cell type (e.g. WT) and the treatment (e.g. Ctrl).

This dataset has 4 **biological replicates** and 2 **technical replicates** for each condition represented in the **replicate** column. For example, replicate 1.2 is in biological replicate group 1 and is the second technical replicate for that sample.

Finally, you may have noticed that, in addition to the three expected columns, we also have two extra columns, **`cell_type`** and **`treatment`**. That's because the tutorial dataset had two factors, cell type (**WT**/**KO**) and treatment (**Ctrl**/**IL22**). These are here just for our information and DEAGO will ignore them.

3.1.4 Annotation file

Because we want to see the gene symbols that are linked to each gene identifier and to perform gene ontology (GO) term enrichment analyses we also need an annotation file. You should see two annotation files in your tutorial directory:

- **ensembl_mm10.tsv**
- **ensembl_mm10_deago_formatted.tsv**

The gene annotation for the mouse genome (mm10) from Ensembl BioMart, one line per annotation, is in **ensembl_mm10.tsv**.

Let's take a look at "ensembl_mm10.tsv":

```
head ensembl_mm10.tsv
```

Here you can see that the GO terms for "ENSMUSG00000064370" are split over 39 lines, one per annotation.

```
grep '^ENSMUSG00000064370' ensembl_mm10.tsv | wc -l
```

We'll look at how to get annotations from Ensembl BioMart in the exercise below.

The DEAGO-formatted annotation is in **ensembl_mm10_deago_formatted.tsv**. The difference between this and the previous file is that the DEAGO-formatted annotation has one line per gene. Essentially, we bring together all the gene names and GO terms associated to a gene together.

Let's take a look at "ensembl_mm10_deago_formatted.tsv":

```
head ensembl_mm10_deago_formatted.tsv
```

Now, if we look for the same gene "ENSMUSG00000064370" there is only one line.

```
grep '^ENSMUSG00000064370'
```

There are three **tab-delimited** columns:

- **gene identifier**
- **gene name**
- **GO terms**

Where there are multiple gene names or GO terms associated with a gene, they will be concatenated together with a **semi-colon** (';').

Note: the gene identifiers must match those found in the count data files.

Let's see how many GO terms were associated with our gene.

```
grep '^ENSMUSG00000064370' ensembl_mm10_deago_formatted.tsv | \
        cut -f 3 | tr ';' '\n' | wc -l
```

There were 36 go terms associated with ENSMUSG00000064370.

Note: we used `tr` here because `grep -c "GO:"` wouldn't work as it doesn't count multiple occurences of the same phrase in a single line.

3.2 Exercise 2

Let's take a quick look at how to get annotations for mm10 in Ensembl BioMart.

Either click on the link below or type the URL into a web browser to go the Ensembl page for the mm10 genome.

https://www.ensembl.org/Mus_musculus

Follow the steps below to download your Ensembl annotation file.

1. Click **BioMart** on the top menu
2. Select **Ensembl Genes 93** as the database (the version number may change with Ensembl updates)
3. Select **Mus musculus genes (GRCm38.p5)** as the dataset
4. Select **Attributes** from the left-hand menu
5. Click on the **+** symbol next to **GENE**: and select **Ensembl Gene ID** and **Associated Gene Name**
6. Click on the **+** symbol next to **EXTERNAL**: and select **GO Term Accession**
7. Click the **Results** button
8. Check **Export all results** to is set to **File** and **TSV** and click the **Go** button to download the annotations

The downloaded file will be called **mart_export.txt** and is the same as **ensembl_mm10.tsv**.

While `deago` is the main running command for DEAGO, there are also some other commands built in which perform the intermediate steps in a DEAGO analysis. One of those is `mart_to_deago` which will convert a BioMart annotation into a DEAGO-formatted annotation.

Let's take a look at the usage for `mart_to_deago`.

```
mart_to_deago -h
```

So, we need to specify the annotation file we want to convert using the `-a` option.

Let's try converting ensembl_mm10.tsv into a DEAGO-formatted annotation file using `mart_to_deago`.

```
mart_to_deago -a ensembl_mm10.tsv
```

This will create the file **deago_annotation.tsv** which is the same as **ensembl_mm10_deago_formatted.tsv**.

Note: Sanger pathogens users can get associated GO terms with `farm_interproscan` giving it the prokka annotation file (.gff) for the reference used to generate the count files from the RNA-Seq Expression pipeline.

3.3 Questions

Q1: How many genes are there in each of the count files?
Hint: use `grep` and/or `wc -l` to count the number of lines (minus headers) or gene identifiers in one of the count files

Q2: How many genes have associated annotations?
Hint: count the number of genes in the DEAGO-formatted annotation file

Q3: How many of those genes have associated gene names?
Hint: use `awk` and its built-in variable `NF` to count the number of lines where at least one of the three fields is missing and whose second columns contain GO terms as the gene name is missing

Q4: How many of those genes have associated GO terms?
Hint: use `awk` and its built-in variable `NF` to count the number of lines where at least one of the three fields is missing and whose second columns don't contain GO terms as they are missing

3.4 What's next?

For a quick recap of what the tutorial covers head back to the Introduction.

If you want a reintroduction to the tutorial dataset, head back to introducing the tutorial dataset.

Otherwise, let's continue on to running a quality control (QC) analysis.

4 Running a quality control (QC) analysis

4.1 Introduction

One of the most important steps in any RNA-Seq analysis is to quality control (QC) check your data. By default, DEAGO will run both a QC and a DE analysis. You can ask DEAGO to build a quick QC report by using the `--qc` option. This is a great way to get a first look at your data to try and identify any issues e.g. batch effects and outliers.

The objectives of this part of the tutorial are:

- run a QC analysis with DEAGO
- interpret the output QC report from DEAGO

4.1.1 Input files

We will need to give DEAGO two bits of information:

- *the name/location of the directory containing our gene count files (counts)*
- *the name/location of our sample/condition mapping file (targets.txt)*

4.1.2 Running a QC analysis with DEAGO

To run a quick, QC analysis with DEAGO the command would be:

```
deago -c <counts_directory> -t <targets file> --qc
```

As our count files were generated by featureCounts for this tutorial, we need to also tell DEAGO the count format with the `--count_type` option:

```
deago -c <counts_directory> -t <targets file> --count_type featurecounts --
qc
```

4.1.3 QC output files

Once your QC analysis has finished, you should see several new files:

- **`deago.config`**
  config file with key/value parameters defining the analysis
- **`deago.rlog`**
  log of the R output generated when converting the R markdown to HTML
- **`deago_markdown.Rmd`**
  R markdown used to run the analysis
- **`deago_markdown.html`**
  HTML report generated from the R markdown

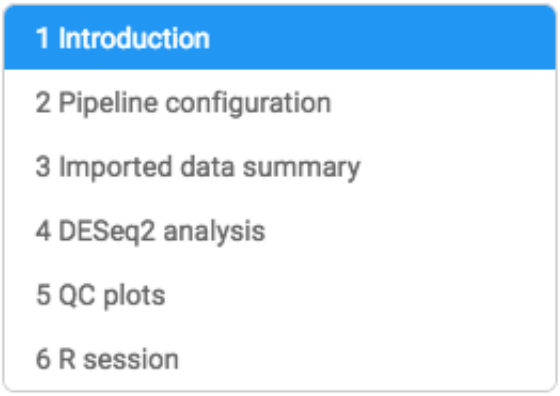
See our user guide more information about the output files from DEAGO.

4.1.4 QC analysis report

The output file we're interested in is **`deago_markdown.html`** which is your QC analysis report. Go ahead and open it in a web browser (e.g. Chrome, Firefox, IE, Safari…). You can do this by going to "File -> Open" in the top navigation or (if you have Firefox installed, use the command:

```
firefox deago_markdown.html
```

All DEAGO reports have a sidebar on the left so you can quickly navigate the report.



Navigation panel

If you click on **`Pipeline configuration`** you will see that the report contains the commands used to run the analysis (grey boxes) and their output. The **`Pipeline configuration`** section shows the parameters that were used for the analysis. This can be useful for finding input/output data, troubleshooting and debugging.

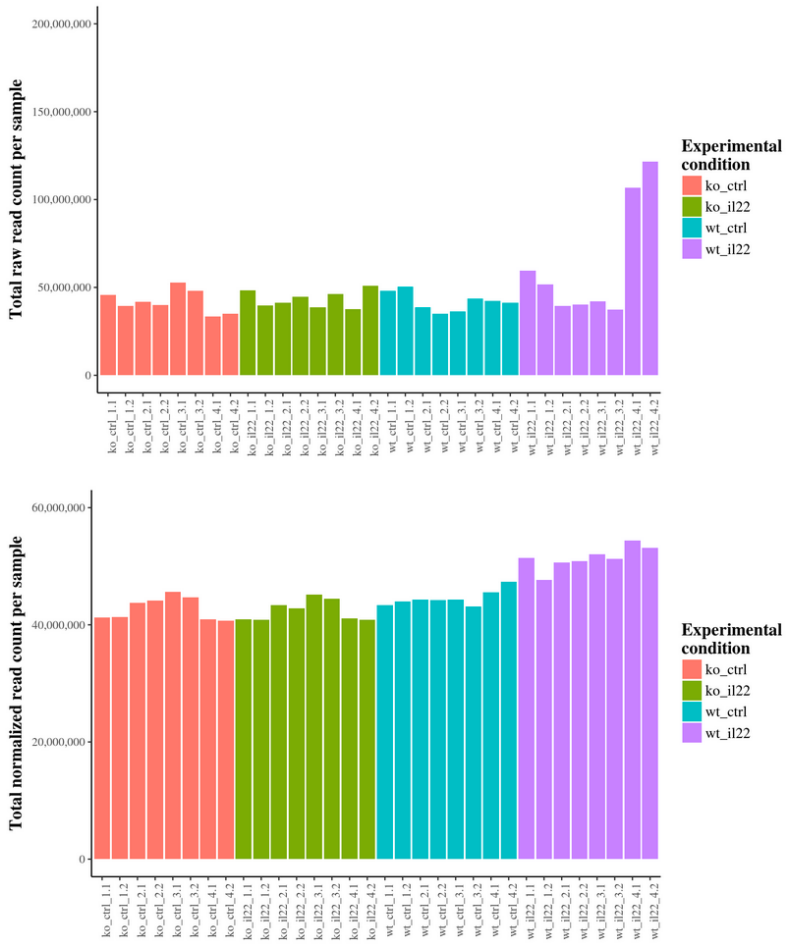
```
parameters <- importConfig("/lustre/scratch118/infgen/pathdev/vo1/deago_tutorial/qc/deago.config")
resultsDir <- makeResultDir(parameters$results_directory, parameters$keep_images)
parameters[['results']] <- resultsDir
```

parameter	value
count_column	7
count_delim	\t
count_type	featurecounts
counts_directory	/lustre/scratch118/infgen/pathdev/vo1/deago_tutorial/counts
gene_ids	Geneid
go_analysis	0
go_levels	all
keep_images	0
qc_only	1
qvalue	0.05
results_directory	/lustre/scratch118/infgen/pathdev/vo1/deago_tutorial/qc
skip_lines	1
targets_file	/lustre/scratch118/infgen/pathdev/vo1/deago_tutorial/targets.txt
alpha	0.05
columns	condition
results	/lustre/scratch118/infgen/pathdev/vo1/deago_tutorial/qc/result_20180322172446

Pipeline configuration

To take a look at the QC plots, you can click on **`QC Plots`** in the left-hand panel.

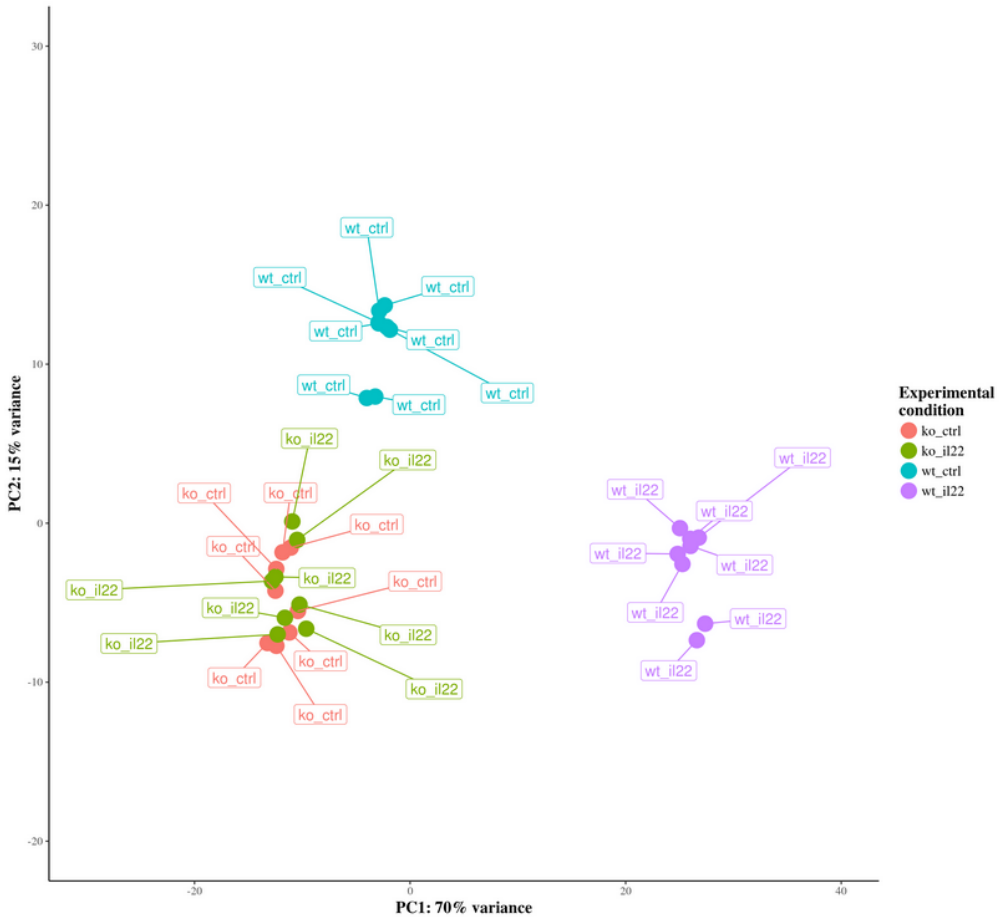
First, check to see if there were any problems with the sequencing. The *Total read counts per sample* plot does what it says on the tin and shows you the number of reads overlapping features in each sample. If one sample has a particularly low count compared to the others, it may indicate an issue and so you should take a closer look at that sample in some of the other QC plots.



Total read counts per sample plot

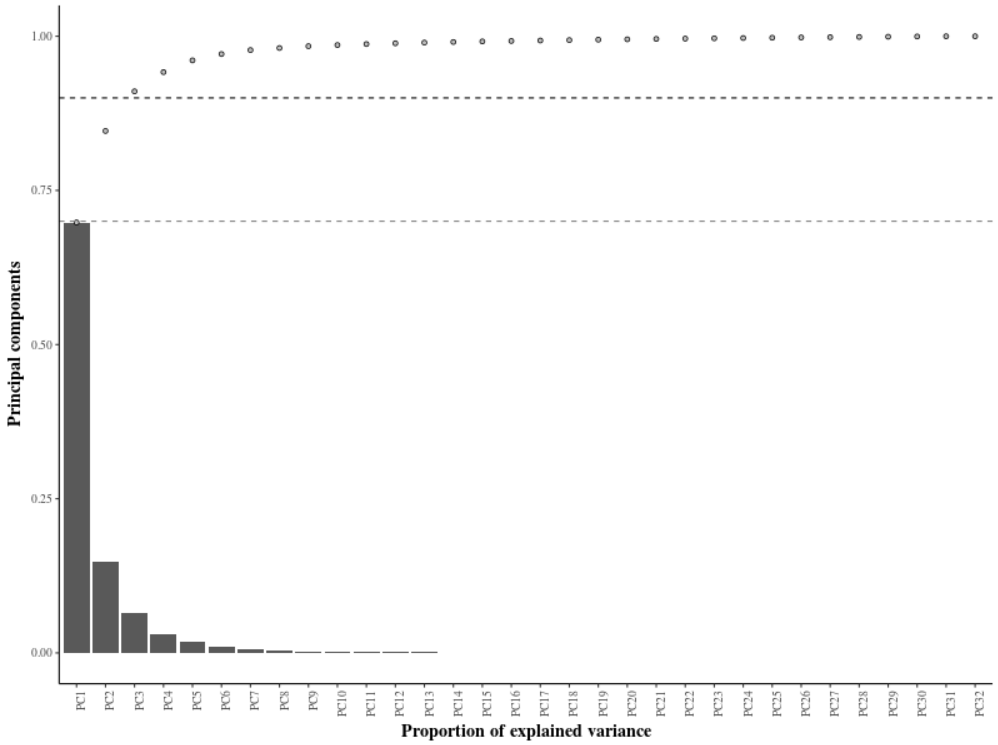
The principal component analysis (PCA) and sample-to-sample distances plot are indicators of variation and will show how well the samples cluster together. These are a quick way of spotting outliers and potential batch effects.

In the PCA plot, samples are coloured by condition and the sample labels are a combination of the sample condition and the replicate number. Here you will see whether your samples cluster reasonably well with distinct groups for each condition: **wt_ctrl**, **wt_il22** and **ko_ctrl**/**ko_il22**.



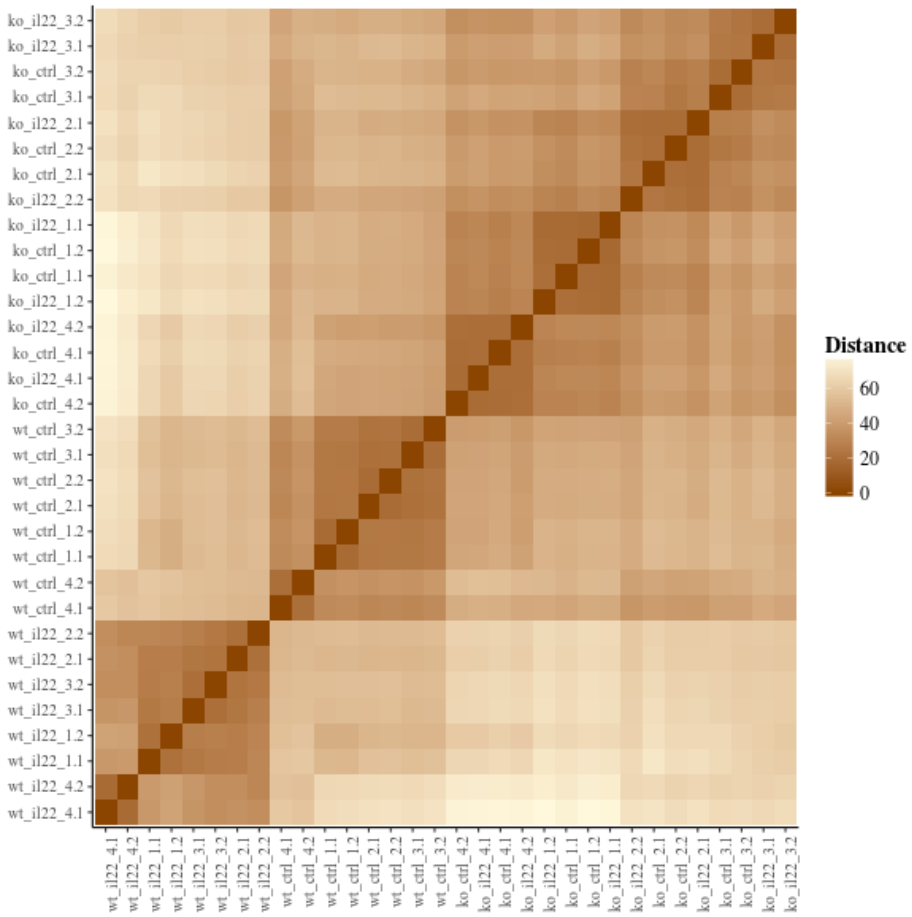
PCA plot

To compliment the PCA plot, we also have a scree plot which is a histogram of the percentage contribution of each of the principal components (PC). The points indicate the cumulative total and the lines represent broad cutoffs of 70% and 90%. Ideally, in a two-factor analysis we'd hope to see the cumulative total of PC1 and PC2 reaching 70%, although this may not always be the case.



PCA scree plot

There is also a sample-to-sample distances plot shaded by the distance (or variability) between samples. The darker the box, the more similar the samples (or the smaller the distance between them).



Sample distances plot

4.2 Exercise 3

First, let's make sure we're in the `data` directory.

```
cd data
```

Each DEAGO analysis should be self-contained, so let's create a new directory for our QC analysis.

```
mkdir qc
```

```
cd qc
```

As this is the first time we're running `deago` in this tutorial, let's take a look at the usage.

```
deago -h
```

Here's a brief explanation of the options we want to use and what they mean:

- **`--build_config`**
  tells DEAGO that we want to build a new config file using the command line parameters
- **`-c`**
  tells DEAGO the location of the folder containing our count files
- **`-t`**
  tells DEAGO the location of our sample/condition mapping file
- **`--count_type`**
  *tells DEAGO the format of the count data (e.g. featurecounts) setting the values for **count_column**, **skip_lines**, **gene_ids** and **count_delim***
- **`--qc`**
  tells DEAGO to only run the QC analysis

Now, let's get our QC report.

```
deago -build_config -c ../counts -t ../targets.txt \
          -count_type featurecounts -qc
```

Note: because we are running the analysis from our new `qc` directory, we need to us `../` to say that our directory is one level above where we are now.

4.3 Questions

Q1: Do all the samples have similar total read counts?

Q2: Look at the PCA plot. How many clusters have the samples grouped into?

Q3: Do you notice anything in the PCA and sample-to-sample distances plot that you might want to look closer at?
Hint: look at the groupings in both plots, are there sub-groupings, do they relate to anything other than the condition...

Q4: What does the `--keep_images` option do?
Hint: look at the DEAGO usage with `deago -h`

4.4 What's next?

If you want a recap of input file preparation, head back to preparing input data.

Otherwise, let's continue on to running a differential expression (DE) analysis.

5 Running a differential expression (DE) analysis

5.1 Introduction

By default, DEAGO will run both a quality control (QC) and a differential expression (DE) analysis. DE analyses try to identify genes whose expression levels differ between experimental conditions. We don't normally have enough replicates to do traditional tests of significance for RNA-Seq data. So, most methods look for outliers in the relationship between average abundance and fold change and assume most genes are not differentially expressed.

Rather than just using a fold change threshold to determine which genes are differentially expressed, DEAs use a variety of statistical tests for significance. These tests give us a **p-value** which is an estimate of how often your observations would occur by chance.

However, we perform these comparisons for each one of the thousands of genes/transcripts in our dataset. A p-value of 0.01 estimates a probability of 1% for seeing our observation just by chance. In an experiment like ours with 5,000 genes we would expect 5 genes to be significantly differentially expressed by chance (i.e. even if there were no difference between our conditions). Instead of using a p-value we can use an **adjusted p-value**, also known as the **q-value**, which accounts for the multiple testing and adjusts the p-value accordingly.

The objectives of this part of the tutorial are:

- run a DE analysis with DEAGO
- interpret the output DE report from DEAGO

5.1.1 Input files

We will need to give DEAGO two bits of information:

- *the name/location of the directory containing our gene count files (counts)*
- *the name/location of our sample/condition mapping file (targets.txt)*

We can optionally give DEAGO a formatted annotation file which contains gene names. These are often more recognisable than the unique gene identifiers found in the counts files. To do this, we use the `-a` option.

5.1.2 Running a DE analysis with DEAGO

To run a quick, DE analysis with DEAGO the command would be:

```
deago -c <counts_directory> -t <targets file>
```

As our count files were generated by featureCounts for this tutorial, we need to also tell DEAGO the count format with the `--count_type` option:

```
deago -c <counts_directory> -t <targets file> --count_type featurecounts
```

As we want to have the gene names in our output tables and plots, we need to provide our formatted annotation file using the `-a` option.

Finally, we will be using the `--control` option which tells DEAGO the condition you want to use as your reference or control, in this case **WT_Ctrl**. We use **`--control`** to define our reference condition because, by default, R chooses the reference condition based on alphabetical order. It would assume that from our four conditions (**KO_Ctrl**, **KO_IL22**, **WT_Ctrl** and **WT_IL22**) that KO_Ctrl is our reference condition because it is first alphabetically. The value you use **must** be in the condition column in your targets file and is *case insensitive*.

```
deago -c <counts directory> -t <targets file> --count_type featurecounts \
    -a <annotation file> --control <control>
```

DEAGO also makes an assumption that you want the FDR cutoff (alpha) to be **0.05** (default). If you are expecting to use a different cutoff in your downstream filtering, use the **`-q`** option to define the FDR cutoff (e.g. -q 0.01).

5.1.3 Output files/directories

Once your DE analysis has finished, you should see several new files and directories:

- **`deago.config`**
  config file with key/value parameters defining the analysis
- **`deago.rlog`**
  log of the R output generated when converting the R markdown to HTML
- **`deago_markdown.Rmd`**
  R markdown used to run the analysis
- **`deago_markdown.html`**
  HTML report generated from the R markdown
- **`results_<timestammp>`**
  directory containing unfiltered DE analysis results and normalised counts for all genes, one file per contrast

Results directory The report tables are limited to genes with an adjusted p-value < 0.01 and a log2 fold change >= 2 or <= -2. However, you are likely to want to explore and filter these results using different thresholds. So, DEAGO also writes the unfiltered results table containing all genes to individual files, one per contrast in your timestamped results directory.

So, for the full results of the contrast between WT and KO cells treated with IL22 you would look at:

```
results_<timestamp>/wt_il22_vs_ko_il22_q0.05.txt
```

5.1.4 DE analysis report

The output file we're interested in is **`deago_markdown.html`** which is your DE analysis report. Go ahead and open it in a web browser (e.g. Chrome, Firefox, IE, Safari…). You can do this by going to "File -> Open" in the top navigation or (if you have Firefox installed, use the command:

```
firefox deago_markdown.html
```

In addition to the QC sections we saw before, you should now see a new option in the left-hand sidebar called **`Pairwise contrasts`**. Click on it and it will take you to your DE analysis results.

First there is a **`Contrast summary`** section which contains a summary table showing how many genes are up-regulated or down-regulated in each contrast (comparison between two sample groups). We can see that there were no differentially expressed (DE) genes between the knock-out (**KO**) samples induced with IL22 (**ko_il22**) and the control knockout samples. However, there were 860 DE genes between wildtype (**WT**) and knockout (**KO**) samples induced with IL22, 510 up-regulated in the WT samples compared to the KO samples and 350 down-regulated.

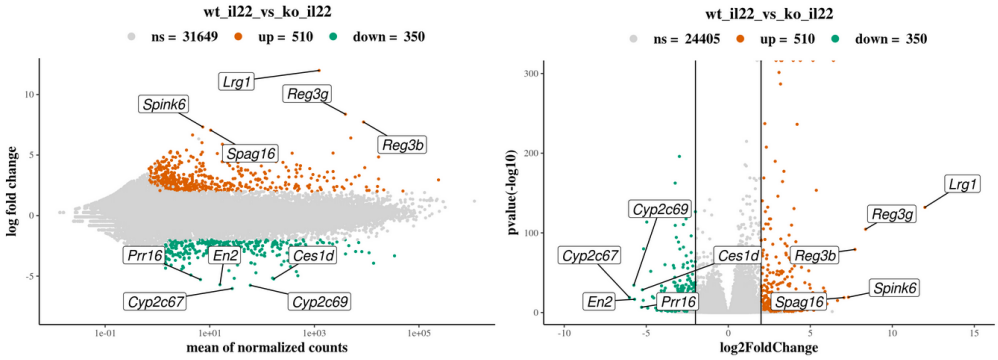
	up-regulated	down-regulated	total
ko_ctrl_vs_wt_ctrl	132	131	263
ko_il22_vs_wt_ctrl	109	130	239
wt_il22_vs_wt_ctrl	312	160	472
ko_il22_vs_ko_ctrl	0	0	0
wt_il22_vs_ko_ctrl	471	376	847
wt_il22_vs_ko_il22	510	350	860

DE summary table

If there are 2-4 contrasts in the analysis, there will also be a Venn diagram showing the overlap/differences in total DE genes between contrasts. We have 6 contrasts in this analysis, so no Venn diagram was generated, but an example would be:

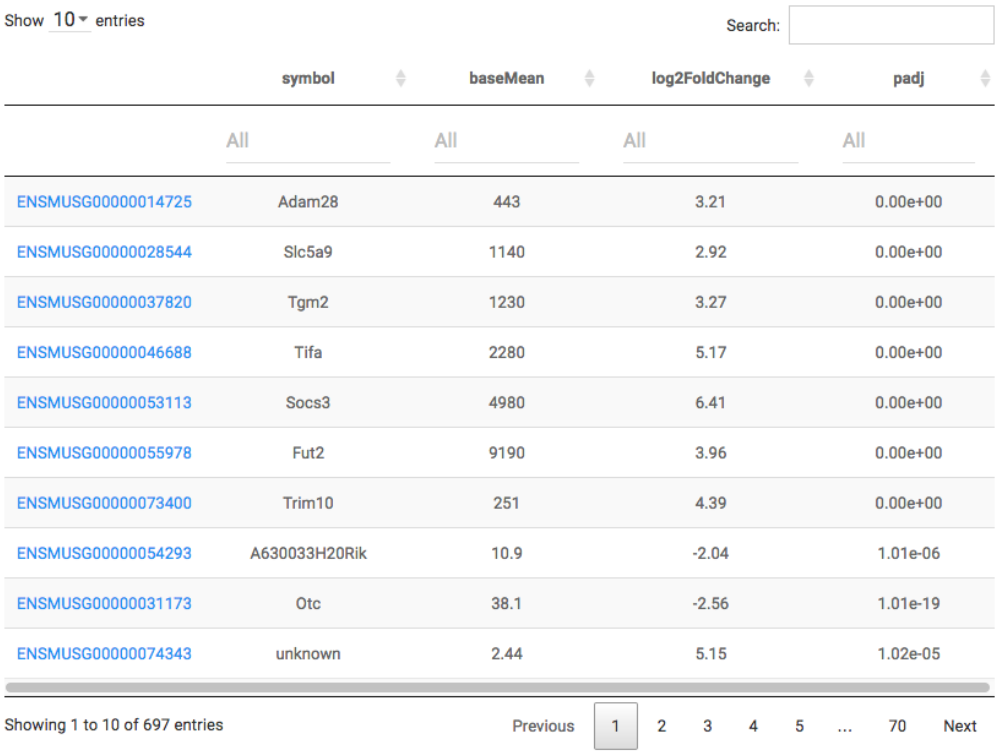
DE Venn diagram

The DE analysis report then has a series of subsections, one per contrast. Each contrast section has an MA plot and a volcano plot. The top 5 up- and down-regulated gene identifiers are labelled on plots. If an annotation with gene symbols was used then the point labels will be the gene symbols and not the gene identifiers.



DE volcano and MA plots

Each contrast section will also have a DE results table which contains genes with an adjusted p-value < 0.01 and a log2 fold change >= 2 or <= -2. This is to reduce the number of genes in the table so that the HTML report is compact and sharable.



Show 10 entries Search:

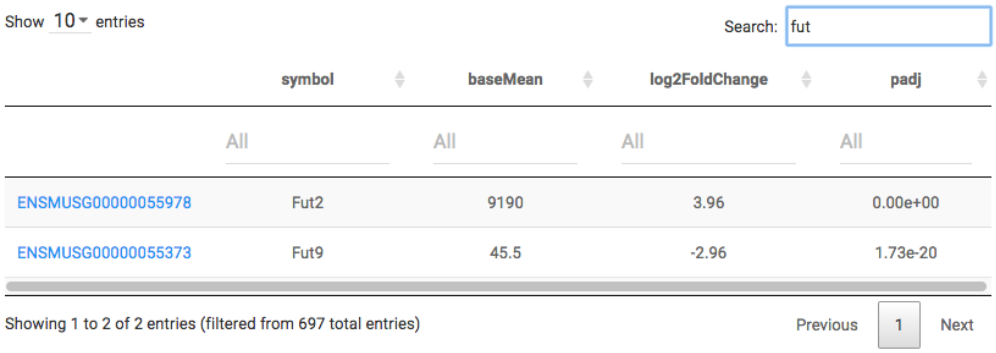
	symbol	baseMean	log2FoldChange	padj
	All	All	All	All
ENSMUSG00000014725	Adam28	443	3.21	0.00e+00
ENSMUSG00000028544	Slc5a9	1140	2.92	0.00e+00
ENSMUSG00000037820	Tgm2	1230	3.27	0.00e+00
ENSMUSG00000046688	Tifa	2280	5.17	0.00e+00
ENSMUSG00000053113	Socs3	4980	6.41	0.00e+00
ENSMUSG00000055978	Fut2	9190	3.96	0.00e+00
ENSMUSG00000073400	Trim10	251	4.39	0.00e+00
ENSMUSG00000054293	A630033H20Rik	10.9	-2.04	1.01e-06
ENSMUSG00000031173	Otc	38.1	-2.56	1.01e-19
ENSMUSG00000074343	unknown	2.44	5.15	1.02e-05

Showing 1 to 10 of 697 entries Previous 1 2 3 4 5 ... 70 Next

DE contrast results table

These tables contain the DESeq2 results and are where you will find your adjusted p-values and log2 fold change values. Also, as our gene identifiers are Ensembl identifiers, they have been converted to a link and if you click on one, it will take you to the current Ensembl page for that gene stable ID. In this example, we had include an annotation in the analysis and so the gene symbols are also shown.

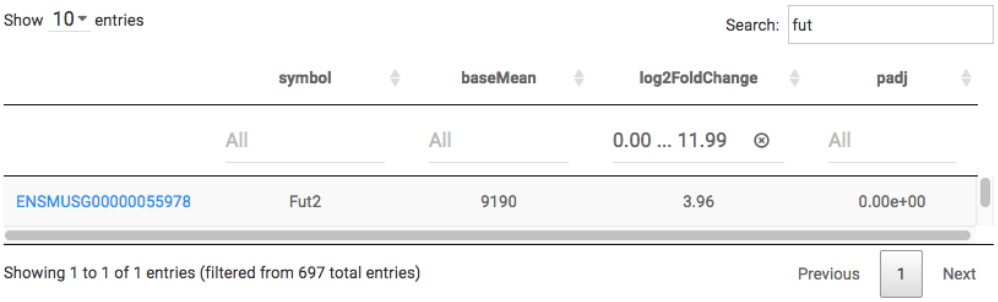
All of the tables are interactive and can be searched or filtered. The paper describes the up-regulation of *Fut2* by IL-22RA1 signalling, so let's take a look. The search box at the top right searches the whole table, so we can use it to search for any *Fut* genes.



Searching DE tables

This gives us two genes: *Fut2* and *Fut9*.

Now, say we wanted to only see the up-regulated *Fut* genes. We can limit searches and filters to a single column by using the search/filter boxes at the top of each column. Use the selector at the top of the **`log2FoldChange`** column to only include values greater than 0 (i.e. drag the left selector).



Searching DE tables

We are now left with the only up-regulated *Fut* gene, *Fut2*.

5.2 Exercise 4

First, let's make sure we're in the `data` directory.

```
cd data
```

```
(no output)
```

Each DEAGO analysis should be self-contained, so let's create a new directory for our DE analysis.

```
mkdir de_analysis
```

```
cd de_analysis
```

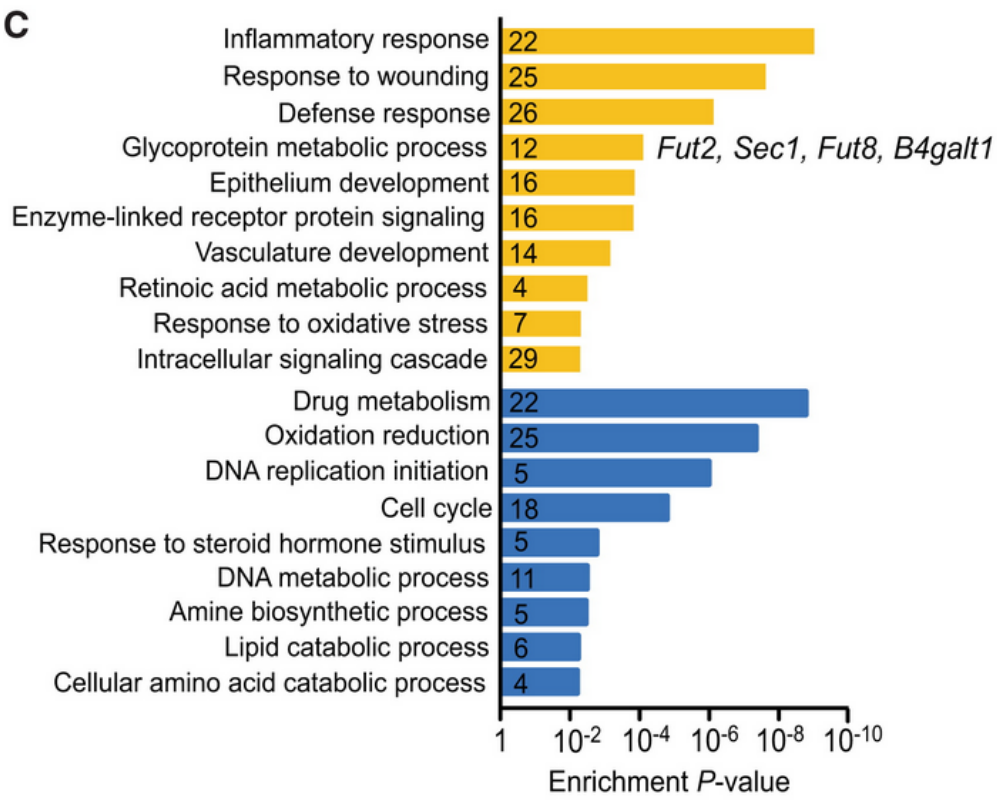
```
(no output)
```

Now, let's get our DE report.

```
deago -build_config -c ../counts -t ../targets.txt \
          -count_type featurecounts \
          -a ../ensembl_mm10_deago_formatted.tsv \
          -control WT_Ctrl
```

5.3 Questions

In Figure 5C (below), the authors have highlighted four genes: ***Fut2***, ***Sec1***, ***Fut8*** and ***B4galt1*** which are associated with glycosylation.



GO term enrichment

We have already found *Fut2*, answer the following questions for the remaining 3 genes using the contrast results for WT and KO cells treated with IL22.

Q1: What is the gene identifier (geneID)?

Q2: What is the log2 fold change?

Q3: What is the adjusted p-value?

Hint: you may want to use `awk` *to look at columns 36 and 40 in the unfiltered results files for that contrast if the genes are not found in the report tables*

5.4 What's next?

If you want a recap of input file preparation, head back to running a quality control (QC) analysis.

Otherwise, let's continue on to running a GO term enrichment analysis.

6 Running a gene ontology (GO) term enrichment analysis

6.1 Introduction

Once you have your list of differentially expressed (DE) genes, the next step is often to ask whether the DE genes associated with a particular biological process or function. To do this, we can perform a gene ontology (GO) term enrichment analysis. A gene ontology (GO) is a controlled vocabulary which describes gene products from in any organism (i.e. species agnostic). These descriptions are broken down into three categories:

- **molecular function (MF)** *what the gene product does*
- **biological process (BP)** *why the gene product does this*
- **cell component (CC)** *where the gene product acts*

DEAGO uses **topGO** to compare a target gene list (e.g. your DE genes) to the background (all genes) and identifies GO terms which are significantly enriched or over-represented in your gene list.

The objectives of this part of the tutorial are:

- run a GO analysis with DEAGO
- interpret the output GO report from DEAGO

6.1.1 Input files

We will need to give DEAGO three bits of information:

- *the name/location of the directory containing our gene count files (counts)*
- *the name/location of our sample/condition mapping file (targets.txt)*
- *the name/location of our formatted annotation file (ensembl_mm10_deago_formatted.tsv)*

6.1.2 Running a GO analysis with DEAGO

To simplest command to run a GO analysis with DEAGO the command would be:

```
deago -c <counts_directory> -t <targets file> -a <annotation file> --go
```

To indicate that we want to run a GO analysis, we use the `--go` option. Notice here that we **must** provide a formatted annotation file using the `-a` option as we need to know the GO terms assoicated with each gene for the analysis.

However, as before, our count files were generated by featureCounts for this tutorial, so we need to also tell DEAGO the count format with the `--count_type` option:

```
deago -c <counts_directory> -t <targets file> -a <annotation file> --
go \
    --count_type featurecounts
```

We're also using the `--control` option, as before, which tells DEAGO the condition you want to use as your reference or control, in this case **WT_Ctrl**.

```
deago -c <counts_directory> -t <targets file> -a <annotation file> --
go\
    --count_type featurecounts --control <control>
```

6.1.3 Output files

Once your GO analysis has finished, you should see several new files and directories:

- **`deago.config`**
  config file with key/value parameters defining the analysis
- **`deago.rlog`**
  log of the R output generated when converting the R markdown to HTML
- **`deago_markdown.Rmd`**
  R markdown used to run the analysis
- **`deago_markdown.html`**
  HTML report generated from the R markdown
- **`results_<timestammp>`**
  directory containing unfiltered DE analysis results and normalised counts for all genes, one file per contrast

Results directory DEAGO also writes the GO term enrichment analysis results tables containing the top 30 significantly enriched GO terms to individual files, split by GO term level (MF and BP), in your timestamped results directory. If possible, there will be three GO tables for each GO term level containing the results for analyses using all genes, up-regulated genes only and down-regulated genes

- [contrast]_BP.tsv
- [contrast]_BP_up.tsv
- [contrast]_BP_down.tsv
- [contrast]_MF.tsv
- [contrast]_MF_up.tsv
- [contrast]_MF_down.tsv

6.1.4 GO analysis report

The output file we're interested in is deago_markdown.html which is your GO analysis report. Go ahead and open it in a web browser (e.g. Chrome, Firefox, IE, Safari…). You can do this by going to "File -> Open" in the top navigation or (if you have Firefox installed, use the command:

```
firefox deago_markdown.html
```

You'll already be familiar with the **`QC plots`** and **`Pairwise contrasts`** sections. Click on **`Pairwise contrasts`** and then go to the contrast subsection for **`wt_il22_vs_ko_il22`**. You should now see six new subsections, *6.7.2 - 6.7.7*, which contain your GO term enrichment analysis results for that contrast.

The first three subsections have the prefix **`GO term enrichment – BP`** and contain the results for the *Biological Processes (BP)*. The remaining three subsections have the prefix **`GO term enrichment – MF`** and contain the results for the *Molecular Functions (MF)*.

For each GO level (BP or MF) there is a subsection containing the results from GO term enrichment analyses which were run using all DE genes, up-regulated genes only and down-regulated genes only.

Let's take a look at the BP (upregulated genes only) table for the contrast wt_il22_vs_ko_il22 (subsection 6.7.3). All of the results tables are interactive. Despite performing a single-factor analyis, we can still find the top GO description for up-regulated genes *inflammatory response* in our results table (GO:0006954).

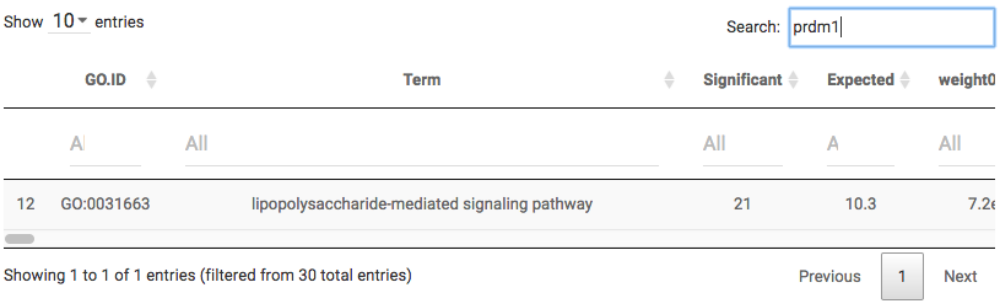
Show 10 entries Search:

	GO.ID	Term	Significant	Expected	weight0
	All	All	All	All	All
1	GO:0015031	protein transport	554	371.42	8.0e
2	GO:0006886	intracellular protein transport	288	203.35	1.5e
3	GO:0043123	positive regulation of I-kappaB kinase/NF-kappaB signaling	68	35.72	1.6e
4	GO:0018345	protein palmitoylation	19	6.14	2.5e
5	GO:0016567	protein ubiquitination	228	161.72	4.5e
6	GO:0045944	positive regulation of transcription from RNA polymerase II promoter	314	240.39	8.0e
7	GO:0051291	protein heterooligomerization	53	26.51	9.9e
8	GO:0007030	Golgi organization	51	23.01	1.1e
9	GO:0018105	peptidyl-serine phosphorylation	90	65.52	4.8e
10	GO:0035023	regulation of Rho protein signal transduction	47	26.73	5.1e

Showing 1 to 10 of 30 entries Previous 1 2 3 Next

GO term enrichment results table for wt_il22_vs_ko_il22 BP (upregulated genes only)

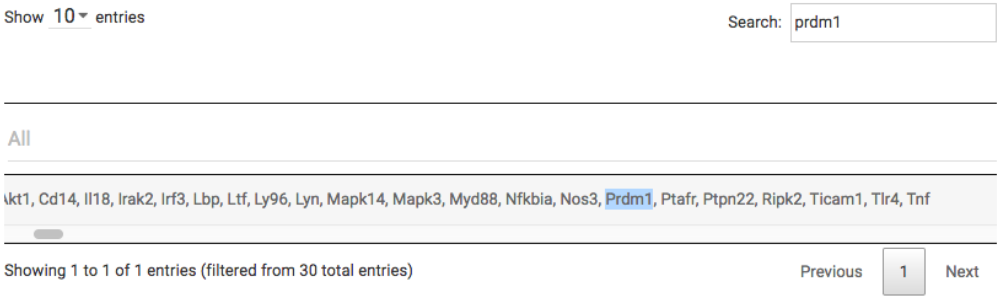
The paper also mentions *Prdm1* which is upregulated by IL-22RA1 signalling and is a susceptibility gene in to inflammatory bowel disease. Let's see whether it's associated with any of the enriched GO terms. Type "prdm1" in the top right search box to search the whole table.



Search for Prdm1 (left)

We can see that one GO term matches: GO:0031663 (lipopolysaccharide-mediated signaling pathway).

Scrolling to the right, we can see how the search found this match. For each GO term, the DE genes which are associated with that term are also shown in the table.



Search for Prdm1 (right)

6.2 Exercise 5

First, let's make sure we're in the data directory.

```
cd data
```

Each DEAGO analysis should be self-contained, so let's create a new directory for our GO analysis.

```
mkdir go_analysis
```

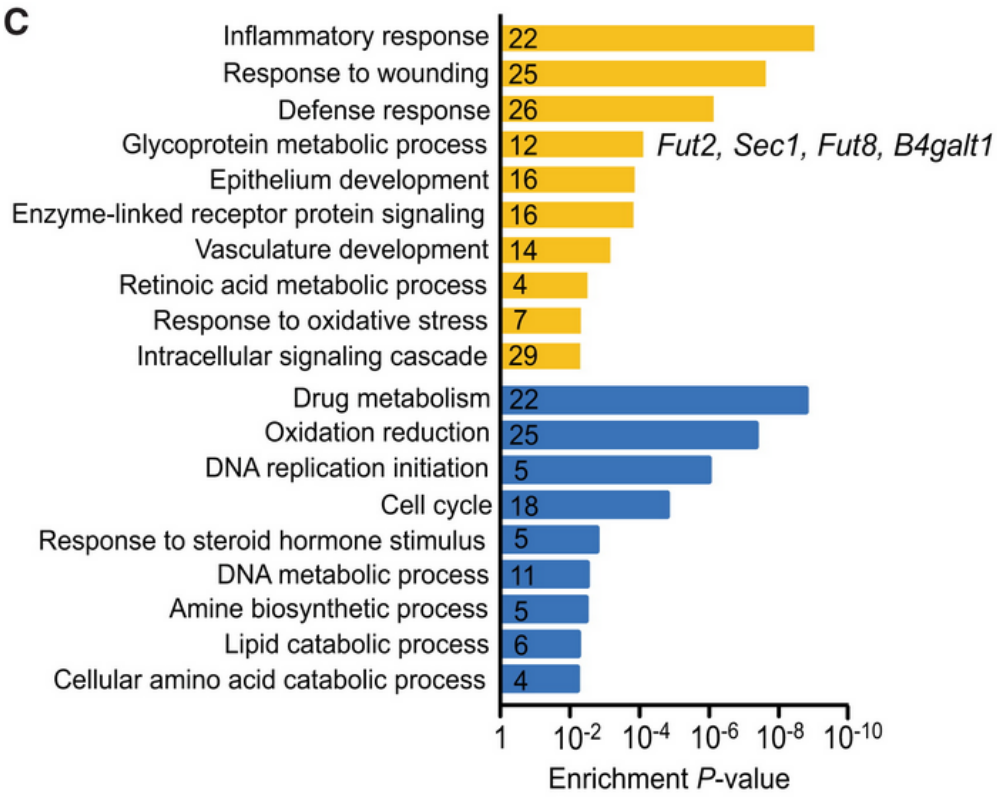
```
cd go_analysis
```

Now, let's get our GO report.

```
deago -build_config -c ../counts -t ../targets.txt \
          -count_type featurecounts \
          -a ../ensembl_mm10_deago_formatted.tsv \
          -control WT_Ctrl -go
```

6.3 Questions

In Figure 5C (below), the authors have highlighted four genes: ***Fut2***, ***Sec1***, ***Fut8*** and ***B4galt1*** which are associated with glycosylation.



GO term enrichment

Answer the following question for all four genes:

Q1: Which biological processes is this upregulated gene associated with?

6.4 What's next?

If you want a recap of input file preparation, head back to running a differential expression (DE) analysis.

Otherwise, let's continue on to troubleshooting.

7 Troubleshooting

If **deago_markdown.html** isn't generated, the first thing to do is check the end of the R log file (**deago.rlog**). If successful it should be similar to:

```
output file: deago_markdown.knit.md

/software/pathogen/external/apps/usr/bin/pandoc +RTS -K512m -
RTS deago_markdown.utf8.md --to html --from
markdown+autolink_bare_uris+ascii_identifiers+tex_math_single_backslash --
output /lustre/scratch118/infgen/pathdev
/vo1/deago_tutorial_test/deago_analysis/deago_markdown.html --smart --
email-obfuscation none --self-contained
--standalone --section-divs --table-of-contents --toc-depth 3 --
variable toc_float=1 --variable
toc_selectors=h1,h2,h3 --variable toc_collapsed=1 --
variable toc_smooth_scroll=1 --variable toc_print=1
--template /software/pathogen /external/lib/R-3.4/rmarkdown/rmd/h/default.html --
no-highlight
--variable highlightjs=1 --number-sections --variable 'theme:paper' --
include-in-header
/tmp/RtmpyTxHuS/rmarkdown-str1782209e9bc2.html --mathjax --variable
'mathjax-url:https://mathjax.rstudio.com/latest/MathJax.js?config=TeX-
AMS-MML_HTMLorMML'

In addition: Warning messages:
1: Removed 778 rows containing non-finite values (stat_density).
2: Removed 778 rows containing non-finite values (stat_density).
```

If not, it will show the error which stopped the report being generated. For example:

Quitting from lines 206-208 (deago_markdown.Rmd)

```
Error in .local(.Object, ...) : allGenes must be a factor with 2 levels
Calls: <Anonymous> ... prepareGOdata -> new -> initialize -> initialize -> .local
In addition: Warning messages:
1: Removed 778 rows containing non-finite values (stat_density).
2: Removed 778 rows containing non-finite values (stat_density).

Execution halted
```

This suggests you should look at what's on lines 206-208 in **deago_markdown.Rmd** and try to work out the cause of the error. Once you've fixed it, you can use **deago_markdown_to_html** to try and convert your modified markdown file to a new HTML report. If this doesn't work, you can always try running the commands from the markdown file in R manually to debug the issue.

If this doesn't work and you think there might be bug, please email path-help@sanger.ac.uk attaching your R markdown file, R log file and config file.

Congratulations, you have now reached the end of this tutorial!

To go back to the beginning, you'll want to head to the introduction. Or, for the answers to the tutorial questions, head to the answer section (answers.ipynb).