BIO8068 Data visualisation and management

Handling messy eDNA or gene expression data in R

Roy Sanderson

## 1. Introduction

When analysing phylogenetics data, eDNA samples, or microarray data you are generally best using one of the standard tools provided, that you have already covered in your Phylogenetics module. There is also a specialist set of packages in R, called Bioconductor <https://www.bioconductor.org> although this contains thousands (literally!) of individual packages, many of which need to be linked together for an analysis pipeline. Good packages that might be of interest in Bioconductor include phyloseq <https://bioconductor.org/packages/release/bioc/html/phyloseq.html> which is designed for high-throughput microbiome data, but also integrates effectively with the vegan package for multivariate analysis that I covered with you on the data analysis module. Another good package is ape (Analysis of Phylogenetics and Evolution) <http://ape-package.ird.fr/> which also has a related text of the same name (Emmanuel Paradis, 2012, Springer).

However, it is still useful to understand how some of these analyses can be undertaken using tidyverse packages. This exercise demonstrates the challenges of these types of data, using a dataset from Brauer et al (2008) Coordination of growth rate, cell cycle, stress response and metabolic activity in yeast <https://doi.org/10.1091/mbc.e07-08-0779>. The exercise shows how dplyr, tidyr and ggplot can be used to your advantage with gene expression data, and is adapted from an exercise by David Robinson of the <https://varianceexplained.org> website.

## 2. Background

In gene regulation, a cell can control which genes are transcribed from DNA to RNA, and hence ‘expressed’. When a gene is expressed it might speed up or slow down growth, nutrient exchange, how the cell responds to stimuli etc. Gene expression microarrays allow us to measure how much of each gene is expressed under a particular condition, and hence what is its function. Brauer used microarrays to test the effect of starvation and growth rate on yeast, as it is easy to manipulate e.g. supplies of glucose (energy), leucine (essential amino acide) or ammonium (nitrogen). By restricting or ‘starving’ the availability of these you can find genes that:

* raise or lower their activity in response to growth rate
* respond differently when different nutrients are being limited

### Getting started

Download the Brauer 2008 file from Blackboard, create an R project, and store the file in a data sub-folder within the project folder.

## 3. Tidy the data with dplyr and tidyr

Import the data, view it and understand its size; it is tab-delimited so we’ll use read\_delim with \t to indicate tabs:

# Read and examine the data  
library(readr)  
original\_data <- read\_delim("data/Brauer2008\_DataSet1.tds", delim="\t")

## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## GID = col\_character(),  
## YORF = col\_character(),  
## NAME = col\_character()  
## )

## See spec(...) for full column specifications.

original\_data

## # A tibble: 5,537 x 40  
## GID YORF NAME GWEIGHT G0.05 G0.1 G0.15 G0.2 G0.25 G0.3 N0.05  
## <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 GENE… A\_06… SFB2… 1 -0.24 -0.13 -0.21 -0.15 -0.05 -0.05 0.2   
## 2 GENE… A\_06… " … 1 0.28 0.13 -0.4 -0.48 -0.11 0.17 0.31  
## 3 GENE… A\_06… QRI7… 1 -0.02 -0.27 -0.27 -0.02 0.24 0.25 0.23  
## 4 GENE… A\_06… CFT2… 1 -0.33 -0.41 -0.24 -0.03 -0.03 0 0.2   
## 5 GENE… A\_06… SSO2… 1 0.05 0.02 0.4 0.34 -0.13 -0.14 -0.35  
## 6 GENE… A\_06… PSP2… 1 -0.69 -0.03 0.23 0.2 0 -0.27 0.17  
## 7 GENE… A\_06… RIB2… 1 -0.55 -0.3 -0.12 -0.03 -0.16 -0.11 0.04  
## 8 GENE… A\_06… VMA1… 1 -0.75 -0.12 -0.07 0.02 -0.32 -0.41 0.11  
## 9 GENE… A\_06… EDC3… 1 -0.24 -0.22 0.14 0.06 0 -0.13 0.3   
## 10 GENE… A\_06… VPS5… 1 -0.16 -0.38 0.05 0.14 -0.04 -0.01 0.39  
## # … with 5,527 more rows, and 29 more variables: N0.1 <dbl>, N0.15 <dbl>,  
## # N0.2 <dbl>, N0.25 <dbl>, N0.3 <dbl>, P0.05 <dbl>, P0.1 <dbl>,  
## # P0.15 <dbl>, P0.2 <dbl>, P0.25 <dbl>, P0.3 <dbl>, S0.05 <dbl>,  
## # S0.1 <dbl>, S0.15 <dbl>, S0.2 <dbl>, S0.25 <dbl>, S0.3 <dbl>,  
## # L0.05 <dbl>, L0.1 <dbl>, L0.15 <dbl>, L0.2 <dbl>, L0.25 <dbl>,  
## # L0.3 <dbl>, U0.05 <dbl>, U0.1 <dbl>, U0.15 <dbl>, U0.2 <dbl>,  
## # U0.25 <dbl>, U0.3 <dbl>

dim(original\_data)

## [1] 5537 40

It is a big dataset, with columns G0.05 and N0.3 indicating gene expression values. The G0.05 indicates glucose was limiting nutrient, and the growth rate was 0.05. The higher the value in the column, the more the gene is expressed. Use the View command to explore the data more. The experiment had 6 limiting nutrients, at 6 growth rates, giving 36 samples (i.e. 36 columns) of gene expression data.

**Question** : what is messy about this data??

* The column headings are values (G, N, P etc. and 0.05-0.3) rather than variable names
* The NAME column contains a muddle of information, e.g. the first few entries:

# First entry of NAME  
original\_data$NAME[1:3]

## [1] "SFB2 || ER to Golgi transport || molecular function unknown || YNL049C || 1082129"   
## [2] " || biological process unknown || molecular function unknown || YNL095C || 1086222"   
## [3] "QRI7 || proteolysis and peptidolysis || metalloendopeptidase activity || YDL104C || 1085955"

The NAME column contains:

* **Gene name** e.g. SFB2, although not all genes have names
* **Biological process** e.g. “proteolysis and peptidolysis”
* **Molecular function** e.g. “metalloendopeptidase activity”
* **Systematic ID** e.g. YNL049C. Unlike a gene name, every gene in this dataset has a systematic ID.3
* **Another ID number** e.g. 1082129. I don’t know what this number means!

The NAME column contains a muddle about systematic ID’s, biological information etc. So we need to sort these problems out. Remember, do this in R, don’t be tempted to try it in Excel.

We can now use the tidyr function separate to split up this column; there are || bars between each entry, and as this is treated as a special character in R we’ll need **2** backslash symbols (unlike the single backslash needed before ‘t’ for ‘tab’ in the read\_delim earlier).

# Separate the NAME  
library(dplyr)  
library(tidyr)  
  
cleaned\_data <- original\_data %>%  
 separate(NAME, c("name", "BP", "MF", "systematic\_name", "number"), sep = "\\|\\|")

If you look at some of the BP entries you’ll see that irritatingly there is now whitespace at the end of some, but not all, columns. We need to be consistant, so use mutate\_each with trimws. We can also drop the GID, YORF, GWEIGHT and mysterious number columns:

# Separate NAME, cleanup whitespace  
cleaned\_data <- original\_data %>%  
 separate(NAME, c("name", "BP", "MF", "systematic\_name", "number"), sep = "\\|\\|") %>%  
 mutate\_each(funs(trimws), name:systematic\_name)

## Warning: funs() is soft deprecated as of dplyr 0.8.0  
## please use list() instead  
##   
## # Before:  
## funs(name = f(.)  
##   
## # After:   
## list(name = ~f(.))  
## This warning is displayed once per session.

# Separate NAME, cleanup whitespace and drop columns  
cleaned\_data <- original\_data %>%  
 separate(NAME, c("name", "BP", "MF", "systematic\_name", "number"), sep = "\\|\\|") %>%  
 mutate\_each(funs(trimws), name:systematic\_name) %>%  
 select(-number, -GID, -YORF, -GWEIGHT)

The next problem we have is that the column headers contain values, not variables. We need to do 3 things:

* Get the limiting nutrients into a column. These are glucose (G), ammonium (N), sulfate (S), phosphate (P), uracil (U), leucine (L)
* Get the growthrate into a column. 0.05 (slow growth) through to 0.3 (rapid growth)
* Get the gene expression into a column. Currently these values are spread across multiple columns

What to do next: the clue is in the last bullet point. The opposite of spread is of course gather which is a nice function in the tidyr package. We’ll add it as an extra line:

# Separate NAME, cleanup whitespace, drop columns and gather  
cleaned\_data <- original\_data %>%  
 separate(NAME, c("name", "BP", "MF", "systematic\_name", "number"), sep = "\\|\\|") %>%  
 mutate\_each(funs(trimws), name:systematic\_name) %>%  
 select(-number, -GID, -YORF, -GWEIGHT) %>%  
 gather(sample, expression, G0.05:U0.3)  
  
cleaned\_data

## # A tibble: 199,332 x 6  
## name BP MF systematic\_name sample expression  
## <chr> <chr> <chr> <chr> <chr> <dbl>  
## 1 SFB2 ER to Golgi … molecular functio… YNL049C G0.05 -0.24  
## 2 "" biological p… molecular functio… YNL095C G0.05 0.28  
## 3 QRI7 proteolysis … metalloendopeptid… YDL104C G0.05 -0.02  
## 4 CFT2 mRNA polyade… RNA binding YLR115W G0.05 -0.33  
## 5 SSO2 vesicle fusi… t-SNARE activity YMR183C G0.05 0.05  
## 6 PSP2 biological p… molecular functio… YML017W G0.05 -0.69  
## 7 RIB2 riboflavin b… pseudouridylate s… YOL066C G0.05 -0.55  
## 8 VMA13 vacuolar aci… hydrogen-transpor… YPR036W G0.05 -0.75  
## 9 EDC3 deadenylylat… molecular functio… YEL015W G0.05 -0.24  
## 10 VPS5 protein rete… protein transport… YOR069W G0.05 -0.16  
## # … with 199,322 more rows

We still have one last problem. The column sample actually contains two variables, so we need to separate it after the first character (the letter for the nutrient) and the number. We’ll use separate again, but indicate column 1, and to convert any numeric values into a number. We use the stringr library for this extra feature:

# Separate NAME, cleanup whitespace, drop columns, gather and separate nutrient from rate  
library(stringr)  
  
cleaned\_data <- original\_data %>%  
 separate(NAME, c("name", "BP", "MF", "systematic\_name", "number"), sep = "\\|\\|") %>%  
 mutate\_each(funs(trimws), name:systematic\_name) %>%  
 select(-number, -GID, -YORF, -GWEIGHT) %>%  
 gather(sample, expression, G0.05:U0.3) %>%  
 separate(sample, c("nutrient", "rate"), sep = 1, convert = TRUE)

## 4. Visualise with ggplot

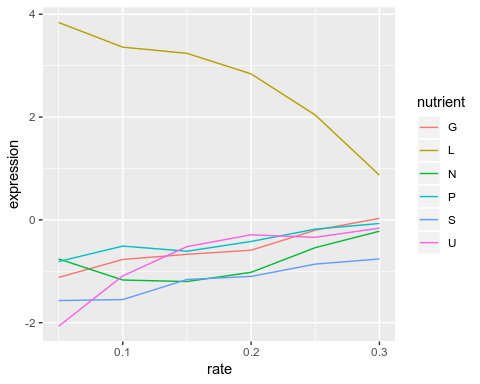
The great advantage of getting your data into tidy format is that it now makes it easier to subset and visualise. Suppose you want to look at the leucine sysntesis pathway (LEU1), it is easy to filter out those values:

# Leucine synthesis  
cleaned\_data %>%  
 filter(name == "LEU1")

## # A tibble: 36 x 7  
## name BP MF systematic\_name nutrient rate expression  
## <chr> <chr> <chr> <chr> <chr> <dbl> <dbl>  
## 1 LEU1 leucine b… 3-isopropylm… YGL009C G 0.05 -1.12  
## 2 LEU1 leucine b… 3-isopropylm… YGL009C G 0.1 -0.77  
## 3 LEU1 leucine b… 3-isopropylm… YGL009C G 0.15 -0.67  
## 4 LEU1 leucine b… 3-isopropylm… YGL009C G 0.2 -0.59  
## 5 LEU1 leucine b… 3-isopropylm… YGL009C G 0.25 -0.2   
## 6 LEU1 leucine b… 3-isopropylm… YGL009C G 0.3 0.03  
## 7 LEU1 leucine b… 3-isopropylm… YGL009C N 0.05 -0.76  
## 8 LEU1 leucine b… 3-isopropylm… YGL009C N 0.1 -1.17  
## 9 LEU1 leucine b… 3-isopropylm… YGL009C N 0.15 -1.2   
## 10 LEU1 leucine b… 3-isopropylm… YGL009C N 0.2 -1.02  
## # … with 26 more rows

The nice thing about the tidyverse is you can pipe the output from above straight into ggplot. **Note** remember to switch from %>% to + when you go to ggplot:

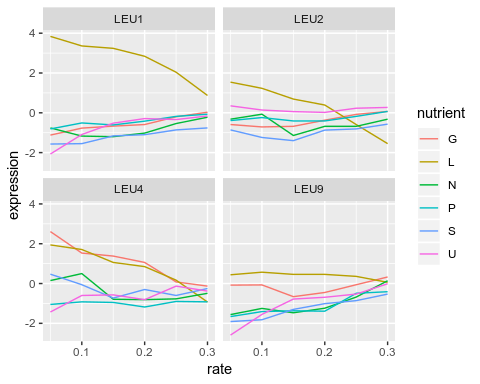
library(ggplot2)  
  
# ggplot of leucine  
cleaned\_data %>%  
 filter(name == "LEU1") %>%  
 ggplot(aes(rate, expression, color = nutrient)) +  
 geom\_line()



I’m not a geneticist, but even I can spot that this gene is switched on when the cells are starved of leucine. The cell has to synthsise its own leucine; as the amount of leucine available (rate) increases, the gene expression decreases.

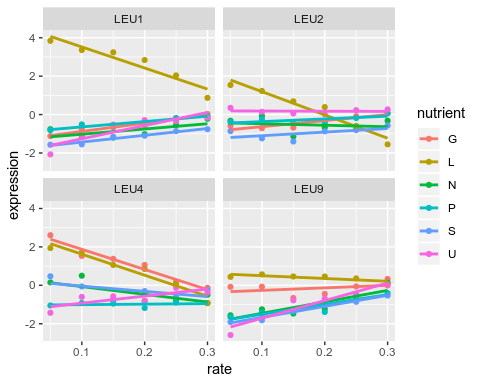
There is actually more than one gene in the leucine biosynthesis process (BP column) so let’s look at all of them:

# All the leucine BP  
cleaned\_data %>%  
 filter(BP == "leucine biosynthesis") %>%  
 ggplot(aes(rate, expression, color = nutrient)) +  
 geom\_line() +  
 facet\_wrap(~name)



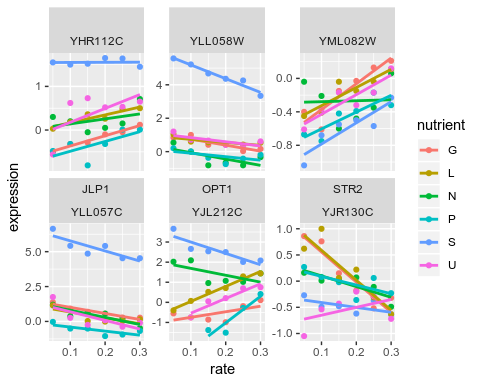
LEU1, 2 and 4 seem to respond most. Perhaps it would be clearer if we fit a linear model and add to our plots:

# All the leucine BP plus linear model  
cleaned\_data %>%  
 filter(BP == "leucine biosynthesis") %>%  
 ggplot(aes(rate, expression, color = nutrient)) +  
 geom\_point() +  
 geom\_smooth(method = "lm", se = FALSE) +  
 facet\_wrap(~name)



We can do the same for sulphur metabolism, although as not all genes in this process have traditional names, we also have to facet the plot by the systematic\_name code:

# Sulphur metabolism  
cleaned\_data %>%  
 filter(BP == "sulfur metabolism") %>%  
 ggplot(aes(rate, expression, color = nutrient)) +  
 geom\_point() +  
 geom\_smooth(method = "lm", se = FALSE) +  
 facet\_wrap(~name + systematic\_name, scales = "free\_y")



**Comments** : hopefully this shows you how useful the tidy format is. In effect we went from the messy raw data to valuable plots in 12 lines of code, and you wouldn’t have been able to produce the gene-expression plots from the raw data:

# Core code  
library(dplyr)  
library(tidyr)  
library(ggplot2)  
  
cleaned\_data <- original\_data %>%  
 separate(NAME, c("name", "BP", "MF", "systematic\_name", "number"), sep = "\\|\\|") %>%  
 mutate\_each(funs(trimws), name:systematic\_name) %>%  
 select(-number, -GID, -YORF, -GWEIGHT) %>%  
 gather(sample, expression, G0.05:U0.3) %>%  
 separate(sample, c("nutrient", "rate"), sep = 1, convert = TRUE)  
  
cleaned\_data %>%  
 filter(BP == "leucine biosynthesis") %>%  
 ggplot(aes(rate, expression, color = nutrient)) +  
 geom\_point() +  
 geom\_smooth(method = "lm", se = FALSE) +  
 facet\_wrap(~name)

This might be a genetics example, but could just as easily be applied anywhere you have messy data.