Demo yeast mutant analysis

Jean-Yves Sgro

February 20, 2018

Contents

1 Analysis of yeast growth data

Based on webinar by Dr. Jeremy Chacon Beginners Introduction to R Statistical Software[1]

The mock yeast experiment table used in the webinar (`yeast_example.txt`) can be obtained from this short link: https://go.wisc.edu/mc5d52

1.1 Set working directory

It is always good practice to keep projects wihtin a separate directory.

Change directory to the one on the desktop with `setwd()` and verify with `getwd()`. This commands assumes that the directory exists already. Create it on your computer first if necessary, and download the `yeast_example.txt` (see above) within it.

```
setwd("~/Desktop/R_intro_2018/Yeast_demo")
getwd()
```

> *Note*: On a Windows computer it would be something like this: `C:/Users/etc/etc/etc/` (using the forward slash `/`)

[1] https://bitesizebio.com/webinar/beginners-introduction-to-r-statistical-software/

1.2 List all files in directory

```
list.files()
```

```
 [1] "Demo_yeast_cache"          "Demo_yeast_files"
 [3] "Demo_yeast.html"           "demo_yeast.R"
 [5] "Demo_yeast.Rmd"            "growth_rate.pdf"
 [7] "RStudio_yeast_demo.Rproj" "yeast_example.md"
 [9] "yeast_example.txt"         "yeast_example.xlsx"
```

Note: the command `dir()` would give the same result.

```
dir()
```

1.3 List "txt" files and read data

List `*.txt` files wihtin the directory with either `list.files()` or `dir()` specifying the pattern searched:

```
dir(pattern = ".txt")
```

```
[1] "yeast_example.txt"
```

Read data, specifying that the first line is a header, into variable named `yeast_eg`

```
yeast_eg = read.table('yeast_example.txt', header=T)
```

2 Examine data

The first 6 lines of the data look like this:

```
head(yeast_eg)
```

```
  genotype       drug     treatment OD_change
1       WT       none    WT_no_drug       3.2
2       WT       none    WT_no_drug       2.8
3       WT       none    WT_no_drug       3.1
4       WT       none    WT_no_drug       3.3
5       WT       none    WT_no_drug       2.6
6       WT nocodazole WT_nocodazole       1.2
```

During an interactive session the following command will open a spreadsheet-like tab or window showing all the data in tabular format.

```
View(yeast_eg)
```

The structure and summary of the data look like this:

```
str(yeast_eg)
```

```
'data.frame':   20 obs. of  4 variables:
 $ genotype : Factor w/ 2 levels "mad2_del","WT": 2 2 2 2 2 2 2 2 2 2 ...
 $ drug     : Factor w/ 2 levels "nocodazole","none": 2 2 2 2 2 1 1 1 1 1 ...
 $ treatment: Factor w/ 4 levels "mad2_del_no_drug",..: 3 3 3 3 3 4 4 4 4 4 ...
 $ OD_change: num  3.2 2.8 3.1 3.3 2.6 1.2 1.5 1.3 1.9 0.7 ...
```

```
summary(yeast_eg)
```

```
    genotype          drug                   treatment   OD_change
 mad2_del:10   nocodazole:10   mad2_del_no_drug   :5   Min.   :0.700
 WT      :10   none      :10   mad2_del_nocodazole:5   1st Qu.:2.125
                               WT_no_drug         :5   Median :2.650
                               WT_nocodazole      :5   Mean   :2.425
                                                       3rd Qu.:2.925
                                                       Max.   :3.300
```

Optionally we can alaos create a nice looking table with some added command (that may require loading additional `R` pacakges, so it it does not work now that's OK.) Here is the complete dataset wihtin the table:

```
library(knitr)
kable(yeast_eg)
```

genotype	drug	treatment	OD_change
WT	none	WT_no_drug	3.2
WT	none	WT_no_drug	2.8
WT	none	WT_no_drug	3.1
WT	none	WT_no_drug	3.3
WT	none	WT_no_drug	2.6
WT	nocodazole	WT_nocodazole	1.2
WT	nocodazole	WT_nocodazole	1.5
WT	nocodazole	WT_nocodazole	1.3
WT	nocodazole	WT_nocodazole	1.9
WT	nocodazole	WT_nocodazole	0.7
mad2_del	none	mad2_del_no_drug	2.7
mad2_del	none	mad2_del_no_drug	2.9
mad2_del	none	mad2_del_no_drug	3.0
mad2_del	none	mad2_del_no_drug	2.5
mad2_del	none	mad2_del_no_drug	3.1
mad2_del	nocodazole	mad2_del_nocodazole	2.2
mad2_del	nocodazole	mad2_del_nocodazole	2.4
mad2_del	nocodazole	mad2_del_nocodazole	2.9
mad2_del	nocodazole	mad2_del_nocodazole	2.5
mad2_del	nocodazole	mad2_del_nocodazole	2.7

3 Data exploration

3.1 Accessing column data

Accessing specific columns in the data table can be done in 2 ways:

- Using the `$` sign between the name of the dataset and the name of the column. For example: `yeast_eg$genotype`
- The `with()` function allows a more elegant writing. The first argument is the dataset, here `yeast_eg`. The second command will be typically be a function into which is specified the name of the column to use. For example: `with(yeast_eg,summary(genotype))`.

```
with(yeast_eg,summary(genotype))
```

```
mad2_del       WT
      10       10
```

3.1.1 Exploratory plots

The following comman with plot the **genotype** on the horizontal `x` axis and the **OD change** on the vertical `y` axis:

```
with(yeast_eg, plot(genotype, OD_change))
```

Note: Using the `$` nomenclature would create the exact same plot: `plot(yeast_eg$genotype, yeast_eg$OD_change)`.

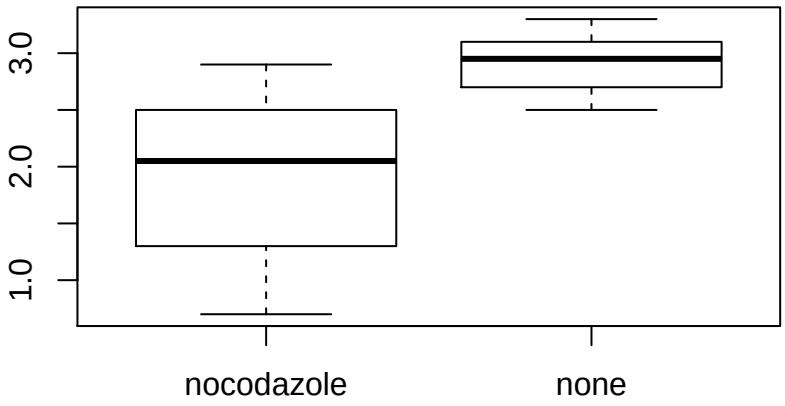
We can observe that the OD change is higher, on average for `mad2_del` as indicated by the thick line within the box representing the `median`.

Thus for now it appear that the growth rate is greater in `mad2_del` even when we add the drug *nocodazole* which should sop the cells from growing.

But to confirm this hypothesis we need to look at the data a few different more ways.

We can now look at the effect of the **drug** on the **OD change**.

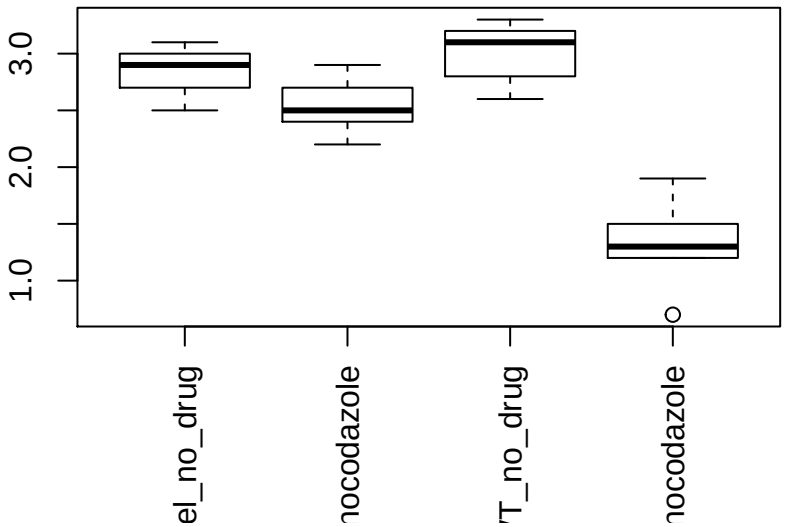
```
with(yeast_eg, plot(drug, OD_change))
```



Now we see on the plot that the growth rate is lower in the presence of the drug *nocodazole* compared to no drug added. This suggests that treatment worked as expected as the drug on average should stop growth rate.

The *treatment* variable, the 3rd column of data, summarizes all 4 treatments into one variable (genotype and presence of drug). This is making plotting easier to see all treatments on the same plot.

```
with(yeast_eg, plot(treatment, OD_change, las=3))
```



Note: the rotating command (`las`) can alter the style of axis labels: 0=parallel, 1=all horizontal, 2=all perpendicular to axis, 3=all vertical. If labels are horizontal and not showing, simply extend the size of the display with the mouse.

The plot indicates that:

- `WT` cells *without* drug grow very fast.
- `WT` cells *with* drug grow much more slowly, as expected.
- mutant cells grow faster in *absence* of drug
- mutant cells grow almost as fast in *presence* of drug.

Therefore for mutants cells this suggest that the drug is not causing the celle cycle to stop as we expect for the `WT` cells further suggesting that gene `mad2` may be important for cell cycle rest.

BUT can we trust our eyes with this plot?
Are the differences we see real?
The next step is to perform some statistics.

4 Stastistical test: two-way ANOVA

4.1 Calculate ANOVA

We can do a 2-way ANOVA because we have a *continuous response* variable (`OD_change`) and 2 factors (`genotype` and `drug treatment`) and we want to look at the *interaction* between those 2 factors.

In summary **we want to know if the effect of drug differs depending on the genotype**.

`R` was created as a statistical language and therefore it is easy to calculate an ANOVA with the `aov()` function. However, it is necessary to know how to tell `R` the *model formulation* for the experiment.

```
response ~ predictor1 + predictor2
```

- `~` : separates response from predictor
- `+` : adds more predictors

The formula with `+` would give answer for main effect.

To specify interaction use `*` to specify all possible interactions and main effects:

```
response ~ predictor1 * predictor2
```

For our experiment we would want the *main effects* and also the *interactions* between the predictors.

In our case we would write the formula with the `response` as `OD_change` separated by tilde `~` and then `genotype` as the first main effect, then asterisk `*` and then `drug` which is the second main effect.

By using the asterisk `*` the ANOVA results will include both main effects and the interactions between the 2 factors.

The ANOVA result is stored in variable `m1` specifying *interactions* between `genotype` and `drug` written in the format using the `with()` function:

```
m1 <- with(yeast_eg, aov(OD_change ~ genotype * drug))
```

We can look at the ANOVA results:

```
m1
```

```
Call:
   aov(formula = OD_change ~ genotype * drug)

Terms:
                genotype   drug genotype:drug Residuals
Sum of Squares    1.4045 4.9005        2.3805    1.6320
Deg. of Freedom        1      1             1        16

Residual standard error: 0.3193744
Estimated effects may be unbalanced
```

However the ANOVA table itself has more information. It is obtained with the `summary()` function:

Show ANOVA summary:

```
summary(m1)
```

```
              Df Sum Sq Mean Sq F value   Pr(>F)
genotype       1  1.405   1.405   13.77 0.001899 **
drug           1  4.901   4.901   48.04 3.38e-06 ***
genotype:drug  1  2.380   2.380   23.34 0.000184 ***
Residuals     16  1.632   0.102
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The table shows the degrees of freedom (`Df`,) the sum of squares (`Sum Sq`,) the mean square (`Mean Sq`,) the *F-statistic* (`F value`,) and the *p-value* (`Pr>(F)`). At theb bottom the `Signif. codes` provides a level of statistical significance for the results highlighted in the table next to the *p-value*.

The *p-value* indicates that the `genotype` has a significant effect on growth rate, as expected, since we observed that on average the `WT` grows slower than the mutant because it got arrested in its growth with drug.

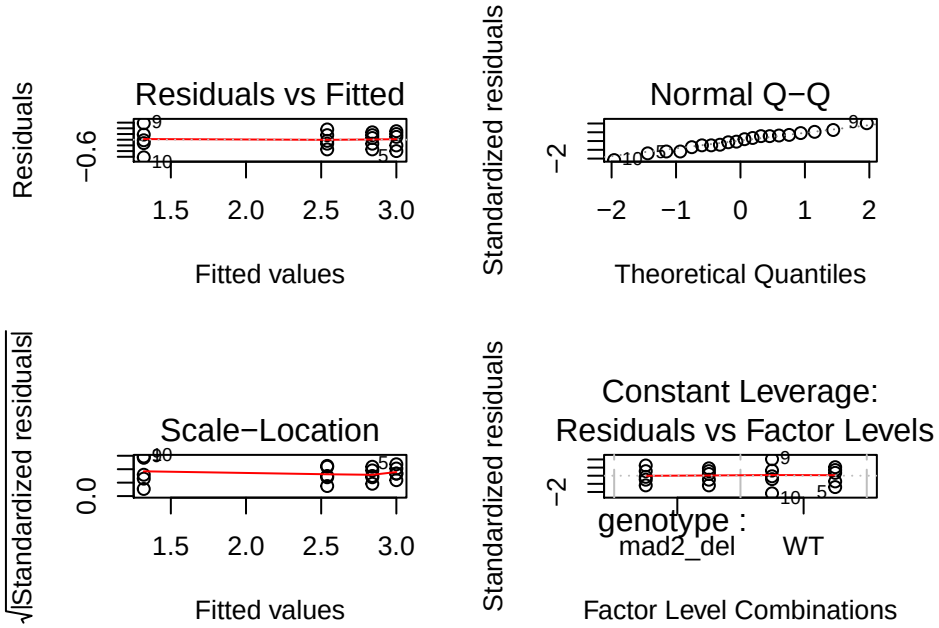
We also see a significant of the `drug`, as indeed the `drug` on average decreases the growth rate of the cells.

Importantly, we see a *significant interaction* between the`genotype` and the `drug`. While `WT` cells stopped growing when we added the drug, the mutant `mad2_del` cells did not stop growing with the drug added.

4.2 Plot ANOVA

It is important to check the *model assumptions*. This can be done visually with the plot command on the ANOVA model.

```
par(mfrow=c(2,2))
plot(m1)
```




```
par(mfrow=c(1,1))
```

The parameter function `par()` instructs how to split the plot area in rows and columns, and how many of which. Here we'll split in 4 quadrants: 2 rows and 2 columns. After the plot the area is reset to a single spot.

5 Publication quality plots

5.1 Installation of ggplot

The package `ggplot` is a modern package that makes beautiful plots. *gg* stands for "*grammar of graphics*."

It may be necessary to install the package on your computer. The following command will do that, or it can also be done within RStudio with a graphical interface.

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

Note: installation other, dependent packages is likely to occur as well.

5.2 Plot with ggplot

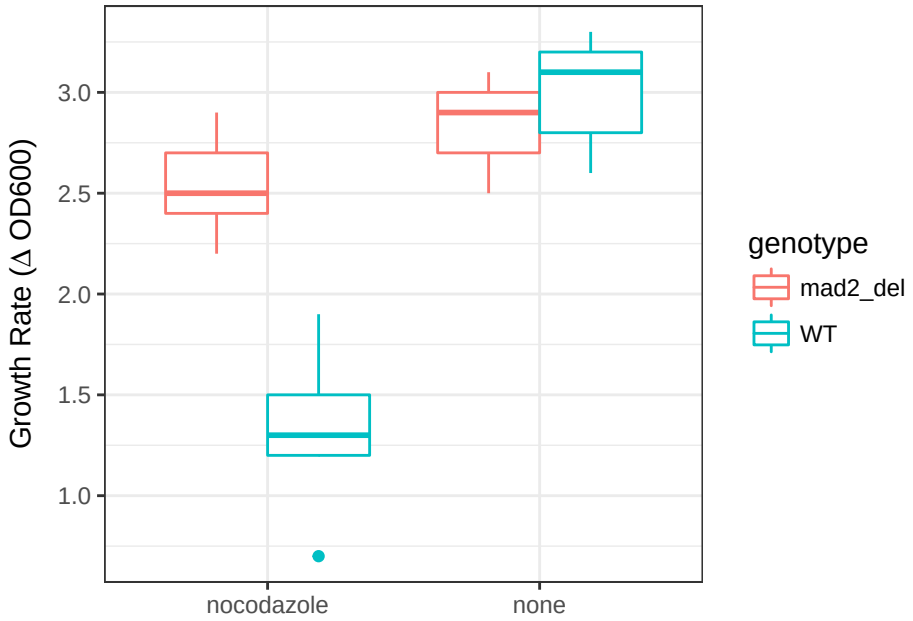
The `library()` command will load the requested package and others dependencies.

```
library(ggplot2) # laad the package for this session.
```

On the original video, the plot command was shorter for a less fancy plot:

```
ggplot(yeast_eg, aes(x =  drug, y =  OD_change, color = genotype))+
  geom_boxplot()+
  theme_bw()+
```

```
  scale_x_discrete('')+
  scale_y_continuous(expression(paste('Growth Rate (', Delta, ' OD600)')))
```

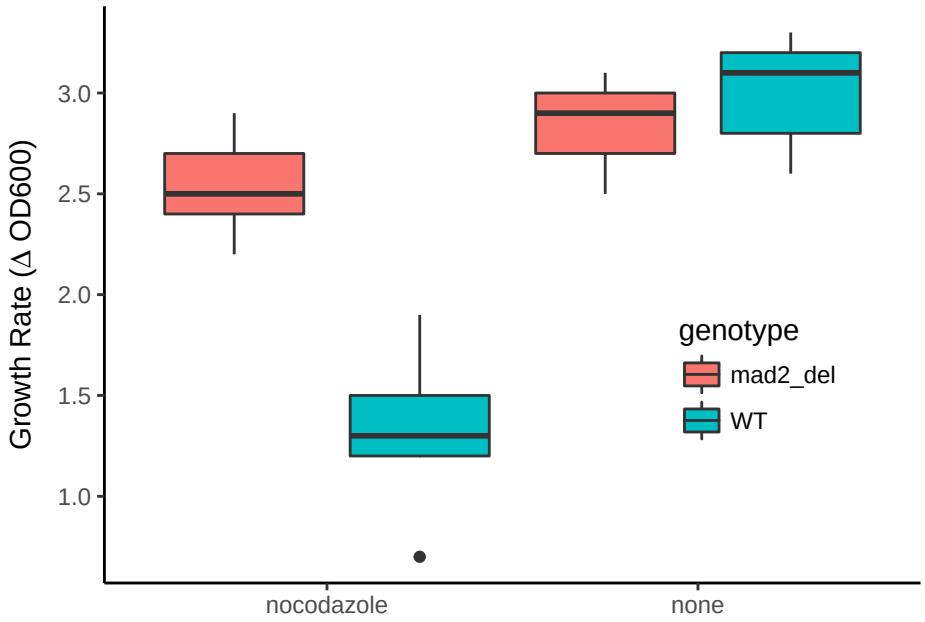



```
# +opts(legend.position=c(0.8,0.35)) # opts is no longer supported
```

However the `opts` option is no longer supported and an error could occur if run. The only difference is that the legend is outside the plot rather than within the plotting area.

The following code provides a colored fill within the box and separates the boxes so that they do not touch. Some of the background grids are also removed.

```
ggplot(yeast_eg, aes(x =  drug, y =  OD_change, fill = genotype))+
  geom_boxplot(position=position_dodge(1))+
  theme_bw()+
  scale_x_discrete('')+
  scale_y_continuous(expression(paste('Growth Rate (', Delta, ' OD600)')))+
  theme(legend.position=c(0.8,0.35), panel.border = element_blank(), panel.grid.major = element_blank()
panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))
```



Note: in RStudio the plot will appear within the bottom right quadrant by default. To make the plot appear in a ***separate*** window, the following commands cane be used depending on the type of computer used. In additon, specific dimensions can be specified in inches.

Computer type /OS	Command
Windows	`windows`
Macintosh	`quartz()`
Linux	`x11`

In the video the command is `windows(height=3, width-3*1.3)`

6 Saving a plot in PDF

While there are different ways to saving to PDF, artifacts of errors can occur. The `Cairo` package is a specific package for creating PDF reproducibly.

6.1 Installation of Cairo

If you need to install this package you can run the following command or use the RStudio graphical interface:

```
install.packages('Cairo')
```

Load the package with:

```
library(Cairo)
```

Initialize the PDF file. Specified sizes are in inches by default.

```
CairoPDF(file='growth_rate.pdf', width=3*1.3, height=3)
```

Now `CairoPDF` is waiting for the plot to be given.

Issue the `ggplot` above command again.

Note that no plot will be visible at that time.

It is then necessary to "close" the PDF file called a "device" hence the name of the function used.

```
dev.off()
```

7 Conclusions

- This was a complete data analysis in very few lines of code
- The `R` code is *re-usable*
- The `R` code is *reproducible*
- Essential points to remember:
- write code in a *script*
- run lines by highlighting and the use `control`+`enter` to run it
- Get help on functions with `?` or `??`
- Web search engines can provide many solutions to questions