

R Small Group: Class 3

Amy Allen & Dayne Filer

June 28, 2016

Using this document

- Code blocks and R code have a grey background (note, code nested in the text is not highlighted in the pdf version of this document but is a different font).
- `#` indicates a comment, and anything after a comment will not be evaluated in R
- The comments beginning with `##` under the code in the grey code boxes are the output from the code directly above; any comments added by us will start with a single `#`
- While you can copy and paste code into R, you will learn faster if you type out the commands yourself.
- Read through the document after class. This is meant to be a reference, and ideally, you should be able to understand every line of code. If there is something you do not understand please email us with questions or ask in the following class (you're probably not the only one with the same question!).

Class 3 expectations

1. Know how to import data from a csv file format
2. Understand all of the arguments of the `plot` function
3. Be able to make basic plots

Importing data from a csv file

A csv is a comma separated values file, which allows data to be saved in a table structured format. Excel files can often be saved as csv files. R has a function to read these files called `read.csv`. We will use this function to read in data from a csv file.

You have been provided with the data set “yeastmutants.csv”. This data set includes data on the response to mating pheromone over time for following three cell types: wildtype (wt), an sst2 mutant (sst2), and a gpa1 mutant (gpa1). For each mutant there is data on the mean response of a population (avg) as well as the standard deviation of the response (stdev). The data set also includes the time for each data point. Let's import it into R and store it as `data`. First we need to set our working directory to the location where the file is stored. We do this by using the `setwd` function. First locate the file under the files tab, then hover over the file name and it should give you the path to that directory; for me it is “~/Documents/rclass”. Make sure that the path name is in quotes so R knows it is a character data type and execute the function as follows:

```
setwd("~/Documents/rclass")
getwd()
## [1] "/Users/AmyAllen/Documents/rclass"
```

You can use the `getwd()` function to check that you have successfully changed directories. Now that we are in the correct working directory we can load the data using the `read.csv` function. Here the argument `header` is set to `TRUE` because the data set has a header labeling each column.

```
data <- read.csv("yeastmutants.csv", header = TRUE)
```

Now we can take a look at our data.

```

class(data)
## [1] "data.frame"
data
##      Time sst2.avg sst2.stdev  wt.avg  wt.stdev gpa1.avg gpa1.stdev
## 1      0 1.000000 0.00000000 1.000000 0.00000000 1.000000 0.00000000
## 2      5 1.136837 0.09389147 1.072776 0.09115671 1.057325 0.08164994
## 3     10 1.287260 0.17363721 1.203336 0.14969849 1.168357 0.14578347
## 4     15 1.468307 0.28511297 1.355571 0.23767208 1.295126 0.23039342
## 5     20 1.743784 0.37276442 1.502573 0.32856109 1.431064 0.32072659
## 6     25 1.969451 0.48747293 1.681677 0.42366003 1.614628 0.40513027
## 7     30 2.238040 0.61199152 1.891336 0.51515710 1.759505 0.49942606
## 8     35 2.562568 0.72005169 2.093010 0.60034859 1.891254 0.56851048
## 9     40 2.882724 0.84589976 2.259807 0.68484877 2.105130 0.68865793
## 10    45 3.173244 0.92524386 2.475061 0.76737933 2.268349 0.79249475
## 11    50 3.492047 1.03084217 2.662022 0.83065122 2.468349 0.89058264
## 12    55 3.777006 1.11674734 2.895606 0.90892127 2.635299 0.98186805
## 13    60 4.085648 1.20400677 3.101005 0.98064725 2.784591 1.03479761
## 14    65 4.371034 1.28763041 3.285258 1.02267586 2.946033 1.09631276
## 15    70 4.665376 1.37728041 3.455390 1.05638858 3.113485 1.17051632
## 16    75 4.850383 1.46856287 3.652236 1.10244107 3.314269 1.25634935
## 17    80 5.073138 1.51458449 3.839015 1.13939773 3.466327 1.31108736
str(data)
## 'data.frame':    17 obs. of  7 variables:
##  $ Time      : int  0 5 10 15 20 25 30 35 40 45 ...
##  $ sst2.avg   : num  1 1.14 1.29 1.47 1.74 ...
##  $ sst2.stdev : num  0 0.0939 0.1736 0.2851 0.3728 ...
##  $ wt.avg     : num  1 1.07 1.2 1.36 1.5 ...
##  $ wt.stdev   : num  0 0.0912 0.1497 0.2377 0.3286 ...
##  $ gpa1.avg   : num  1 1.06 1.17 1.3 1.43 ...
##  $ gpa1.stdev : num  0 0.0816 0.1458 0.2304 0.3207 ...

```

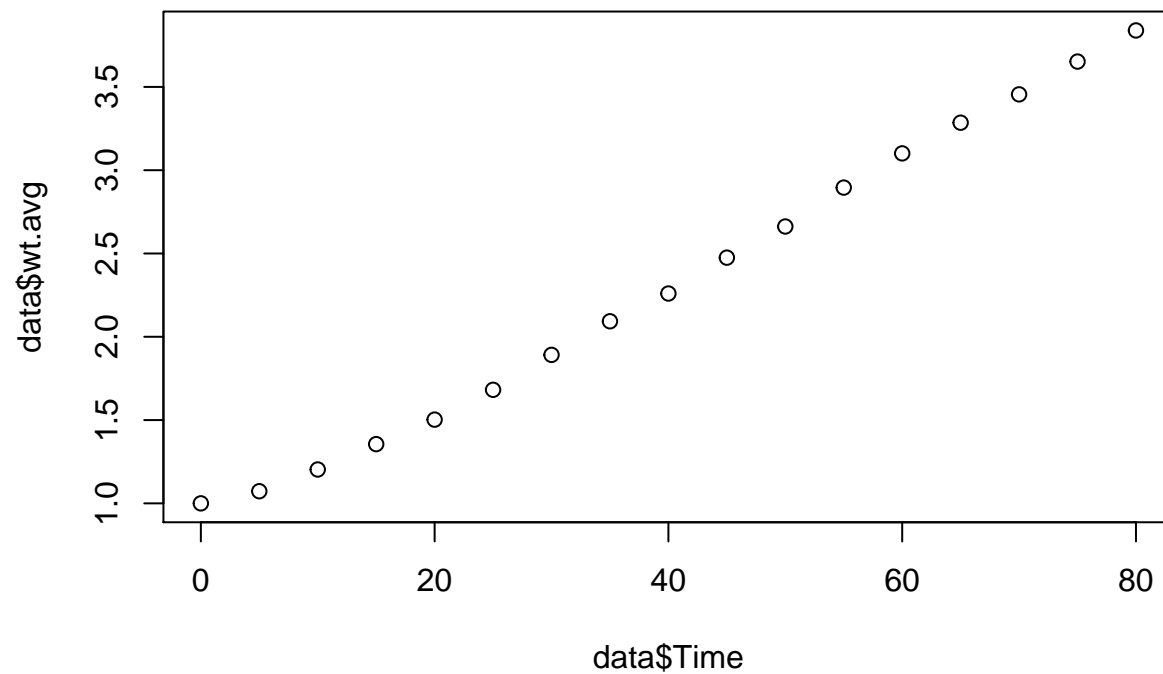
Using the `class()` function we can tell that `data` is a dataframe. This means it is a list of vectors each with the same length. Now, let's look closer at the data itself; we see it looks like a table with 7 columns (each with a header) and 17 rows. We can also look at the structure of this data frame using the `str()` function. It tells us that we have 17 observations of 7 variables and lists the variables and their classes. `Time` is an integer and the rest of the variables are numeric.

Basic plotting

Now we have imported some data, so how can we visualize it? One of the most basic ways to visualize data is the `plot()` function. We can learn more about this function using `?plot`. The `plot` function is used for generic X-Y plotting. It requires two arguments `x` and `y`, which represent the x and y coordinates for the plot respectively. Then the function also has lots of optional arguments. `type` is used to indicate what type of plot should be drawn; all of the options for this are listed in the help menu. `main` and `sub` are used to give the plot an overall and a subtitle, respectively. `xlab` and `ylab` are used to label the x and y axes. Finally, `asp` is used to assign the y/x aspect ratio. There are other arguments that can be passed to `plot()`. You can google these if you're interested or use `?par`; one we will be particularly interested in is `col` which assigns a color.

Now let's do a basic plotting example. We want to plot the average response of wildtype cells versus time. We will need to subset the average response of `wt` and `time` from the data frame and plot them. We can do this using the `$` operator.

```
plot(x = data$Time, y = data$wt.avg)
```



We also want to give the plot a title and axis labels.

```
plot(x = data$Time, y = data$wt.avg, main = "Yeast Response to Pheromone",  
     xlab = "Time (minutes)", ylab = "Mean pheromone response")
```

Yeast Response to Pheromone

