Graphing with ggplot

George Savva

23/07/2020

In this demonstration we’ll create some plots using ggplot2 to do some descriptive analysis.

We will use the data found in the ‘combined’ sheet of the walking speed Excel file. First we’ll load the data then check it looks OK.

library(ggplot2)  
library(ggpubr)

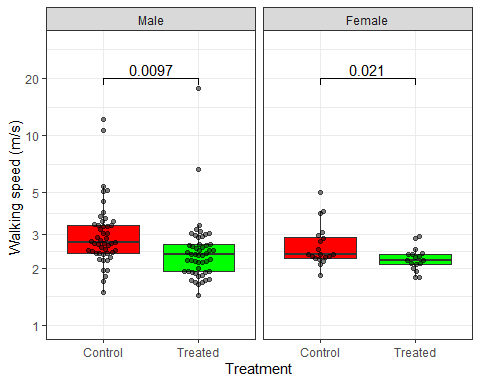
## Loading required package: magrittr

library(ggbeeswarm)  
library(readxl)

alldata <- read\_excel("walkingspeed.xlsx", sheet="combined", range="A1:F139")  
head(alldata)

## # A tibble: 6 x 6  
## patid time sex age department treated  
## <dbl> <dbl> <chr> <dbl> <dbl> <chr>   
## 1 1 1.90 M 53 3 treated  
## 2 2 3.54 M 61 3 control  
## 3 3 2.93 M 65 1 treated  
## 4 4 1.82 M 48 2 control  
## 5 5 2.20 M 62 2 treated  
## 6 6 3.04 M 62 4 control

Our aim is to make a good visualisation of the difference in walking speed between control patients and treated patients, stratified by sex. Something like this:

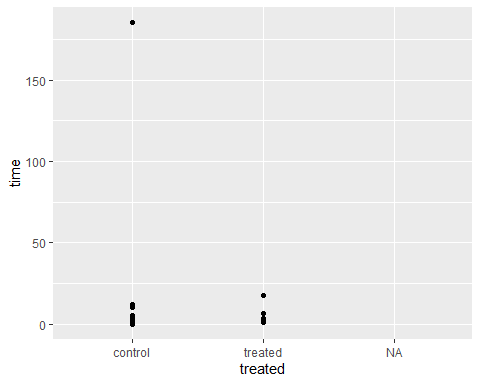


## Getting started

First we’ll need to make a simple scatter plot. The minimum we can do to specify a plot is to supply (1) a dataset (2) an aesthetic mapping from that dataset to plot coordinates (3) a geom to display. We’ll do this as below. Note the use of the ‘+’ operator to add components together. This is how we will add all of the layers of our graph:

ggplot(alldata, aes(x=treated, y=time)) + geom\_point()

## Warning: Removed 2 rows containing missing values (geom\_point).



## Dropping the ‘missing’ category

This is a good start but we have some way to go

First, note the warning that the rows with missing y values (times) have been removed. Note also that the rows with missing x values have not been removed. Discrete scales can easily incorporate missing values, and so they do so by default.

Suppose we want not to show the missing values. There are lots of ways we could do this, but in short we can either manipulate our dataset to remove the missing values, or we can ask ggplot not to show them.

First we’ll look at changing the data. ggplot2 comes with the ‘remove\_missing’ function, that takes away any row with any missing data.

ggplot(remove\_missing(alldata), aes(x=treated, y=time)) + geom\_point()

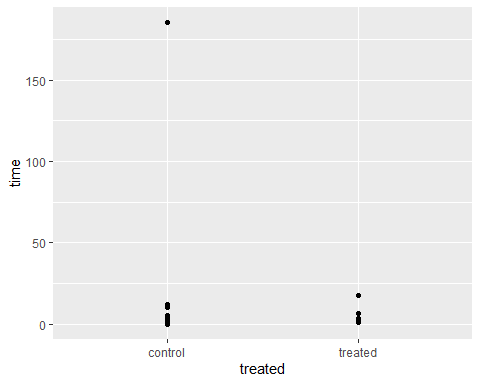
Or we could run ‘remove\_missing’ first to create a new dataset without these rows, then pass that to ggplot.

alldata2 <- remove\_missing(alldata)

## Warning: Removed 2 rows containing missing values.

Another method is to use the base command ‘subset’. Subset returns the rows of a dataset that meet the condition specified in it’s second argument. Here we will ask for the rows of ‘alldata’ where ‘treated’ is not NA, and where time is not NA.

alldata <- subset(alldata, !is.na(treated))  
ggplot(alldata, aes(x=treated, y=time)) + geom\_point()



Finally the best way is to request that ggplot ignores missing values in the x axis. To do this we can use the ‘scale\_x\_discrete’ function. This is the function that we use to specify how the x axis should be organised, that is exactly how are the values in our data to be mapped onto the axis. Look at the help for scale\_x\_discrete, can you see how to stop the missing category from appearing?

##Exercise:

1. Make a scatter graph showing the relationship between age and walking speed,
2. Make the the shape of the points corresponding to the different genders
3. Make the the colour of the points corresponding to the different treatments

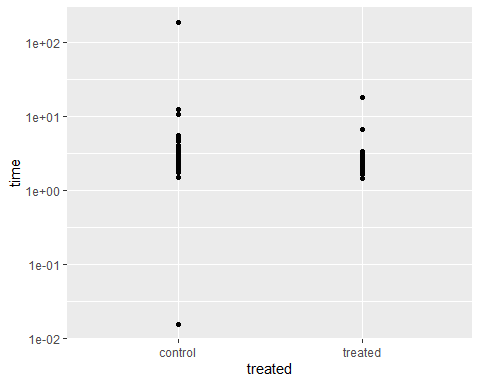
## Removing outliers

The plot above suggests an outlier. We should use our knowledge of the experiment and our research question to decide what do with this point.

Handling outliers always requires some thought. It is often the case that what seems like an outlier on one scale might not look like an outlier on another, and might actually be an data point. Before identifying and removing erroneous outliers it is usually wise to change your scale first, to see if the outlying point makes sense under a different model.

Switching to a log-scale for the y-axis we now see a second possible problem observation:

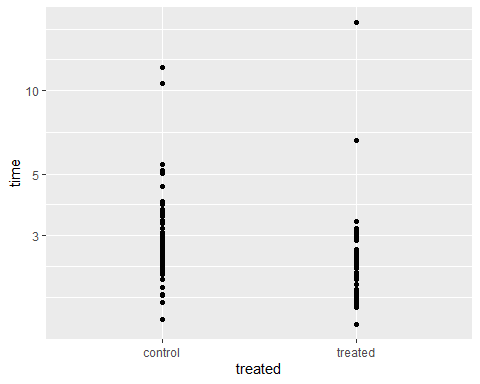
ggplot(alldata, aes(x=treated, y=time)) + geom\_point() + scale\_y\_log10()



Our experiment is a walking time experiment over a very short distance. It is not feasible that anybody completed this task in less than a second, or took more than a couple of minutes (or is it?), so we can safely remove these observations from our graph.

# This code sets the outlying points to 'NA' (missing)  
# This is fine if you're sure you want the change to affect all subsequent analysis  
# But if you only want to affect the graph but otherwise leave the data the way it was,  
# then use a 'subset' within the ggplot call or use 'y\_scale\_continuous' to set the limits.  
alldata$time[alldata$time>100] <- NA  
alldata$time[alldata$time<0.1] <- NA  
ggplot(remove\_missing(alldata), aes(x=treated, y=time)) + geom\_point() + scale\_y\_log10()

## Warning: Removed 2 rows containing missing values.

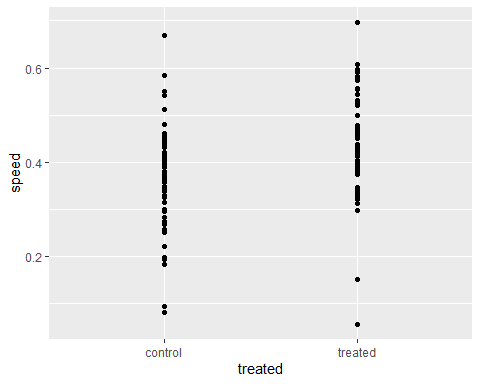


Even on a log scale some of the data points look unusual. Another possible transformation is the inverse transformation. This would turn our ‘time’ variable into a ‘speed’ variable.

We can make a new column in our dataframe, then plot this instead:

alldata$speed <- 1 / alldata$time  
ggplot(remove\_missing(alldata), aes(x=treated, y=speed)) + geom\_point()

## Warning: Removed 2 rows containing missing values.



This does look better from some points of view. For a regression model I think I would prefer the ‘speed’ variable as opposed to a log-transformed ‘time’ variable, because it makes more intuitive sense to me and it seems more ‘normally’ distributed than the time variable. Both graphs are OK

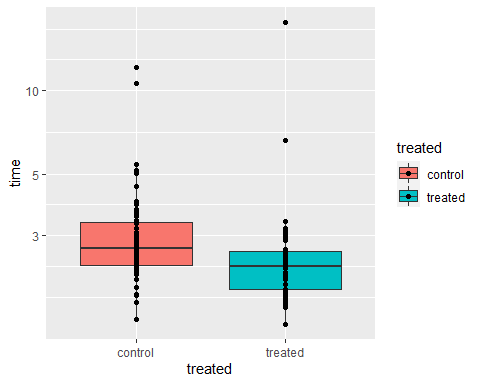
## Adding a second geom

We might be tempted to replace the points with a mean and standard error for each group. This is very common in biology journals but it is usually a very bad idea. A much better way to provide summary statistics for large datasets is to use a boxplot. To add a boxplot we just add a component with the geom\_box() function.

ggplot(alldata, aes(x=treated, y=time, fill=treated)) +  
 geom\_boxplot() +  
 geom\_point() +   
 scale\_y\_log10()

## Warning: Removed 2 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 2 rows containing missing values (geom\_point).



Things to note here: First, I’ve added the geom\_box() before the geom\_point() because I want the points on top of the box. I have added a new aesthetic mapping, the ‘fill’ aesthetic is now mapped to the ‘treated’ variable in my data. Finally I’ve broken the command over several lines to make it easier to read. Remember when R gets to the end of the line but a command isn’t finished it will continue reading.

## Labelling the scales

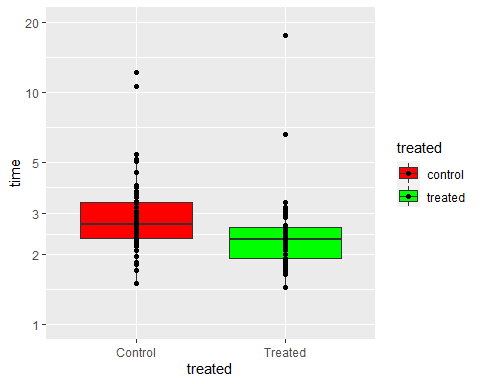
We have already seen two uses of the scale functions, to change the y axis to a log-scale and to remove the missing values from the x axis. ‘scales’ in ggplot2 control the way the values in our data are mapped to specific values in our aesthetic, and so we will also use them for changing the axis ticks and labels, and setting specific colours in our plots. We can use one scale function for each aesthetic in our graph:

We will use: scale\_fill\_manual() to set specific colours scale\_x\_discrete() to arrange and relabel the x-axis scale\_y\_continuous() to set specific tick marks and limits on the y-axis

g <- ggplot(alldata, aes(x=treated, y=time, fill=treated)) +  
 geom\_boxplot() +  
 geom\_point() +  
 scale\_fill\_manual(values=c(`treated`="green",`control`="red")) +  
 scale\_y\_log10(breaks=c(1,2,3,5,10,20),   
 limits=c(1,20)) +   
 scale\_x\_discrete(na.translate=FALSE,   
 labels=c(`treated`="Treated",`control`="Control"))  
g

## Warning: Removed 2 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 2 rows containing missing values (geom\_point).



Points to note: I have used named vectors to get the colours and to relabel the x-axis. I could have used unnamed vectors and relied on the order of the elements, but this risks making a mistake. As well as entering each component on a new line, I also split some of the individual components over lines. This helps to keep the function readable. I have assigned the value of the ggplot function to an object ‘g’, then called ‘g’ to print the graph. This is helpful later because now I can add extra components easily to the graph as in the next section:

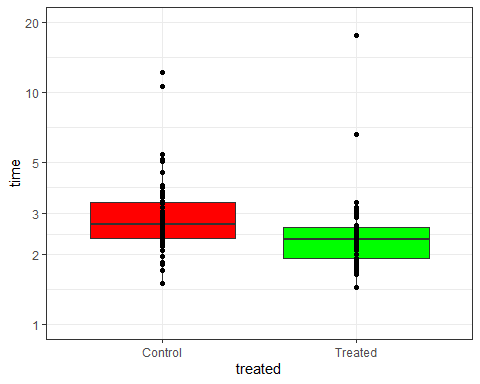
## Using themes to control the look of the graph:

Now our graph is almost the way we want it, but the overall look could be improved and we don’t want the legend. The look of a graph is determined by the ‘theme’ functions. There are several preset themes that will change the entire look, and then we can set specific elements individually with the theme() function.

g <- g + theme\_bw() + theme(legend.position="none")  
g

## Warning: Removed 2 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 2 rows containing missing values (geom\_point).



theme\_bw() sets a black-and-white theme that is good for scientific publishing.

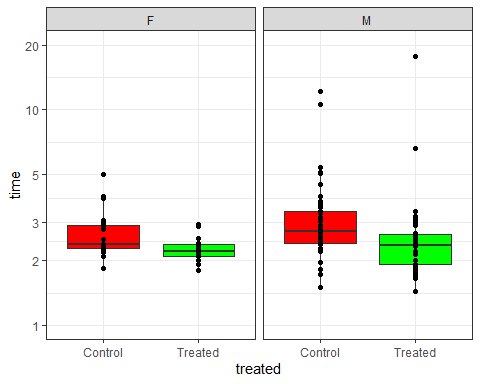
## Facets

Now suppose we want to stratify our graphs by sex. This is called ‘facetting’ in ggplot. There are two main ways to do this, facet\_wrap() acts on one dimension, while facet\_grid() creates a rectangle of plots along two dimensions. We’ll use facet\_wrap() here to split the graph by sex.

g + facet\_wrap(~sex)

## Warning: Removed 2 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 2 rows containing missing values (geom\_point).

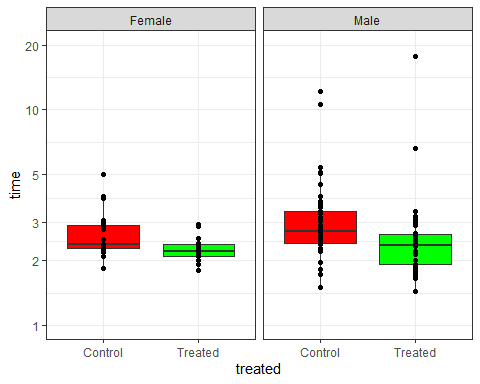


The graph is now split into two pieces that are placed side-by-side and share a common y-axis. Changing the facet titles is a bit tricky, but we can change the labels on the facets using a ‘labeller’:

g <- g + facet\_wrap(~sex, labeller=labeller(.rows=c(`F`="Female", `M`="Male")))  
g

## Warning: Removed 2 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 2 rows containing missing values (geom\_point).

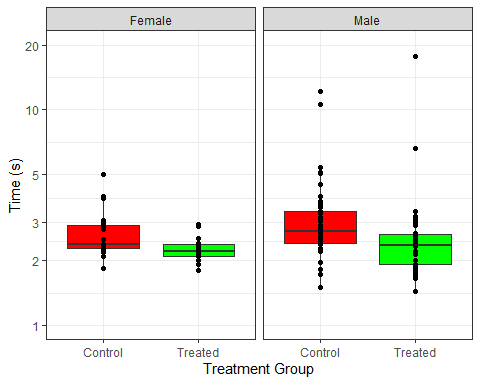
 ## Changing titles

You can set the titles for aesthetics (x-axis, y-axis, fill legend etc) using the scale functions, but I prefer to do it using the labs command:

g <- g + labs(x="Treatment Group", y="Time (s)")  
g

## Warning: Removed 2 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 2 rows containing missing values (geom\_point).



## Saving your graph

You can save your graph in any number of graphics formats using ggsave(), or by hitting ‘export’ in the plot window. Note that the relative size of the text, markers etc is determined by the shape and size of the graphics device your graph is drawn in. When you use ggsave you will have to specify the height and width of your plot in inches (or cm or mm), and you may have to use some trial and error to make the markers and text the appropriate size.

ggsave(filename = "testgraph1.png", g, height=10, width=5)

## Warning: Removed 2 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 2 rows containing missing values (geom\_point).

ggsave(filename = "testgraph2.png", g, height=4, width=4)

## Warning: Removed 2 rows containing non-finite values (stat\_boxplot).  
  
## Warning: Removed 2 rows containing missing values (geom\_point).

## Using beeswarm instead of point, and adding p-values

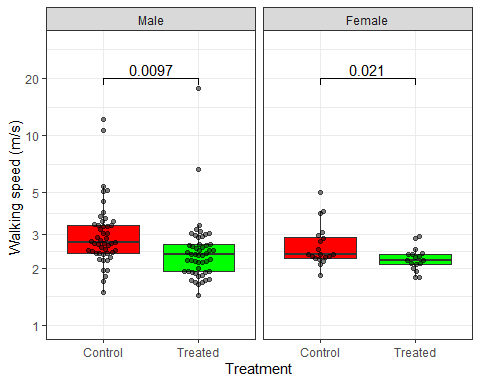
ggplot doesn’t make beeswarm graphs, but there is a package called ‘ggbeeswarm’ that does! Once we have installed the ggbeeswarm library and added it to the search path we can change geom\_point for geom\_beeswarm, and use its cex argument to control the amount of ‘jitter’, and alpha to control the transparency of the points.

I have also added ‘stat\_compare\_means()’ from the ggpubr package to add p-values. Note I do not think this is a good idea in general, we should do our statistical testing apart from our graphing. Eg here it is not clear exactly what stat\_compare\_means() has done to get the p-values, it turns out to be dependent on the scales of the graph! It’s also not clear why we should want a separate p-value for men and women, doing subgroup analyses like this is not recommended.

In any case, here is our final graph:

library(ggbeeswarm)  
ggplot(remove\_missing(alldata), aes(x=treated, y=time, fill=treated)) +  
 geom\_boxplot(outlier.shape = NA) + #   
 geom\_beeswarm(cex=2,alpha=0.5) +  
 facet\_wrap(~factor(sex, levels=c("M", "F")), labeller=labeller(.rows=c(M="Male",F="Female"))) +  
 scale\_fill\_manual(values=c("red","green")) +  
 scale\_y\_log10(breaks=c(1,2,3,5,10,20), limits=c(1,30)) +  
 scale\_x\_discrete(labels=c(`treated`="Treated",`control`="Control"),  
 breaks=c("treated", "control"),  
 na.translate=FALSE)+  
 stat\_compare\_means(method="t.test",label = "p.format", comparisons = list(c("control","treated"))) +  
 labs(x="Treatment", y="Walking speed (m/s) ", fill="Treatment") +  
 theme\_bw() +   
 theme(legend.position = "none")

## Warning: Removed 2 rows containing missing values.



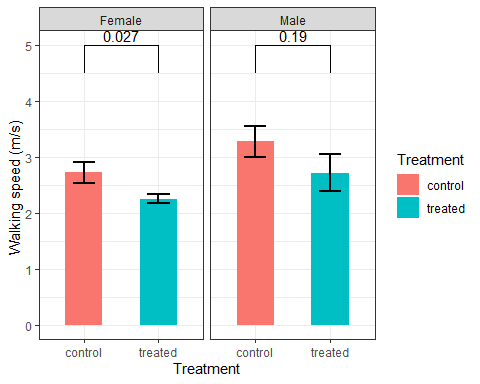
## Plotting summary statistics

You might want to plot only summary statistics, or place summaries along with points on the same graph. You may also want some p-values added. I don’t generally think this is a good idea, but the command for doing this is shown below.

You can do this with ‘stat\_summary()’ as follows:

ggplot(remove\_missing(alldata), aes(x=treated, y=time, fill=treated)) +  
 facet\_wrap(~sex, labeller=labeller(sex=c(M="Male",F="Female"))) +  
 stat\_summary( geom="bar", fun=mean,width=0.5) +  
 stat\_summary( geom="errorbar", fun.data=mean\_se,width=0.3 ,size=1) +   
 stat\_compare\_means(method="t.test",   
 comparisons = list(c("control","treated")) ,  
 label.y = 5) +  
 labs(x="Treatment", y="Walking speed (m/s) ", fill="Treatment") +  
 theme(legend.position = "none") + theme\_bw()

## Warning: Removed 2 rows containing missing values.



But note that the p-values are different to the p-values above! ggpubr has made different choices about the test depending on the other settings of the graph. This is another reason I would avoid the use of stat\_compare\_means() or related functions.