Interestingly, I recently encountered someone who referred to herself a "*ggplot programmer*" as opposed to, for example, an "*R Programmer*". While it does not bother me that the *tidyverse* provides a well-conceived point of entry into the world of R, it *DOES* bother me that someone might then suggest that the previous ways of dealing with things have somehow become invalid or inferior. Many of my students go on to work at places like the CDC which has a large body of R code that involves liberal use of Base R functions and supporting logic, thus I wouldn't be doing my job unless I made sure they knew something about it. Put another way, Base Graphics is not the "drunk uncle" of R visualization tools – *a beloved aging family member who refuses to acknowledge that their prime occurred years ago* – (I'm thinking Uncle Rico from Napoleon Dynamite as I write this). I tend to use a combination of approaches (both old and new) but then, in my family, I'm the "peace maker" / diplomat so I try to focus on the positive aspects contributed by each.

An Example Is In Order

I have an example that I hope will provoke some thought. The **mlbench** package has a data frame called *PimaIndians* which contains mostly measured / continuous data on 768 participants who have also been labelled as having diabetes or not.

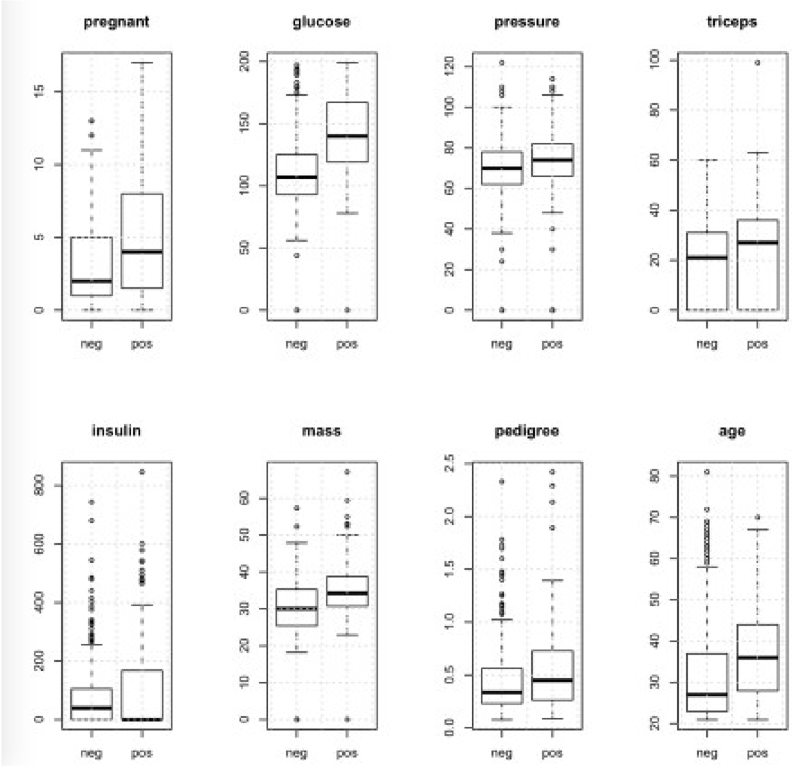
```
# Load mlbench as well as the data set of interest

library(mlbench)
data(PimaIndiansDiabetes)

# Get a shorter name

pm <- PimaIndiansDiabetes

str(pm)
'data.frame':    768 obs. of  9 variables:
 $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
 $ pressure: num  72 66 64 66 40 74 50 0 70 96 ...
 $ triceps : num  35 29 0 23 35 0 32 0 45 0 ...
 $ insulin : num  0 0 0 94 168 0 88 0 543 0 ...
 $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
 $ age     : num  50 31 32 21 33 30 26 29 53 54 ...

 $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

Respect Your Elders

I will use Base graphics to make a series of **boxplot**s that compare all features for each level of the diabetes factor which is either "pos" or "neg". The biggest "hack" here is the use of the **as.formula()** function to coerce a character string into something that can be then be parsed / recognized by the **boxplot()** function. To me, this isn't a hack at all, it's merely using a long existing function as it was intended. In fact most of the cognitive load here relates to understanding the **as.formula()** function and perhaps the **par()** function neither of which produces the actual boxplot. I also realize that the resulting figures do not share a common axis which is something that one would typically want although, again, in this case, I just wanted to get a look at the data with each feature reflecting its respective scale.

```
# Load mlbench as well as the data set of interest

library(mlbench)
data(PimaIndiansDiabetes)

# Get a shorter name
```

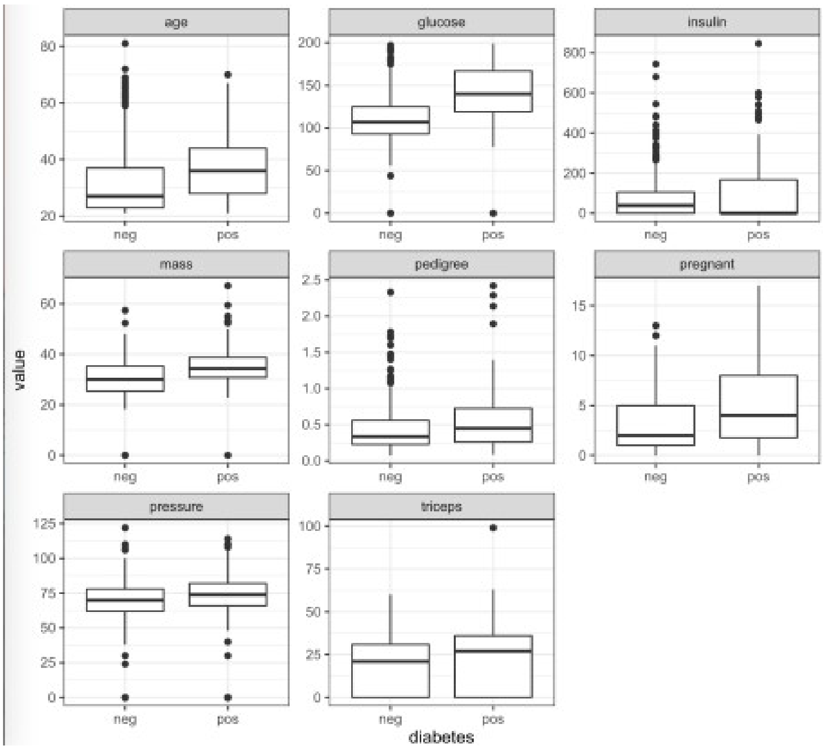
```
pm <- PimaIndiansDiabetes

# Set up the Plot Window manually

par(mfrow=c(2,4))

for (ii in 1:(nrow(pm)-1)) {
  form <- as.formula(paste(names(pm)[ii]," ~ diabetes",sep=""))
  boxplot(form,data=pm,main=names(pm)[ii])
  grid()
}
par(mfrow=c(1,1)) # Reset the plot window
```



What About The Tidyverse ?

Now consider the following code which uses elements of the *tidyverse* to produce a similar plot. In particular, we use the **gather()** function to reshape the data into a tidier format that lends itself to plotting with **ggplot()**

```
# Make the data tidy and then pipe to ggplot

library(tidyverse)

gather(pm,key="variable",val="value",-diabetes) %>%
       ggplot(aes(x=diabetes,y=value)) +
       geom_boxplot() +
       facet_wrap(~variable,scales="free") +
       theme_bw()
```



Which is the better figure ? Note, I'm *NOT* asking which figure *LOOKS* better but rather which approach is the best. Sorry for the abuse of all caps but I needed to make a point.

Looks Aren't Everything

The Base graphics solution feels very natural to me as someone who entered the world of graphics programming by using "*primitives*" to build plots in layers which works well with the "*pen on paper*" model adopted by Base graphics. Once you draw something in the plot window then it's there. You can draw over it , but sometimes it's just as easy to start over. When I want to reproduce a plot I've seen in a scientific journal, (or even a consumer graphic in a newspaper). I can usually knock it out using a combination of high and low level Base graphics commands in about half the time it would take to produce a comparable **ggplot** or **lattice** version.

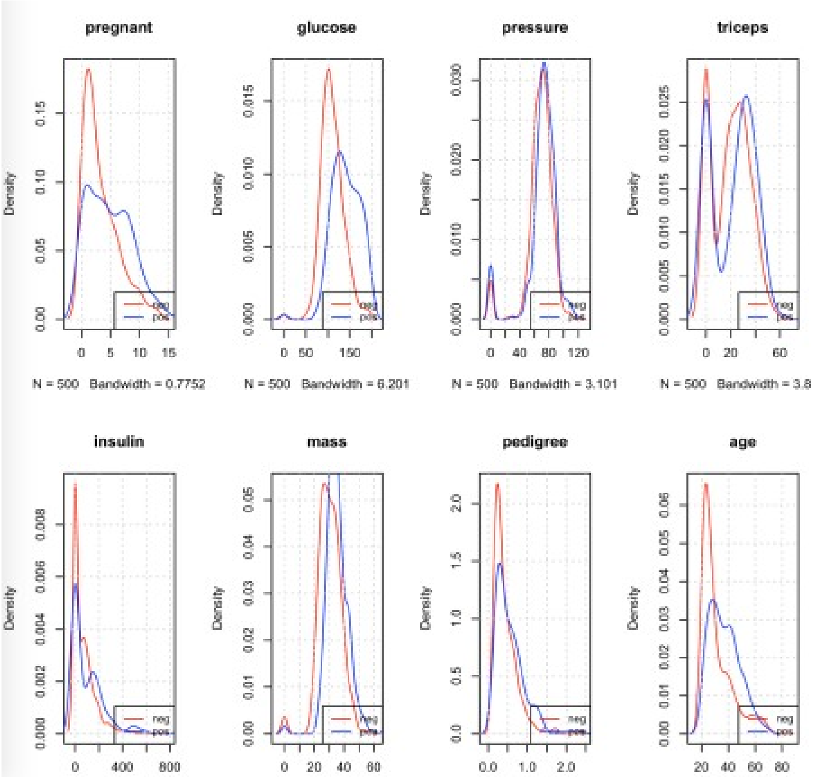
On the other hand, I appreciate the elegance and flow of *tidyverse* packages which allows me to think more about shaping data (and what I want to do with it) more so than the underlying options specific to a given geometry (well most of the time). To be fair, **ggplot** also allows one to build a plot in layers which becomes essential when reproducing against a reference example. Both **lattice** and **ggplot** will do a lot of the heavy lifting for you which is truly a marquee feature though sometimes you will definitely need to first decompose an intended plot into the appropriate aesthetics, mappings, geometries, etc. It could be argued that that is an essential step anyway. I once had a student ask, "*How do I keep ggplot from doing so much for me* ". Of course, as one acquires more experience then custom annotations and plots become much easier so there is an answer to that question.

In any case, were I to give someone an assignment to reproduce the plots above, I wouldn't care what they used. Particularly ***for exploratory work – use whatever you darn well want.***

Another Example

Here is another Base graphics solution for creating density plots from the same data. I showed this to a ggplot2 advocate who then left the room ! Granted there are some ways to clean up the code but then I didn't spend very much time on it all and still managed to get a decent result. As with the previous Base example, most of the confusion one might experience in looking at this code relates to prepping the data and making the input variables palatable to the density function. Nothing here is a "hack" if you already know about manipulating formulae in R which I believe is a reasonable expectation.

```
splitdf <- split(pm,pm$diabetes)
par(mfrow=c(2,4))

 for (ii in 1:8) {
    a <- paste("splitdf[[1]]$",names(pm)[ii],sep="")
    plot(density(eval(parse(text=a))),main=names(pm)[ii],col="red")
    a <- paste("splitdf[[2]]$",names(pm)[ii],sep="")
    lines(density(eval(parse(text=a))),col="blue")
    legend("bottomright",
           legend=c("neg","pos"),
           col=c("red","blue"),lty=1:1,cex=0.8)
    grid()

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



Lastly, I made a comment in this thread about how using Base R graphics simplifies the ordering or bar charts which is something that takes a few extra steps in ggplot. As an example, I tossed out the following of how to do it in Base R using the composite function approach.

```
barplot(rev(sort(table(mtcars$cyl))),horiz = TRUE)
```

I received a response that the following would work although it, like the example above, horrified my *tidyverse* colleague who views it as an entirely inappropriate application of pipes. However, for those of us with a UNIX background, using pipes to string together anything and everything is a completely legitimate thing to do – pipes don't cease to work in the presence of Base R commands.

```
mtcars$cyl %>%
     table() %>% sort() %>% rev() %>% barplot(horiz = TRUE)
```

Happy Holidays

So there is room at the family table for all graphics packages. No need to bicker about the differences which we should learn to appreciate and accept as strengths. None are perfect and some are younger than others but the wisdom of the older siblings (and parents) should not be hastily discarded as things tend to recycle. To wit – I never though I would see pipes in R but I'm glad it happened. And hey ! I didn't even get into talking about the **lattice** package but I have a friend who uses nothing else and is amazingly productive with it).