Research Post 01

Samir Hassanin

10-31-17

gridExtra: Working Smarter with Data Visualization

Note: packages will be referred to in **bold** while functions will be referred to in *italics*

Introduction

Ggplot2 is undisputedly the most elegant way to visualize data in RStudio. It can go above and beyond the functionalities of regular R plotting. However, one of its shortcomings is the inability to create multiple panels easily and visualize multiple plots on the same figure that are **not** connected by facetwrapping. Facetwrapping, in short, splits a plot into sub-plots grouped by one or more specific variable(s). **gridExtra** is an excellent package in combination with **Ggplot2** that most efficiently solves this problem, and can allow multiple data visualizations on a single figure that do not have to be connected via facetwrapping. In terms of replicable code for future use, **gridExtra** cuts out the fat (i.e. no need for reshaping, rescaling, melting, etc.) and creates an aesthetically appealing product with minimal codework. **In this post, we will be delving into plotting multiple graphs in the same figure with Ggplot2, and comparing what that process looks like with and without gridExtra. We will be focusing on the functions *grid.table*, *tableGrob*, and *grid.arrange.***

Familiarizing ourselves with gridExtra

Note: **gridExtra's** CRAN descriptions and manual documentation uses the term "grob" which is jargon for "grid graphical object" or description of a graphical object

1. CRAN description: **gridExtra** Provides a number of user-level functions to work with ``grid'' graphics, notably to arrange multiple grid-based plots on a page, and draw tables
2. Colloquial description: All functions in **gridExtra** are made to further improve the visualization and manipulation of plots
3. There are *16* main functions in the **gridExtra** package. We will focus on *grid.arrange*, *tableGrob*, and *grid.table*
4. *grid.arrange* function: Arrange multiple grobs on a single figure. This can be done by row or column.
5. *grid.table* function: Create a table containing text representing a character matrix.
6. *tableGrob* function: Create a table that can be grouped with other plots into a single figure using **gridExtra**. *tableGrob* makes a grob data table, rather than a text data table.

Now that we're a bit more familiar with what **gridExtra** does, and have the definition of *grid.arrange's*, *tablegrob's*, and *grid.table's* functions, we are going to jump right into seeing how this package will make data visualization easier.

Learning by doing

This section will be the meat of the document. We will be looking at plotting with **GGplot2**, and what we can do with multiple plots on a panel without **gridExtra**. Then we will do the same thing using **gridExtra**, observing the stark difference between the two. Again, *grid.arrange*, *grid.table*, and *tableGrob* will be the main functions from **gridExtra** that we will use. First let's load all the packages we will be using throughout:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.4.2
```

```
##
## Attaching package: 'gridExtra'
```

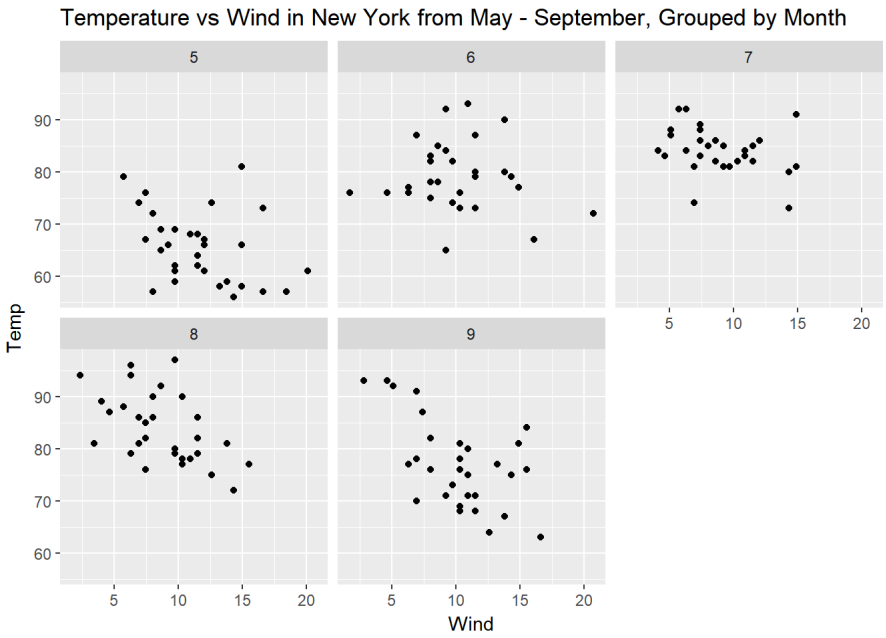
```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ggplot2)
library(datasets)
#datasets is for example data that we will be working with
```

Example 1: Multiple plots on a panel *without* gridExtra

I personally like the airquality dataset that is provided by R, so we will use that for this example. Let's say I wanted to make some plots, looking at temperature vs wind, facetwrapping by month.

```
ggplot(data = airquality, aes(x = Wind, y = Temp)) + geom_point()+ facet_wrap(~Month) + ggtitle("Temperature vs Wind in New York from May - September, Grouped by Month")
```

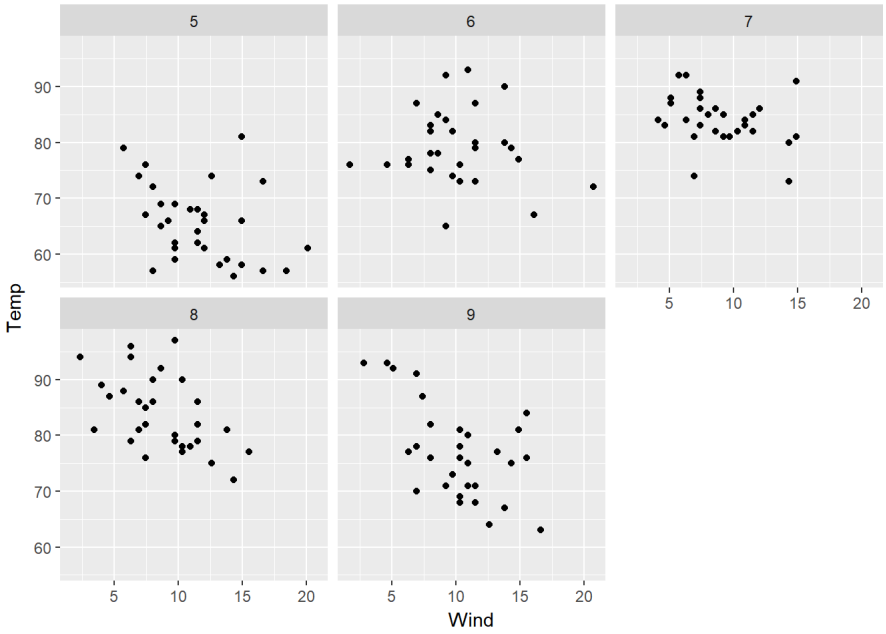


Great! We are able to see the temperature vs wind patterns and how they vary by month. Aside from some graphs (months 5 & 6) not having a labeled x-axis, and some graphs (6, 7, & 9) not having a y-axis, there really isn't an issue with this grid visualization from **GGplot2** since they're facetwrapped and have the same axes.

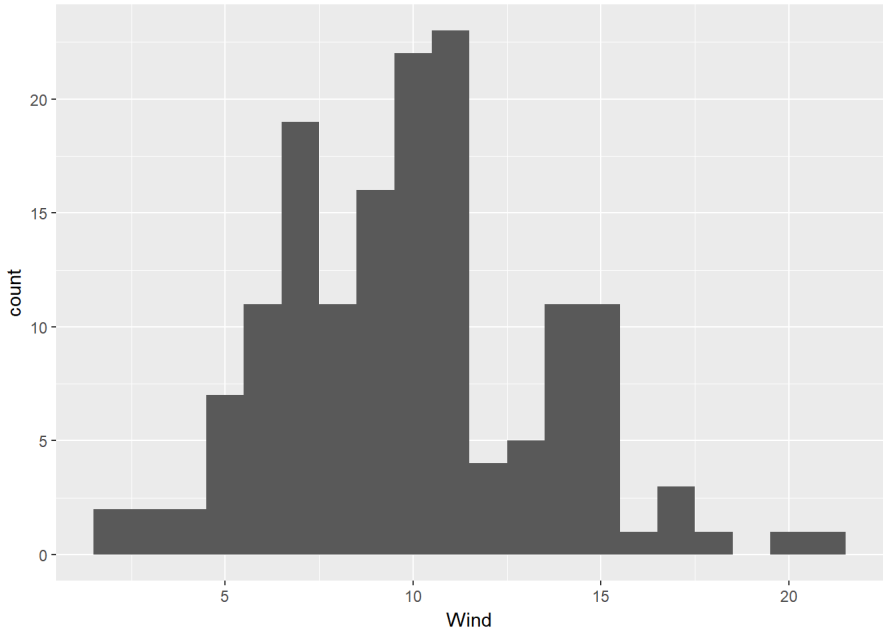
However, let's say I also wanted to look at a histogram of wind distribution, along with the current plots that I have, in a single figure. This becomes tricky. Or heck, let's say I wanted to get crazy, and look at a data table that summarizes wind and temperature averages per month on the same figure too!

This is where **GGplot2** has a difficult time holding its own. Let's try it.

```
q<- ggplot(data = airquality, aes(x = Wind, y = Temp)) + geom_point()+ facet_wrap(~Month)
r<- ggplot(data = airquality, aes(x = Wind)) + geom_histogram(binwidth = 1)
# q, r does not work
# must call q, and r, separately
q
```




```
r
```



As you can see, we are unable to visualize both the histogram and the facetwrapped temp vs. wind pattern plots together. There is no function in ggplot2 that allows us to combine the two separate visualizations into one image. Aside from that kink, let's try to make a data table image of averages per month for temperature and wind…

…oh, wait. **GGplot2** doesn't have a function that allows you to create a data table image! Let's jump into our next two examples and see how **gridExtra** can give us a little edge in terms of data visualization.

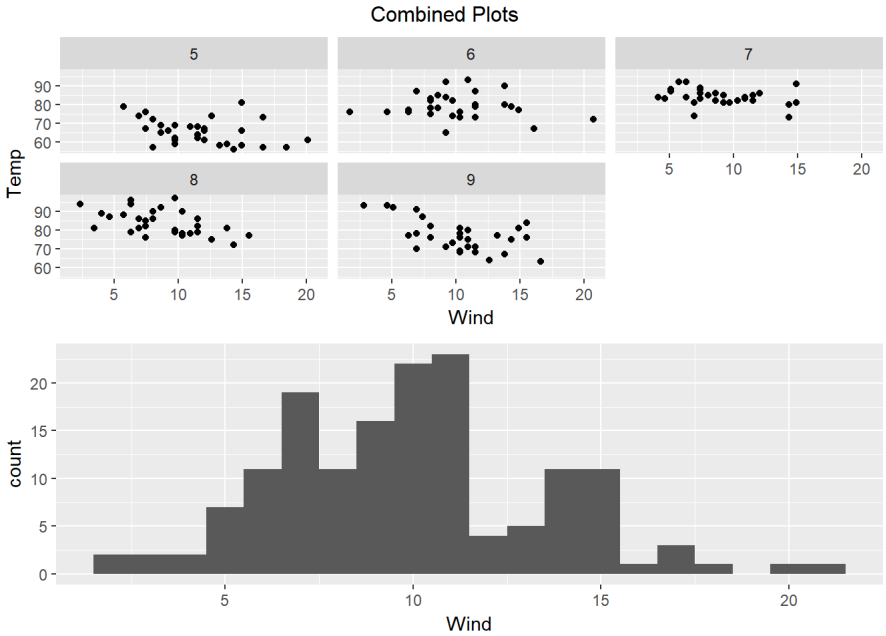
Example 2: Multiple plots on a panel *with* gridExtra

Since we will be doing a comparison of **Ggplot2** alone and **Ggplot2** combined with **gridExtra**, it would only make sense to continue using the same data set and graphs. Let's take a look.

```
q<- ggplot(data = airquality, aes(x = Wind, y = Temp)) + geom_point()+ facet_wrap(~Month)
r<- ggplot(data = airquality, aes(x = Wind)) + geom_histogram(binwidth = 1)
```

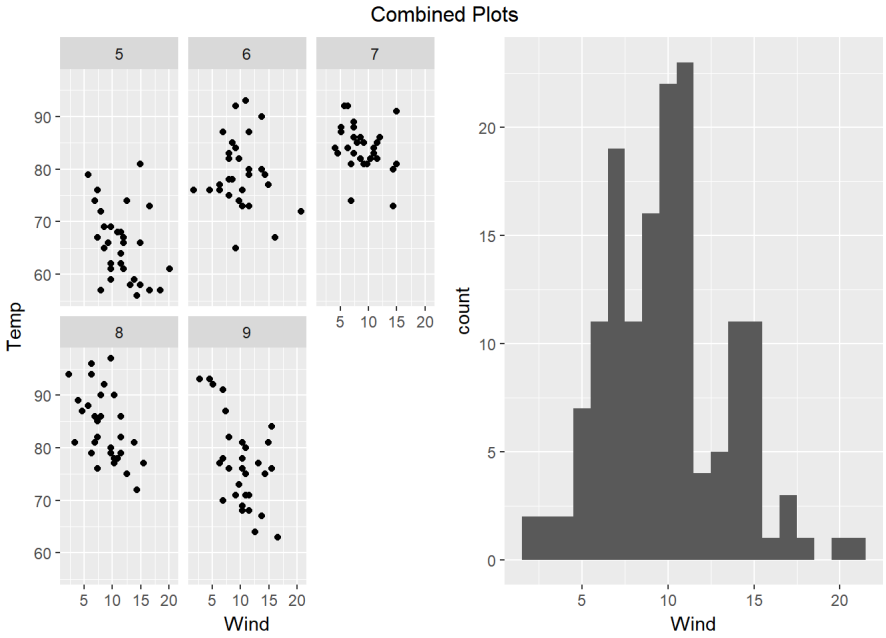
These are the same graphs from *example 1*. Remember how I've been itching to get these two separate graphs on the same figure? With **gridExtra**, we're going to do just that.

```
grid.arrange(q, r, nrow=2, top = "Combined Plots")
```



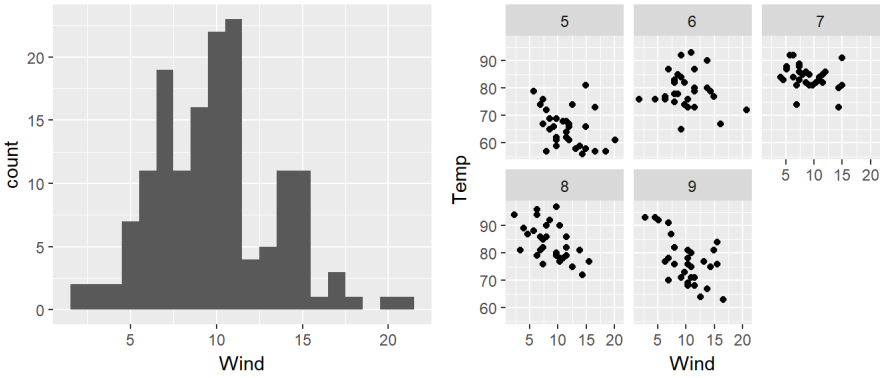
Would you look at that! Now, thanks to **gridExtra**, we have both of those plots on the same figure, making it much more clean, concise, and simple to visualize our different kinds of data in the same sphere. With **gridExtra**, we can also choose how to arrange the plots. Let's take a look

```
grid.arrange(q, r, ncol = 2, top = "Combined Plots")
```



As you could probably infer from the codes above, adding *nrow* or *ncol* determines how the plots are arranged. Also, you can order your plots by which one you want to see first (i.e. q then r, and vice versa). *grid.arrange* is a versatile function. We can also use "heights" to change the dimensions of the plots we are calling. Let's take a look

```
grid.arrange(r, q, ncol= 2, heights = c(0.3, 0.2))
```



As you can see, this drastically changed the output of our figure's plots, as they are an entirely different (and probably too small) height. You can use "heights" to toy with how you want the plot dimensions to look. This is useful if you want to emphasize one data visualization, while using another as merely a supplement. The possibilities are endless! Now that we've seen some basics on how to arrange and use *grid.arrange*, let's combine this with *grid.table* and *tablegrob* and see how we can make a complete figure.

Example 3: *grid.arrange*, *tablegrob*, and *grid.table*

To top off this figure of multiple plots, let's add a data table image that summarizes the average temperature grouped by month. *grid.table* is a very easy function that can get us a nice text-based data table. However, we can't do much with *grid.table* aside from look at a summary. When we want to place a table into a figure with other plots, We have to use *tablegrob* to make a table that will be interpreted as a graphical object. Then we can stick it with our plots!(again, you can use anything for the *grid.table* function, this is merely an example):

```
tempmean<-airquality%>%group_by(Month)%>%summarise(mean = mean(Temp))
```

Now that we have this data summary, let's make the table using *grid.table*

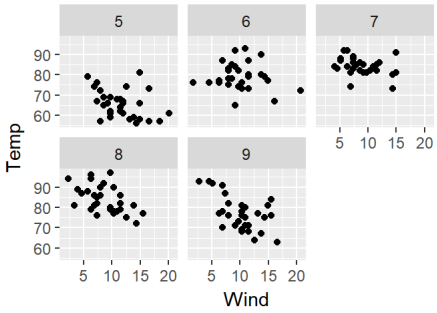
```
grid.table(tempmean)
```

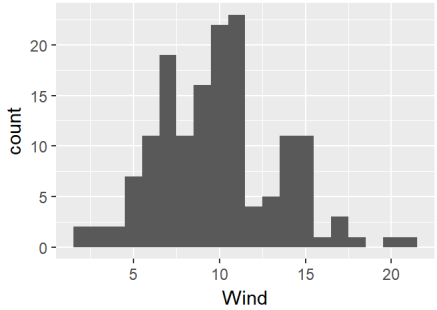
	Month	mean
1	5	65.5483870967742
2	6	79.1
3	7	83.9032258064516
4	8	83.9677419354839
5	9	76.9

As you can see, *grid.table* is pretty smart and efficient. We can now add this to the previous figure we made by using *tablegrob*, creating a coherent image of some summary statistics for our data. The reason we can't just throw the *grid.table* results up onto our figure of plots, is because it draws text. Text can't be placed on the same panel, but a grob can! And that's where *tablegrob* comes in handy.

```
t<-tableGrob(tempmean)
grid.arrange(t, q, r, ncol = 2)
```

	Month	mean
1	5	65.5483870967742
2	6	79.1
3	7	83.9032258064516
4	8	83.9677419354839
5	9	76.9





Voila! We've successfully used **gridExtra**'s functions *grid.arrange*, *grid.table*, and *tableGrob* to summarize data with **GGplot2**. Even better is that we were able to solve an issue with **GGplot2** that would prevent future statisticians, researchers, and data analysts from being able to create figures with multiple panels of data visualization types.

Take Home Message

As mentioned in the introduction, **Ggplot2** is an incredibly powerful tool for data visualization. However, when it comes to placing multiple **unfacetwrapped** plots in a single figure, **Ggplot2** falls short. **gridExtra** is another extremely powerful tool that performs as an addendum to **Ggplot2** and fixes this shortcoming. **In short, gridExtra helps both summarize data easily, AND plot different graphs on a single figure, in a grid-like format**. We learned *three* main functions of **gridExtra** that helps us better visualize grobs(remember, **a grob is just jargon for a graphical object!**:

1. *grid.arrange*: This function allows us to take multiple plots/grobs, and put them together on a single figure. Refer back to Examples for a recap.
2. *grid.table*: This function allows us to make a *text* data table with minimal effort. All we need to do is put *grid.table()* with the matrix in the parentheses to get a very nicely created data table (Recall that it is text and cannot be put on a figure of grobs!). Refer back to Examples for a recap.
3. *tableGrob*: This function allows us to make a *grob* data table with minimal effort (just like *grid.table* except in different format). The function works exactly the same as *grid.table's*, but the result allows you to take the data table and use *grid.arrange* to put it on a figure with other plots/grobs. Refer back to Examples for a recap.

Motivation

My motivation for selecting this specific package to write a post on lies in my interest in biomedical research and scientific presentation. I took this class in order to pick up R studio and become better at working with scientific data to make aesthetically pleasing, clear, and concise graphs for a lab that I work in. When I found out about **gridExtra** while researching a topic for this post, it was hard to turn it down. **gridExtra** makes data visualization easier, more practical, and more communicable.

References

- https://www.magesblog.com/post/2015-04-14-plotting-tables-alsongside-charts-in-r/
- https://www.r-bloggers.com/extra-extra-get-your-gridextra/
- http://rstudio-pubs-static.s3.amazonaws.com/2852_379274d7c5734f979e106dcf019ec46c.html
- https://github.com/tidyverse/ggplot2/wiki/Mixing-ggplot2-graphs-with-other-graphical-output
- https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html
- http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/81-ggplot2-easy-way-to-mix-multiple-graphs-on-the-same-page/
- http://stat545.com/block020_multiple-plots-on-a-page.html#use-the-cowplot-package
- https://stackoverflow.com/questions/29067541/rmarkdown-how-to-change-the-font-color
- https://stackoverflow.com/questions/2822089/how-to-link-to-part-of-the-same-document-in-markdown