Creating Graphs with ggformula

### 1. Introduction

It is often necessary to create graphs to effectively communicate key patterns within a dataset. Many software packages allow the user to make basic plots, but it can be challenging to create plots that are customized to address a specific idea. While there are numerous ways to create graphs, this tutorial will focus on the R package **ggformula**, created by Danny Kaplan and Randy Pruim.

# This tutorial will use the following packages  
library(ggplot2)  
library(ggformula)  
library(mosaic)

**Data**: In this tutorial, we will use the [AmesHousing](http://www.amstat.org/publications/jse/v19n3/decock.pdf) data, which provides information on the sales of individual residential properties in Ames, Iowa from 2006 to 2010. The data set contains 2930 observations, and a large number of explanatory variables involved in assessing home values. A full description of this dataset can be found [here](http://www.amstat.org/publications/jse/v19n3/Decock/DataDocumentation.txt).

# The csv file should be imported into RStudio:  
AmesHousing <- read.csv("data/AmesHousing.csv")  
# str(AmesHousing)

To start, we will focus on just a few variables:

* SalePrice: The sale price of the home
* GrLivArea: The above ground living area in the home
* Fireplaces: The number of fireplaces in the home
* KitchenQual: The quality rating of the kitchen

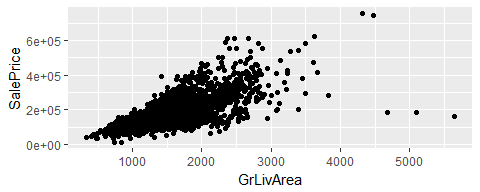
### 2. The structure of the functions in the ggformula package

The **ggformula** package is based on another graphics package called **ggplot2**. It provides an interface that makes coding easier for people new to coding in R. One primary benefit is that it follows the same intuitive structure provided by the creators of the [mosaic package](https://cran.r-project.org/web/packages/mosaic/mosaic.pdf).

goal ( y ~ x , data = mydata, … )

For example, to create a scatterplot in **ggformula**, we use the function called gf\_point(), to create a graph with points. We then use SalePrice as our y variable and GrLivArea as our x variable. The ... indicates that we have an option to add additional code, but it is not required.

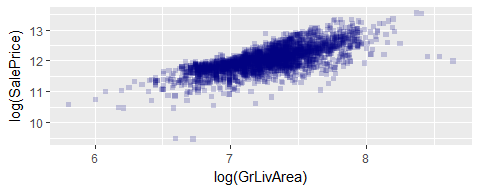
# Create a scatterplot of above ground living area by sales price  
gf\_point(SalePrice ~ GrLivArea, data = AmesHousing)



#### Making Modifications to Plots with ggformula

It is easy to make modifications to the color, shape and transparency of the points in a scatterplot.

# Create a scatterplot with log transformed variables, coloring by a third variable  
gf\_point(log(SalePrice) ~ log(GrLivArea), data = AmesHousing, color = "navy", shape = 15, alpha = .2)



**Questions**:

1. Based upon the data documentation, what are the five different levels for kitchen quality?
2. In the code above, adjust alpha to any values between 0 and 1. How does adding alpha = .2 modify the points on a graph?
3. What type of shape corresponds to using shape = 1?
4. Instead of using color = "navy", run the code using color = ~ KithchenQual. Notice that fixed colors are given in quotes while a variable from our data frame is treated as an explanatory variable in our model.

The scatterplots above suffer from overplotting, that is, many values are being plotted on top of each other many times. We can use the alpha argument to adjust the transparency of points so that higher density regions are darker. By default, this value is set to 1 (non-transparent), but it can be changed to any number between 0 and 1, where smaller values correspond to more transparency. Another useful technique is to use the facet option to render scatterplots for each level of an additional categorical variable, such as kitchen quality. In **ggformula**, this is easily done using the gf\_facet\_grid() layer.

We can use the pipe operator %>% to add a new layer into a graph. This pipe operator is an easy way to create a chain of processing actions by allowing an intermediate result (left of the %>%) to become the first argument of the next function (right of the %>%). Below we start with a scatterplot and then assign that scatterplot to the gf\_facet\_grid() function to create distinct panels for each type of kitchen quality. Then the result is again passed to the gf\_labs() function, which adds titles and labels to the graph.

# Create distinct scatterplots for each type of kitchen quality  
gf\_point(SalePrice/100000 ~ GrLivArea, data = AmesHousing) %>%  
 gf\_facet\_grid(KitchenQual ~ . ) %>%  
 gf\_labs(title = "Figure 3: Housing Prices in Ames, Iowa",  
 y = "Sale Price (in $100,000)",  
 x = "Above Ground Living Area")

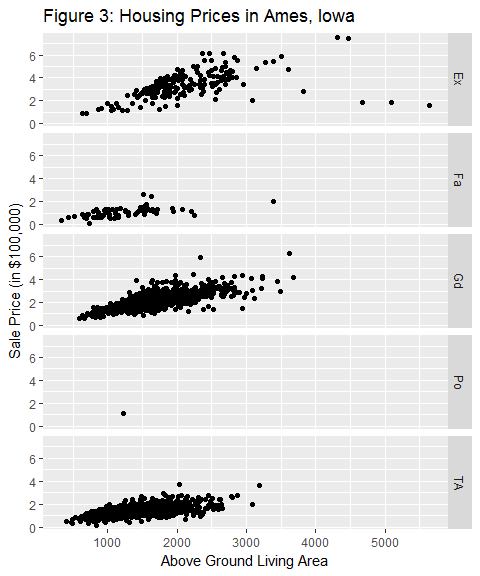
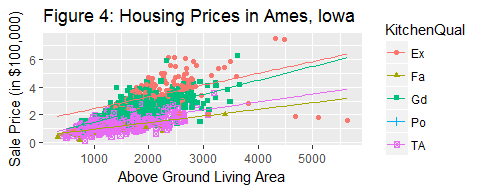


Figure 3 facets the scatterplot by Kitchen Quality. In Figure 4, we overplot these graphs, and use color and shape to identify the Kitchen Quality. Both graphs allow us to look at Sale Price by Above Ground Living Area and Kitchen Quality at the same time. Often, researchers create multiple graphs to determine which best shows patterns within the data. Figure 4 allows us to more clearly see the effect of Kitchen Quality on the Sale Price, however, it is still difficult to see many of the points.

gf\_point(SalePrice/100000 ~ GrLivArea, data = AmesHousing, shape = ~ KitchenQual, color = ~ KitchenQual) %>%   
 gf\_lm() %>%   
 gf\_labs(title = "Figure 4: Housing Prices in Ames, Iowa",  
 y = "Sale Price (in $100,000)",   
 x = "Above Ground Living Area")



**Remarks:**

* The y ~ x format of **ggformula** is required even when there is just one variable. For example, in univariate graphs, such as histograms, we should write gf\_histogram(~ SalePrice, data = AmesHousing) to indicate that there is no y variable in our graph. Similiarly, gf\_facet\_grid(KitchenQual ~ . ) indicates that we should facet by the y axis and not by the x axis.
* Multiple shapes can be used as points. The [Data Visualization Cheat Sheet](https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf) lists several shape options`.
* Note that gf\_facet\_grid and gf\_facet\_wrap commands are used to create multiple plots. Try incorporating gf\_facet\_grid(~KitchenQual) or gf\_facet\_grid(Fireplaces~KitchenQual) into a scatterplot to see how it separates each graph by a categorical variable.
* When assigning fixed characteristics (such as color, shape or size), the commands include quotes, such as color="green". When characteristics are dependent on the data, the command should occur without quotes, such as color= ~ KitchenQual.
* In the above examples, only a few functions are listed. The [ggformula description](https://cran.r-project.org/web/packages/ggformula/ggformula.pdf) lists each function and gives detailed examples of how they are used. You can also use the tab completion feature in RStudio (type gf\_ and hit the Tab key on your console) to see options for most graphs.
* At the beginning of an r chunk you can specify the figure size, for example {r fig.height=6}.
* The mosaic package includes an mplot() function that involves a helpful pull-down menu for graphic options. In the RStudio Console, type > mplot(AmesHousing) and select 2 for a two-variable plot. Select the gear symbol in the top right corner of the graphics window and choose x and y variables. After selecting these items, click the Show Expression to see the **ggformula** code used to make a boxplot. You can then modify the code to include an appropriate title to the plot.

**Questions:**

1. What does the gf\_lm() command do? How is it different than gf\_smooth()?
2. Try incorporating gf\_facet\_grid(Fireplaces~KitchenQual) into the scatterplot. What does this command do?
3. Create a scatterplot using Year.Built as the explanatory variable and SalePrice as the response variable. Include a regression line, a title, and labels for the x and y axes.
4. Use the following code to create a boxplot that incorporates two explanatory variables, KitchenQual and Fireplaces. We then use the pipe operator to add points to the boxplot. Note that you will need to adjust the fig.height as done in previous graphs.

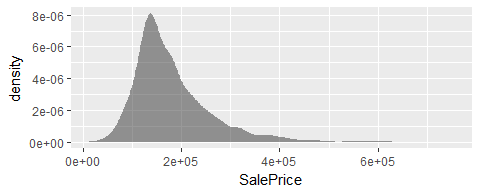
gf\_boxplot(log(SalePrice) ~ KitchenQual | Fireplaces, data = AmesHousing) %>%  
 gf\_jitter(width = 0.2, alpha = 0.1, color = "blue")

1. The graph in question 8 uses a slightly different syntax. Recreate the graph in quesiton 8, but instead of using KitchenQual | Fireplaces, use the gf\_facet\_wrap() function.
2. What is the difference between gf\_jitter and gf\_point?

### 3. Additional Considerations with R graphics

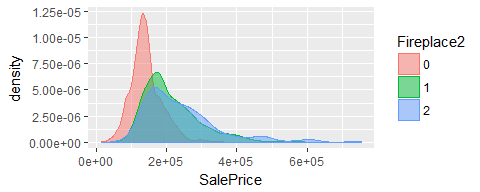
**Influence of data types on graphics:** If you use the str command after reading data into R, you will notice that each variable is assigned one of the following types: Character, Numeric (real numbers), Integer, Complex, or Logical (TRUE/FALSE). In particular, the variable **Fireplaces** in considered an integer. In the code below we try to color and fill a density graph by an integer value. Notice that the color and fill commands appear to be ignored in the graph.

# str(AmesHousing)  
gf\_density(~ SalePrice, data = AmesHousing, color = ~ Fireplaces, fill = ~ Fireplaces)



In the following code, we use the dplyr package to modify the AmesHousing data; we first restrict the dataset to only houses with less than three fireplaces and then create a new variable, called **Fireplace2**. The as.factor command creates a **factor**, which is a variable that contains a set of numeric codes with character-valued levels. Notice that the color and fill command now work properly.

# Create a new data frame with only houses with less than 3 fireplaces  
AmesHousing2 <- filter(AmesHousing, Fireplaces < 3)  
# Create a new variable called Fireplace2  
AmesHousing2 <-mutate(AmesHousing2,Fireplace2=as.factor(Fireplaces))  
#str(AmesHousing2)  
  
gf\_density(~ SalePrice, data = AmesHousing2, color = ~ Fireplace2, fill = ~ Fireplace2)



**Questions:**

1. Pick another quantitative variable from the AmesHousing2 data set and describe its distribution using the gf\_histogram() function. Use a secondary variable to see more detailed pattersn, such as fill = ~KitchenQual. Consider different bin widths to see if the shape of the distribution changes. Type ?gf\_histogram in the command line and then learn how to use the bin and binwidth arguement.

### 4. On Your Own

In order to complete this activity, you will need to use the dplyr package to manipulate the dataset before making any graphics.

* Restrict the AmesHousing data to only sales under normal conditions. In other words, Condition.1 == Norm
* Create a new variable called TotalSqFt = GrLivArea + TotalBsmtSF and remove any homes with more than 3000 total square feet.
* Create a new variable, where No indicates no fireplaces in the home and Yes indicates at least one fireplace in the home.
* With this modified data file, create a graphic involving no more than three explanatory variables that portrays how to estimate sales price. For example, in previous graphics we used kitchen quality, above ground square footage, and number of fireplaces to estimate sale price.

### Additional Resources

* <https://www.rdocumentation.org/packages/ggformula/versions/0.7.0>: A full description of the ggformula package.
* <https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>: Data Visualization with ggplot2 Cheat Sheet. NOte that this sheet lists several shape scales and color scales that can be used within **ggformula**.
* The default color schemes are often not ideal. In some cases, it can be difficult to perceive differences between categories. To learn more about color schemes, see Roger Pengâs discussion of plotting and colors in R, <https://www.youtube.com/watch?v=HPSrjKt-e8c>, or the R Graph Gallery, <https://www.r-graph-gallery.com/>.