Creating Graphs with ggplot2

Lab 1, Fall 2016

Complete this lab at your own pace with your assigned lab group. You should create a new R project on the R Studio server for this lab and share the project with me. Within this R project, you should create an R markdown file with your solutions. An R markdown template for this lab can be found at bit.ly/math107-lab1. Please complete your solutions by 4 pm on Wednesday, September 21.

1. Introduction

It is often necessary to create graphics to effectively communicate key patterns within a data set. While many software packages allow the user to make basic plots, 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 `ggplot2`, created by Hadley Wickham.

In this lab, we will focus on building plots in layers using the `ggplot2` package. This approach uses a particular grammar inspired by Leland Wilkinson's landmark book, *The Grammar of Graphics*, that focused on thinking about, reasoning with and communicating with graphics. It enables layering of independent components to create custom graphics for tidy data sets.

To begin, load the `ggplot2` package using the following command

```
library(ggplot2)
```

If you have not yet installed the package, you can do so with the following command

```
install.packages("ggplot2")
```

Data: In this tutorial, we will use the AmesHousing data set, 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 data set can be found here.

If you have saved the data file in the data folder of your R Studio project, then you can load it using the following command. Alternatively, you can use "Import Dataset" button in the Environment tab.

```
AmesHousing <- read.csv("data/AmesHousing.csv")
# str(AmesHousing)
```

2. Building your first plot

To see how plots are constructed, let's first create a histogram of the sales price (`SalePrice`) of homes in Ames, IA. To do this, we will use the `ggplot` function which expects:

- the data frame where the variables exist (the `data` argument) and
- the names of the variables to be plotted (the `mapping` argument).

The names of the variables will be entered into the aes function as arguments where aes stands for "aesthetics".

With that in mind, we start by creating the base layer, or the backdrop on which we will layer the elements of our histogram.

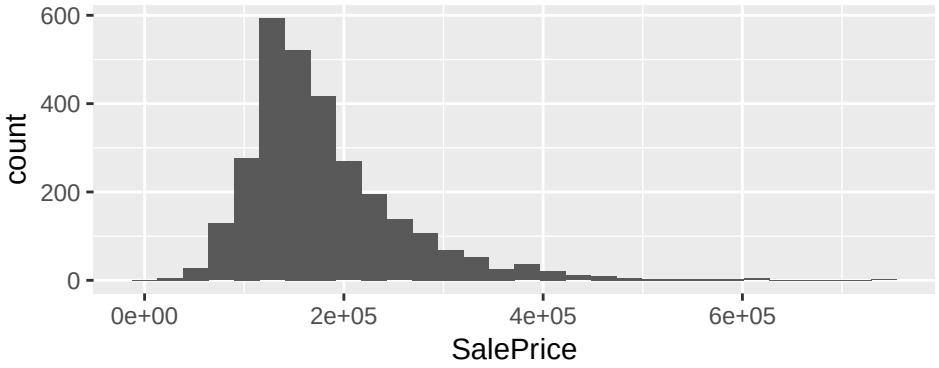
```
ggplot(data = AmesHousing, mapping = aes(x = SalePrice))
```

We next proceed by adding a layer—hence, the use of the + symbol—to the plot to produce a histogram.

Note: You are encouraged to enter **Return** on your keyboard after entering the `+`. As we add more and more elements, it will be nice to keep them indented as you see below. Note that this will not work if you begin the line with the `+`.

```
ggplot(data = AmesHousing, mapping = aes(x = SalePrice)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Here, we are warned that the histogram was been drawn with 30 bins, a rather arbitrary choice. We can customize our histogram by adding a `binwidth` argument to the `geom_histogram` command:

```
ggplot(data = AmesHousing, mapping = aes(x = SalePrice)) +
  geom_histogram(binwidth=10000)
```

We can also add some color to the plot by invoking the fill and color arguments.

```
ggplot(data = AmesHousing, mapping = aes(x = SalePrice)) +
  geom_histogram(binwidth=10000, fill = "steelblue2", color = "black")
```

Currently, the x-axis label does not contain units. To change the axis labels we add a layer to our plot

```
ggplot(data = AmesHousing, mapping = aes(x = SalePrice)) +
  geom_histogram(binwidth=10000, fill = "steelblue2", color = "black") +
  labs(x = "Sale Price (in dollars)")
```

Note that we can also tweak the y-axis label with the addition of a `y` argument in the `labs` function.

Finally, we can easily add a title to our plot with the addition of another layer

```
ggplot(data = AmesHousing, mapping = aes(x = SalePrice)) +
  geom_histogram(binwidth=10000, fill = "steelblue2", color = "black") +
```

```
  labs(x = "Sale Price (in dollars)") +
  ggtitle("Sales Prices of Homes in Ames, IA")
```

Throughout the remainder of this lab we will explore how to create and customize the basic plot types:

- histograms,
- bar charts,
- density plots,
- scatterplots,
- and side-by-side plots.

3. Building basic univariate plots

Displaying a quantitative variable

In the previous section you learned how to create a histogram, which can be used to display the distribution of a quantitative variable. Before moving on, answer the following questions:

Questions

1) Create a histogram of the above-grade living area (in ft^2). This is the `Gr.Liv.Area` variable in the data set. Remember to experiment with the bin width!
2) Would you classify the distribution of above-grade living area as symmetric or skewed?
3) What would you guess is the "center" value in this distribution? Why did you make that choice?

Density plots are also useful in exploring the shape of the distribution of a quantitative variable. To create a density plot, we start with the same base layer as a histogram, but add the `geom_density` layer rather than the `geom_histogram` layer. For example, the below code creates a density plot for the sale price.

```
ggplot(data = AmesHousing, mapping = aes(x = SalePrice)) +
  geom_density()
```

We can customize a density plot using the same options as a histogram.

Question

4) Use the color and fill arguments to customize the density plot or sale price. You should also change the x-axis label to "Sale Price (in dollars)."

Displaying a categorical variable

Both histograms and density plots display the distribution of a quantitative variable. To display the distribution of a categorical variable we can use bar charts.

As a first example, we'll explore the number of houses in Ames with central air (`Central.Air`). To do this, we'll use the `geom_bar` layer:

```
ggplot(data = AmesHousing, mapping = aes(x = Central.Air)) +
  geom_bar()
```

Clearly, the vast majority of houses sold in Ames, IA have central air.

One common customization for bar charts is to make each bar a different color. To do this we need to add the fill argument to the aesthetic mapping:

```
ggplot(data = AmesHousing, mapping = aes(x = Central.Air, fill = Central.Air)) +
  geom_bar()
```

Questions

5) Create a bar chart of the type of heating system (`Heating`).
6) Which type of heating system is the most common?

Categorical variables are often coded numerically. For example, in the **AmesHousing** data set the garage capacity (`Garage.Cars`, the number of cars the garage can hole) is coded as an integer (0, 1, 2, 3, 4, or 5). This variable can be considered categorical, as the number of cars can be viewed as a label. To create a bar chart for the `Garage.Cars` variable, we must tell `ggplot` to treat it as a factor

```
ggplot(data = AmesHousing, mapping = aes(x = as.factor(Garage.Cars))) +
  geom_bar() +
  labs(x = "Garage capacity in cars")
```

Question

7) Create a bar chart of the number of `Fireplaces`. What does the bar chart reveal?

4. Building bivariate plots

Now that we know how to construct the basic univariate plots, we need to explore ways to display two variables.

Two categorical variables

When we wish to compare the distribution of two categorical variables, we can again use bar charts. As a first example, let's consider comparing the distribution of full bathrooms (`Full.Bath`) and bedrooms (`Bedroom.AbvGr`).

One option is to create a segmented bar chart. A segmented bar chart displays a bar chart for one of the categorical variables, and uses color to fill in the bars proportional to the occurrence of the other variable within that category. For our example, we can create a bar chart of the number of bedrooms and use color to represent the number of full bathrooms.

```
ggplot(data = AmesHousing, mapping = aes(x = as.factor(Bedroom.AbvGr), fill = as.factor(Full.Bath))) +
  geom_bar() +
  labs(x = "Number of bedrooms above grade")
```

Questions

8) Recreate the above bar chart.
9) What kinds of questions are not easily answered by looking at the above figure?
10) What can you say about the relationship between the number of bedrooms and the number of full bath rooms?

Instead of displaying counts on the y-axis, we could also display proportions. To do this, we add `position = "fill"` to the `geom_bar` layer.

```
ggplot(data = AmesHousing, mapping = aes(x = as.factor(Bedroom.AbvGr), fill = as.factor(Full.Bath))) +
  geom_bar(position = "fill") +
  labs(x = "Number of bedrooms above grade")
```

Questions

11) Recreate the above plot.
12) What kinds of questions are more easily answered by looking at the above figure?

Another option would be to create a side-by-side bar chart, which is often called a faceted bar chart. Faceting is used when we'd like to create small multiples of the same plot over a different categorical variable. By default, all of the small multiples will have the same vertical axis.

```
ggplot(data = AmesHousing, mapping = aes(x = as.factor(Bedroom.AbvGr), fill = as.factor(Full.Bath))) +
  geom_bar() +
  facet_wrap(~ Full.Bath) +
  labs(x = "Number of bedrooms above grade")
```

It is often easier to compare small multiples if they are either displayed in a row or a column. To specify the layout of the panels, you can add arguments specifying the number of rows (`nrow`) and the number of columns (`ncol`). For example, we can arrange all of the panels in one column:

```
ggplot(data = AmesHousing, mapping = aes(x = as.factor(Bedroom.AbvGr), fill = as.factor(Full.Bath))) +
  geom_bar() +
  facet_wrap(~ Full.Bath, ncol = 1) +
  labs(x = "Number of bedrooms above grade")
```

Questions

13) Recreate the above plot.
14) What kinds of questions are more easily answered by looking at the above figure?

Two quantitative variables

To display the relationship of two quantitative variables, we can use a scatterplot. For example, we can look at the relationship between the above grade living area and sales price of a home. To create this scatterplot, we need to add a `geom_point` layer to our base layer.

```
ggplot(data = AmesHousing, mapping = aes(x = Gr.Liv.Area, y = SalePrice)) +
  geom_point() +
  labs(x = "Above grade living area (sq. ft)", y = "Sale price ($)")
```

Questions

15) Recreate the above scatterplot.
16) Describe the relationship between the above grade living area and sales price of a home.

The large mass of points near the bottom left of the scatterplot can cause some confusion. This is the result of a phenomenon called over-plotting. As one may guess, this corresponds to values being plotted on top of each other over and over again. It is often difficult to know just how many values are plotted in this way when looking at a basic scatterplot as we have here.

One way of relieving this issue of over-plotting is to make the points somewhat transparent. The amount of transparency of a point can be set using the `alpha` argument in `geom_point`. By default, this value is set to 1 (non-transparent), but can be changed to any number between 0 and 1. Smaller values correspond to more transparency.

```
ggplot(data = AmesHousing, mapping = aes(x = Gr.Liv.Area, y = SalePrice)) +
  geom_point(alpha = 0.2) +
  labs(x = "Above grade living area (sq. ft)", y = "Sale price ($)")
```

Questions

17) Recreate the above scatterplot.
18) Why is setting the alpha argument value useful with scatterplots? What further information does it give you that a regular scatterplot cannot?

One categorical and one quantiative variable

Many times we are interested in comparing the distribution of a quantitative variable over the levels of a categorical variable. To do this we can still use our basic plot types (histograms, density plots, and boxplots)

but we must create a plot for each category. For example, we can explore the relationship between sale price (`SalePrice`) and whether a house has central air (`Central.Air`).

One way to create a plot for each category is to use faceting. For example, we can create faceted histograms of sale price by the number of bedrooms:

```
ggplot(data = AmesHousing, mapping = aes(x = SalePrice)) +
  geom_histogram() +
  facet_wrap(~ Central.Air, ncol = 1)
```

Another option is to overlay density plots

```
ggplot(data = AmesHousing, mapping = aes(x = SalePrice, fill = Central.Air)) +
  geom_density(alpha = 0.5)
```

Questions:

19) Recreate the faceted histogram and overlaid density plots.
20) What do these plots reveal about the impact of central air on the price of a home?
21) Which plot did you find most helpful? Why?

A third option is to create side-by-side boxplots. To do this we use the `geom_boxplot` layer, where we display the categories on the x-axis and the quantitative variable on the y-axis.

```
ggplot(data = AmesHousing, mapping = aes(x = Central.Air, y = SalePrice)) +
  geom_boxplot()
```

Questions:

22) Create side-by-side boxplots of sale price by building type (`Bldg.Type`).
23) Describe what you can learn from this plot about the relationship between sale price and building type.

5. Customizing colors

For all of the graphics created so far we have used the default color options. Unfortunately, this color scheme is not always the best. For example, it is not colorblind safe—that is, people with certain forms of colorblindness may be unable to perceive differences in some of the colors. Luckily, there is an easy way to change the color scheme used. To demonstrate this, reconsider the segmented bar chart of the number of bedrooms and the number of full bathrooms.

One color scheme option is to use `RColorBrewer` to generate a better color palette. There are many color palette options, but one reasonable choice for categorical data is the `Dark2` palette. To use this palette as the fill color, we utilize the `scale_fill_brewer` function. An additional perk of changing the color palette is that we can also change the legend title.

```
ggplot(data = AmesHousing, mapping = aes(x = as.factor(Bedroom.AbvGr), fill = as.factor(Full.Bath))) +
  geom_bar(position = "fill") +
  labs(x = "No. bedrooms above grade") +
  scale_fill_brewer("No. bathrooms", palette = "Dark2")
```

While this is a nicer palette, it is not totally colorblind safe. A better palette to ensure that everyone can perceive the differences in categories is to use the `scale_fill_colorblind` function found in the `ggthemes` package. (You'll need to install this package first!)

```
library(ggthemes)

ggplot(data = AmesHousing, mapping = aes(x = as.factor(Bedroom.AbvGr), fill = as.factor(Full.Bath))) +
  geom_bar(position = "fill") +
```

```
  labs(x = "No. bedrooms above grade") +
  scale_fill_colorblind("No. bathrooms")
```

The color can be change using analogous functions as fill. For example, we can color points on a scatterplot by a categorical variable.

```
ggplot(data = AmesHousing, mapping = aes(x = Gr.Liv.Area, y = SalePrice, color = Central.Air)) +
  geom_point(alpha = 0.2) +
  labs(x = "Above grade living area (sq. ft)", y = "Sale price ($)") +
  scale_color_colorblind("AC?")
```