Visualizing Cross Tabulations

R Tutorials for Applied Statistics

Note on required packages: The following code requires the packages `tidyverse`, `scales`, `stringr`, and `ggthemes`.

- The `tidyverse` package contains many packages that allow you to organize, summarize, and plot data.
- We use the `scales` library to customize the scales of our axes.
- The `stringr` package allows us to manipulate strings, which we use to manipulate string labels.
- The `ggthemes` package provides multiple themes, which are combinations of parameters to change a plots look and feel

If you have not already done so, download, install, and load the libraries with the following code:

```
# These lines only need to be executed once for your machine
install.packages("tidyverse")
install.packages("scales")
install.packages("stringr")
install.packages("ggthemes")

# Thiese lines need to be executed every time you load R
library("tidyverse")
library("scales")
library("stringr")
library("ggthemes")
```

Cross-tabulations or contingency tables are tables used to describe relationships between two *categorical*variables. This table is also used in the computation of the *Chi-square test of independence*, which you can read more about here.

1 Example Data Set

1.1 Download the Data

We will discuss cross tabulations in the context of an example data set on financial behaviors and financial literacy from The National Financial Capability Study. The study is performed by the FINRA Investor Education Foundation. The survey was administered in 2009, 2012, and 2015. The 2015 study includes responses by more than 27,000 individuals from every demographic group across the United States.

The data set `findata.RData` below includes a subset of the variables and observations from the full study. This subset includes 98 financial and demographic variables on 10,000 individuals across the country.

The code below downloads the data and opens it, creating a data frame object, `df`.

```
load(url("https://murraylax.org/datasets/findata.RData"))
```

In this tutorial, we will focus on two categorical variables, `FinDifficulty` and `LivingSituation`.

1.2 Examining the Data: Financial Difficulty

`FinDifficulty` is an ordinal categorical variable that is a description of how difficult a person finds it to meet their expenses in a typical month.

In R, categorical variables are of class `factor` and ordinal variables are of class `ordered` `factor`. We can confirm the class for `FinDifficulty` with a call to `class()`.

```
class(df$FinDifficulty)
```

```
## [1] "ordered" "factor"
```

R refers to the possible categories a factor can take as *levels*. We can learn the levels for `FinDifficulty`with a call to the function `levels()`

```
levels(df$FinDifficulty)
```

```
## [1] "Very difficult"       "Somewhat difficult"   "Not at all difficult"
```

We can see that there are three categories corresponding to individuals finding it very difficult, somewhat difficult, or not at all difficult to meet their monthly expenses.

We can call the `table()` function to learn how many observations fall into each of these categories.

```
table(df$FinDifficulty)
```

```
## 
##       Very difficult   Somewhat difficult Not at all difficult 
##                 1082                 3859                 4834
```

If is it more useful to see these in percentages of the full data set, we can save the table to an object (which we call `findiff.tb` below) and use that object to create a proportions table with the function, `prop.table()`.

```
findiff.tb <- table(df$FinDifficulty)
prop.table(findiff.tb)
```

```
## 
##       Very difficult   Somewhat difficult Not at all difficult 
##            0.1106905            0.3947826            0.4945269
```

We can see here that approximately 11.1% of the sample finds it very difficult to meet their expenses in a typical month, approximately 39.5% find it somewhat difficult, and 49.4% of people find it not at all difficult.

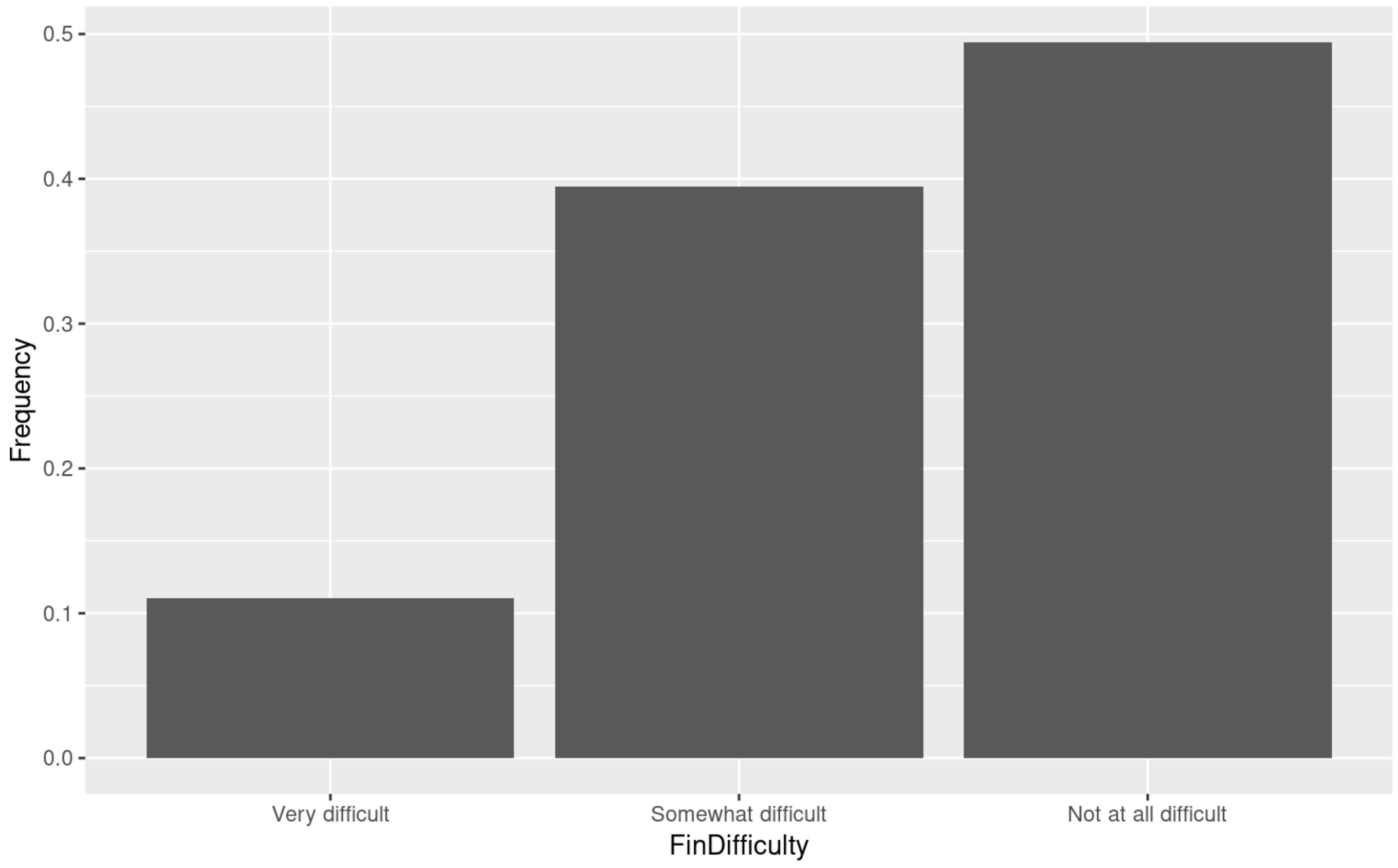
Instead of looking at a table, we can visualize it with a bar graph. The code below uses the `ggplot()`function to plot a proportion table like the one we created above. (Note: This is not the only way, nor possibly even the most common way of creating a bar graph with counts or percentages for categorical variables, but it aligns with the strategy in the subsequent section of this tutorial for visualizing cross tabulations.)

```r
# Save the proportion table in an object
findiff.ptb <- prop.table(findiff.tb)

# Convert the proportion table to a data frame
findiff.df <- as.data.frame(findiff.ptb)

# Converting to a data frame lost the names
names(findiff.df) <- c("FinDifficulty", "Frequency")

# Use the proportion table as the data frame in a call to ggplot()
ggplot(data=findiff.df, mapping=aes(x=FinDifficulty, y=Frequency)) +
  geom_col()
```



In the first line we again compute the proportions for the table information, but this time save the result in `findiff.ptb`.

In the next line we call the function `as.data.frame()` to convert `findiff.ptb` to a data frame object and call this data frame `findiff.df`. We do this because we need to use an object of class `data.frame` in our call to `ggplot()`. An object of class `table` will not work.

In the next line we set the variable names of our data frame to `FinDifficulty` and `Frequency`.

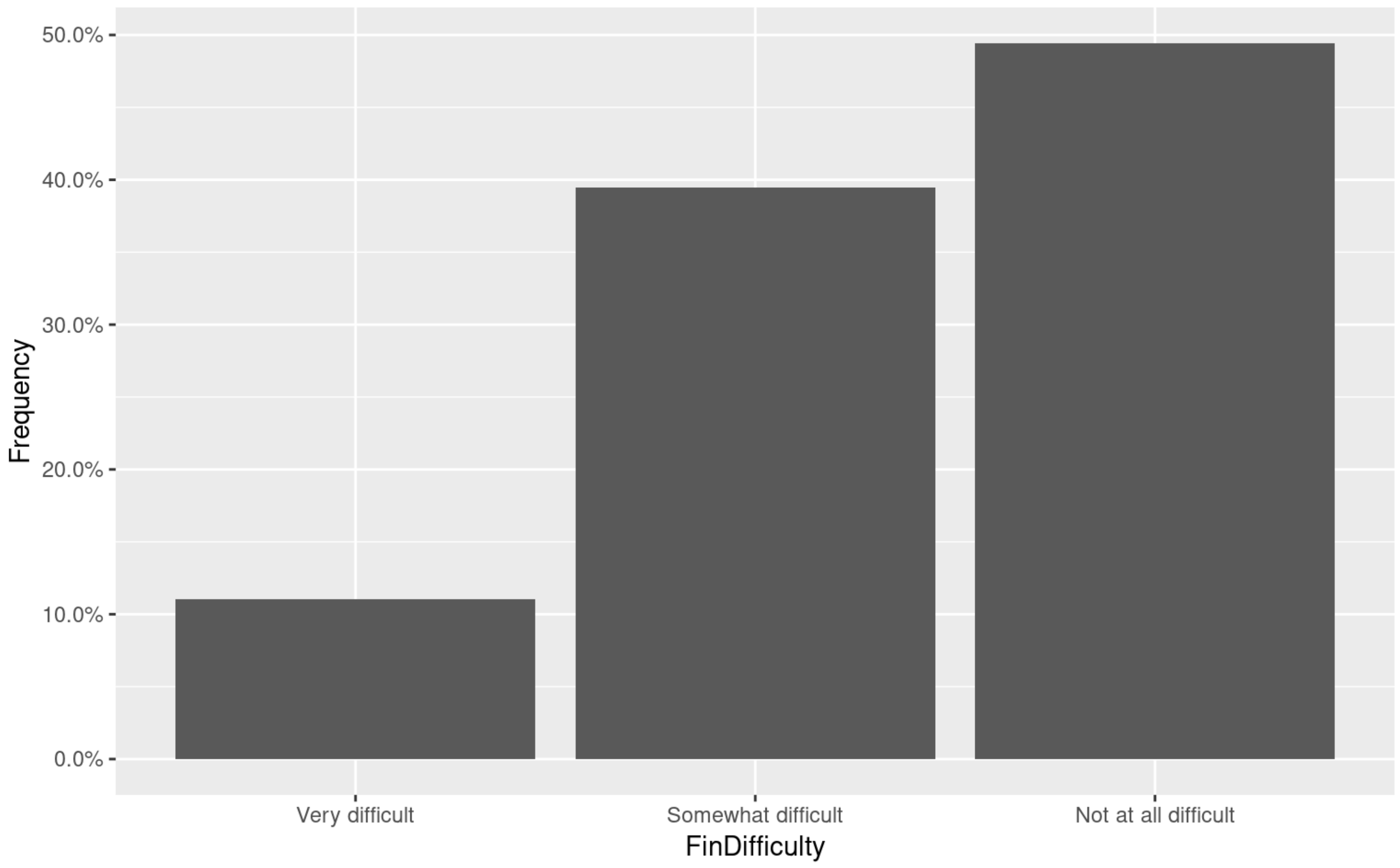
Finally, we call `ggplot()` to create our plot. We set the data layer equal to our newly constructed `findiff.df`. We set the aesthetics layer by mapping the variable `FinDifficulty` to the x-axis and `Frequency` to the y-axis. We add the geometry layer that creates the columns with `geom_col()`.

1.3 Improving the Data Visualization

The plot is useful visualization to the researcher, but it should be cleaned up some if you wanted to present this to an audience.

First, we should change the labels on the vertical axis so that it is clear these proportions are meant to communicate a percentage. We can change the y-scale with a call to `scale_y_continuous()` and change the labels by appropriately setting the `label` parameter to the function.

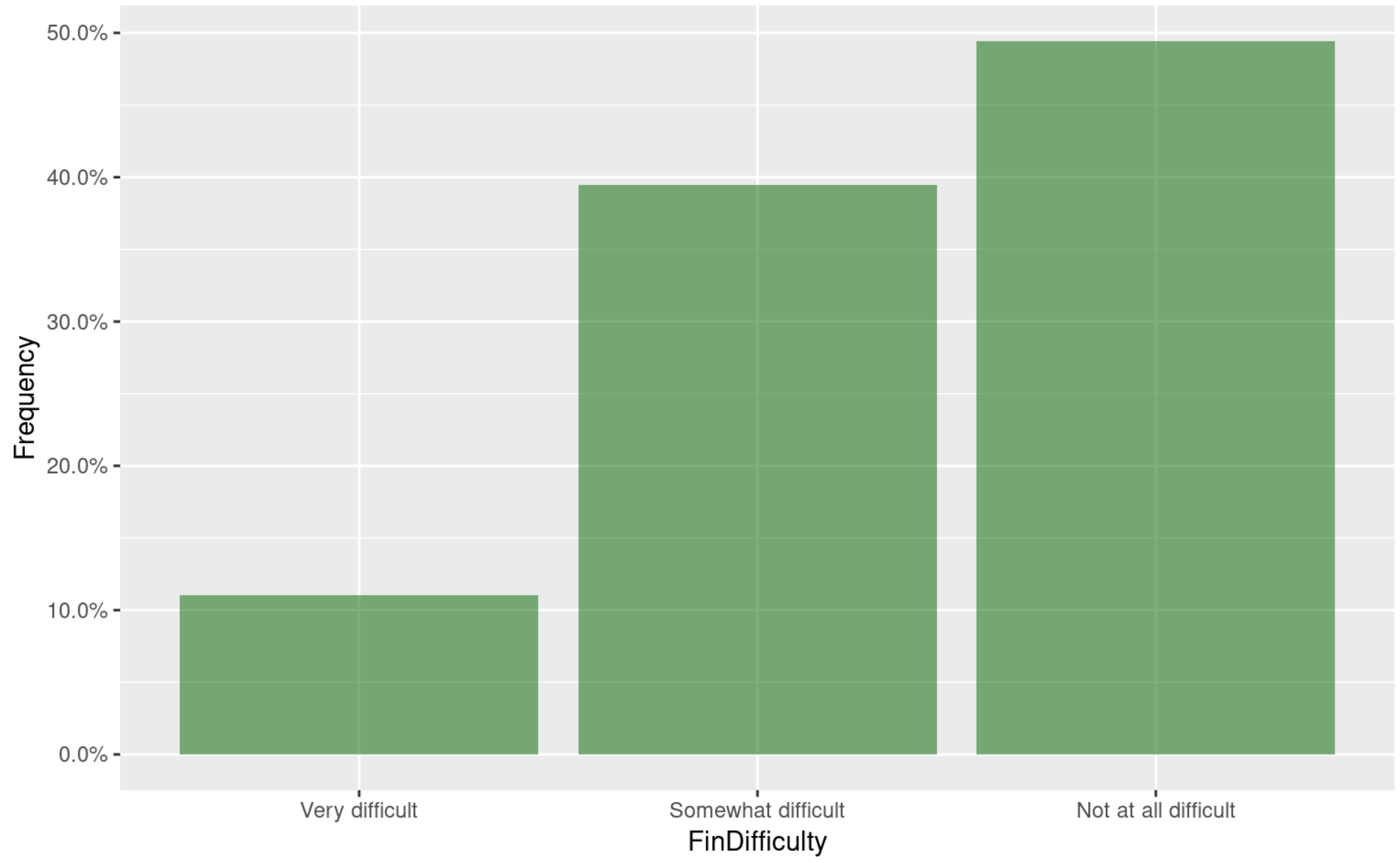
```
ggplot(data=findiff.df, mapping=aes(x=FinDifficulty, y=Frequency)) +
  geom_col() +
  scale_y_continuous(label=percent)
```



The parameter `percent` is actually a function that is included in the `scales` package. It converts fractions to a percent and presents the result with a percent sign.

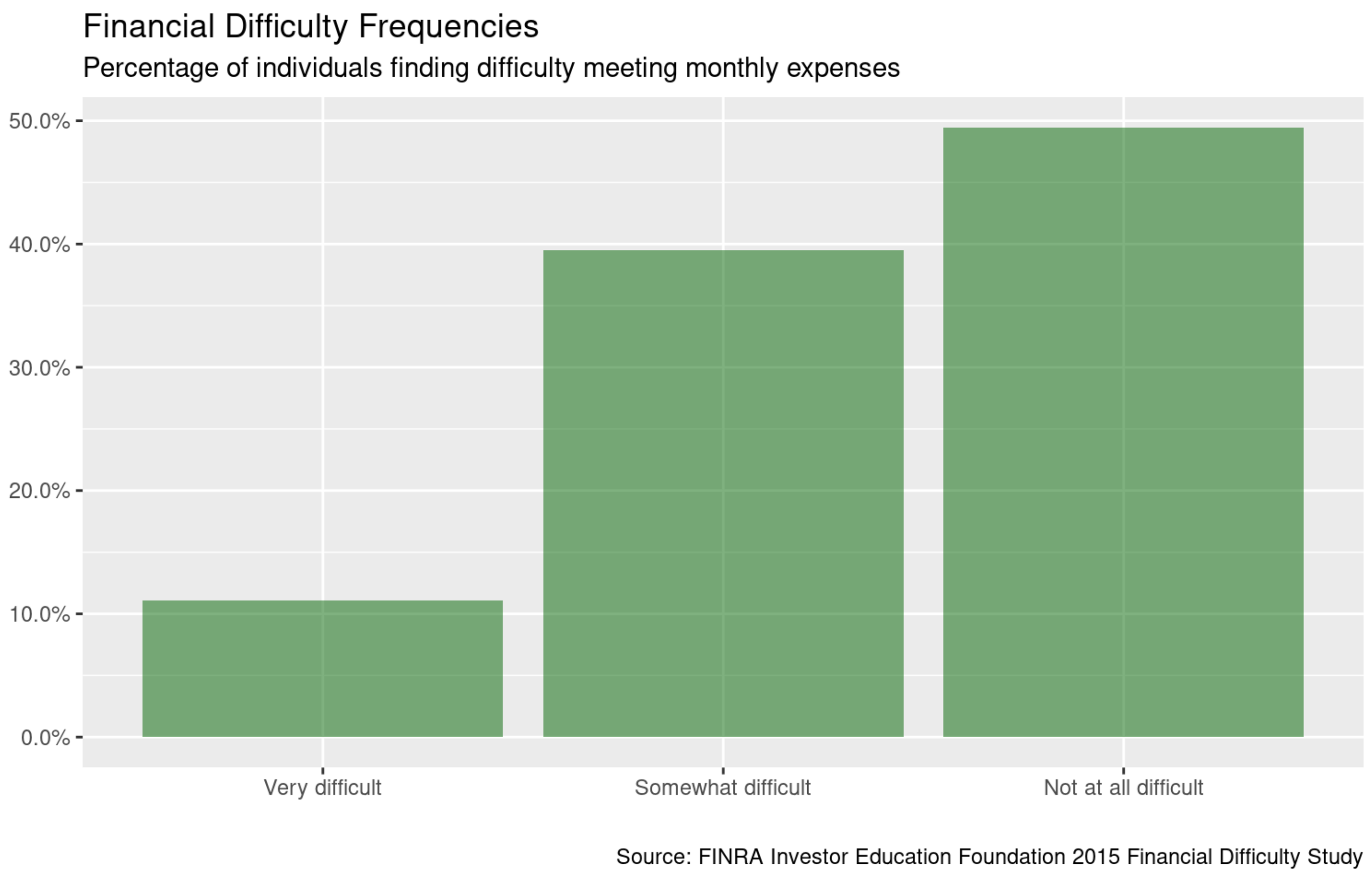
Next we give our graph some color by setting some optional parameters in `geom_col()`. We set `fill="darkgreen"` to change the color of the columns, and we set `alpha=0.5` to make the green bars partially transparent. With partially transparent bars, we can see through the bars to see the grid if necessary.

```r
ggplot(data=findiff.df, mapping=aes(x=FinDifficulty, y=Frequency)) +
  geom_col(fill="darkgreen", alpha=0.5) +
  scale_y_continuous(label=percent)
```



Next, we add some labels with a call to the `labs()` function. We set the title, a subtitle, and a caption to credit our source. We also remove the labels for the x-axis and y-axis because it is obvious from our other descriptions what these axes mean.

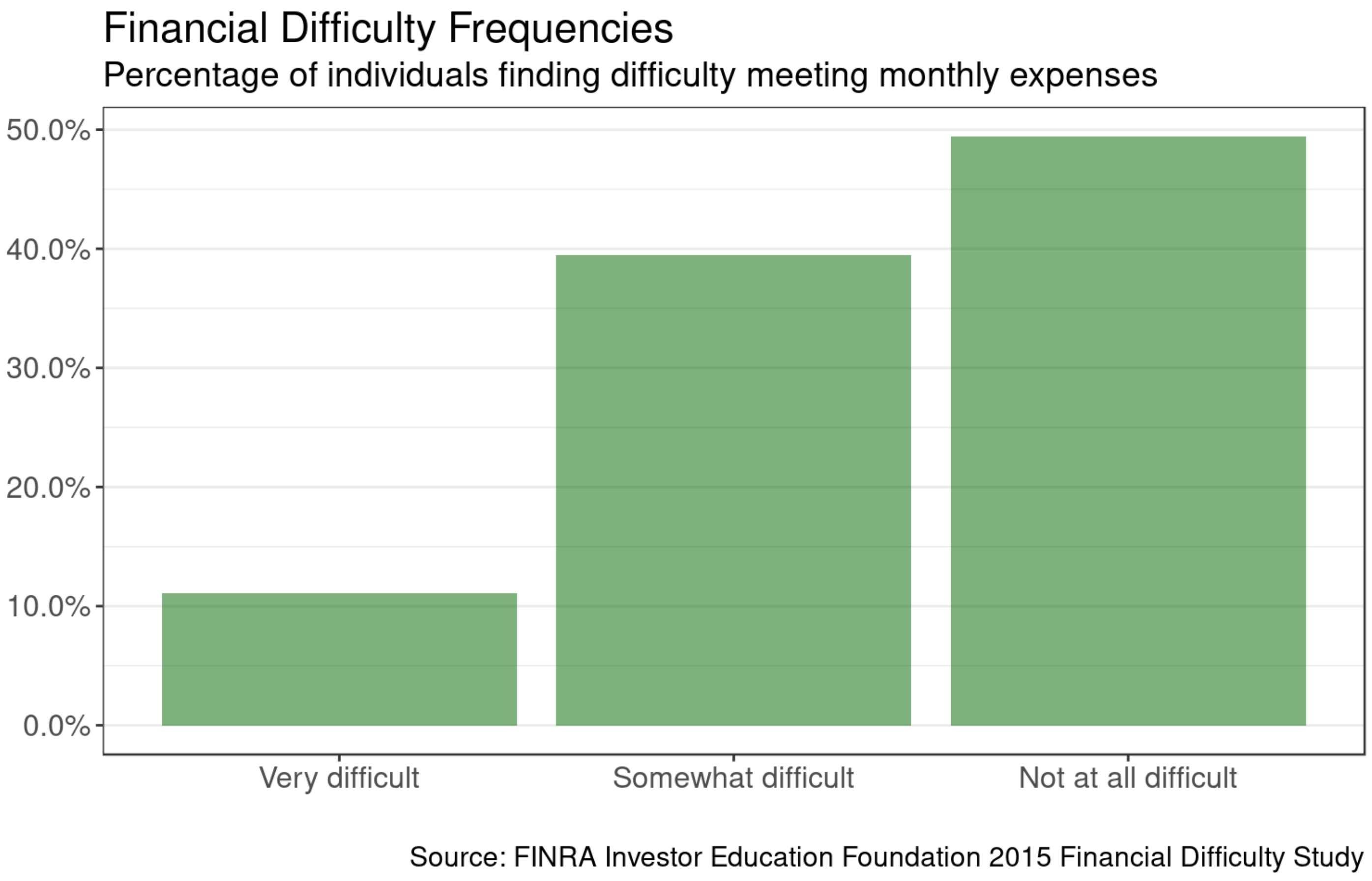
```r
ggplot(data=findiff.df, mapping=aes(x=FinDifficulty, y=Frequency)) +
  geom_col(fill="darkgreen", alpha=0.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequencies",
    subtitle="Percentage of individuals finding difficulty meeting monthly expenses",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    x="", y="")
```



We next call some theme functions. In the code below, we use a black-and-white theme with a call to `theme_bw()` to get rid of the gray background. If this visual is to be used in a PowerPoint presentation, we will want to make the text larger. We do this by calling the `theme()` function and configuring the title and axis text. Finally, we remove the vertical grid lines in a call to the `theme()` function by setting `panel.grid.major.x` to `element_blank()`.

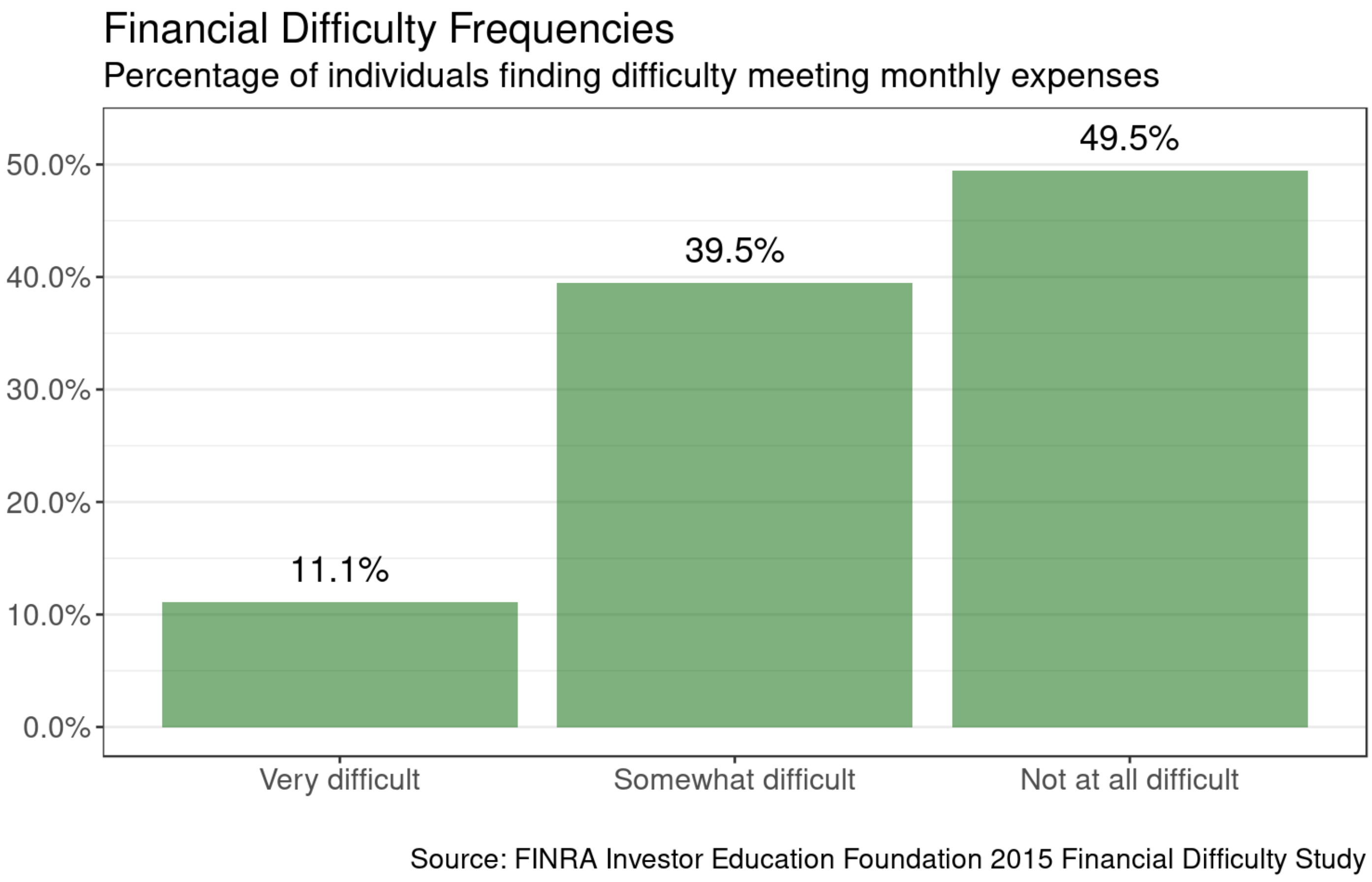
The code below recreates the same graph with a number of other measures to enhance it for effective visual communication.

```r
ggplot(data=findiff.df, mapping=aes(x=FinDifficulty, y=Frequency)) +
  geom_col(fill="darkgreen", alpha=0.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequencies",
    subtitle="Percentage of individuals finding difficulty meeting monthly expenses",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    x="", y="") +
  theme_bw() +
  theme(title=element_text(size=14), axis.text=element_text(size=12)) +
  theme(panel.grid.major.x = element_blank())
```



Finally, let us add the frequencies to the top of the bars with a call to `geom_text()`. In the `geom_text()`call, the parameters `x` and `y` determine where on the plot the text will be placed, the parameter `label`determines what text will be written,
the `size` parameter scales the size of the text (does not correspond perfectly to font sizes), and `nudge_y` tells `geom_text()` to move the text up just a little but above the top of the bar.

```r
ggplot(data=findiff.df, mapping=aes(x=FinDifficulty, y=Frequency)) +
  geom_col(fill="darkgreen", alpha=0.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequencies",
    subtitle="Percentage of individuals finding difficulty meeting monthly expenses",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    x="", y="") +
  theme_bw() +
  theme(title=element_text(size=14), axis.text=element_text(size=12)) +
  theme(panel.grid.major.x = element_blank()) +
  geom_text(mapping=aes(x=FinDifficulty, y=Frequency,
                        label=percent(Frequency)), size=5, nudge_y=0.03)
```



1.4 Examining the Data: Living Situation

The variable `LivingSituation` refers to who, if anyone, that the individual answering the survey lives with. It is a nominal categorical variable. We can confirm the class and learn the levels of the variable with calls to `class()` and `levels()`.

```r
class(df$LivingSituation)
```

```
## [1] "factor"
```

```r
levels(df$LivingSituation)
```

```
## [1] "Alone"               "With Parnter/Spouse" "With Parents"
## [4] "With Roommates"
```

We can see that living situation includes living alone, living with a roommate, living with one’s parents, and living with a spouse.

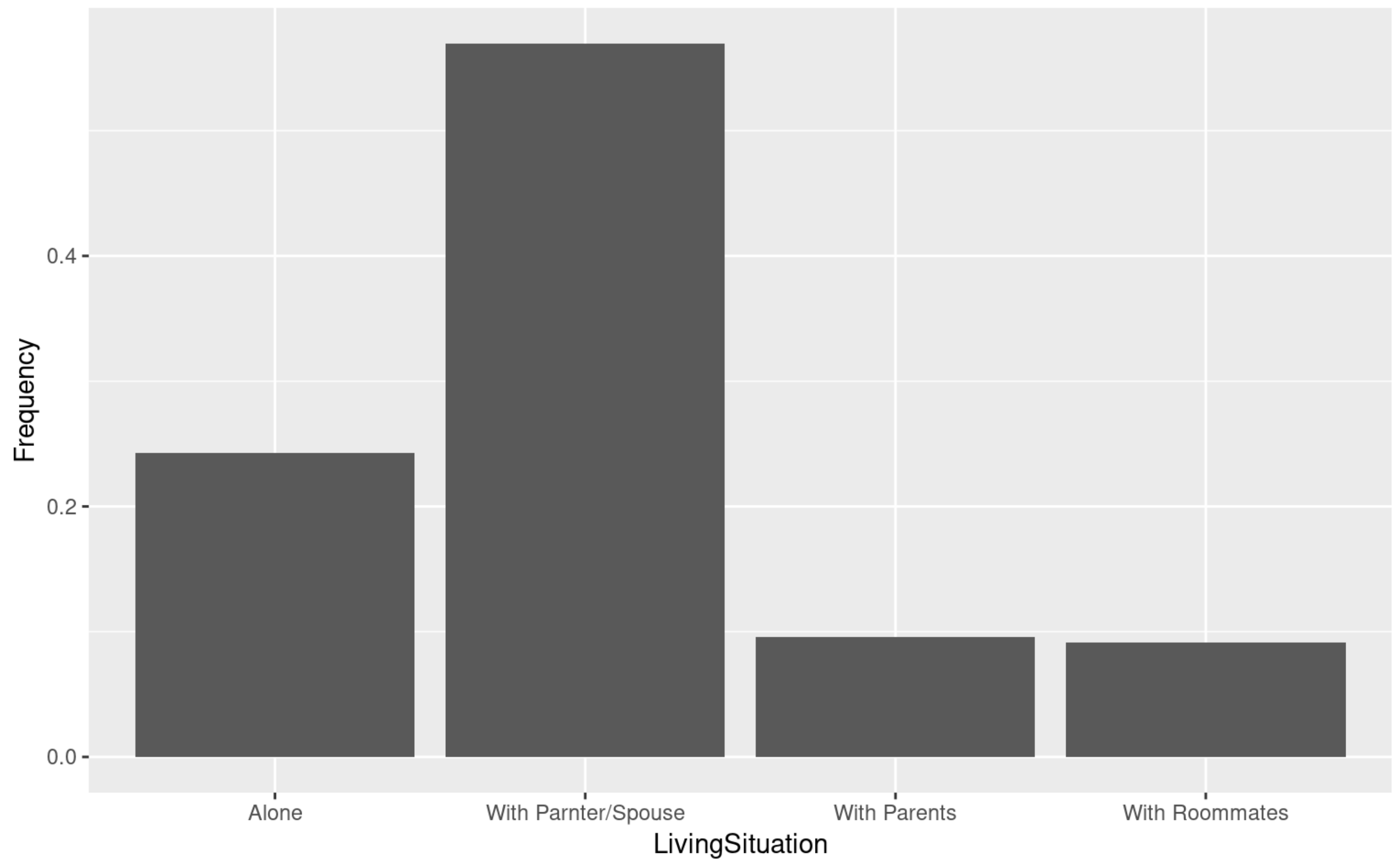
In the code that follows we discover what proportion of our sample is in each living situation.

```r
livingsit.tb <- table(df$LivingSituation)
prop.table(livingsit.tb)
```

```
## 
##               Alone With Parnter/Spouse        With Parents 
##              0.2430              0.5698              0.0955 
##      With Roommates 
##              0.0917
```

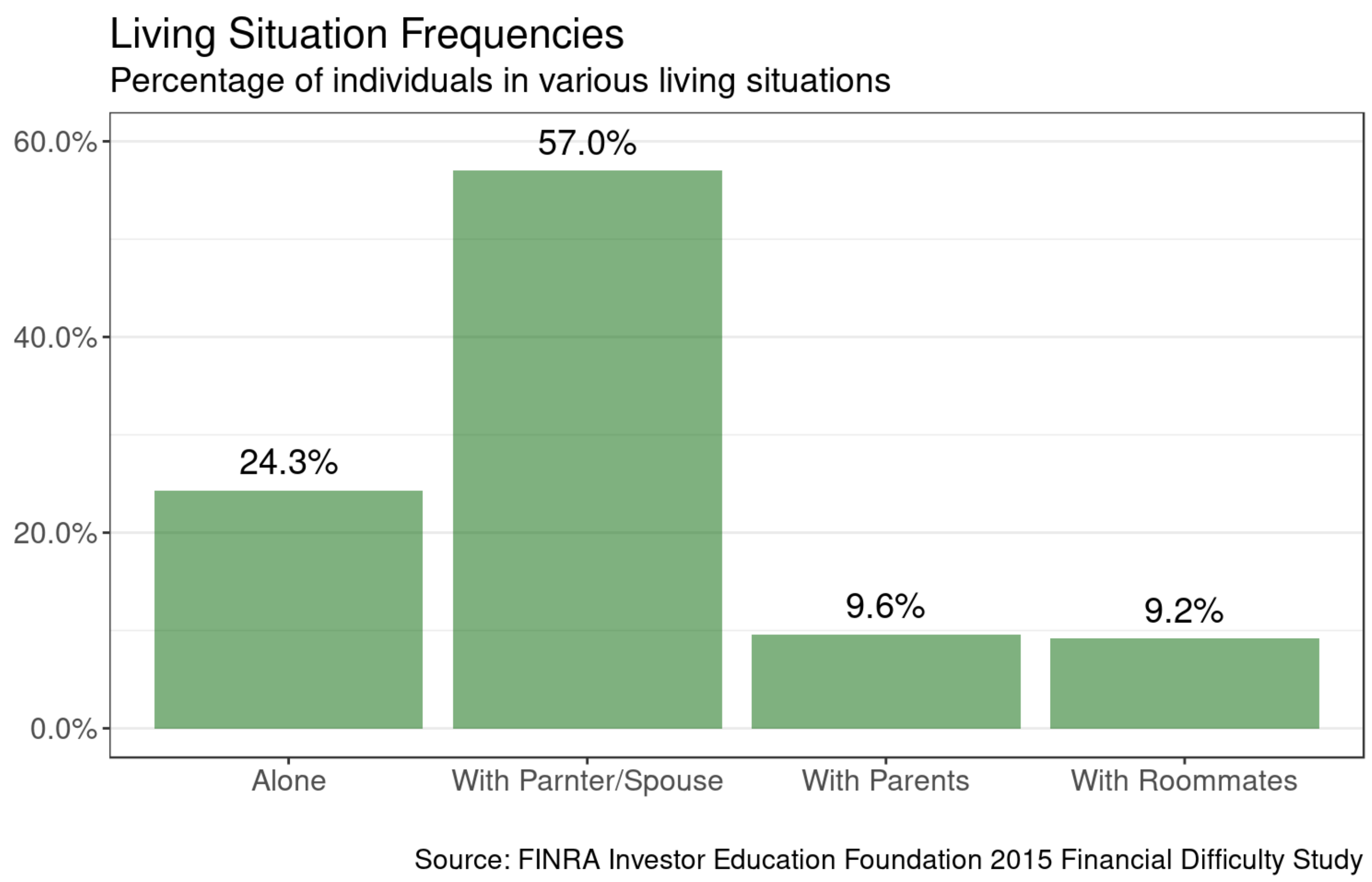
We can make a bar plot for living situation like we did above.

```r
livingsit.ptb <- prop.table(livingsit.tb)
livingsit.df <- as.data.frame(livingsit.ptb)
names(livingsit.df) <- c("LivingSituation", "Frequency")
ggplot(livingsit.df, aes(x=LivingSituation, y=Frequency)) + geom_col()
```



We can configure this plot to make our communication more effective and visually appealing following the same strategies above.

```r
ggplot(data=livingsit.df, mapping=aes(x=LivingSituation, y=Frequency)) +
  geom_col(fill="darkgreen", alpha=0.5) +
  scale_y_continuous(label=percent) +
  labs(title="Living Situation Frequencies",
    subtitle="Percentage of individuals in various living situations",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    x="", y="") +
  theme_bw() +
  theme(title=element_text(size=14), axis.text=element_text(size=12)) +
  theme(panel.grid.major.x = element_blank()) +
  geom_text(mapping=aes(x=LivingSituation, y=Frequency,
                        label=percent(Frequency)), size=5, nudge_y=0.03)
```



2 Contingency Table

2.1 Computing the Contingency Table

A contingency table or cross-tabulation breaks both variables into its categories and reports how many observations in the data set fall into each paired subgroup. We can also use the function `table()` to create a contingency table. We create the contingency table and save the output to an object we call `tb`.

```r
tb <- table(df$LivingSituation, df$FinDifficulty)
tb
```

```
##
##                         Very difficult Somewhat difficult
##    Alone                           344                988
##    With Parnter/Spouse             463               2016
##    With Parents                    131                421
##    With Roommates                  144                434
##
##                         Not at all difficult
##    Alone                                1056
##    With Parnter/Spouse                  3140
##    With Parents                          322
##    With Roommates                        316
```

The table reveals that of the 10,000 individuals in the sample, there are 316 that are living alone and report meeting monthly expenses is typically very difficult, 991 individuals are living along and report meeting monthly expenses is somewhat difficult, etc.

It is usually more convenient to view a contingency tables in terms of row percentages or column percentages. Whether to choose the row depends on what makes more sense in your application and how you ordered your variables in the call to `table()`.

We will take row percentages, that is, *for each living situation*, determine what percentage of those in that living situation finds meeting monthly expenses very difficult, somewhat difficult, or not at all difficult. In the code below, we call the `prop.table()` function, pass in our contingency table, and instruct the function to compute proportions long dimension `1`. We use dimension `1` because `LivingSituation` is the *first* variable in the `table()` call. We save the output to an object we call `tb.prob`.

```
tb.prop <- prop.table(tb, 1)
tb.prop
```

```
##
##                        Very difficult Somewhat difficult
##   Alone                     0.1440536          0.4137353
##   With Parnter/Spouse       0.0823990          0.3587827
##   With Parents              0.1498856          0.4816934
##   With Roommates            0.1610738          0.4854586
##
##                        Not at all difficult
##   Alone                           0.4422111
##   With Parnter/Spouse             0.5588183
##   With Parents                    0.3684211
##   With Roommates                  0.3534676
```

If we look along the first row, we see that *of those that live alone*, approximately 13.4% find it very difficult to meet monthly expenses, 42.1% of people find it somewhat difficult, and 44.5% of people do not find it difficult at all.

If you look in the next row, you can see the story is somewhat different for individuals living with a partner or spouse. Of those, only 8% find it very difficult to meet monthly expenses, 35.7% find it somewhat difficult, and a majority, 56.2%, do not find it difficult at all.

2.2 Testing for Independence: The Chi-Square Test of Independence

In this tutorial we describe the Chi-Square Test of Independence for determining statistical evidence for whether two categorical variables are related to one another.

Our casual examination of the contingency table in the previous subsection suggested that a person's degree of financial difficulty (measured by `FinDifficulty`) may be related to one's living situation (measured by `LivingSituation`).

We can formally test this with a call to `chisq.test()`. In the call below, we save the output to an object we call `cht` which we will use later in the tutorial.

```
cht <- chisq.test(df$LivingSituation, df$FinDifficulty)
cht
```

```
##
##  Pearson's Chi-squared test
##
## data:  df$LivingSituation and df$FinDifficulty
## X-squared = 277.5, df = 6, p-value < 2.2e-16
```

The p-value for the test is less than 2.2×10^{-16}2.2×10−16 (which is machine-zero), which implies we have statistical evidence that there is a relationship between financial difficulty and living situation.

What remains now is to construct some visualizations that can effectively communicate what that relationship is.

3 Visualizing the Cross Tabulation Proportions

3.1 Converting Contingency Table to Data Frame

The `ggplot()` function requires a data frame for creating a plot, so we must first convert our contingency table from a class `table` to a class `data.frame`. In the code that follows, we convert the proportions table to a data frame with the function `as.data.frame()` and examine it with the function `glimpse()` from the tidyverse.

```
tb.df <- as.data.frame(tb.prop)
glimpse(tb.df)
```

```
## Observations: 12
## Variables: 3
## $ Var1 <fct> Alone, With Parnter/Spouse, With Parents, With Roommates, A…
## $ Var2 <fct> Very difficult, Very difficult, Very difficult, Very diffic…
## $ Freq <dbl> 0.1440536, 0.0823990, 0.1498856, 0.1610738, 0.4137353, 0.35…
```

We can see that we have three variables, named `Var1`, `Var2`, and `Freq`. From the first few values, we can see that `Var1` is the living situation, `Var2` is the financial difficulty variable, and `Freq` is the proportion in our contingency table.

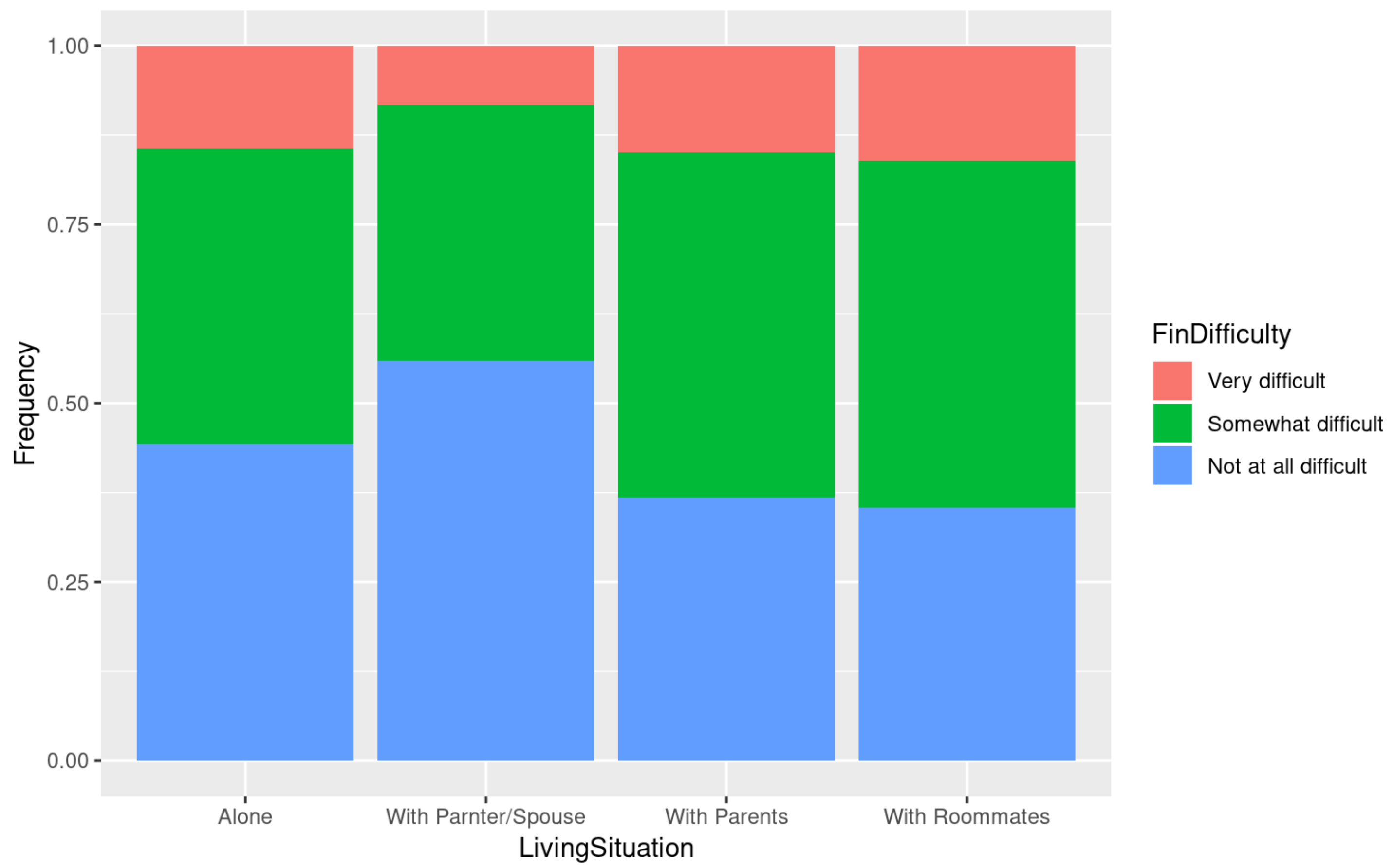
Let us rename these variables appropriately.

```
names(tb.df) <- c("LivingSituation", "FinDifficulty", "Frequency")
```

3.2 Bar Charts

We can make a very ugly stacked-bar chart with our new data frame. This kind of visualization is very common, but not very effective. Still, it makes a good start.

```
ggplot(tb.df, aes(x=LivingSituation, y=Frequency, fill=FinDifficulty)) + geom_col()
```



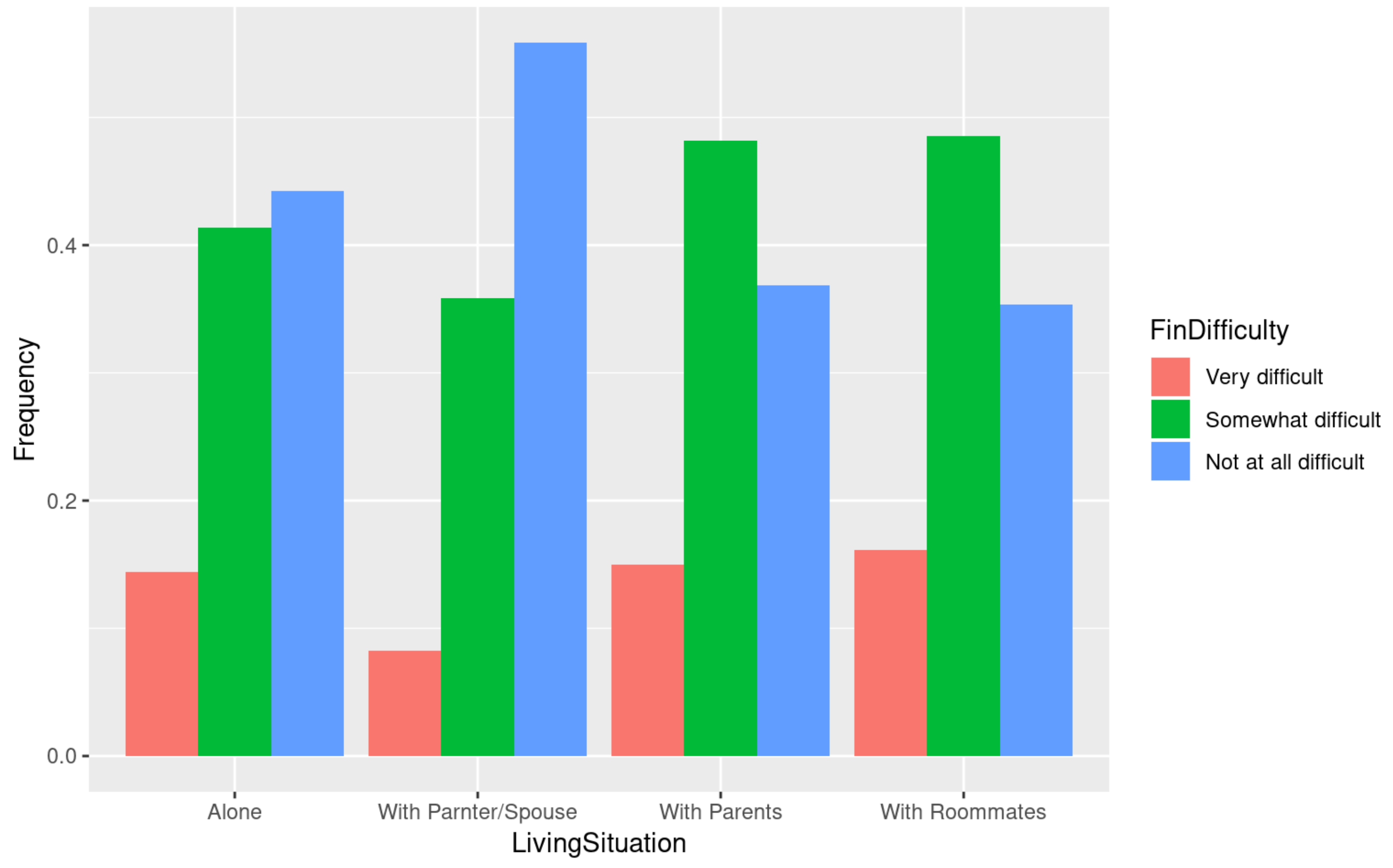
The first parameter to `ggplot()` is the data frame where are contingency table proportions are stored, `tb.df`. The second parameter calls `aes()` to set up the aesthetics layer. We map the x-axis to the variable `Living Situation` to give us the living situation categories along the bottom. We map the y-axis to `Frequency` so that the height of the bars are the proportions. The last aesthetic we set is `fill`, which is the color the bars are filled with. We map `fill` to the variable `FinDifficulty` so each category for financial difficulty is shown with a different color.

A stacked-bar chart stacks the bars on top of one another. This is often convenient when it makes intuitive sense to add up the responses within each x-axis category. In this case, it does makes sense that within each living situation, the proportions add up to 100% (or 1.00).

This stacked bar chart is convenient for comparing the bottom (blue) bars between living situation categories, or even the top (red) bars. It is more difficult to compare the lengths of the middle (green) bars, and it is extremely to eye the difference in the length of the different color bars within a category.

Instead of a stacked-bar chart, we can make a multiple-bar chart. We do this by setting the optional parameter, `position` in the `geom_col()` function. By setting `position="dodge"`, we tell `ggplot()` to not let the columns stack on top of each other (dodge each other).

```r
ggplot(tb.df, aes(x=LivingSituation, y=Frequency, fill=FinDifficulty)) +
  geom_col(position="dodge")
```



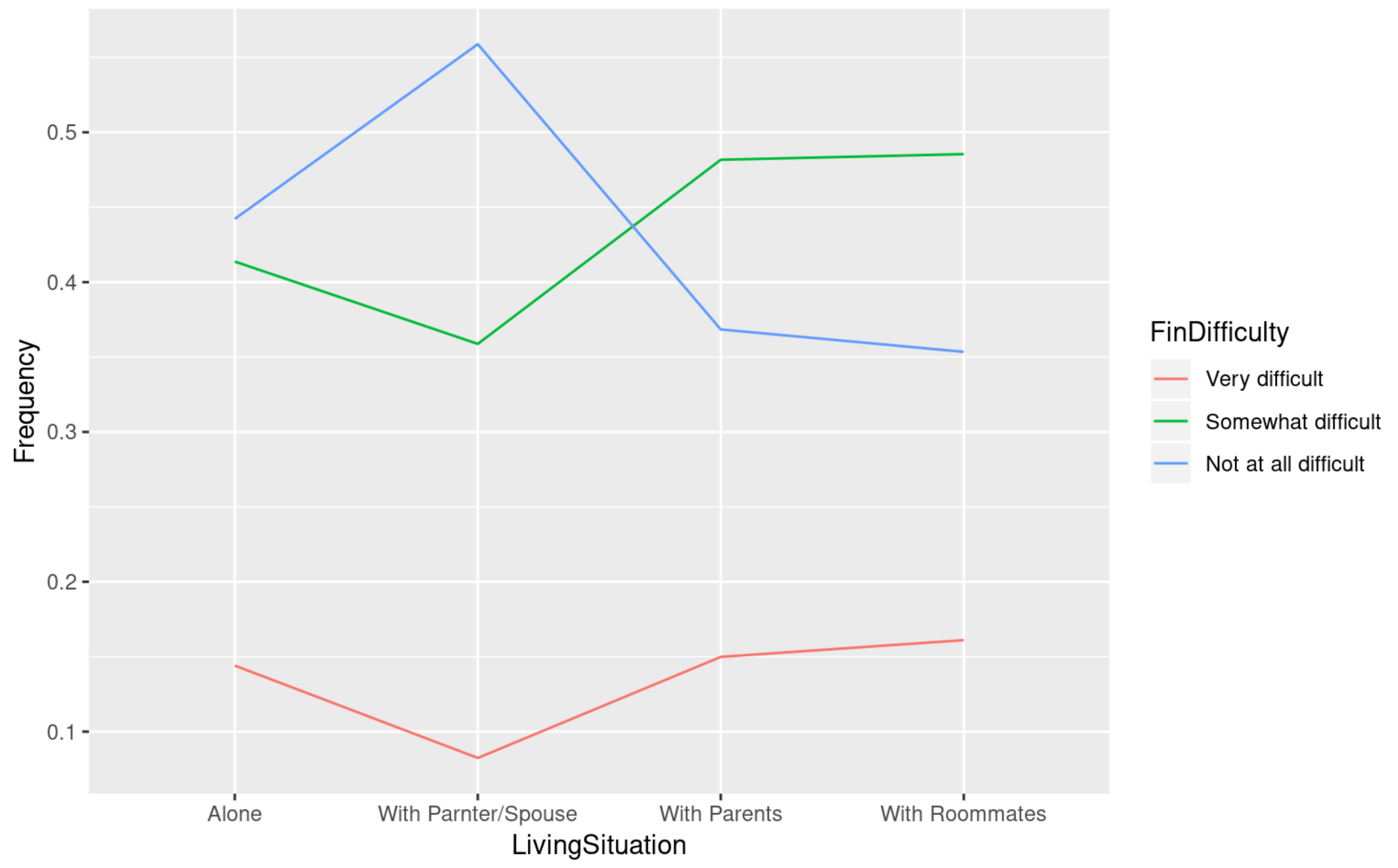
This makes it easier to compare heights of bars within a living situation. For example, we can see among people with partners or spouses, many more of them have no difficulty meeting monthly expenses (blue bar) than finding it somewhat difficult (green bar). That situation is reversed for people living with parents.

3.3 Line Plots

Generally, line plot are reserved for time-series variables, where time moves forward as you move to the right along the x-axis, and the y-axis shows the value the a variable takes over time.

However, time plots are often useful for comparing across categories, for multiple categorical variables. The code below creates another plot with many of the same aesthetics as above.

```r
ggplot(tb.df, aes(x=LivingSituation, y=Frequency, color=FinDifficulty, group=FinDifficulty))
  geom_line()
```



Instead of a `fill` aesthetic, we have a `color` aesthetic for the color of the line. We map `FinDifficulty` to color. We also have a `group` aesthetic that we also map to `FinDifficulty`. The `group` aesthetic is used to draw different lines, instead of having all the points connect is a single line. We want a separate line for each `FinDifficulty` category, so we set `group=FinDifficulty`.

Finally, instead of calling `geom_col()` to create columns, we call `geom_line()` to create lines.

This line plot is useful for a researcher, and with some work, can be useful to communicate to an audience. The red line on the bottom illustrates the proportions of people that find it very difficult to meet their monthly finances. You can see that these have lower proportions than the other two categories (somewhat and not-at-all). You can also follow it with ease to see with what living situations it is highest (living with roommates) and where it is lowest (living with a partner/spouse).

The blue line illustrates those who have no difficulty, you can see that with two living situations, the blue line is highest (alone and with partner/spouse), and with two situations is is the second most common (with parents and with roommates).

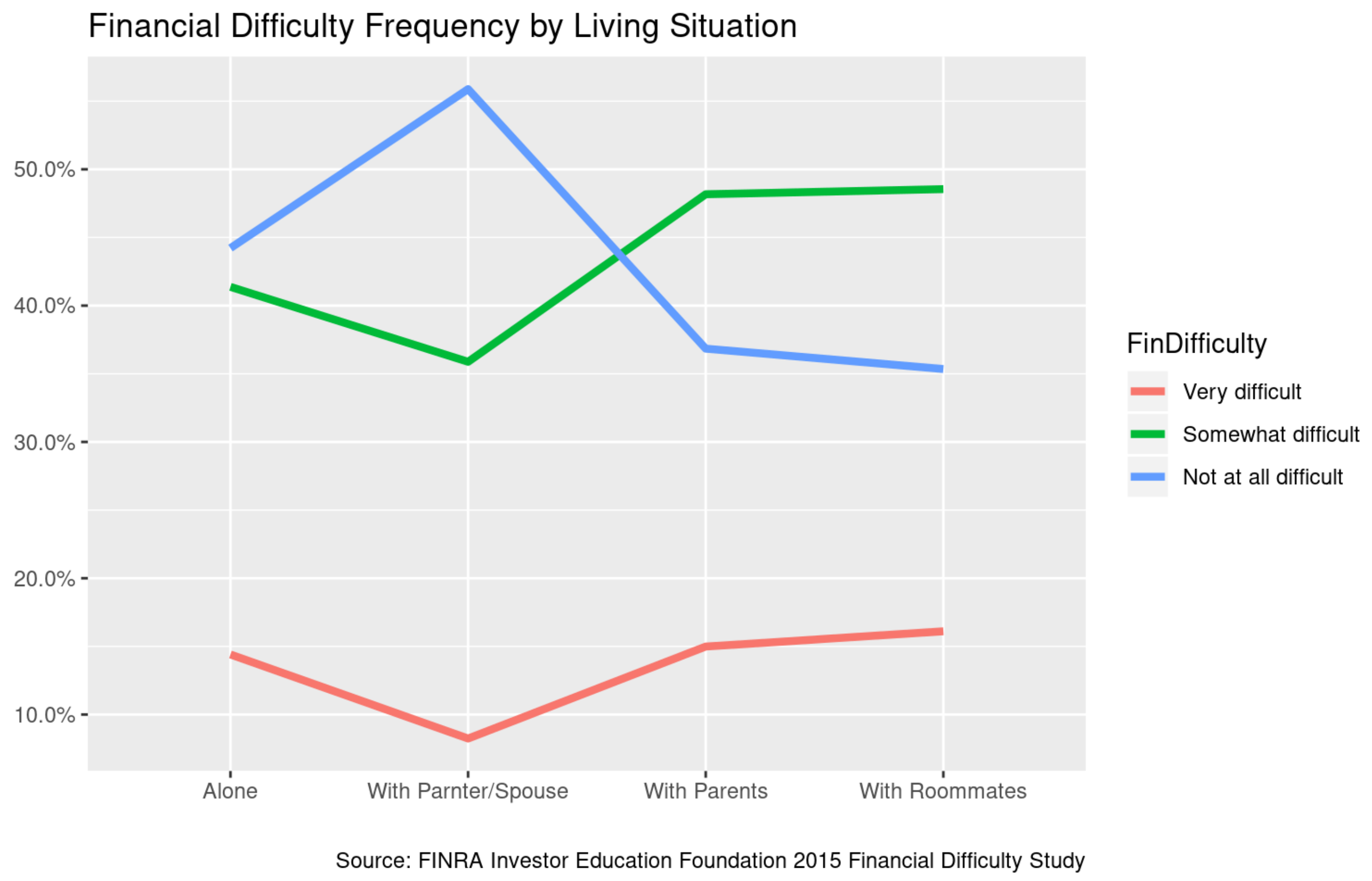
That the lines cross are not all horizontal, and that they sometimes cross, are visual clues that there is a relationship between living situation and financial difficulty. If there were no relationship, we would expect the lines to be of possibly different heights, but to not change height with each living situation category. If there were no relationship, then we should see approximately the same percentage of people having financial difficulty regardless of their living situation.

3.4 Improving the Data Visualization

Some Basics: Thicker Lines, Titles, and Labels

Let us make the lines thicker, make the y-axis labels percentages, and give some descriptive titles and labels:

```r
ggplot(tb.df, aes(x=LivingSituation, y=Frequency,
                  color=FinDifficulty, group=FinDifficulty)) +
  geom_line(size=1.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequency by Living Situation",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    x="", y="")
```



Include the Chi-Square Test Results

In Section 2.2, we computed the Chi-Square Test of Independence and found out that there is statistically significance evidence of a relationship between financial difficulty and living situation. We saved the results in an object we called `cht`. In the code below, we remind ourselves of this object, and view the `statistic` object and `p.value` object with it.

```
cht
```

```
##
##  Pearson's Chi-squared test
##
## data:  df$LivingSituation and df$FinDifficulty
## X-squared = 277.5, df = 6, p-value < 2.2e-16
```

```
cht$statistic
```

```
## X-squared
##  277.4957
```

```r
cht$p.value
```

```
## [1] 5.398428e-57
```

Let us report this statistical evidence in a subtitle of our graph. In the code below, we use the `sprintf()` function to create a formatted string. We save the string in an object that we call, `sbtitle`.

```r
sbtitle <-
  sprintf("Chi-Square Test of Independence: Test Statistic = %.1f, P-value = %.3f",
          cht$statistic,
          cht$p.value)
```

The first parameter to `sprintf()` is our text, which includes the format codes, `%.1f` and `%.3f`. The '%' indicates what follows is going to be a format code, the 'f' in these codes means to print a floating point number (i.e. a number with one or more decimal places), the number after the decimal point indicates how many decimal places to show.

The numbers that are shown are the parameters that follow. The first number that will be shown rounded to one decimal point will be the result of `cht.statistic`. The second number that will be shown rounded to three decimal points will be the result of `cht.p.value`.

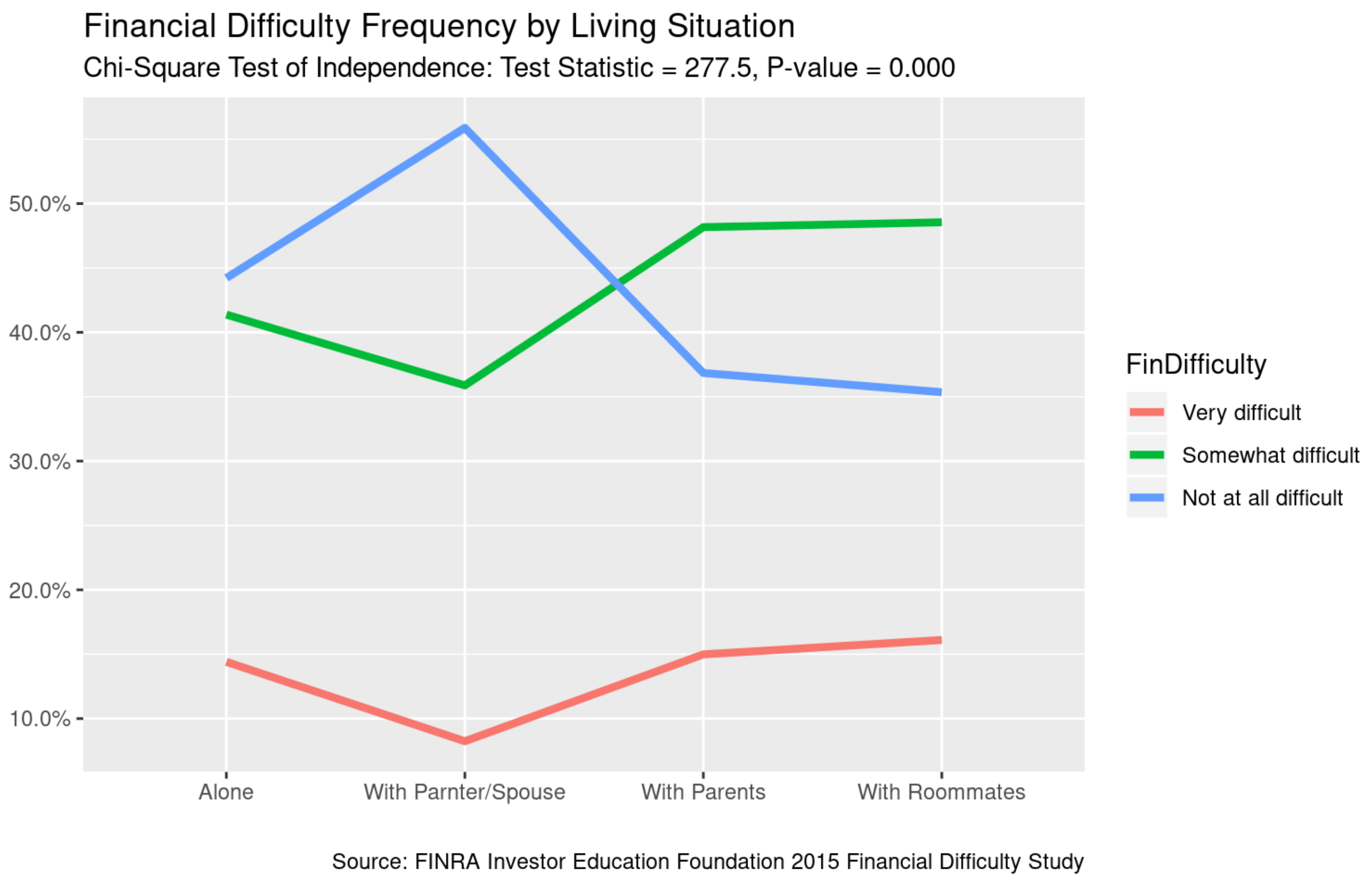
Let us look at the string we created.

```r
sbtitle
```

```
## [1] "Chi-Square Test of Independence: Test Statistic = 277.5, P-value = 0.000"
```

Let us make this string our subtitle. We recreate the `ggplot()` code above, this time adding a parameter for `subtitle` in the `labs()` function.

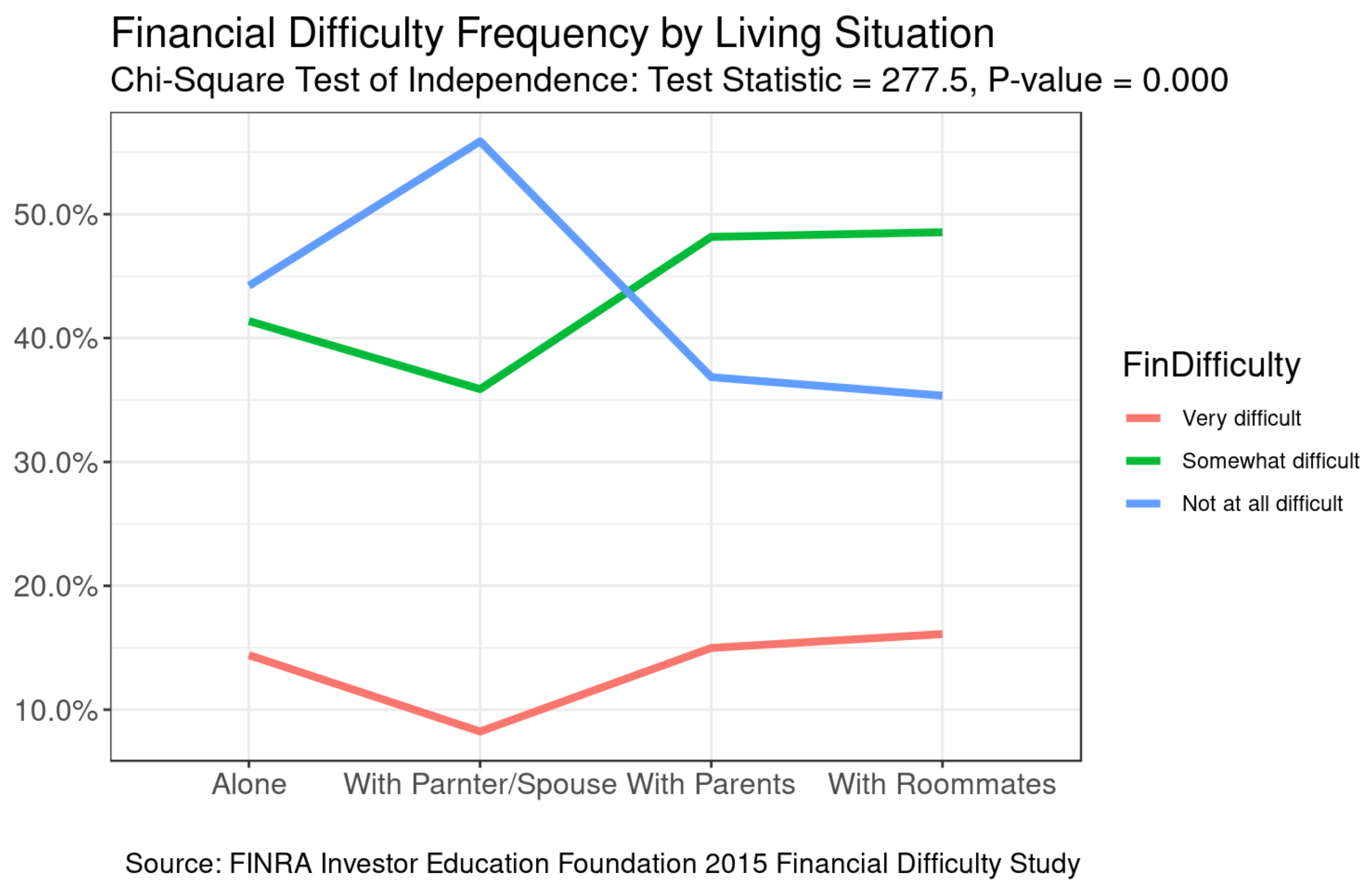
```
ggplot(tb.df, aes(x=LivingSituation, y=Frequency,
                  color=FinDifficulty, group=FinDifficulty)) +
  geom_line(size=1.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequency by Living Situation",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    subtitle=sbtitle,
    x="", y="")
```



Themes: Background and text-size

We can alter the theme so that the background is white instead of gray, and that the text is larger, which can be useful if the graph is to be used for presentation slides.

```r
ggplot(tb.df, aes(x=LivingSituation, y=Frequency,
                  color=FinDifficulty, group=FinDifficulty)) +
  geom_line(size=1.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequency by Living Situation",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    subtitle=sbtitle,
       x="", y="") +
  theme_bw() +
  theme(title=element_text(size=14), axis.text=element_text(size=12))
```

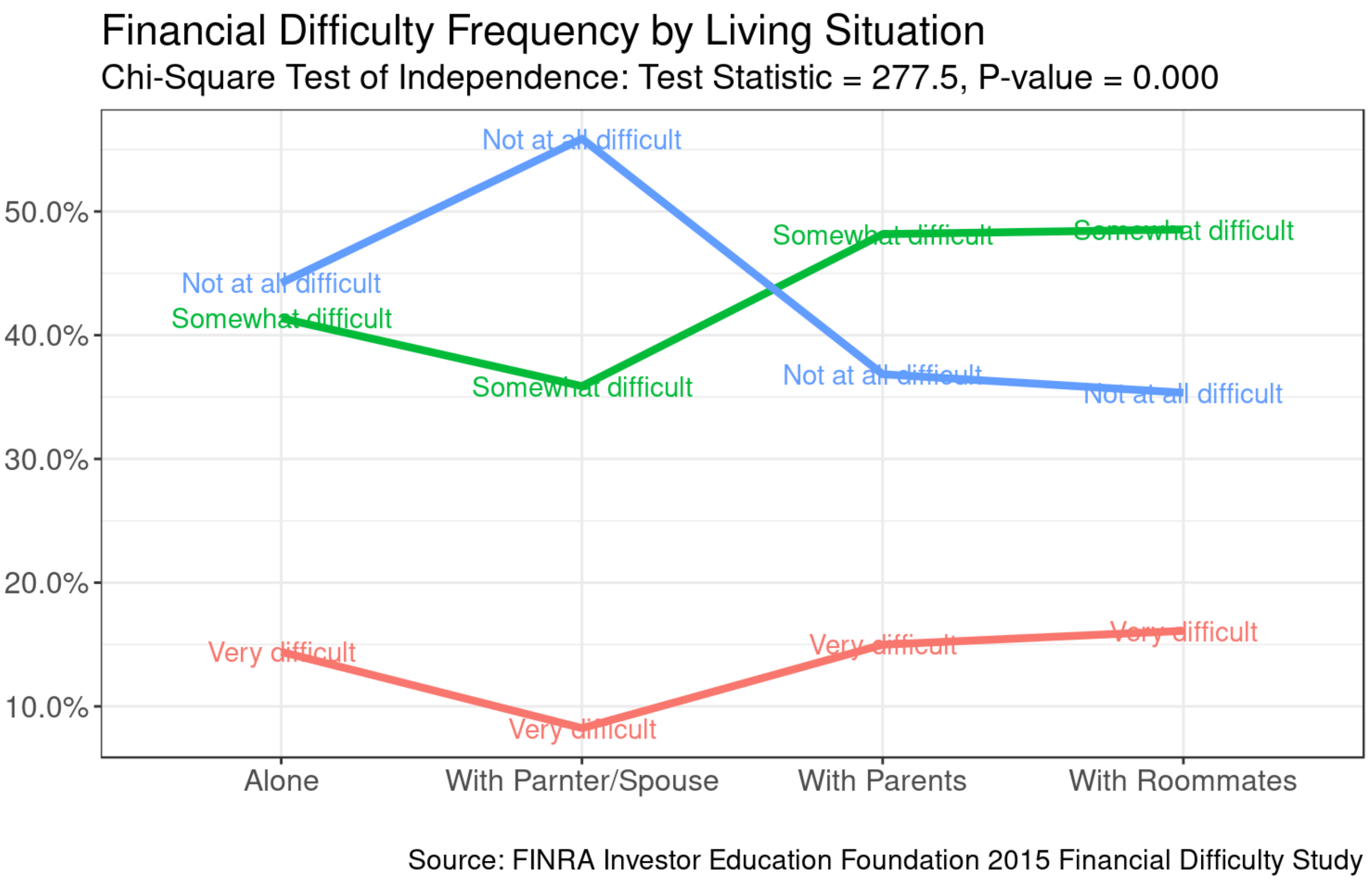


Avoid Legends if Possible

The legend takes up a lot of space that could be used to illustrate the data. It also takes time for the audience to move back and forth between a legend and a graph. This is especially problematic in oral presentations, but less problematic in written communication. While legends reveal important information, there may be more effective ways of communicating the same information.

Let us try to remove the legend, and embed the labels directly on the graph next to the lines with `geom_text()`. In the code below, the call to `theme(legend.position="none")` removes the legend, and `geom_text()` places this information on the graph.

```
ggplot(tb.df, aes(x=LivingSituation, y=Frequency,
                  color=FinDifficulty, group=FinDifficulty)) +
  geom_line(size=1.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequency by Living Situation",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    subtitle=sbtitle,
       x="", y="") +
  theme_bw() +
  theme(title=element_text(size=14), axis.text=element_text(size=12)) +
  theme(legend.position = "none") +
  geom_text(aes(x=LivingSituation, y=Frequency, label=FinDifficulty))
```



That is not quite what we wanted. We have labels next to the lines, but more labels than we wanted. Let us look at the call to `geom_text()` in the last line and see if we understand what happened.

We set the aesthetics in the call to `aes()` within `geom_text()`. The parameters `x=LivingSituation` and `y=Frequency` tells `geom_text()` the location to place the labels. In this case, we place the labels at every point on the graph. The label we placed is given in the variable `FinDifficulty`.

This is almost what we want, but we only want each label to appear once. How about we put labels only near the end of the line, that is only near the “With Roommates” category.

Let us create a new data frame that is only for our labels. We will take our existing data frame `tb.df` and create from that a new data frame that only includes the rows for “With Roommates”.

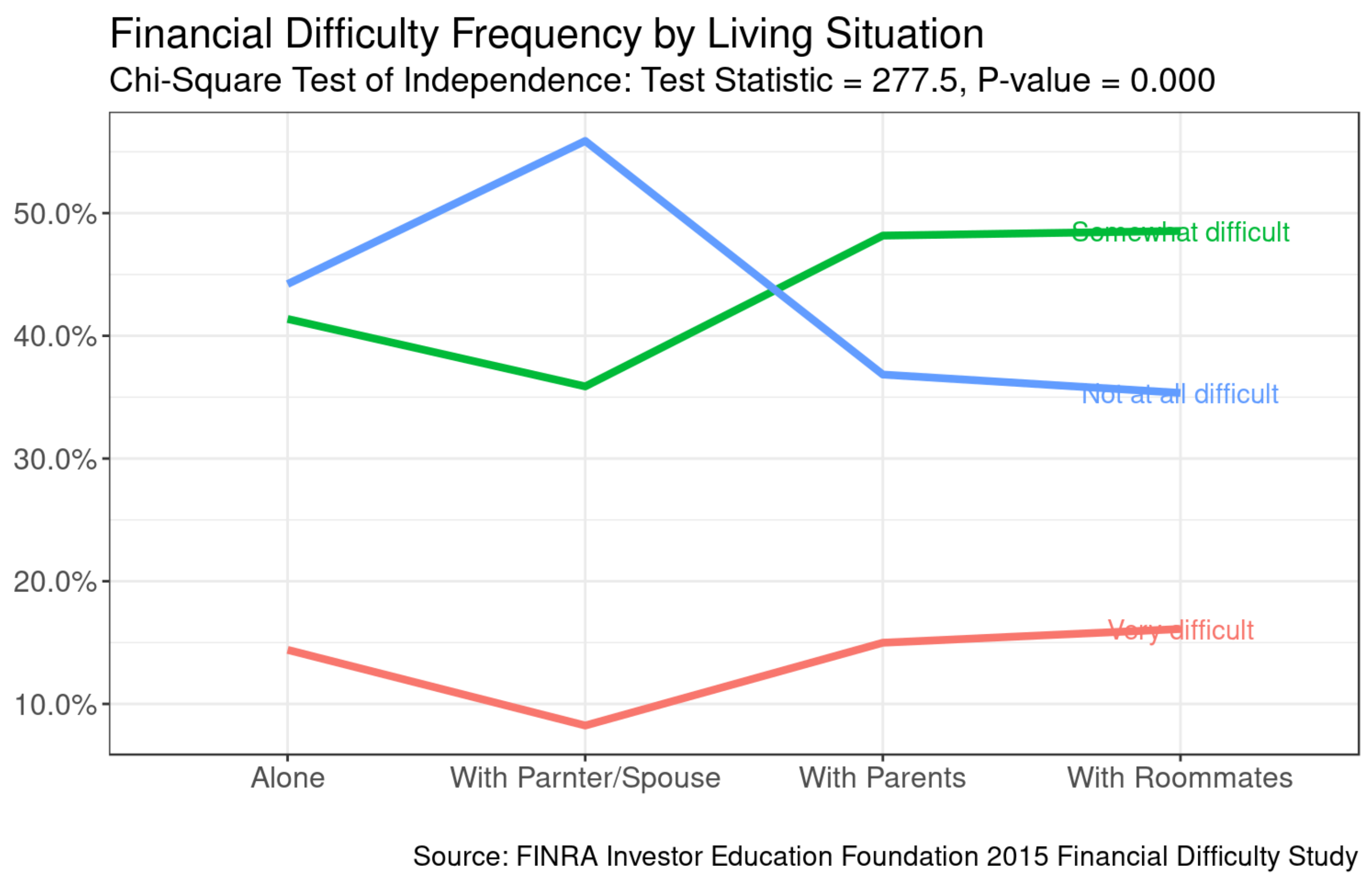
```
labels.df <- tb.df %>% filter(LivingSituation=="With Roommates")
labels.df
```

```
##   LivingSituation        FinDifficulty Frequency
## 1  With Roommates       Very difficult 0.1610738
## 2  With Roommates   Somewhat difficult 0.4854586
## 3  With Roommates Not at all difficult 0.3534676
```

You can see that our new data frame includes only three rows, the rows where `Living Situation="With Roomates"`.

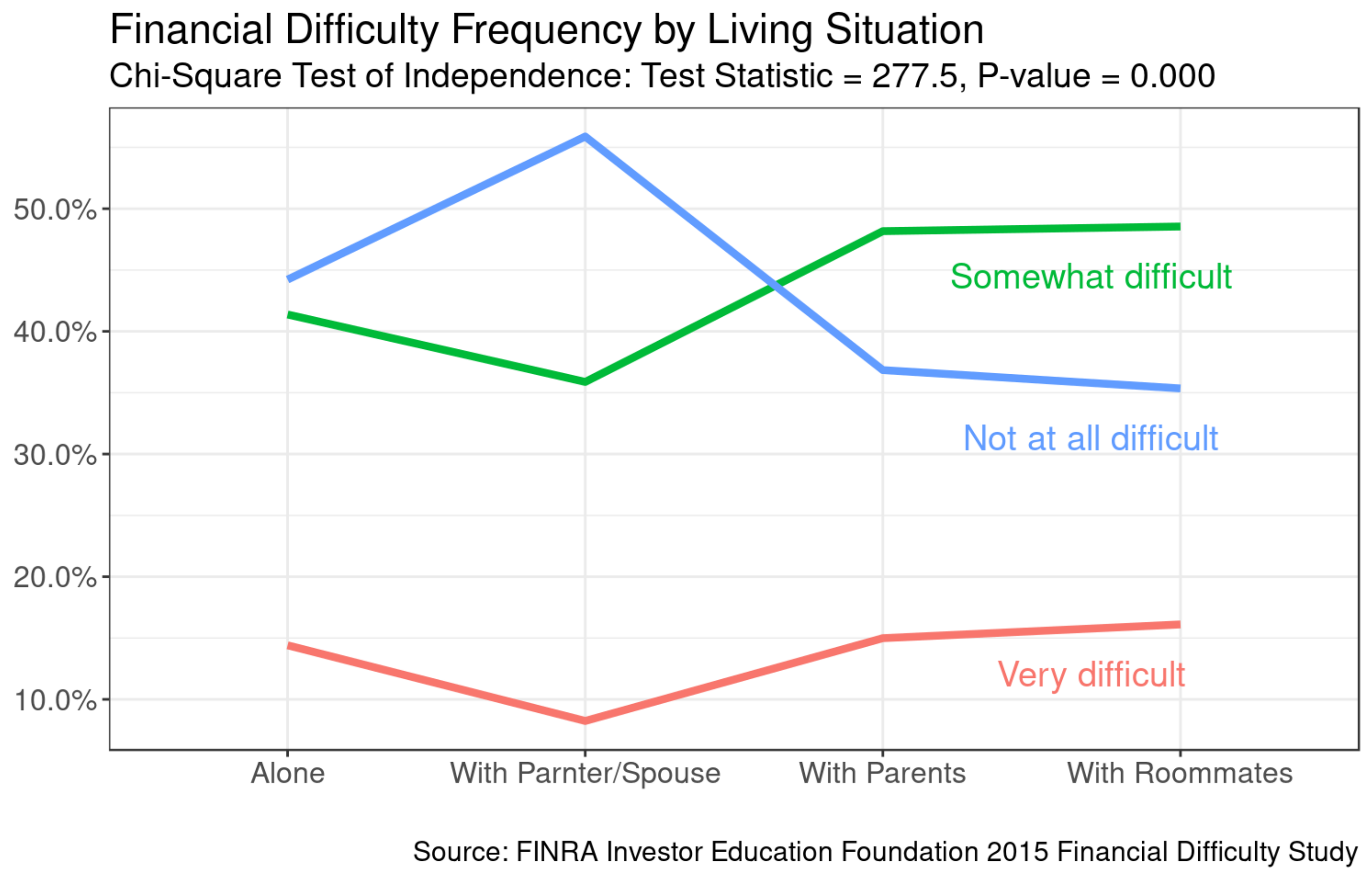
Now, let us repeat the code to plot above, put this time, use only the data frame `labels.df` in the `geom_text()` call.

```
ggplot(tb.df, aes(x=LivingSituation, y=Frequency,
                  color=FinDifficulty, group=FinDifficulty)) +
  geom_line(size=1.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequency by Living Situation",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    subtitle=sbtitle,
    x="", y="") +
  theme_bw() +
  theme(title=element_text(size=14), axis.text=element_text(size=12)) +
  theme(legend.position = "none") +
  geom_text(data=labels.df, aes(x=LivingSituation, y=Frequency, label=FinDifficulty))
```



That looks closer to what we wanted. Let us make the text a little bigger, and nudge the text down a little bit (with `nudge_y`) and nudge the text to left a little bit (with `nudge_x`)

```
ggplot(tb.df, aes(x=LivingSituation, y=Frequency,
                  color=FinDifficulty, group=FinDifficulty)) +
  geom_line(size=1.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequency by Living Situation",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    subtitle=sbtitle,
    x="", y="") +
  theme_bw() +
  theme(title=element_text(size=14), axis.text=element_text(size=12)) +
  theme(legend.position = "none") +
  geom_text(data=labels.df, aes(x=LivingSituation, y=Frequency, label=FinDifficulty),
            size=5, nudge_y = -0.04, nudge_x = -0.3)
```

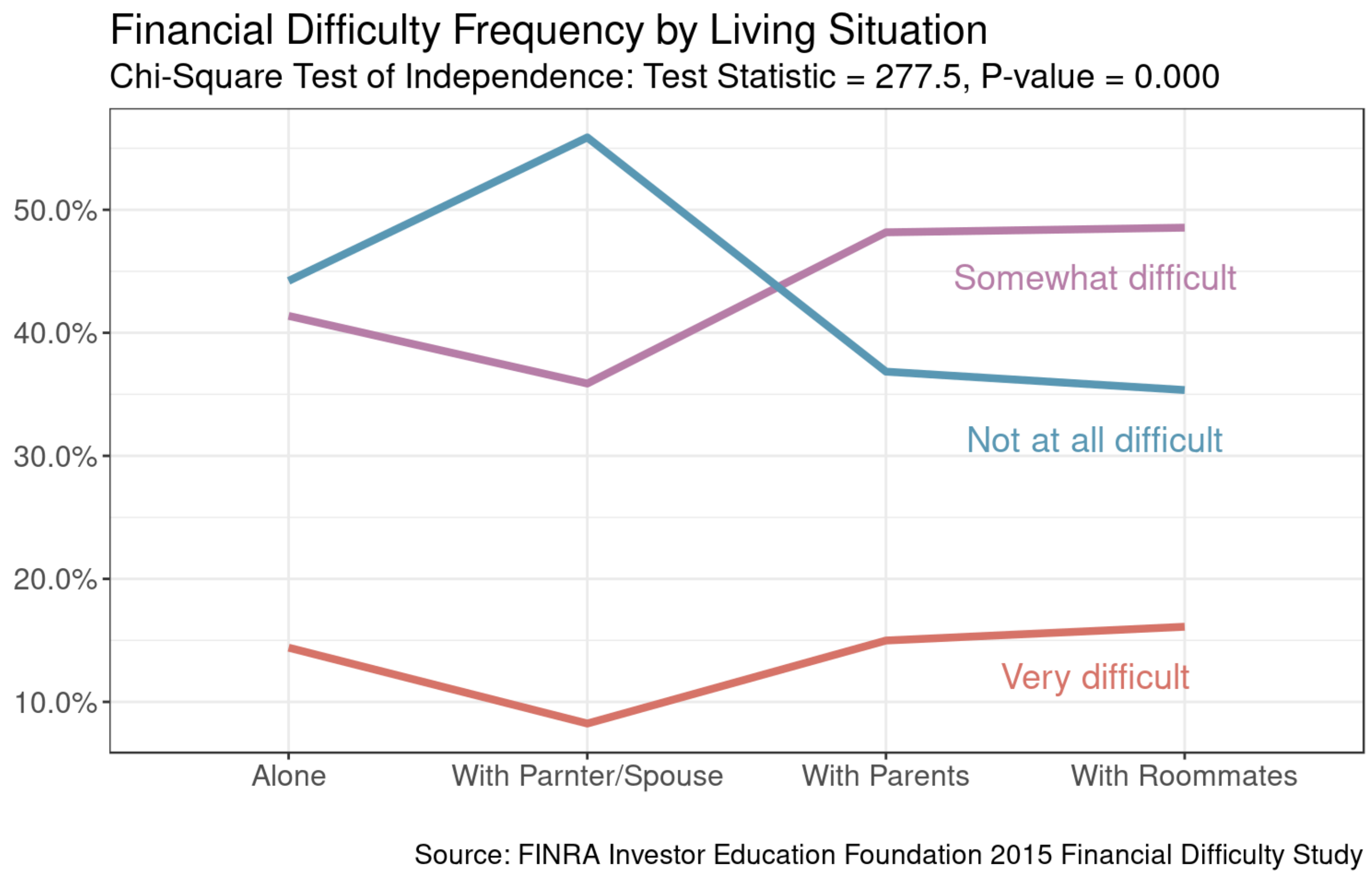


Change Colors

Finally, we can change the color scale if we are interested in doing so. It is best to avoid both red and green in the same visual for those who are colorblind. With a call to `scale_color_manual()`, we can manually set some colors. I obtained the color codes below using the colorpicker tool,

http://tristen.ca/hcl-picker/#/hlc/3/1/D67368/5997B3.

```r
ggplot(tb.df, aes(x=LivingSituation, y=Frequency,
                  color=FinDifficulty, group=FinDifficulty)) +
  geom_line(size=1.5) +
  scale_y_continuous(label=percent) +
  labs(title="Financial Difficulty Frequency by Living Situation",
    caption="Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study",
    subtitle=sbtitle,
    x="", y="") +
  theme_bw() +
  theme(title=element_text(size=14), axis.text=element_text(size=12)) +
  theme(legend.position = "none") +
  geom_text(data=labels.df, aes(x=LivingSituation, y=Frequency, label=FinDifficulty),
          size=5, nudge_y = -0.04, nudge_x = -0.3) +
  scale_color_manual(values=c("#D67368", "#B67DA7", "#5997B3"))
```


Financial Difficulty Frequency by Living Situation
Chi-Square Test of Independence: Test Statistic = 277.5, P-value = 0.000
Somewhat difficult
Not at all difficult
Very difficult
Alone
With Parnter/Spouse
With Parents
With Roommates
Source: FINRA Investor Education Foundation 2015 Financial Difficulty Study