Homework 01 - getting started

Melinda K. Higgins, PhD.

August 30, 2017

This document provides some suggested code and examples to get started on your homework 01.

Get the data

The following code uses the `tidyverse` packages and workflow. As such these examples load (using the `library()` command) these packages:

- `tidyverse`
- `readxl`
- `forcats`

```
# load tidyverse and readxl packages
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
library(readxl)

# used read_excel() from the readxl package
dat <- read_excel("Dataset_02_fixq2.xlsx")
```

do a basic summary

```
# the summary() function gives you some basic summary stats
summary(dat)
```

```
##    SubjectID          Age          WeightPRE       WeightPOST
##  Min.   : 1.00   Min.   :24.00   Min.   : 60.0   Min.   :108.0
##  1st Qu.: 5.75   1st Qu.:35.75   1st Qu.:166.5   1st Qu.:154.8
##  Median :15.00   Median :44.00   Median :190.0   Median :190.0
##  Mean   :15.30   Mean   :44.80   Mean   :185.2   Mean   :184.7
##  3rd Qu.:23.25   3rd Qu.:50.00   3rd Qu.:230.0   3rd Qu.:216.2
##  Max.   :32.00   Max.   :99.00   Max.   :260.0   Max.   :240.0
##
##      Height           SES        GenderSTR          GenderCoded
##  Min.   :2.600   Min.   :1.00   Length:20          Min.   :1.000
##  1st Qu.:5.475   1st Qu.:2.00   Class :character   1st Qu.:1.000
```

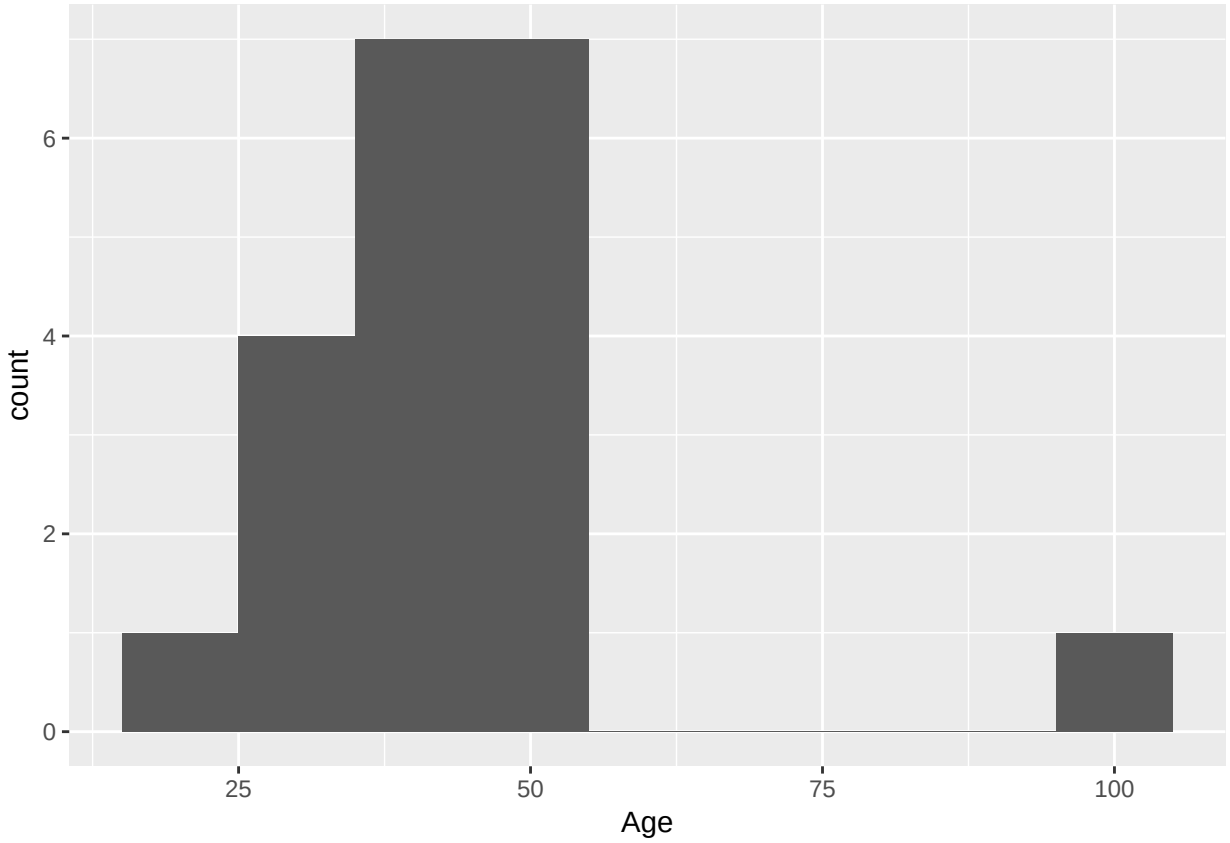
```
##  Median :5.750   Median :2.00   Mode  :character   Median :1.000
##  Mean   :5.650   Mean   :1.95                      Mean   :1.421
##  3rd Qu.:6.125   3rd Qu.:2.00                      3rd Qu.:2.000
##  Max.   :6.500   Max.   :3.00                      Max.   :2.000
##                                                    NA's   :1
##        q1              q2              q3              q4
##  Min.   : 1.00   Min.   :1.000   Min.   :1.000   Min.   : 1.000
##  1st Qu.: 1.75   1st Qu.:2.000   1st Qu.:2.000   1st Qu.: 2.000
##  Median : 3.00   Median :4.000   Median :4.000   Median : 3.000
##  Mean   : 3.35   Mean   :3.421   Mean   :3.706   Mean   : 5.062
##  3rd Qu.: 4.25   3rd Qu.:4.000   3rd Qu.:5.000   3rd Qu.: 4.000
##  Max.   :11.00   Max.   :9.000   Max.   :9.000   Max.   :40.000
##                  NA's   :1       NA's   :3       NA's   :4
##        q5
##  Min.   :1.000
##  1st Qu.:2.000
##  Median :4.000
##  Mean   :3.882
##  3rd Qu.:5.000
##  Max.   :9.000
##  NA's   :3
```

get some plots

Using the `tidyverse` workflow, the following code uses "pipes" which is the `%>%` symbol to take the output from one line as the input to the next line. So the code below can be read as "take the dataset called `dat` and send it (pipe it) into the `ggplot()` graphics environment and then add `+` a `geom` layer for a histogram using the `geom_histogram()` function." To learn more about `tidyverse`, see https://www.tidyverse.org/. The "R Graphics Cookbook" is another fantastic resource for learning how to use `ggplot()` - see http://www.cookbook-r.com/Graphs/. There is a lot of great info and examples online, but you can also purchase the book on Amazon.

Histogram of Age

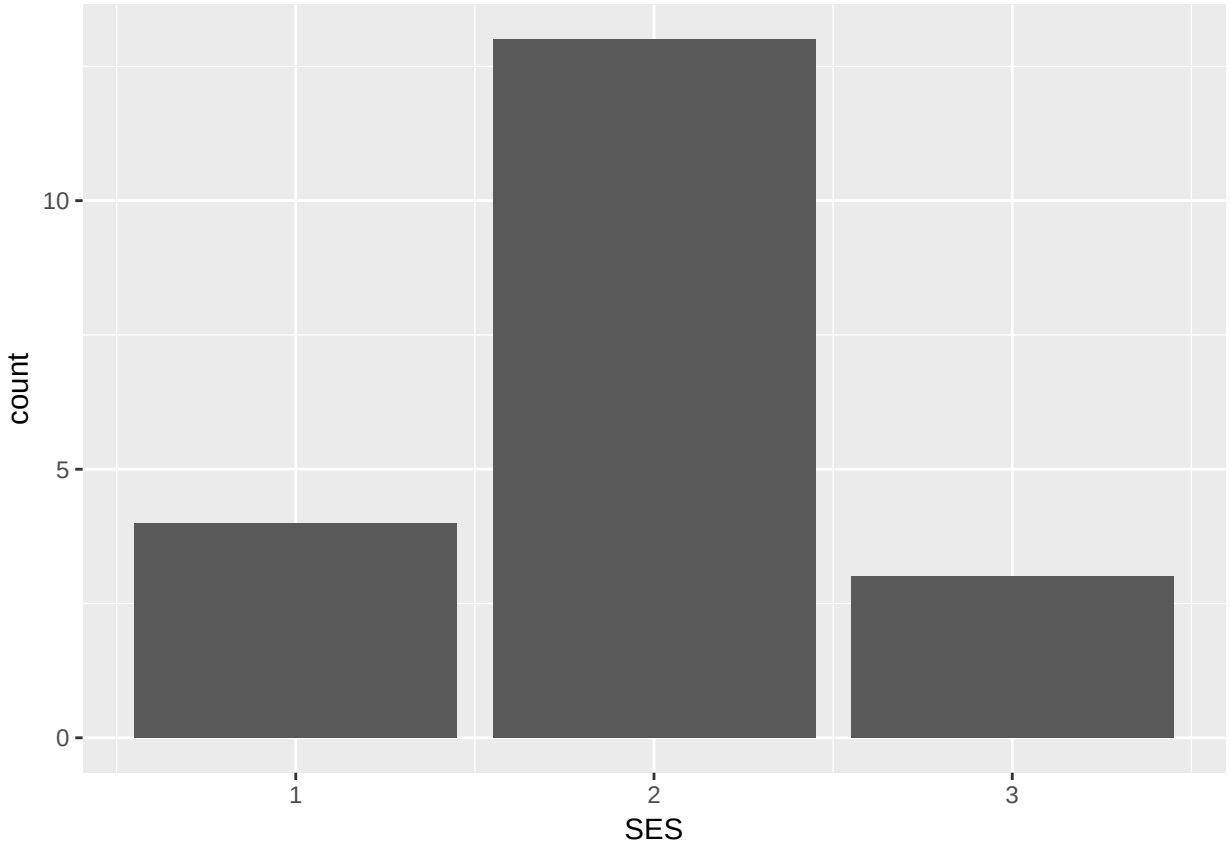
```
dat %>%
  ggplot() +
  geom_histogram(mapping = aes(x = Age),
                 binwidth = 10)
```



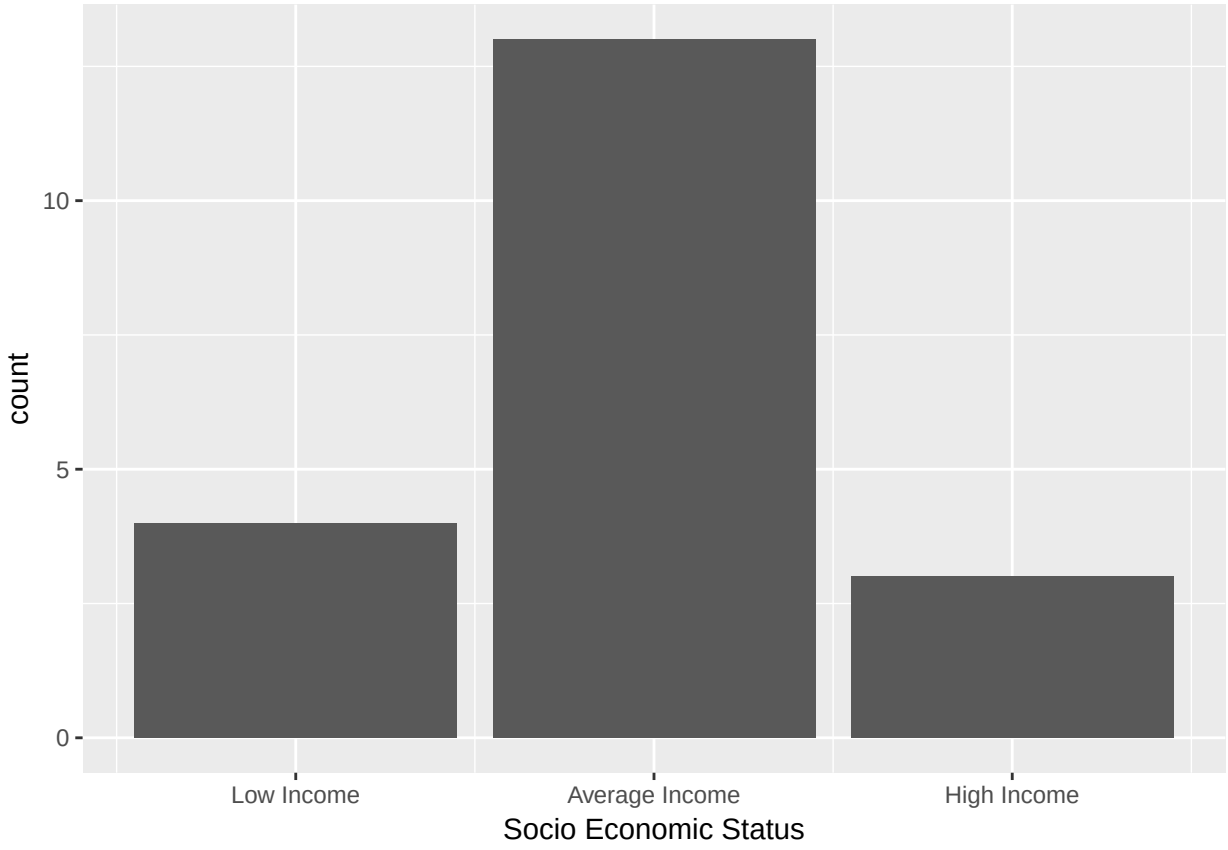
Bar chart of SES

This plot uses the `geom_bar()` graphics layer. However, you'll notice that the categories still show the numeric codes. We could add labels manually, but it gets tedious to do this everytime.

```
dat %>%
  ggplot() +
  geom_bar(mapping = aes(x = SES))
```




```
# you can add an x-axis title and labels for the categories
dat %>%
  ggplot() +
  geom_bar(mapping = aes(x = SES)) +
  scale_x_continuous(name = "Socio Economic Status",
                     breaks = c(1,2,3),
                     labels = c("Low Income","Average Income",
                                "High Income"))
```



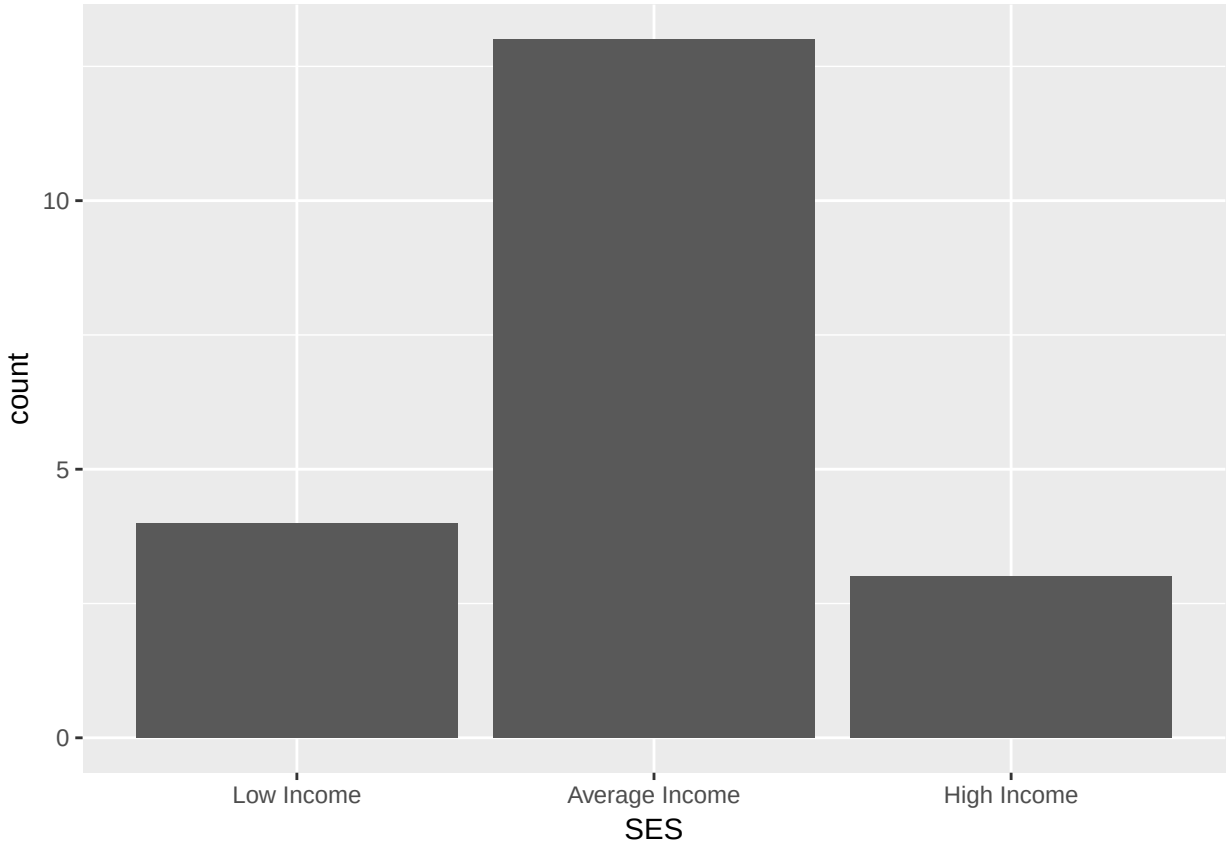
add some factor level info - categorical and ordinal data

We can add the labels for the categorical and ordinal data by changes these variables into "factors". In the `tidyverse` world, the package for this is `forcats`.

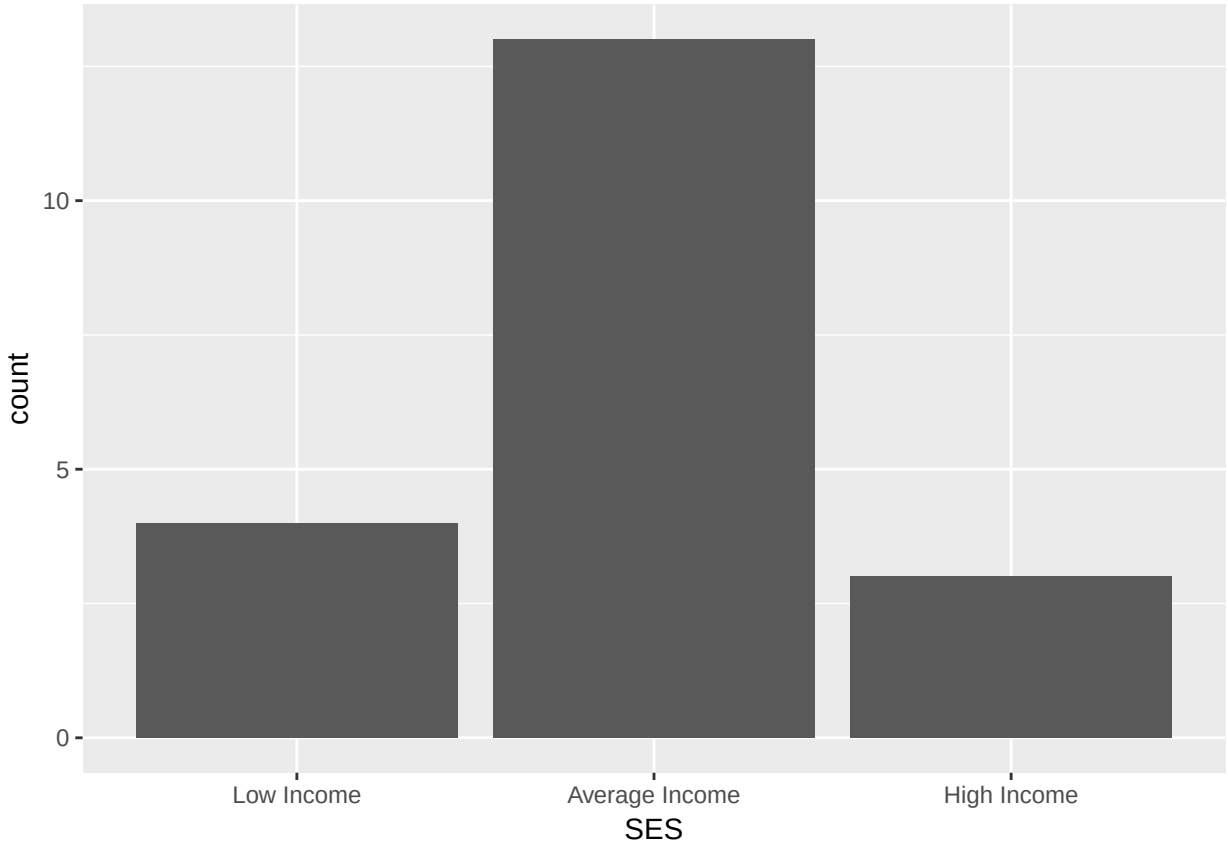
```
# load the forcats package
library(forcats)

# create objects for the labels and levels for SES
sesLabels <- c("Low Income","Average Income", "High Income")
sesLevels <- c(1,2,3)

# apply the labels and levels for SES using the factor() function
dat$SES <- factor(dat$SES,
                  levels = sesLevels,
                  labels = sesLabels)
```

Now that we've create a factor type variable for SES, we can redo our bar chart but now the labels are automatically applied. We do not have to type them in separately using the `scale_x_continuous()` function in the `ggplot2` package.

```
# redo bar plot
dat %>%
  ggplot() +
  geom_bar(mapping = aes(x = SES))
```




```
# another way to do the bar plot
dat %>%
  ggplot(aes(SES)) +
  geom_bar()
```



adding labels for the 5 LiKert scale questions q1,q2,q3,q4,q5

Using a similar workflow to what we did above for SES, let's do it again for q1.

```
# Likert scale coding for q1-q5
qLevels <- c(1,2,3,4,5)
qLabels <- c("None of the time","a little of the time","some of the time",
             "a lot of the time","all of the time")

# apply to q1 - you can redo this for the remaining 4 questions
dat$q1 <- factor(dat$q1,
                 levels = qLevels,
                 labels = qLabels)
```

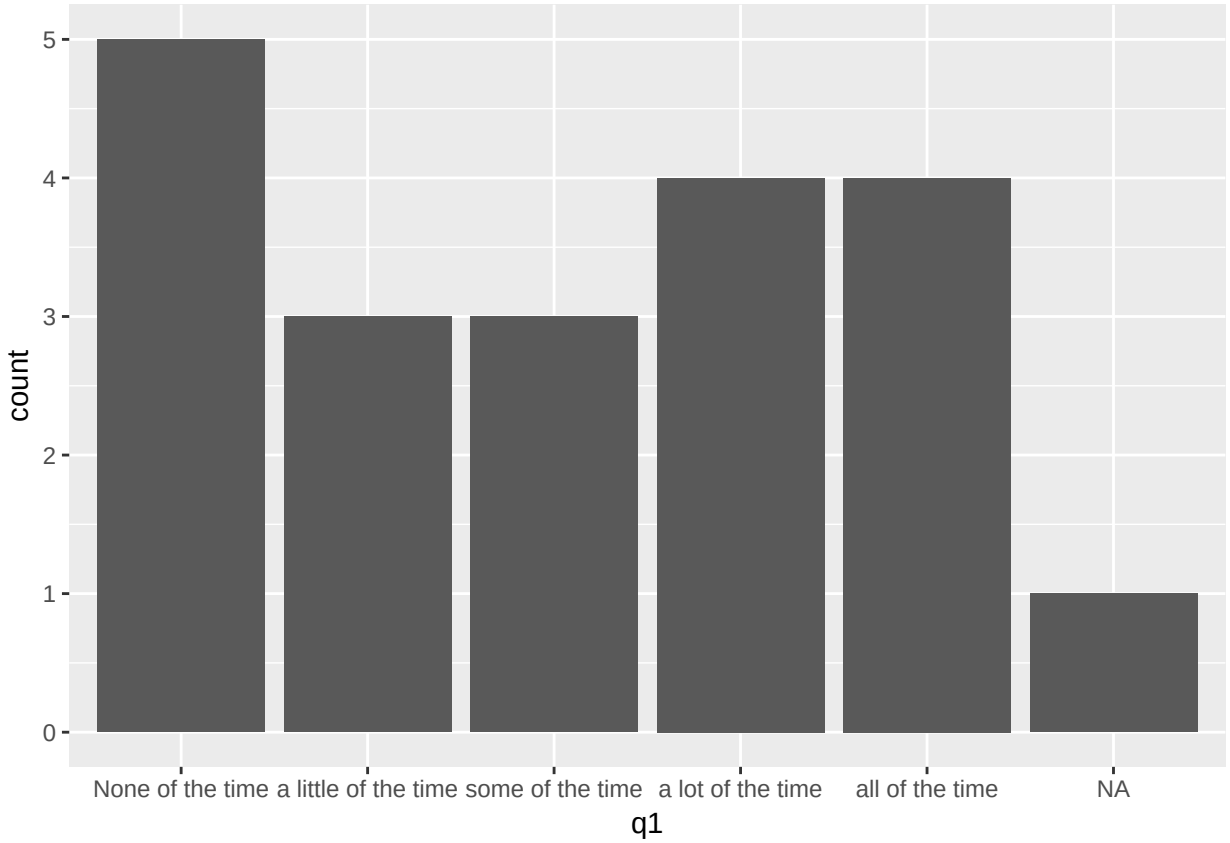
In addition to providing labels in plotting routines, the factor coding also helps with the summary stats.

```
summary(dat$q1)
```

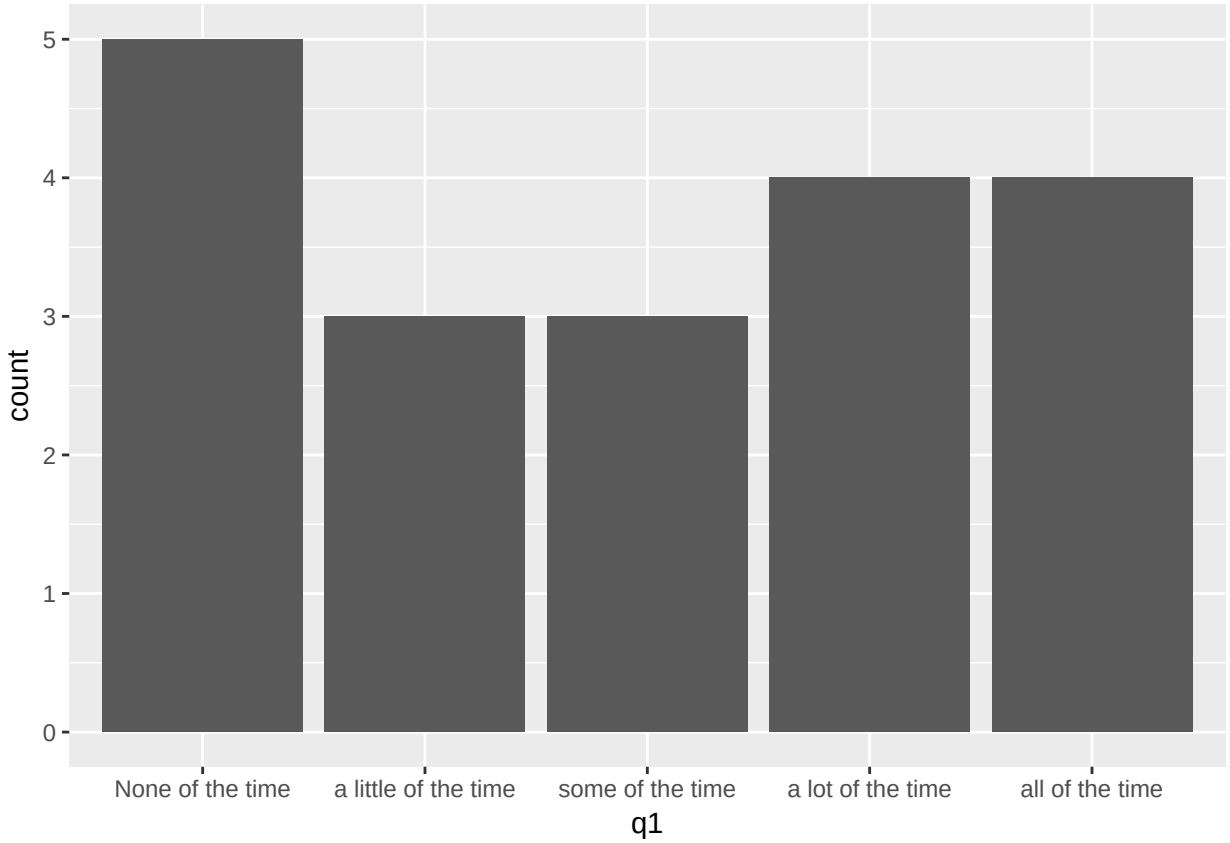
```
##     None of the time a little of the time     some of the time
##                    5                     3                     3
##    a lot of the time      all of the time                  NA's
##                    4                     4                     1
```

And here is a plot of the q1 responses - notice that there are some missing values, `NA`s. There is a second plot made where these are removed using the `is.na()` function and the actual code uses `!is.na()` which says find the data that are NOT (`!`) missing.

```r
dat %>%
  ggplot(aes(q1)) +
  geom_bar()
```




```r
# remove NAs and replot
dat %>%
  subset(!is.na(q1)) %>%
  ggplot(aes(q1)) +
  geom_bar()
```


5
4
3
2
1
0
count
None of the time
a little of the time
some of the time
a lot of the time
all of the time
q1