Discussion Week 1

Jacques Uber

4/2/2020

The Excercise

These are my solutions to problem 2.34 in the Black textbook. I used the packages `tidyverse`, `ggplot2`, `knitr` and `gridExtra`.

Load data

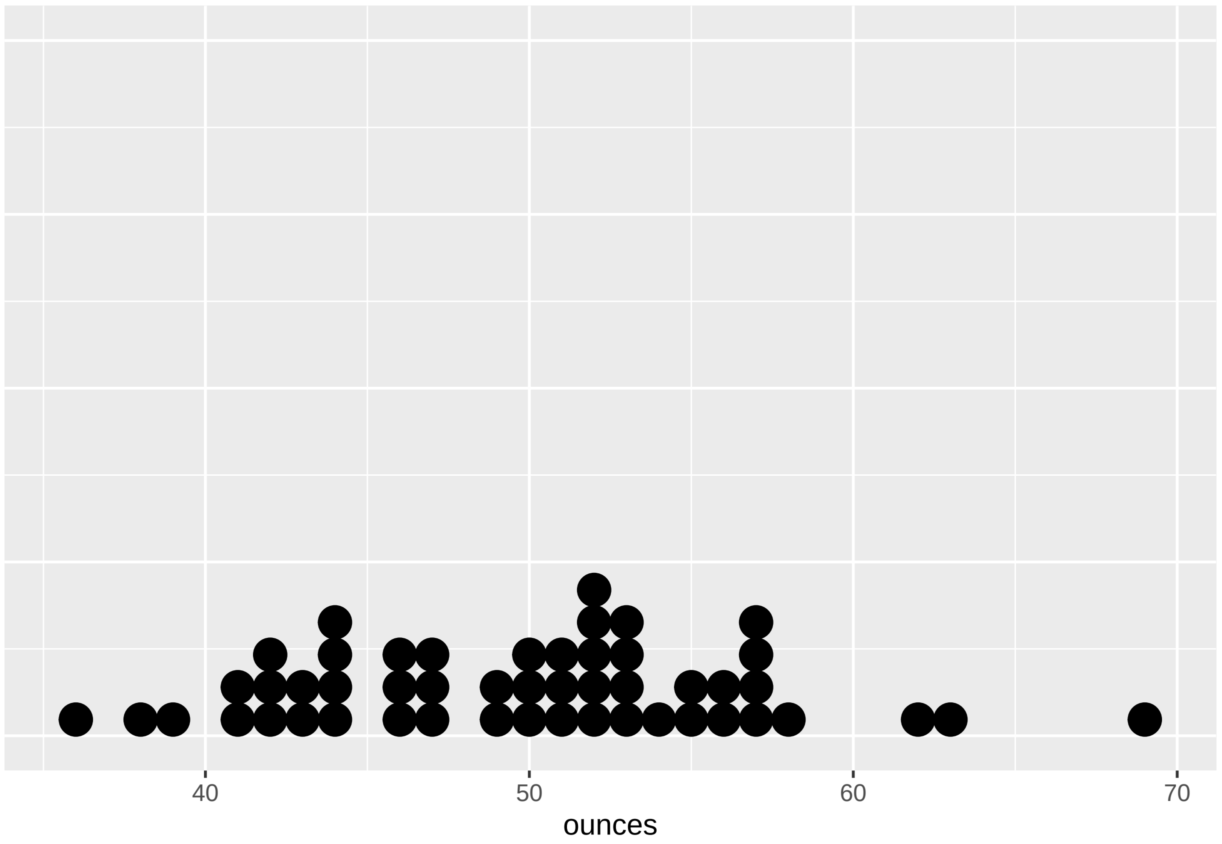
The data we are going to deal with is the weights (in ounces) of 50 random sample from a manufacturing process.

```
data <- tibble(
  ounces=c(
    51,53,56,50,44,47,
    53,53,42,57,46,55,
    41,44,52,56,50,57,
    44,46,41,52,69,52,
    57,51,54,63,42,47,
    47,52,53,46,36,58,
    51,38,49,50,62,39,
    44,55,43,52,43,42,
    57,49
  )
)
```

Part A

Here is a dotplot of our data:

```
ggplot(data, aes(ounces)) +
  geom_dotplot(binwidth=1) + theme(axis.title.y=element_blank(),
                                   axis.text.y=element_blank(),
                                   axis.ticks.y=element_blank())
```



The plot shows that all our sample values are greater than 30 and less than 70 ounces. We can also see that most of the samples fall between 40 and 60 ounces.

Part B

Here is a frequency table of our samples:

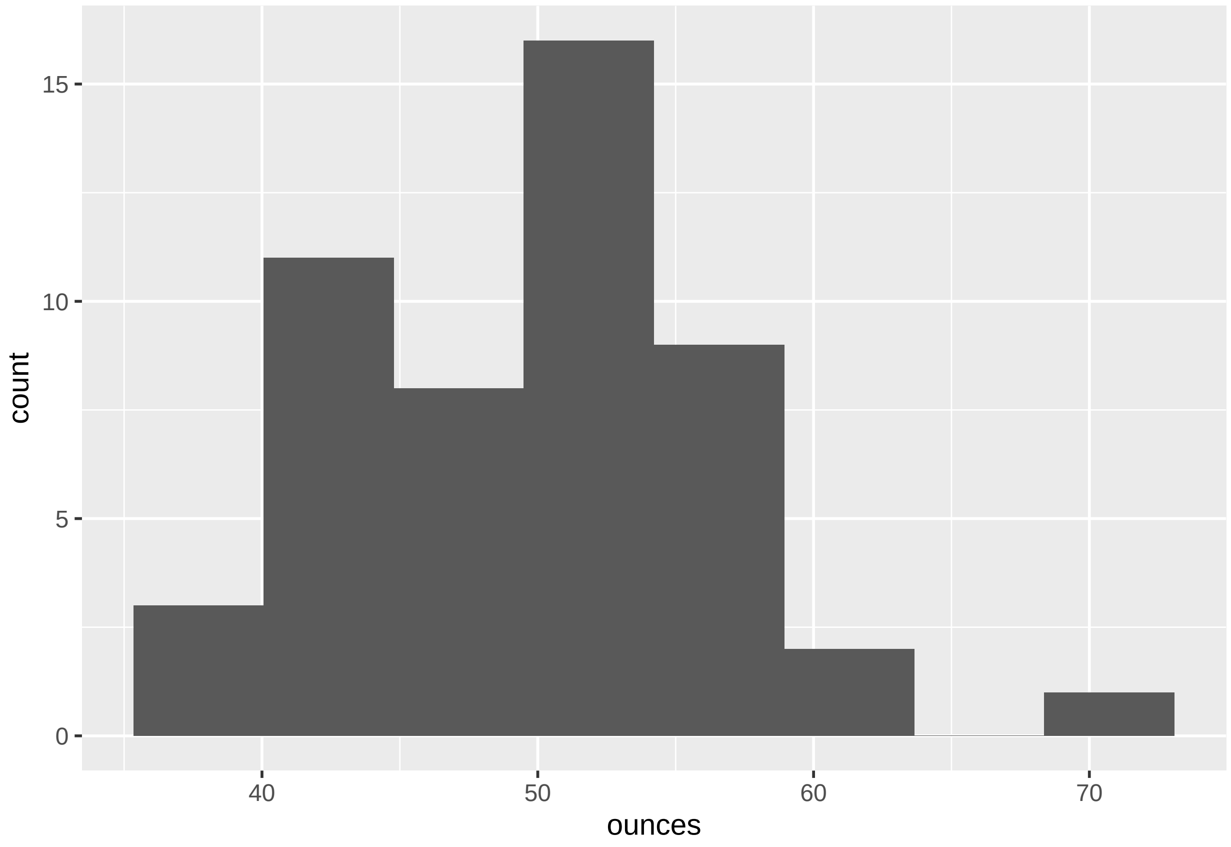
```
library(knitr)
table(data$ounces)%>%
  kable(col.names = c('Ounces', 'Freq'))
```

Ounces	Freq
36	1
38	1
39	1
41	2
42	3
43	2
44	4
46	3
47	3
49	2
50	3
51	3
52	5
53	4
54	1
55	2

Ounces	Freq
56	2
57	4
58	1
62	1
63	1
69	1

The counts in the "Freq" column are all between 1 and 5, which suggest no large concentration in any one ounce value. I would consider this good evidence that our sample set is indeed random.

```r
ggplot(data, aes(ounces)) +
  geom_histogram(bins=8)
```



The histogram above makes me think we are dealing with a normally distributed sample.

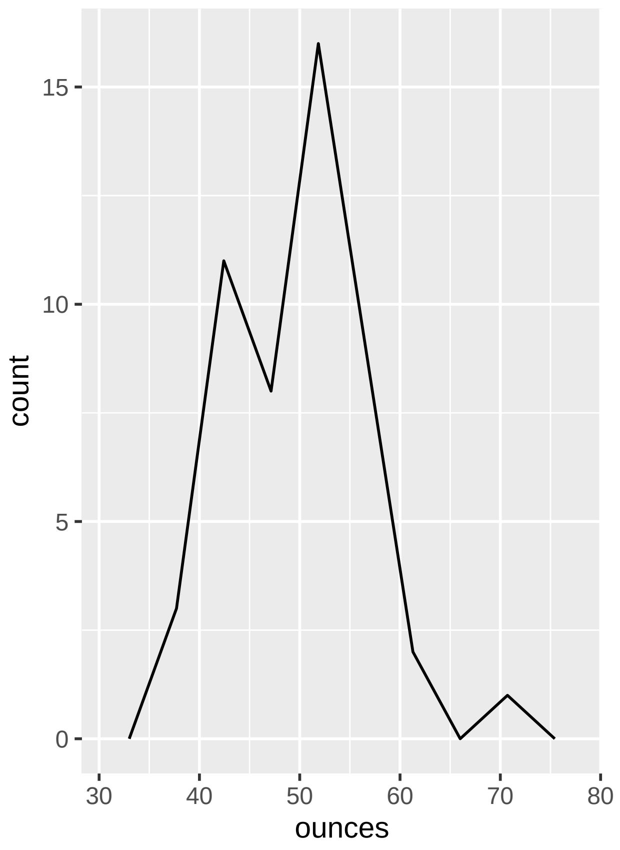
Part C

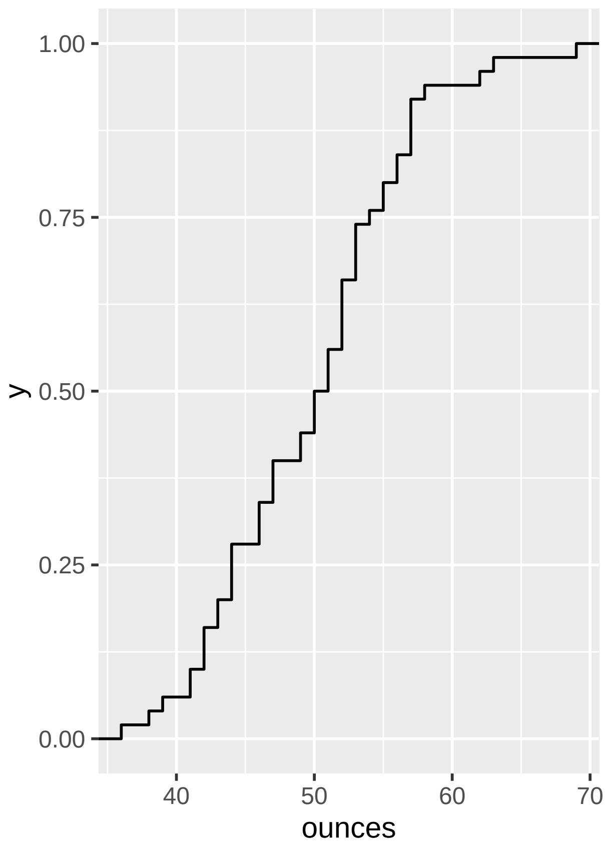
Next we look at a frequency polygon and a cumulative frequency chart (ogive).

```r
plot1 = ggplot(data, aes(ounces)) +
  geom_freqpoly(bins=8)

plot2 = ggplot(data, aes(ounces)) +
  stat_ecdf()

grid.arrange(plot1, plot2, ncol=2)
```





These charts are more confirmation that the sample might be coming from a normally distributed process. The fastest rate of change in the cumulative graph is right near the middle of the dataset, which again tells me most of the data is near the center of the range.

Thoughts

R seems nice – I've never used it before. I especially liked working with RStudio with its multi-pane system; built in spell check (in RMarkdown); the ability to call `help()` and get a nice help page up; and the relatively sane defaults. I was also presently surprised with how easy it was to Google things about `ggplot2`. The built in support for LaTeX was seamless and makes me want to use RMarkdown whenever I need to do LaTeX work (normally I've got a `vim` setup that gets the job done). I'm looking forward a more in-depth dive into R-world in the weeks to come.

A few R experiments I ran

One thing I really like about R so far is its "pipe" operator (`%>%`). I'm still exploring, but so far it seems to be a good way to do data pipelines and even function composition:

In python, you could do something like:

```
f = lambda x: x + 1
g = lambda x: x * 3

g(f(5))
```

```
## 18
```

Or in a shell:

```
echo 5 | { read x; expr 1 + $x; } | { read x; expr 3 \* $x; }
```

```
## 18
```

In R, you can do something like:

```r
f <- function (x) {
  x + 1
}
g <- function (x) {
  x * 3
}
5 %>% f %>% g
```

```
## [1] 18
```

I also wondered how R deals with closures and variable binding:

```r
y <- 3
f <- function(x) {
  x * y
}
y <- 0
f(3)
```

```
## [1] 0
```

The `0` result (as apposed to a `9` result) tells me that f is binding to the variable location of y and not the value.

I also wondered if functions could return functions:

```r
f <- function(a) {
  # this is a comment
  function(b) {
    a + b
  }
}
f(1)(7)
```

```
## [1] 8
```

This is encouraging as it means you can curry values and build partial functions. I wonder how well a partial function will vectorize.

I'm looking forward to diving deeper into R as it seems like an interesting language.