Chapter 2: Distributions

Ghislain Nono Gueye, Ph.D.

3/7/2019

Contents

Load useful package and import data

The data used in his chapter is the `2002FemPreg.Rds` data set

```r
library(here)
library(dplyr)
library(ggplot2)
library(forcats)
```

1 Histograms

The `table()` function in `R` computes frequencies and its output is a *named vector*. It helps to convert the output into a data frame for analysis.

2 Representing histograms

The `table_to_df()` function takes in a vector of values and outputs a data frame of the values and their frequencies.

```
# Base R approach
table_to_df <- function(x){
    df <- as.data.frame(table(x))
    colnames(df) <- c("value", "frequency")
    df$value <- as.numeric(as.character(df$value))
    df
}

# Tidyverse approach
table_to_df <- function(x){
    tibble(value = x) %>%
        group_by(value) %>%
        summarize(frequency = n())
}
```

Let's test the function with the following vector called `values`:

```
values <- c(1, 2, 2, 3, 5)
table_to_df(values)
```

```
## # A tibble: 4 x 2
##   value frequency
##   <dbl>     <int>
## 1     1         1
## 2     2         2
## 3     3         1
## 4     5         1
```

Finding the frequency of a specific value can be achieved with the use of the `sum()` function as well as the equality operator (i.e. `==`). Replicating the example in the book:

```
sum(values == 2)
```

```
## [1] 2
```

```
sum(values == 4)
```

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

There is no real need to wrap this code around a function, but if you really need to:

```
# Base R approach
find_freq <- function(x, v){
    sum(x == v)
}

# Tidy approach
find_freq <- function(x, v){
    tibble(values = x) %>%
        filter(values == v) %>%
        nrow()
}

find_freq(values, 2)
```

```
## [1] 2
```

```
find_freq(values, 4)
```

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

The Python `Values()` method used in the book is the same as the `unique()` function in `R`. The main difference is that the R function outputs sorted values.

```
unique(values)
```

```
## [1] 1 2 3 5
```

3 Plotting histograms

Bar charts (`geom_bar()`) and *histograms* (`geom_histogram()`) look similar, but they have subtle differences (check out this article). Even though the visualizations below are histograms in nature, they are actually bar charts because they map specific values on the x-axis to their frequencies on the y-axis. Another hint is that the bars have spaces between them.

4 NSFG variables

The following script imports the dataset in a variable called `fempreg`. Then, the dataset is filtered by live births and the output is stored in `live_births`.

```
fempreg <- readRDS(here("data", "processed", "used-in-book", "2002FemPreg.Rds"))

# Base R approach
live_births <- fempreg[fempreg$outcome == 1, ]

# Tidy approach
live_births <- fempreg %>%
    filter(outcome == 1)
```

Now, I make a function, which makes a frequency table (using the `table_to_df()` function) and then makes a corresponding histogram (i.e. actually a bar chart). The function also allows the user to specify plot labels (i.e. x-axis and plot title).

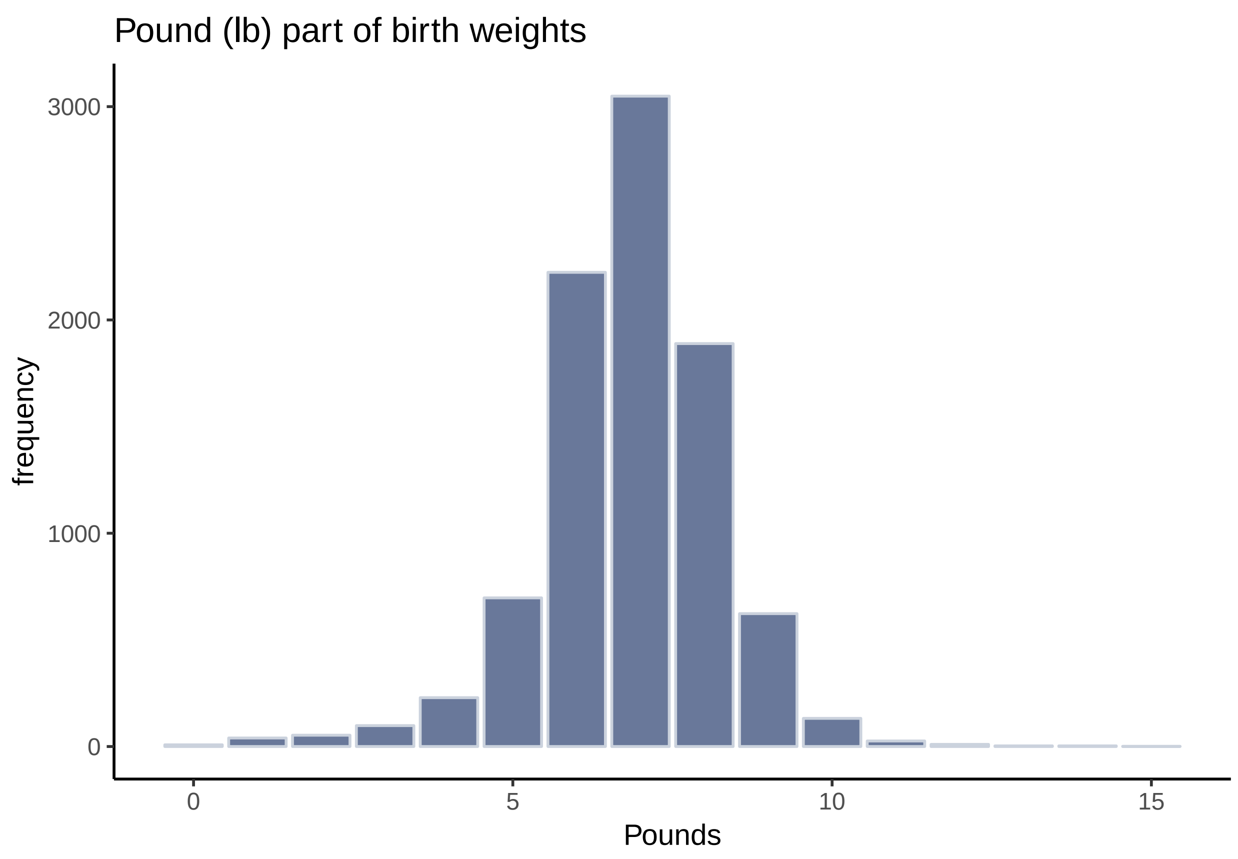
```
make_histogram <- function(variable, x_label, plot_title){

    table_to_df(variable) %>%

    ggplot(aes(x = value, y = frequency)) +
    geom_col(fill = "#69789A", col = "#CBD2DD") +
    theme_classic() +
    labs(x = x_label, title = plot_title)
}
```

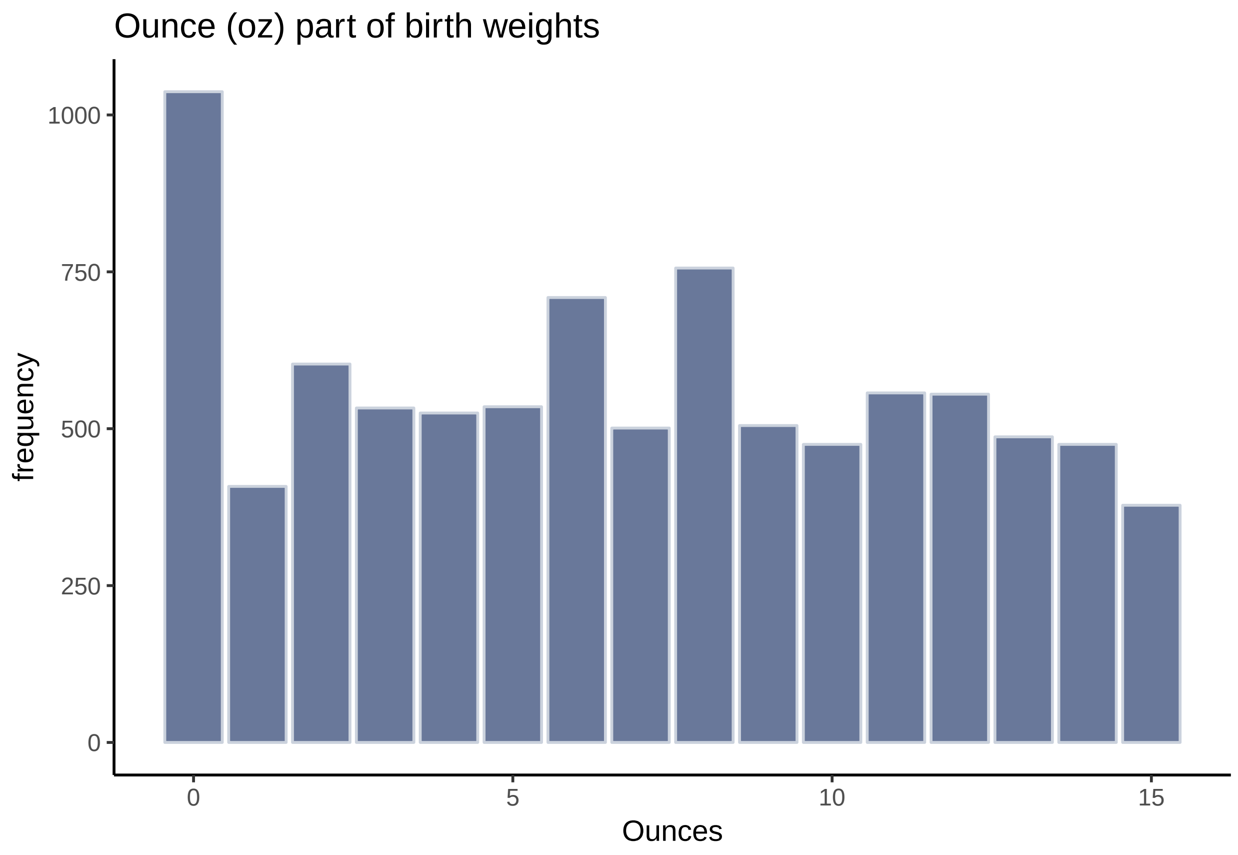
```
make_histogram(live_births$birthwgt_lb, x_label = "Pounds", plot_title = "Pound (lb) part of birth weig
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```

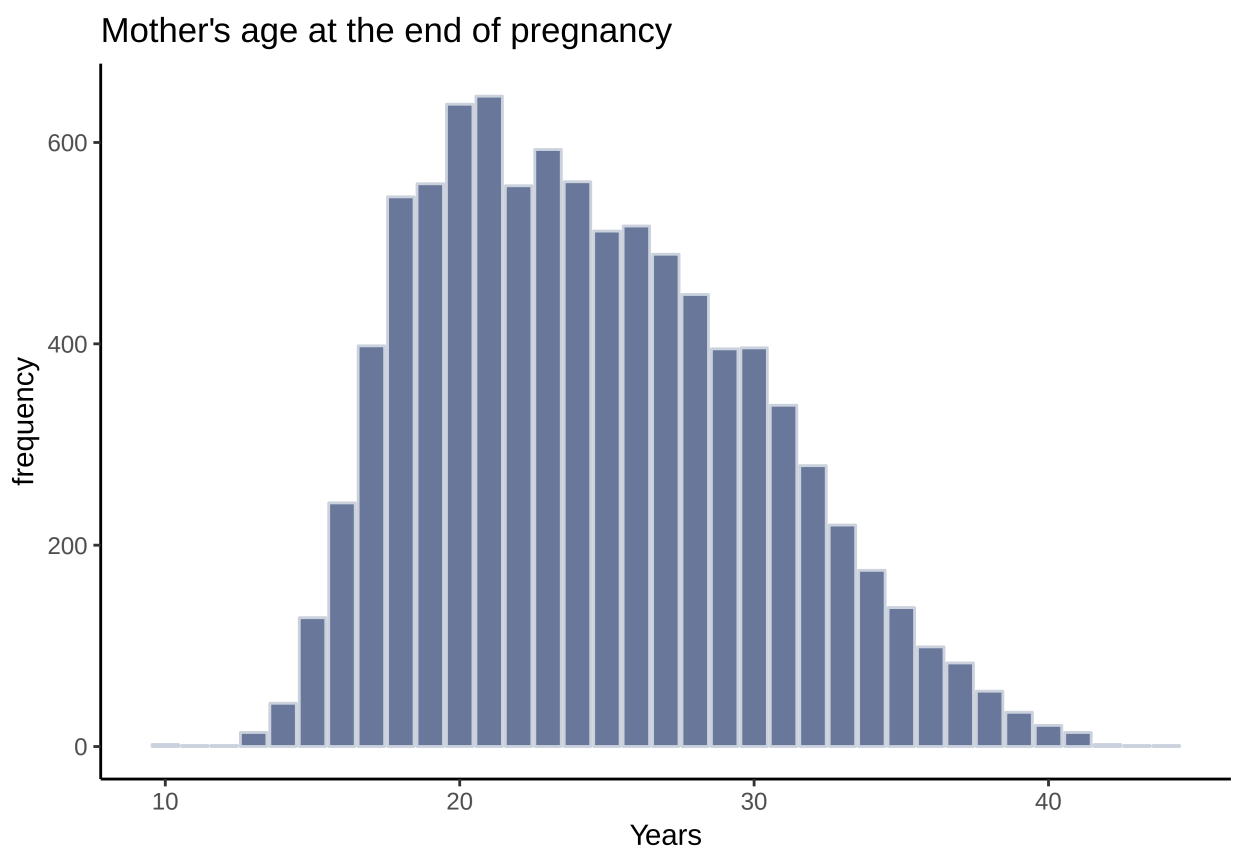


```
make_histogram(live_births$birthwgt_oz, x_label = "Ounces", plot_title = "Ounce (oz) part of birth weig
```

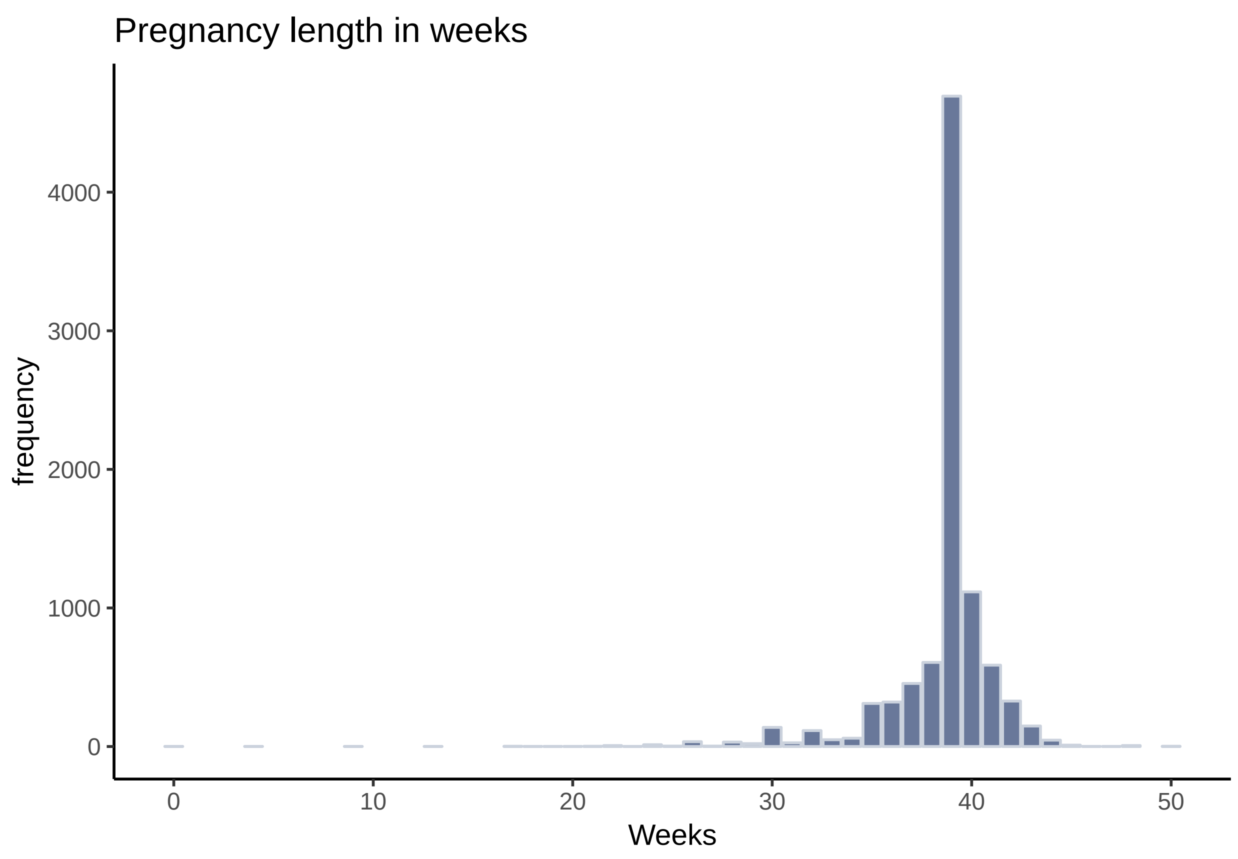
```
## Warning: Removed 1 rows containing missing values (position_stack).
```



```
make_histogram(as.integer(live_births$agepreg), x_label = "Years", plot_title = "Mother's age at the en
```



```
make_histogram(live_births$prglngth, x_label = "Weeks", plot_title = "Pregnancy length in weeks")
```



5 Outliers

The `min_n()` and `max_n()` functions are equivalent to the `Smallest` and `Largest` methods. They display the smallest and largent `n` values of a vector.

```
min_n <- function(x, n){
    head(unique(x), n)
}

max_n <- function(x, n){
    rev(tail(unique(x), n))
}
```

The 10 smallest pregnancy length values:

```
min_n(live_births$prglngth, 10)
```

```
##  [1] 39 38 40 42 35 37 33 41 36 43
```

The 10 largest pregnancy length values:

```
max_n(live_births$prglngth, 10)
```

```
##  [1] 46 23 13 47 17 21 19  0 22 50
```

In order to also view the frequencies of the smallest and largest values, the `table_to_df()` function can be used in conjucntion with the `head()` and `tail()` functions.

```r
# 10 largest pregnancy length values and their frequencies
table_to_df(live_births$prglngth) %>%
    head(10)
```

```
## # A tibble: 10 x 2
##    value frequency
##    <dbl>     <int>
##  1     0         1
##  2     4         1
##  3     9         1
##  4    13         1
##  5    17         2
##  6    18         1
##  7    19         1
##  8    20         1
##  9    21         2
## 10    22         7
```

```r
# 10 largest pregnancy length values and their frequencies
table_to_df(live_births$prglngth) %>%
    tail(10)
```

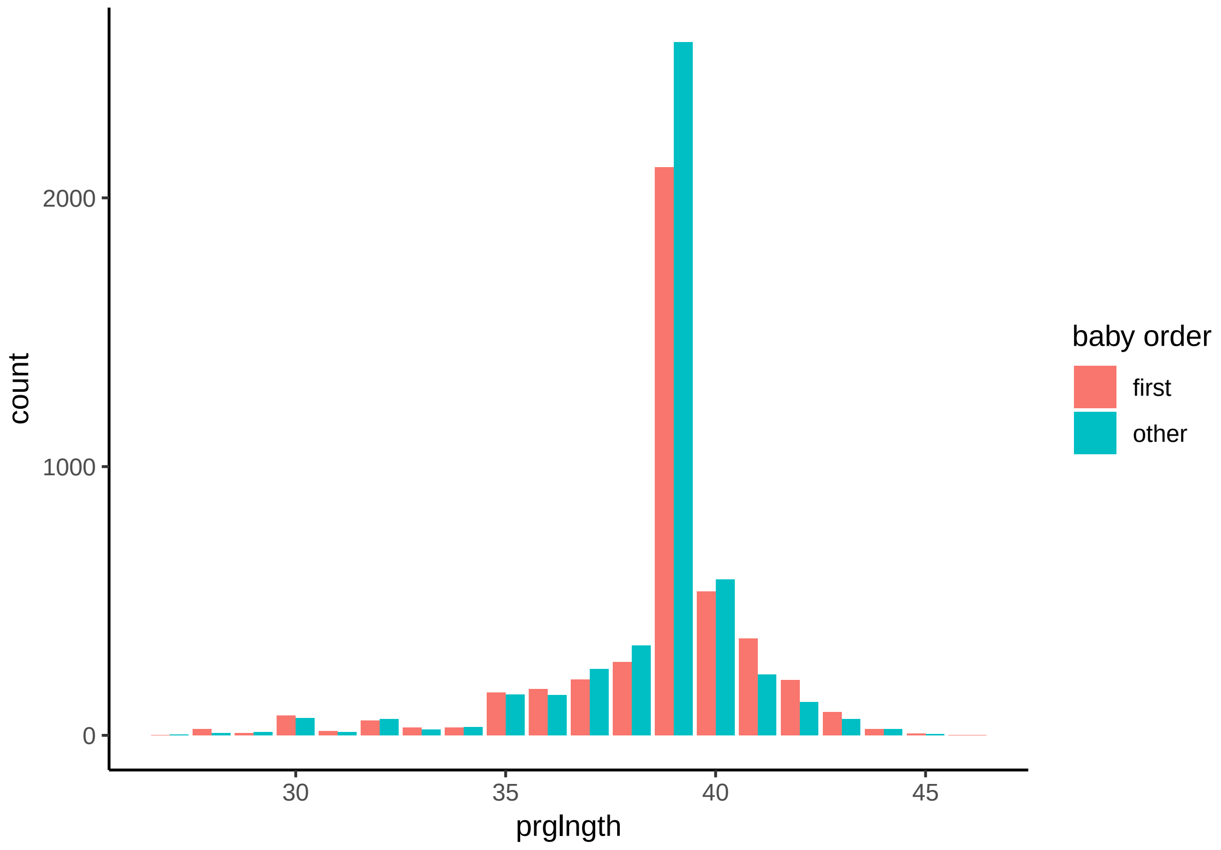
```
## # A tibble: 10 x 2
##    value frequency
##    <dbl>     <int>
##  1    40      1116
##  2    41       587
##  3    42       328
##  4    43       148
##  5    44        46
##  6    45        10
##  7    46         1
##  8    47         1
##  9    48         7
## 10    50         2
```

The `min_freq_n()` and `max_freq_n()` functions will show the smallest and largest values of a given variable as well as their frequencies in the data set. These functions depend on the `table_to_df()` function:

6 First babies

```r
first_or_not <- live_births %>%
    filter(prglngth >= 27 & prglngth <=46) %>%
    mutate(first_baby = factor(if_else(birthord == 1, "first", "other"))) %>%
    group_by(prglngth, first_baby) %>%
    summarize(count = n())

ggplot(data = first_or_not, aes(x = prglngth, y = count, fill = first_baby)) +
    geom_bar(position = "dodge", stat = "identity") +
    labs(fill = "baby order") +
    theme_classic()
```



7 Summarizing distributions

8 Variance

9 Effect size

10 Reporting results

11 Exercises

12 Glossary