R Bootcamp: Data Exploration Solutions

Stat 111 Spring 2019

Warm-up: Probability Review in R

Before jumping into some real data, let's simulate our own data and review a couple probability concepts.

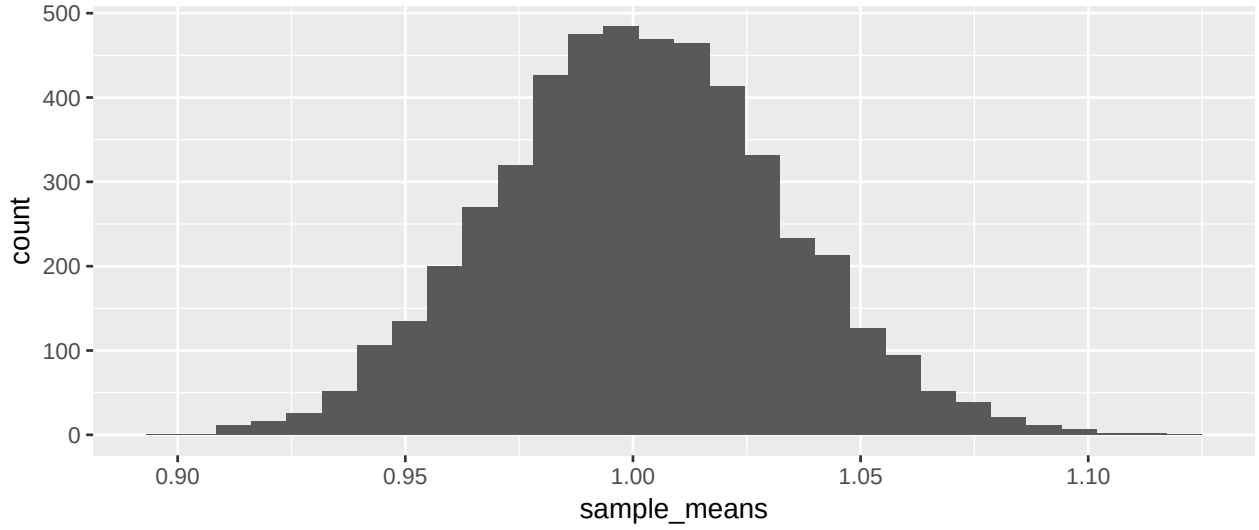
1. Start by generating 1000 independent and identically distributed draws from the Exponential distribution with mean 1 and store these data in a vector. R abbreviates this distribution as "exp." Recall that the Exponential distribution is memoryless, meaning that $\mathbb{P}(X > s + t | X > s) = \mathbb{P}(X > t)$ for $X \sim \text{Expo}(\lambda)$. Verify if this empirically holds true in your simulated data for $s = 0.5$ and $t = 0.25$. Does the phenomenon persist if you change your data-generating distribution to Unif(0, 1)?

```
n <- 1000
data <- rexp(n) # Replace with runif(n) and this is no longer memoryless
s <- 0.5
t <- .25
lhs <- sum(data > s + t)/sum(data > s)
rhs <- sum(data > t)/n
lhs - rhs
```

```
## [1] -0.01247826
```

2. You can use the `mean()` function to compute the sample mean of a vector. Generate 1000 i.i.d. Expo(1) data points as before and compute the sample mean. Repeat this 5000 times and plot a histogram of all the sample means. You may find the `replicate()` function useful or may alternatively write a loop. What shape is the plot and what famous theorem does this exemplify?

```
library(ggplot2)
trials <- replicate(5000, mean(rexp(1000)))
# Roughly Normal histogram because of Central Limit Theorem
ggplot(data.frame(sample_means = trials), aes(sample_means)) + geom_histogram(bins = 30)
```



Data Background

Blue Bikes (formally known as Hubway) is a bike sharing service in greater Boston. The company publishes comprehsive trip data each month. We will be working with data from December 2018, available at https://canvas.harvard.edu/files/7228708/[1]. When you work with data, always perform sanity checks and raise questions. Are the data what you expect? How much variation is there in a column? Do two columns vary together? In this workshop, we will be asking some of these questions and using R to investigate.

Tasks

1. Import the data from the CSV file using `read.csv("201812-bluebikes-tripdata.csv")`. You may need to use `setwd()` to set your working directory to the one containing the CSV file, otherwise, R will not know where to find the data. Make sure to store the data frame in a variable so you can access it later!

```
blue_bikes <- read.csv("201812-bluebikes-tripdata.csv")
```

2. Familiarize yourself with the data set. Useful functions include `head()`, `tail()`, `dim()`, `str()` and `View()`. How many rides were there in December 2018? How many bikes did Blue Bikes use (the `unique()` function will be helpful)?

```
str(blue_bikes, width = 80, strict.width = "cut")
```

```
## 'data.frame':    81961 obs. of  15 variables:
##  $ tripduration           : int  777 313 600 1396 1798 1766 1419 498 886 189 ...
##  $ starttime              : Factor w/ 81957 levels "2018-12-01 00:01:52.7560",..
##  $ stoptime               : Factor w/ 81957 levels "2018-12-01 00:08:38.4730",..
##  $ start.station.id       : int  178 352 54 68 21 21 332 54 352 121 ...
##  $ start.station.name     : Factor w/ 235 levels "175 N Harvard St",..: 148 17..
##  $ start.station.latitude : num  42.4 42.3 42.4 42.4 42.3 ...
##  $ start.station.longitude: num  -71.1 -71.1 -71.1 -71.1 -71.1 ...
##  $ end.station.id         : int  6 57 26 159 21 21 332 26 76 119 ...
##  $ end.station.name       : Factor w/ 234 levels "175 N Harvard St",..: 49 65 ..
##  $ end.station.latitude   : num  42.4 42.3 42.3 42.3 42.3 ...
##  $ end.station.longitude  : num  -71.1 -71.1 -71.1 -71.1 -71.1 ...
##  $ bikeid                 : int  1320 2254 427 3854 3966 4047 3838 3588 2663 2..
##  $ usertype               : Factor w/ 2 levels "Customer","Subscriber": 2 2 2 ..
##  $ birth.year             : int  1996 1989 1986 1996 1969 1969 1998 1991 1967 ..
##  $ gender                 : int  1 1 1 1 0 0 1 2 1 1 ...
```

There were 81961 trips taken in December 2018. There were 2374 bikes.

3. The data has a column for birth year but not for age. Add a new column for age. Then summarize the riders' ages with `summary()`. If you notice any outliers whose ages are clearly unreasonable, remove the corresponding rows from the data frame.

```
blue_bikes$age <- 2018 - blue_bikes$birth.year
summary(blue_bikes$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.00   26.00   30.00   34.33   41.00  130.00
```

[1] We are grateful to Blue Bikes for publishing their data at https://www.bluebikes.com/system-data and to Chi Ting, Low for inspiring parts of this analysis.

```
blue_bikes <- subset(blue_bikes, age < 100)
```

4. Identify the 5 most popular starting stations and their respective number of trips. Also determine how many trips started at the station with ID 104 and the name of that station. You may find the functions `sort()` and `table()` to be useful. As a challenge, see if you can identify the most popular route (that is, pair of starting and ending stations). For this, the `which()` function may be helpful.

```
# Most popular starting stations
sort(table(blue_bikes$start.station.name), decreasing = TRUE)[1:5]
```

```
##
##            MIT at Mass Ave / Amherst St
##                                    2697
## MIT Stata Center at Vassar St / Main St
##                                    2271
##   Central Square at Mass Ave / Essex St
##                                    2183
##         MIT Pacific St at Purrington St
##                                    2023
##                                Kendall T
##                                    1720
```

```
# Mystery station
mystery_station <- subset(blue_bikes, start.station.id == 104)$start.station.name[1]
table(blue_bikes$start.station.name)[mystery_station]
```

```
## Harvard University Radcliffe Quadrangle at Shepard St / Garden St
##                                                               350
```

```
# Most popular route
routes_table <- table(blue_bikes$start.station.name, blue_bikes$end.station.name)
most_popular_route <- which(routes_table == max(routes_table), arr.ind = TRUE)
cbind(rownames(routes_table)[most_popular_route[, 1]],
      colnames(routes_table)[most_popular_route[, 2]])
```

```
##      [,1]
## [1,] "MIT Pacific St at Purrington St"
##      [,2]
## [1,] "MIT Stata Center at Vassar St / Main St"
```

5. Now let's explore the temporal patterns in bike sharing. To make it easier to work with dates, install and load the lubridate package using `install.packages()` and `library()`. Add columns to the data frame for each trip's hour of the day, day of the week, and day of the month, all based on the start time. You can use the functions `ymd_hms()`, `hour()`, `wday()` and `day()`.

```
library(lubridate)
```

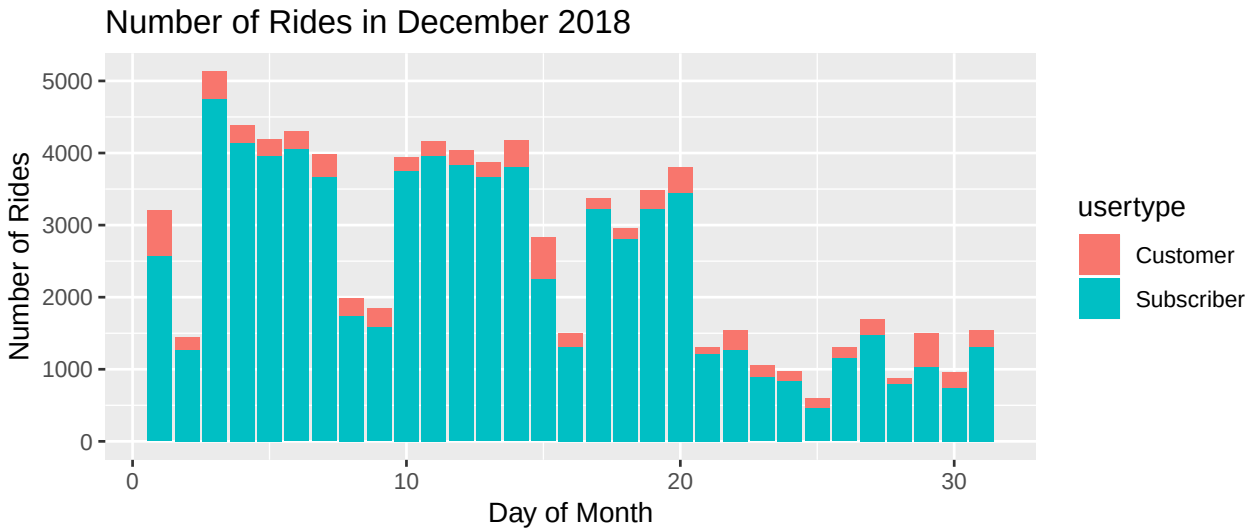
```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
dates = ymd_hms(blue_bikes$starttime)
blue_bikes$hour = hour(dates)
blue_bikes$wday = wday(dates)
blue_bikes$day = day(dates)
```

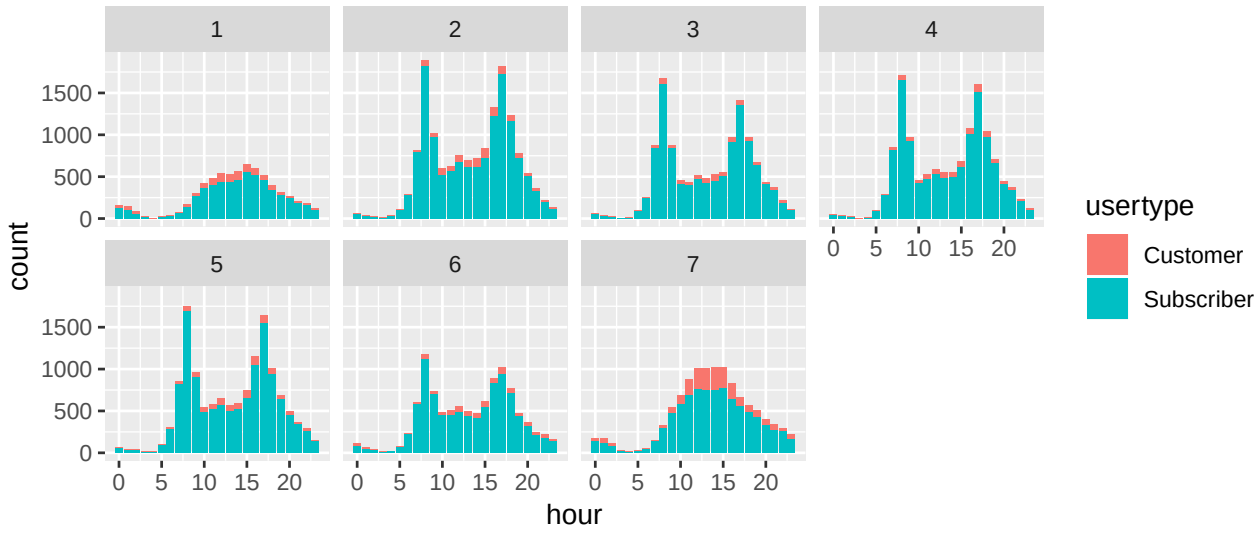
6. Let's visualize these data. Plot a bar plot to see the number of riders per day over the course of the month. Then convert the plot to a stacked bar plot where the bars are split based on the `usertype` of the rider. Also add a plot title and label the axes appropriately. Can you guess what day of the week December 1 was just by looking at the plot?

```
ggplot(blue_bikes, aes(x = day, fill = usertype)) + geom_bar() +
  ggtitle("Number of Rides in December 2018") +
  xlab("Day of Month") + ylab ("Number of Rides")
```



7. Now make seven bar plots, one for each day of the week, showing the number of rides in each hour of the day. Feel free to use a for loop to iterate over the days of the week and make each plot. Alternatively, you can use the `facet_wrap()` function from ggplot2. Continue to split bars by `usertype`. Are the patterns what you expected?

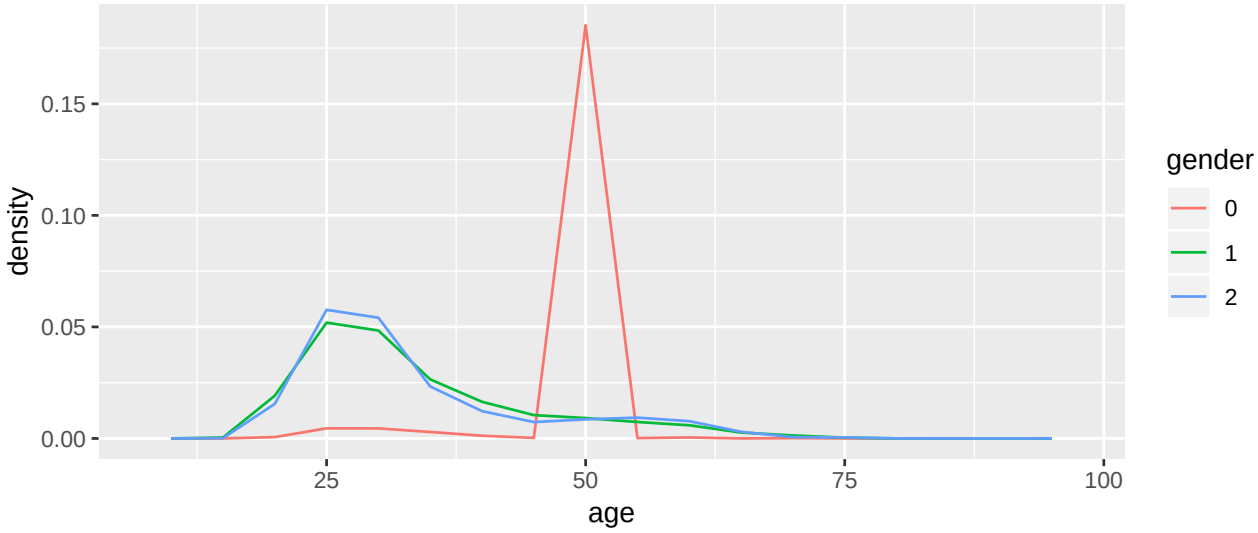
```
# For loop solution:
# for (i in 1:7) {
#   print(ggplot(subset(blue_bikes, wday == i), aes(x = hour, fill = usertype))
#         + geom_bar() + ggtitle(i))
# }

ggplot(blue_bikes, aes(x = hour, fill = usertype)) + geom_bar() +
  facet_wrap(vars(wday), nrow = 2)
```



8. Let's take a look at the `gender` column; unfortunately, the source did not specify what the numbers in the column represent, but we can still explore. In a single plot, show the age distribution (as a density curve) for each gender in a separate color. To do this in ggplot2, try `geom_freqpoly()`. You may need to convert the gender column to the factor data type using `as.factor()` in order to split the data. What data type is gender currently? What is a factor and why is it useful here?

The gender column contains integer data. A factor is a data type for categorical data that may or may not be ordered. In this case, it helps us group our data into riders of a particular gender since we have 3 distinct categories. The color aesthetic expects categorical data.

```
blue_bikes$gender <- as.factor(blue_bikes$gender)
ggplot(blue_bikes, aes(x = age, stat(density), color = gender)) + geom_freqpoly(binwidth = 5)
```



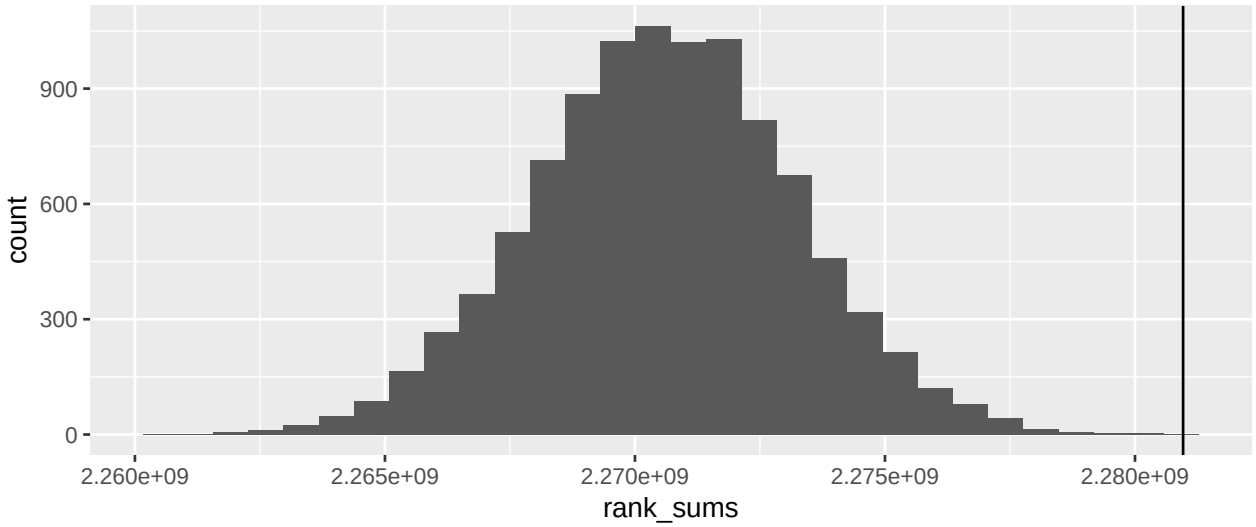
9. You should notice that one of the genders is anomalous in that it overwhelmingly lists an age of 49. This may be the result of filling in missing data, so for now let us focus on the other genders by excluding that data. Your plot may indicate that the distributions of ages are similar between the other two genders, but let's test this hypothesis more rigorously. First, we will compute Wilcoxon's rank-sum statistic as a way to summarize our data (its purpose will be apparent in the next question). Rank the ages of the riders of genders 1 and 2 collectively, breaking ties at random (there is a handy function to do this whose name you can guess) and store the ranks vector. Then, sum together the gender 1 riders' ranks and store this value.

```r
blue_bikes <- subset(blue_bikes, gender != 0)
ranks <- rank(blue_bikes$age, ties.method ="random")
rank_sum <- sum(ranks[blue_bikes$gender == 1])
```

10. Our approach will be to randomly relabel the genders of our data points and see if that affects the distribution of ages (if the genders have the same distribution, then randomizing the labels should not substantially affect the rank sum). Use the same ranks as before, except randomly sample n_1 of them to label as "gender 1" where n_1 is the number of gender 1 individuals in the original data set. Now compute the rank sum and store it in a vector. Repeat the randomization process and continue to store the rank sums in the same vector. Finally, plot a histogram of these randomized rank sums and add a vertical line for the original data's rank sum. What does this say about the hypothesis that the age distribution is the same between genders?

```r
nsims <- 10000
rank_sum_sims <- vector(length = nsims)
number_gender1 <- sum(blue_bikes$gender == 1)
# Alternatively, replace the loop with the replicate() function
for (i in 1:nsims) {
  rank_sum_sims[i] <- sum(sample(ranks, number_gender1))
}
ggplot(data.frame(rank_sums = rank_sum_sims), aes(rank_sums)) +
  geom_histogram() + geom_vline(xintercept = rank_sum)
```

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



The plot gives evidence to reject the hypothesis that the two genders have the same distribution of ages. This is because we assumed the hypothesis to be true in simulating the data, yet the test statistic was more extreme than the vast majority of the simulated statistics, meaning that it is very unlikely we would see such an extreme statistic if the hypothesis were true.

11. We may be interested in repeating this procedure on data from other months. Write a function called `rank_sum_test()` that takes in a data frame and outputs the proportion of simulated rank sums that are greater than or equal to the true rank sum statistic. You can assume the input data frame has a `gender` column with only the genders 1 and 2 as well as an `age` column. Test your function on the current data to see if it works.

```r
rank_sum_test <- function(blue_bikes) {
  ranks <- rank(blue_bikes$age, ties.method ="random")
  rank_sum <- sum(ranks[blue_bikes$gender == 1])
```

```
  nsims <- 10000
  rank_sum_sims <- vector(length = nsims)
  number_gender1 <- sum(blue_bikes$gender == 1)
  for (i in 1:nsims) {
    rank_sum_sims[i] <- sum(sample(ranks, number_gender1))
  }
  sum(rank_sum_sims >= rank_sum)/nsims
}
rank_sum_test(blue_bikes)
```

```
## [1] 1e-04
```

12. Congratulations on completing the guided portion of the data exploration! Now you can continue to play with this data set or download data from other months and study it together. If you choose to download more data, practice using `rbind()` to combine data frames. If you are interested in exploring this data set further, the `leaflet` package gives mapping capabilities that can yield intriguing visualizations.