Prep9S24

Sofia Guttmann

2024-04-13

Reminder: Prep assignments are to be completed individually. Upload a final copy of the .Qmd and renamed .pdf to your private repo, and submit the renamed pdf to Gradescope before the deadline (Tuesday night, 4/16/24, by midnight).

Reading

It's our FINAL prep!

The associated reading for the week is Chapters 7 and 13 on Iteration and Simulation, respectively.

Practice 9 will contain questions about SQL and this week's content on iteration and simulation (Chapters 7 and 13).

1 - Iteration Basics

One of the key concepts in Chapter 7 is that R is designed to work with vectors. A number of functions benefit from this. This means there may be ways to write code more effectively than without using vectors. There are other ways to iterate as well, and iteration is a key concept in simulations. Let's look at a few ideas from the chapter (and your previous Computer science knowledge).

part a - What is the difference between a *for* and a *while* loop, conceptually?

Solution: For loops are used when you know the exact number of times you want to execute the loop, and you want to iterate over a sequence or range of values. While loops are used when you want to iterate as long as a certain condition holds true, and you may not know in advance how many iterations will be needed.

part b - What does the following code from the textbook demonstrate?

```
# you may need to install the bench package
x <- 1:1e5
bench::mark(
  exp(x),
  map_dbl(x, exp)
)
```

```
Warning: Some expressions had a GC in every iteration; so filtering is
disabled.
```

```
# A tibble: 2 x 6
  expression           min   median `itr/sec` mem_alloc `gc/sec`
  <bch:expr>      <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
1 exp(x)            1.54ms   1.69ms    515.      1.53MB     14.0
2 map_dbl(x, exp)  86.69ms  90.34ms      9.39   788.2KB     25.1
```

```
rm(x)
```

Solution: The purpose of this code is to compare the performance of two different methods of calculating the exponential function (**`exp(x)`** and **`map_dbl(x, exp)`**) and see which one is more efficient in terms of execution time. Benchmarking is useful for evaluating the efficiency of different implementations of code and for identifying bottlenecks in performance-critical parts of a program.

part c - At times, we've had to use the *across* function. Suppose you have a data set (mydataset) where the last 5 variables (named myvar1, myvar2, ..., myvar5) are being interpreted by R as characters but they are actually numeric. Write a command to have across help fix this.

Hint: You may need other wrangling commands too!

Solution:

```
library(dplyr)

mydataset <- mydataset %>%
  mutate(across(-1:-5, as.numeric))
```

part d - Describe the uses of *map()* and its variants from the *purrr* package.

Solution:

The basic **map()** function iterates over each element of a list, vector, or data frame, applies a specified function to each element, and returns the results as a list. These variants of **map()** are specialized for specific data types or return formats. They are used when you want the result to be coerced into a logical vector (**map_lgl()**), integer vector (**map_int()**), double vector (**map_dbl()**), character vector (**map_chr()**), data frame row-wise (**map_dfr()**), or data frame column-wise (**map_dfc()**).

part e - The following code to generate some uniform random numbers can be re-written to be more effective. What is the more effective code?

Hint: Remember you can get help about functions by typing ?functionname in the console.

Solution: To generate a vector of **num** uniform random numbers more effectively, you can directly use the **runif()** function without the need for a loop

```
# Keep this value
num <- 100

# Original code
x <- rep(0, num)
set.seed(231)
for(i in 1:num){
  x[i] <- runif(1)
}

# More effective code (provided by you)
set.seed(231)
```

```
x <- runif(num)
```

Generating random values is very important for many simulation ideas. So, let's head to our simulation question!

2 - Simulation

Chapter 13 looks at several different types of simulations. Some are based on actual data sets - to help understand the variability in data, while others involve theoretical situations which are then used to generate data.

If you have taken or are planning to take Probability (Stat 360), the simulation in 13.4.1 about Joan and Sally can be solved with probability concepts, OR you can use a simulation.

For this problem, we'll look at the simulation in 13.4.3, since we've seen that data set before (a long time back). The following code chunks all come from the text, and we'll explore what each one does as you run them.

The data set is the Restaurant Health Violations data set from NYC.

> part a - Explain what the provided code chunk below does. This has nothing to do with simulation, yet. You can answer this by writing sentences, or adding comments to the code.

```
minval <- 7
maxval <- 19
violation_scores <- Violations %>%
  filter(lubridate::year(inspection_date) == 2015) %>%
  filter(score >= minval & score <= maxval) %>%
  select(dba, score)
```

Solution: It sets the minimum value (**minval**) to 7 and the maximum value (**maxval**) to 19.

It filters the dataset **Violations** to include only rows where the year of **inspection_date** is equal to 2015 using **filter(lubridate::year(inspection_date) == 2015)**. It further filters the dataset to include only rows where the **score** is greater than or equal to **minval** and less than or equal to **maxval** using **filter(score >= minval & score <= maxval)**. It selects only the columns **dba** and **score** from the filtered dataset using **select(dba, score)**. The resulting dataset **violation_scores** contains the names of businesses (**dba**) and their corresponding scores within the specified range for the year 2015.

Hint: Being able to describe what the chunk above does in a sentence or two can help you as you work to describe the wrangling process for your final projects.

> part b - Improve the plot generated below to have a title and better name for the y-axis variable.

Be sure you understand what is being plotted!

Solution:

```r
# Improved plot with title and better y-axis label
ggplot(violation_scores, aes(x = dba, y = score)) +
  geom_point() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Violations Scores for Year 2015",
       y = "Score")
```

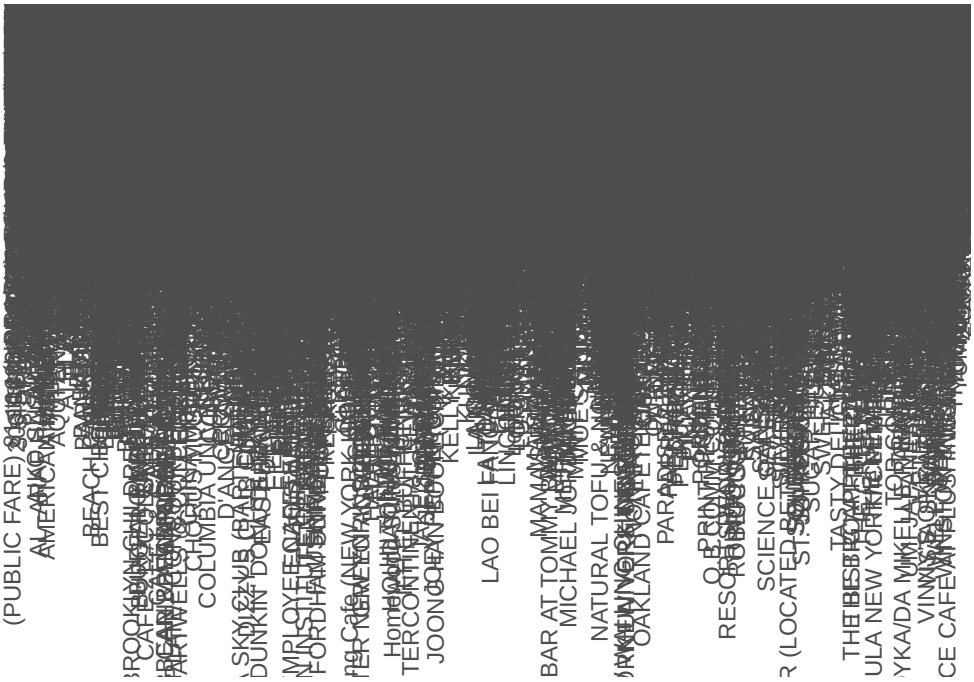
```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89
```

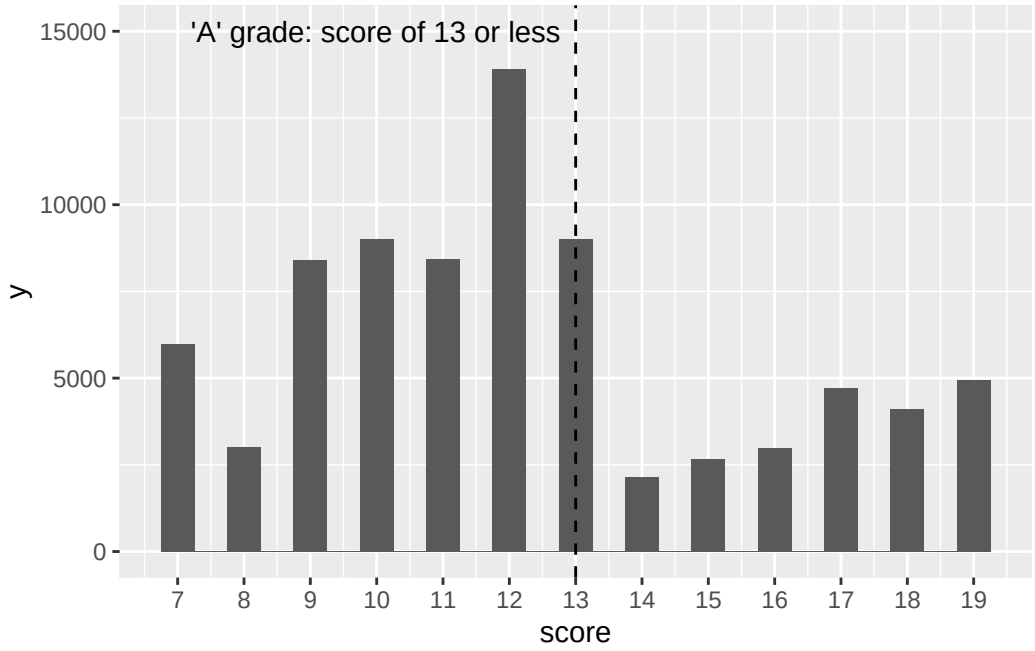
```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x89

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x7f
```

```
Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
width unknown for character 0x1a

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x89

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x7f

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x1a

Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
font width unknown for character 0x1a
```



```
ggplot(data = violation_scores, aes(x = score)) +
  geom_histogram(binwidth = 0.5) +
  geom_vline(xintercept = 13, linetype = 2) +
```

```
  scale_x_continuous(breaks = minval:maxval) +
  annotate(
    "text", x = 10, y = 15000,
    label = "'A' grade: score of 13 or less"
  )
```



part c - In this data set, a score of 13 or below was an "A" grade for the restaurant. Looking at the plot you generated, is there anything suspicious about the transition point?

Solution: No, I don't think so

(This is the motivation for the simulation.)

part d - The next code chunk let's you look at the actual distribution of scores (to see the counts), and then performs a simulation to imagine what would happen if the score values of 13 and 14 were equally likely, and see how many values were returned as "14s" (heads). How many such simulations were performed?

```
# Look at distribution of actual scores
scores <- mosaic::tally(~score, data = violation_scores)
```

```
Registered S3 method overwritten by 'mosaic':
```

```
  method                            from
  fortify.SpatialPolygonsDataFrame ggplot2
```

```
  scores
```

```
score
    7     8     9    10    11    12    13    14    15    16    17    18    19
 5985  3026  8401  9007  8443 13907  9021  2155  2679  2973  4720  4119  4939
```

```
  # Imagine if 13 and 14 were equally likely
  mean(scores[c("13", "14")])
```

```
[1] 5588
```

```
  set.seed(231)
  random_flip <- 1:1000 %>%
    map_dbl(~mosaic::nflip(scores["13"] + scores["14"])) %>%
    enframe(name = "sim", value = "heads")
  # Look at some results
  head(random_flip, 3)
```

```
# A tibble: 3 x 2
    sim heads
  <int> <dbl>
1     1  5632
2     2  5615
3     3  5565
```

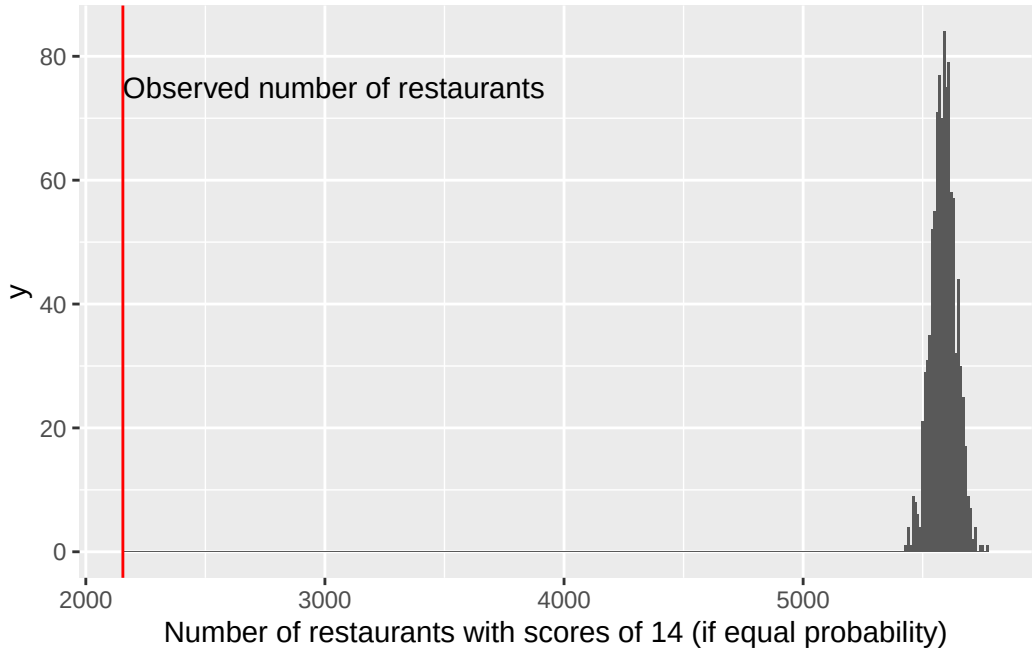
Solution: 5588

part e - To present the results of the simulation, the next plot was produced. Improve the plot by adding: a title, the observed value of restaurants with a score of 14 (add on to the existing label), and making a better y-axis label.

Solution: "Observed number of restaurants"

```
  ggplot(data = random_flip, aes(x = heads)) +
    geom_histogram(binwidth = 10) +
    geom_vline(xintercept = scores["14"], col = "red") +
```

```
  annotate(
    "text", x = 2155, y = 75,
    label = "Observed number of restaurants", hjust = "left"
  ) +
  labs(x = "Number of restaurants with scores of 14 (if equal probability)")
```



part f - What does the plot illustrate? In other words, what do we learn from the simulation? Be sure you can explain this based on the plot (don't just take the answer from the text).

Hint: This is an example of a permutation test. Be sure you are clear on what hypothesis was being tested.

Solution: The plot shows a high number of restaurants with scores of 14 among a wide range of observed values.