Lab Four – Programming in R

- Complete the tasks below. Make sure to start your solutions in on a new line that starts with "**Solution**:".
- Make sure to use the Quarto Cheatsheet. This will make completing and writing up the lab *much* easier.

In this lab, we will build a mealkit recipe generator. I created a small website of about 40 recipes. We will scrape recipes from that website, randomly select three meals, and print a grocery list with recipe cards.

1 Preliminaries

1.1 Part a

Below, I create a numeric vector filled with random observations. Ask for the 3rd item. **Note**: The `set.seed(7272)` portion ensures we all get the same answer.

```
set.seed(7272) # sets the randomization seed for replication
x <- sample(x=1:10,          # sample from 1, 2, 3, ..., 10
            size=10,         # sample of 10
            replace = TRUE)  # allowed to sample the same item multiple times
```

Solution

```
(x[3])
```

```
[1] 5
```

1.2 Part b

Below, I create a data frame filled with random observations.

```
set.seed(7272) # sets the randomization seed for replication
df <- data.frame(x1 = sample(x=1:10,          # sample from 1, 2, 3, ..., 10
                             size=10,         # sample of 10
                             replace = TRUE), # allowed to sample the same item multiple times
                 x2 = sample(x=1:10,          # sample from 1, 2, 3, ..., 10
                             size=10,         # sample of 10
                             replace = TRUE), # allowed to sample the same item multiple times
                 x3 = sample(x=1:10,          # sample from 1, 2, 3, ..., 10
                             size=10,         # sample of 10
                             replace = TRUE)  # allowed to sample the same item multiple times,
)
```

1.3 Part b

Use the `head(...)` function to peek at the data frame.

Solution

```
head(df)
```

```
  x1 x2 x3
1  8  1  8
2  3  7  2
3  5  3  3
4  3  6  2
```

```
5  2  2  3
6  3  8 10
```

1.4 Part c

Ask for the column `x1`.

Solution

```
df[ ,1]
```

```
 [1] 8 3 5 3 2 3 1 2 6 2
```

1.5 Part d

Ask for the fifth row of the data frame.

Solution

```
df[5, ]
```

```
  x1 x2 x3
5  2  2  3
```

1.6 Part e

Ask for the the value of `x1` in the fifth row of the data frame.

Solution

```
df[5, 1]
```

```
[1] 2
```

1.7 Part f

Ask for the the value of in the third column and the fifth row of the data frame.

Solution

```
df[5, 3]
```

```
[1] 3
```

1.8 Part g

Create a sequence from 1 to 10 by 2 and use it to print the odd rows of the data frame.

Solution

```
df[0:4*2+1, ]
```

```
  x1 x2 x3
1  8  1  8
3  5  3  3
5  2  2  3
7  1 10  1
9  6  2 10
```

1.9 Part h

Below, I create an empty column called `x12`. Fill in the details of the `for(...)` loop to ensure `x12` is the product of `x1` and `x2`.

Solution

```r
n <- 10                 # How many rows to we have to fill?
df$x12 <- rep(NA, 10)   # Create an empty column for x12
for(i in 1:10){
 df$x12[i] = df$x1[i]*df$x2[i]
}
print(df)
```

```
   x1 x2 x3 x12
1   8  1  8   8
2   3  7  2  21
3   5  3  3  15
4   3  6  2  18
5   2  2  3   4
6   3  8 10  24
7   1 10  1  10
8   2  6  1  12
9   6  2 10  12
10  2  4  1   8
```

1.10 Part i

Write a function called `calculate.score(...)` that takes three arguments representing `x1`, `x2`, and `x3`) and returns a single value based on the formula:

$$Score = (x_1 \times 2) + x_2 - x_3$$

Use your function to create a new column `total.score` in our data frame `df`.

Solution

```r
n <- 10                 # How many rows to we have to fill?
df$total.score <- rep(NA, 10)   # Create an empty column for total.score
for(i in 1:10){
  calculate.score = (df$x1[i]*2)+df$x2[i]-df$x3[i]
  df$total.score[i] = calculate.score
}
print(df)
```

```
   x1 x2 x3 x12 total.score
1   8  1  8   8           9
2   3  7  2  21          11
3   5  3  3  15          10
4   3  6  2  18          10
5   2  2  3   4           3
6   3  8 10  24           4
7   1 10  1  10          11
8   2  6  1  12           9
9   6  2 10  12           4
10  2  4  1   8           7
```

1.11 Part j

Create a function called `evaluate.row(...)` that takes one argument and returns "low" when the argument is less than 4, "mid" when the argument is between 4 and 7 (inclusive), and "high" when the arguement is 8 or larger. Then, use `sapply(...)` to apply it to column `x1`.

Solution

```r
evaluate.row = function(n){
  if (n < 4){
    return("low")
  }
  if (4 <= n & n <= 7){
    return("mid")
  }
  if (n >= 8){
    return("high")
  }
}


sapply(df$x1, evaluate.row)
```

```
 [1] "high" "low"  "mid"  "low"  "low"  "low"  "low"  "low"  "mid"  "low"
```

1.12 Part k

Did we need to use loops or functions in (h.)-(j.)? That is, can we use vectorization to attain the same results in 1 line each? Where it is possible, write the line of code. Where it is not, explain why.

Solution

1.12.1 Part h

```
(df$x12 = df$x1 * df$x2)
```

```
 [1]  8 21 15 18  4 24 10 12 12  8
```

A loop was not actually needed here to form column x12 (alone), although we cannot replicate it inside the data frame. Since data frames are vectorized, you can use vectorization to multiply x1 and x2 by element.

1.12.2 Part i

```
(df$total.score = (df$x1 * 2) + df$x2 - df$x3)
```

```
 [1]  9 11 10 10  3  4 11  9  4  7
```

A loop was not needed here either to form column total.score (alone), although we cannot replicate it inside the data frame. Since data frames are vectorized, you can use vectorization to perform the arithmetic, applying the function to every row.

1.12.3 Part j

Part j cannot be done without a loop or sapply(), because if() statements in R are not vectorized and require a single logical value. Therefore, there is no way to apply if() statements to a whole column of a data frame.

2 Complete Tasks for One Recipe

2.1 Part a

Install and load the `rvest` package (Wickham 2025).

Solution

```
install.packages("rvest")
```

```
library("rvest")
```

2.2 Part b

Load the html of the `website/KimchiGrilledCheese.html` using the `read_html()` function. We will use the Kimchi Grilled Cheese recipe as our prototype and extend this workflow to all recipes, so try to be as general as possible. If you do look at it, you'll notice it contains the html we saw in the developer tools in class. **Hint**: You can use `read_html(...)` like `read.csv(...)`. Don't forget you can use the documentation to help use it.

Solution

```
library("rvest")
recipe.item = read_html("website/KimchiGrilledCheese.html")
```

2.3 Part c

Open the html file in a web browser and open developer tools. In chrome-based browsers, you can do this by pressing the three verticle dots in the upper-right corner, clicking "more tools", then "developer tools".

Find the name of the Ingredients section of the website and pull the HTML of the function using `html_element(...)` and save the results to an object called `ingredients.section`. **Hint**: You can ask for elements by id using a preceeding "#". See `?html_element(...)` for a helpful example.

Solution

```
ingredients.section = html_element(recipe.item, "#ingredients")
```

2.4 Part d

Now, we want to obtain all of the itemized items. Find the element type the individual ingredients and pull all of them from `ingredients.section` using `html_elements(...)` (note the added s) and save the results to an object called `ingredients`. **Hint**: You can ask for elements by type by simply specifying the tag (e.g., "p" for paragraph). See '?html_element for a helpful example.

Solution

```
ingredients = html_elements(ingredients.section, "li")
```

2.5 Part e

Similar to Part c. Find the name of the Instructions section of the website and pull the HTML of the function using `html_element(...)` and save the results to an object called `instructions.section`.

Solution

```
instructions.section = html_element(recipe.item, "#instructions")
```

2.6 Part f

Now, we want to obtain all of the enumerated items. Find the element type the individual instructions and pull all of them from `instructions.section` using `html_elements(...)` (note the added s) and save the results to an object called `instructions`.

Solution

```
instructions = html_elements(instructions.section, "li")
```

2.7 Part g

Find the class of the recipe image and pull the HTML using `html_element(...)` and save the results to an object called `image.element`. **Hint**: You can ask for elements by class using a preceeding ".". See `?html_element(...)` for a helpful example. Further, note that HTML elements may have more than one class (separated by a space). When that is the case, you need to choose one.

Solution

```
image.element = html_element(recipe.item, ".figure-img")
```

2.8 Part h

Use the `html_attr(...)` function to pull the source link ("src"). Then, use `paste(...)` to prepend the source link with "website/" so we have the full link. **Note**: This would be like adding "https://www.website.com" to get the absolute link.

Solution

```
image.url = html_attr(image.element, "src")
image.url = paste("website/", image.url, sep = "")
```

2.9 Part i

Now that we have all the things we need, let's try to print the recipe. Below, I have written code to print from the objects. One by one, remove `#| eval: false` from the YAML header and add `#| echo: false` and `#| results: 'asis'`, and test. Let me know if you're stuck! **Image**

3 Kimchi Grilled Cheese

Ingredients

- 1/8 tsp chili flakes
- 4 garlic cloves
- 5 oz spinach
- 1 tsp gochujang
- 3 tsp mayonnaise
- 2/3 cup kimchi
- 4 slices of bread
- 2/3 cup cheddar cheese
- 4 tablespoons everything seasoning.

Instructions

1. In a frying pan, heat 2 tablespoons olive oil with chili flakes.
2. Add spinach and cook until wilted.
3. Mix gochujang and mayonnaise in a small bowl and spread on one side of each slice of bread. Sprinkle with everything seasoning and press it into the bread.
4. Build sandwiches by layering spinach, kimchi, and cheese between two slices of bread.
5. Fry in butter until golden brown on both sides.

4 Complete a Full Menu!

Open `Menu.qmd` and add code to complete the following.

1. Randomly select three dinner recipes and one breakfast recipe at random. **Hint:** Use the `sample(...)` function.
2. Pull the image, ingredients, and instructions for each selected recipe.
3. Combine all of the ingredients into a grocery list on the first page.
4. Print the image, ingredients, and instructions for each receipe on the subsequent pages.

When you render this document, no code should be visible. Instead, you should see a five-page document as described above.

Solution

```
[1] "website/GardenTomatoQuiche.html" "website/BBQChili.html"
[3] "website/PastaAndPeas.html"       "website/BakedZiti.html"
```

Figure 1: Garden Tomato Quiche (Serves 6-8)

Ingredients

- 1 egg
- 2 1/2 tablespoons ice water
- 1 1/2 cups flour
- 10 tablespoons butter
- 7 eggs
- 1/2 cups parmesan cheese
- 1 cup swiss and gruyere shredded cheese blend
- 1 cup milk
- 1 teaspoon italian seasoning
- 1/4 teaspoon salt
- 1/8 teaspoon chili flakes
- 1 tsp dried basil (or 4-5 fresh basil leaves)
- 12 ounces cherry tomatoes

Instructions

1. Grease a pie plate.
2. Whisk egg and ice water in a small bowl.
3. Cut butter into small cubes and mix into flour and salt. If you have a food processor, use it. Otherwise mix with a fork until it is a sand-like mixture.
4. Add the egg mixture into the bowl with the flour and mix until a dough forms.
5. Roll out dough to fit a pie plate, pierce with a fork, and press it into the pan.
6. Put the pie plate in the freezer while the oven preheats to 375 degrees.
7. Blind bake quiche crust with pie weights (can use dried beans instead) for 15 minutes.
8. Mix 3 egg yolks, 4 whole eggs, parmesan cheese, shredded cheese, milk, italian seasoning, salt, chili flakes, and basil.
9. Halve cherry tomatoes.
10. Pour egg mixture into the pie plate.
11. Place cherry tomatoes cut-side up, filling the surface area of the quiche. Sprinkle with salt and italian seasoning.

12. Bake for 35-50 minutes, until the edges are set but the center has a slight jiggle or it reaches an internal temperature near 175.

5 BBQ Chili

Ingredients

- 2 cans of baked beans
- 1 can of black beans (drained and rinsed)
- 1 can of white kidney beans (drained and rinsed)
- 1 can of red kidney beans (drained and rinsed)
- 1 large yellow onion
- 1 pound of ground beef
- 1 pound of bacon
- 16 ounces of BBQ sauce

Instructions

1. Dice the yellow onion.
2. Drain and rinse black, white, and red beans.
3. Crumble and cook beef in a frying pan.
4. Cook bacon and cut into bite size pieces.
5. Combine all ingredients in a crock pot. Cook at low heat for 6-7 hours. Stir occasionally.

6 Baked Ziti

Ingredients

- 1 2 oz can of anchovies
- 1 vidalia onion
- 5 garlic cloves
- 1/2 cup red wine (optional)
- 1 small can of tomato paste
- 2 tablespoons butter
- 1 large can of peeled tomatoes
- 2 tablespoons honey
- 3 tsp italian seasoning
- 1/8 tsp red pepper flakes
- 1 pound pasta (e.g., rigatoni or penne)
- 1 pound of ground beef or sausage
- 1 large container of ricotta cheese
- 1/2 cup grated parmesan cheese
- 1 egg
- 8 oz mozzarella

Instructions

1. Dice onion and mince garlic.
2. Heat 2 tablespoons olive oil in a sauce pan. Add anchovies and cook until they break down.
3. Add onions and garlic to the pan and cook until soft.
4. Add red wine and simmer for 3-5 minutes (optional).
5. Add tomato paste and butter, and cook until fully combined.
6. At peeled tomatoes, stir breaking up tomatoes.
7. Once simmering, add honey, 2 tsp italian seasoning, pepper flakes and continue to stir.
8. Brown meat in a frying pan with 1 tsp italian seasoning.
9. Prepare pasta according to package instructions. Save 1 cup of pasta water in case you need to loosen the sauce.
10. Drain meat and put in a large bowl. Mix in ricotta, egg, half the mozzarella, half the parmesan, and a pinch of salt and pepper. Add in half the tomato sauce and all of the pasta.
11. Add mixture to a 9x13 glass pan. Top with remaining tomato sauce, mozzarella cheese, and parmesan.
12. Bake in the oven at 375 degrees for 35 minutes.

7 Pasta and Peas (Serves 4)

Ingredients

- 1 large Vidalia onion
- 6 garlic cloves
- 12 oz bag of peas
- 1 pound regular or mezzi rigatoni
- 1/2 cup grated parmesan cheese
- Italian seasoning and chili flakes

Instructions

1. Chop onion and mince garlic.
2. Boil and cook pasta according to package instructions, reserve 1 cup pasta water if it finishes before step 5.
3. Cook onion and garlic in 2 tablespoons olive oil until translucent, 5-7 minutes. Season with salt, pepper, Italian seasoning, and chili flakes.
4. Add frozen peas to onions and garlic.
5. Add two ladles of pasta water to onion, garlic, and peas and simmer.
6. Add drained pasta back into the stock pot, drizzle with olive oil, and season with Italian seasoning.
7. Mix onion, garlic, and peas into the stock pot with pasta. Mix in parmesan cheese.
8. Serve with additional parmesan cheese.

8 Describe your work!

8.1 Why a fake website?

Read https://www.scrapingbee.com/blog/is-web-scraping-legal/. Originally, I conceived doing this with recipes from Purple Carrot (my favorite subscription service). Being a large company, their terms of service are *very* long and precluded us from copying their recipes. We also checked a few smaller recipe websites we like and even they had terms of service that restricted automated collection of data.

8.2 Conditionals

Did you use conditional statements in your code? If yes, how. If not, are there places you could have used them but did something else?

Solution

Yes, I used conditional statements in part j of part 1. I used if() statements to return "low", "mid", and "high" for their corresponding n values (using logicals, and return()). I did this combined with the function evaluate.row which is a function of n, that later was applied to all of column x1 using sapply.

8.3 Loops

Did you use loops in your code? If yes, how. If not, are there places you could have used them but did something else?

Solution

Yes, I used loops in parts h and i of part 1. In part h, I used a loop to repeat the multiplication of x1 and x2 accross all 10 rows without rewriting the code. This works creating an empty column for x12, then using a for loop for i in $1:10$ and applying the formula throughout. In part i I used a loop in a similar manner by, again, creating an empty column called total score, then using a for loop for i in $1:10$ that applies the calculate.score function to all 10 rows, storing the answers in the total.score column. Part k shows additional ways the x12 and total.score columns could be generated without using loops.

8.4 Functions

Did you use functions in your code? If yes, how. If not, are there places you could have used them but did something else?

Solution

I used functions several times throughout the lab. I used functions in parts i and j of part 1, parts b-h of part 2, and in the similar sections of part 3 (plus the sample() function part). Functions were very useful in this lab, especially for storing functions as objects to keep using them later on. This is because a lot of these activities built on each other, so being able to reference earlier functions easily was necessary.

9 Citations

The (Wickham 2025) package was crucial in creating this fake website. This package is used to "scrape" websites, extracting the information for closer review or for cleaning data. This is a key part of the larger Tidyverse package.

10 Discussion

This lab required a wide variety of different R skills. Activities built on each other, starting with basic navigating of data frames like referencing cells/columns and using the head() function. Then, more advanced skills such as using for loops and conditional statements was required at the end of part 1 to create new columns and interact in more advanced ways with the data frame. Lastly, parts 2 and 3 got into the main task of the lab, recreating the recipe pages. In part 2, the functions read_html, html_element, and html_elements interacted with each other, to pull the different sections of the recipes before printing them. The html_attr and paste functions were used similarly to paste the images specifically. Part 3 was the same except the sample function was used first, storing the breakfast and dinner recipes as vectors and sampling 1 and 3 recipes respectively.

References

Wickham, Hadley. 2025. *Rvest: Easily Harvest (Scrape) Web Pages*. https://doi.org/10.32614/CRAN.package.rvest.