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
```

1.1.1 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.

1.3.1 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`.

1.4.1 Solution

```r
(df[ ,"x1"])
```

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

1.5 Part d

Ask for the fifth row of the data frame.

1.5.1 Solution

```r
(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.

1.6.1 Solution

```r
(df[5,"x1"])
```

```
[1] 2
```

1.7 Part f

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

1.7.1 Solution

```r
(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.

1.8.1 Solution

```r
(df[seq(1,10,2), ])
```

```
  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`.

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

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`.

1.10.1 Solution

```r
calculate.score = function(x1,x2,x3) {
  return((x1 * 2) + x2 - x3)
}
df$total.score = calculate.score(df$x1,df$x2,df$x3)
```

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`.

1.11.1 Solution

```r
evaluate.row = function(val) {
  if (val < 4) {
    return("low")
  } else if (val >= 4 & val <= 7) {
    return("mid")
  } else {
    return("high")
  }
}
df$evaluation.x1 = sapply(df$x1,evaluate.row)
```

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.

1.12.1 Solution

For (h.):

```r
df$x12 = df$x1 * df$x2
```

For (i.):

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

For (j.):

```r
df$evaluation.x1 = ifelse(df$x1 < 4, "low", ifelse(df$x1 <= 7, "mid", "high"))
```

2 Complete Tasks for One Recipe

2.1 Part a

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

2.1.1 Solution

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

```r
library(rvest)
```

2.2 Part b

Load the html of the `website/KimchiGrilledCheese.html` using the `read_html()` function and save the result to an object called `recipe.item`. 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.

2.2.1 Solution

```r
url = "website/KimchiGrilledCheese.html"
(recipe.item = read_html(url))
```

```
{html_document}
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
[1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
[2] <body class="nav-fixed quarto-light">\r\n\r\n<div id="quarto-search-resul ...
```

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.

2.3.1 Solution

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

```
{html_node}
<section id="ingredients" class="level2">
[1] <h2 class="anchored" data-anchor-id="ingredients">Ingredients</h2>
[2] <ul>\n<li>1/8 tsp chili flakes</li>\r\n<li>4 garlic cloves</li>\r\n<li>5  ...
```

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.

2.4.1 Solution

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

```
{xml_nodeset (9)}
[1] <li>1/8 tsp chili flakes</li>
[2] <li>4 garlic cloves</li>
[3] <li>5 oz spinach</li>
[4] <li>1 tsp gochujang</li>
[5] <li>3 tsp mayonnaise</li>
[6] <li>2/3 cup kimchi</li>
[7] <li>4 slices of bread</li>
[8] <li>2/3 cup cheddar cheese</li>
[9] <li>4 tablespoons everything seasoning.</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`.

2.5.1 Solution

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

```
{html_node}
<section id="instructions" class="level2">
[1] <h2 class="anchored" data-anchor-id="instructions">Instructions</h2>
[2] <ol type="1">\n<li>In a frying pan, heat 2 tablespoons olive oil with chi ...
```

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`.

2.6.1 Solution

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

```
{xml_nodeset (5)}
[1] <li>In a frying pan, heat 2 tablespoons olive oil with chili flakes.</li>
[2] <li>Add spinach and cook until wilted.</li>
[3] <li>Mix gochujang and mayonnaise in a small bowl and spread on one side o ...
[4] <li>Build sandwiches by layering spinach, kimchi, and cheese between two  ...
[5] <li>Fry in butter until golden brown on both sides.</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.

2.7.1 Solution

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

```
{html_node}
<img src="images/KimchiGrilledCheese.jpg" class="img-fluid 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.

2.8.1 Solution

```r
(image.url = paste0("website/",html_attr(image.element,"src")))
```

```
[1] "website/images/KimchiGrilledCheese.jpg"
```

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

Figure 1: 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.

3 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.

4 Grocery List

- 1/2 cup pumpkin
- 1/2 cup milk
- 2 eggs
- 2 teaspoons cinnamon
- 1 teaspoon vanilla
- 6-8 slices of bread
- 1 pound pasta (e.g., farfalle or rigatoni)
- 1.5 pounds of stew beef
- 2 bottles of red wine
- 1 vidalia onion
- 1-2 celery sticks
- 1-2 carrots
- 4 garlic cloves
- 4 oz diced pancetta
- 1/2 teaspoon ground cinnamon
- 1/4 teaspoon ground clove (or 1/2 teaspoon nutmeg)
- 1 square dark chocolate
- 1/2 teaspoon dried sage (or 2-3 fresh sage leaves chopped)
- 1 teaspoon dried rosemary (or one fresh rosemary sprig)
- Parmesan cheese to top
- 1/4 tsp ground cinnamon
- 1/4 tsp ginger paste
- 2 tablespoons pine nuts
- 1 cups rolled oats
- 1 tablespoons maple syrup
- 4 teaspoons fig butter
- 1 plums
- 1 cups milk
- 1 bag baby arugula (regular is fine, too)
- 1 pound pasta (e.g., bucatini or farfalle)
- 6-8 ounces of fresh or jarred pesto
- 1 lemon
- Grated parmesan cheese
- Pine nuts

4.1 PumpkinToast

4.1.1 Ingredients

- 1/2 cup pumpkin
- 1/2 cup milk
- 2 eggs
- 2 teaspoons cinnamon
- 1 teaspoon vanilla
- 6-8 slices of bread

4.1.2 Instructions

1. Mix pumpkin, milk, eggs, cinnamon, and vanilla in a medium bowl.
2. Dip a slice of bread, coat in pumpkin mixture, and cook in a generous amount of butter.
3. Repeat until pumpkin mixture runs out.
4. Serve with syrup, berries, and/or whipped cream.

4.2 RedWineRagu

4.2.1 Ingredients

- 1 pound pasta (e.g., farfalle or rigatoni)
- 1.5 pounds of stew beef
- 2 bottles of red wine
- 1 vidalia onion
- 1-2 celery sticks
- 1-2 carrots
- 4 garlic cloves
- 4 oz diced pancetta
- 1/2 teaspoon ground cinnamon
- 1/4 teaspoon ground clove (or 1/2 teaspoon nutmeg)
- 1 square dark chocolate
- 1/2 teaspoon dried sage (or 2-3 fresh sage leaves chopped)
- 1 teaspoon dried rosemary (or one fresh rosemary sprig)
- Parmesan cheese to top

4.2.2 Instructions

1. The night before, place stew beef in a bowl with pepper and enough red wine to cover. Marinate in the refrigerator overnight.
2. Dice onion, carrot, and celery. Mince garlic, and sage and rosemary if using fresh. Finely shave dark chocolate with a knife.
3. Add pancetta to a stock pan with olive oil. Cook for 2-3 minutes.
4. Add onion, carrot, celery and garlic to stock pan and cook until vegetables are cooked.
5. Drain beef marinade.
6. Add beef, sage, rosemary, chocolate, cinnamon, and clove to the sauce pan. Stir.
7. Cover with red wine, add a pinch of salt and pepper, and simmer for about two hours.
8. When ready, boil pasta according to package instructions. Save a cup of pasta water.
9. Add pasta to stock pan with sauce and stir. If needed, add pasta water to loosen the sauce.
10. Serve with grated Parmesan.

4.3 OvernightOats

4.3.1 Ingredients

- 1/4 tsp ground cinnamon
- 1/4 tsp ginger paste
- 2 tablespoons pine nuts
- 1 cups rolled oats
- 1 tablespoons maple syrup
- 4 teaspoons fig butter
- 1 plums
- 1 cups milk

4.3.2 Instructions

1. Combine milk, cinnamon, and ginger, with a pinch of salt in a 1-pint wide mouthed mason jar.
2. Whisk oats and maple syrup. Cover and refrigerate overnight.
3. Thinly slice plums.
4. Split between two 1-pint wide mouthed mason jars. Top with fig butter, plums and pine nuts.

4.4 ArugulaPasta

4.4.1 Ingredients

- 1 bag baby arugula (regular is fine, too)
- 1 pound pasta (e.g., bucatini or farfalle)
- 6-8 ounces of fresh or jarred pesto
- 1 lemon
- Grated parmesan cheese
- Pine nuts

4.4.2 Instructions

1. Boil and cook pasta according to box, strain but reserve 1 cup pasta water.
2. In the stock pot used to cook the pasta, add two tablespoons of olive oil and whisk in pesto over medium heat.
3. Whisk in 1/2 cup pasta water, a little more if needed to create a light sauce.
4. Remove sauce from heat and toss cooked pasta in the sauce.
5. Stir arugula in and cover until wilted.
6. Add juice from half a lemon, add more to taste.
7. Serve with parmesan cheese and pine nuts.

5 Describe your work!

5.1 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.

5.2 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?

5.2.1 Solution

I used a conditional in 1.11 part (j.) to split column x1 into “low”, “mid” and “high” values.

5.3 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?

5.3.1 Solution

I used for-loops in part 3 to apply the code to all of the sampled recipes.

5.4 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?

5.4.1 Solution

I used a function in 1.10 part (i.) that takes three initial values and returns the result of some arithmetic. I also used a function in 1.11 part (j.) that takes an initial value and categorizes it as either “low”, “mid” or “high”.

References

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