Lists

1. Create a list called `my_list` with the three following items. Name them `"char_vec"`, `"char_mat"`, and `"lst"`. Draw this list as a train with cargo in the space below.

```
list_el_1 <- "crane"
list_el_2 <- matrix(c("a", "b", "c", "d"))
list_el_3 <- list(c(3.2, 5.7, 9), c(FALSE, TRUE))
```

For each of the following, write the R code to subset the list and then draw the corresponding train or cargo. You're welcome to subset by index, name, or logical; and use `[]`, `[[]]`, or `$`, as is appropriate.

2. The same list but without `char_mat`.

3. The character string `"crane"`.

4. The element `"d"`.

5. The element `"d"` repeated several times to form a character vector of length 4.

6. The double vector, as the sole element of a list.

7. The value `TRUE`.

Control Flow

8. Write an if statment that takes an exam score like `score <- 93` and returns `"A"` if the score is greater than 90.

9. Write an if statement that takes a single integer `n` and prints "even" if `n` is divisible by 2, "odd" otherwise. Test your code with `n <- 7` and `n <- 16`. (*Hint 1: experiment with the modulo operator `%%`, e.g `3 %% 2`.)

10. Extend your previous answer to return letters grades of A, B, C, or "no pass" if the score is in the 90s, 80s, 70s, or less than 70, respectively.

11. On the lines provided below, complete the code to iterate through the vector `x` and print its contents. For this question, the iterator, `i`, should take as its values the character string elements of `x`. (*Hint: see Option 3 in the slides*)

```
x <- c("apple", "banana", "cherry")
```

```
for (i in _______) {

    _________(________)

}
```

12. Repeat the question but this time, the iterator, `i`, should take as its values the integers from 1 to 3. (*Hint: see Option 1 in the slides*)

```
for (i in _______) {

    _________(________)

}
```

13. Write a for loop that uses `x` and prints the following output:

```
Fruit 1: apple
Fruit 2: banana
Fruit 3: cherry
```

14. Using a for-loop (don't cheat with vectorization!), change each of the temperatures of `temps_c` one-by-one so that they are converted from degrees Celcius to Fahrenheit.

```
temps_c <- c(0, 10, 20, 30)
```

15. For-loops, like if-else statments, can be *nested*: inside the two for-loops below, your code has access to both of the iterators `i` and `j`. Fill in the code below to turn `m` from a matrix of 0s to a multiplication table showing the products of the integers 1, 2, 3 with one another.

```
m <- matrix(rep(0, 9), nrow = 3)

for (i in __________) {

    for (j in __________) {

        m[__________] <- __________
    }
}
```

Run the code in R to check your answer. As a cherry on top, make the row and column names of this multiplication table easier to read by reassigning their names with `rownames(m) <- ____` and `colnames(m) <- ____`.

16. *For fun*: Run the following code and view the result.

```
plot(0, 0)

for(i in 1:100) {
  points(runif(1, min=0, max=1), runif(1, min=0, max=1))
}
```

`plot(0, 0)` makes a scatterplot with a single point at (0, 0). `points(x, y)` will add a point to your plot at (x, y) and `runif()` works like `rnorm()` except it generates 1 random uniform number between 0 and 1.

Modify this code so that your random points appear uniformly along the unit circle (a circle centered at (0, 0) with radius one) instead of uniformly in the upper right quadrant.

17. *For even more fun*: Go onto the class JupyterHub to access the Data In Trees files. Write an R script that uses a for loop to read all of the text file and forms a list where the first element is a character vector named `song` and the second element is a double vector named `distance`. The element index in each vector should correspond to the same student's index card.

If you figure it out, please share your R script on Ed!

Functions

18. Define a function called `f_to_c()` that converts temperature in Fahrenheit to Celcius.

19. Define a function that simulates the result of flipping a coin with sides "heads" and "tails". Generalize it to be able to simulate an *unfair* coin, i.e. a coin that does not land "heads" with probability 1/2. (see `?sample()`)

20. Define a function called `drop_n_lowest(x, n)` that takes two arguments: `x`, an atomic vector and `n`, the number of lowest elements to drop. It should return `x` with the lowest `n` elements removed. Test your function on a short numeric vector to ensure it works.

21. What are the pros and cons of defining this function with a default of `n = 1`?

22. Define a function called `check_water_temp()` that takes a temperature value and does the following:

1. Returns `"freezing"` if the supplied temp is below 32 degrees F.
2. Returns `"boiling"` if the supplied temp is above 212 degrees F[1].
3. Returns the same numerical temperature that was input if its between 32 and 212 F.
4. Allows the user to toggle between Fahrenheit or Celcius (but defaults to F) for the units of their input.

[1] Alternatively, you're welcome to look into how to return the hot face emoji (and its analog for freezing).

Factors and Data Frames

23. Convert the following character vector to a factor and make a barplot to visualize its distribution (provide your code and sketch of the barplot if writing the PS by hand).

```
opinion <- c("dislike", "neutral", "dislike", "love", "like", "like")
```

24. Add the possibility for an additional level named `"despise"`. Then, reorder the factor sensibly from most negative to most positive and remake your barplot.

25. Consider the factor below. Without running the code, predict what `as.integer(size)` returns, then explain why the answer would change if the factor had been created without the `levels` argument.

```
size <- factor(c("M", "S", "L", "S"),
               levels = c("S", "M", "L"),
               ordered = TRUE)
```

26. `summary()` is a function whose behavior depends on the `class` of its argument. Describe, in a sentence each, what `summary()` returns for (a) a numeric vector, (b) a factor, and (c) a data frame. Why is it useful that one function name can do all three?

The following four questions deal with the `iris` data set, which is built in to R. You can view it by just typing `iris`.

27. What are the dimensions of the data frame?

28. What is the unit of observation (i.e. what is in each row)? See `?iris`.

29. What is the type of each of the atomic vectors?

30. How can you create a new data frame that has the same number of rows but excludes the columns dealing with sepals (without using `data.frame()`)?

31. Shift the `name` column in `star_wars` to be the row names of that data frame. That's to say, use it to populate the row names and then remove it as a column.

```
star_wars <- data.frame(
    name = c("Anakin", "Padme", "Luke", "Leia"),
    gender = c("male", "female", "male", "female"),
    height = c(1.88, 1.65, 1.72, 1.50),
    weight = c(84, 45, 77, 49)
)
```

32. Using `star_wars` from the previous question, write code that returns each of the following. Where there is more than one way to do it, give both a list-style and a matrix-style version.

a. The `height` column as a numeric vector.
b. A data frame containing only the rows for characters taller than 1.7 meters.
c. The weight of Leia.

33. A data frame is a named list of vectors of the same length. Given that, predict what happens for each of the following and explain your reasoning.

a. `length(star_wars)`
b. `data.frame(x = 1:4, y = c("a", "b"))`
c. `data.frame(x = 1:4, y = c("a", "b", "c"))`

34. Convert `star_wars` to a tibble with `as_tibble()`, then compare the output of `star_wars[, "height"]` to `star_wars_tbl[, "height"]`. What class is each result? Which behavior do you find less surprising, and why?

Data Wrangling I

The questions below refer to the data frame called `day_at_cal`, with the habits of six students during a day at Cal including the day of the week (`weekday`), their study spot that day (`study_spot`), the number of total hours that they used AI in the course of their work (`ai`), and the number of steps walked, in thousands (`steps_k`). To check your answers in R, you can copy and paste the code that creates this data frame from the last slide from class.

```
day_at_cal
```

```
# A tibble: 6 × 4
  weekday study_spot ai_hours steps_k
  <ord>   <fct>         <dbl>   <dbl>
1 Mon     Moffitt         1.2     8.5
2 Mon     Doe             0.4     6.2
3 Tue     Cory            2.1     9.1
4 Tue     MLK             1       7
5 Wed     CITRIS          0.8     5.8
6 Wed     Soda            1.6    10.3
```

Write a single `dplyr` command that will produce each of the following.

35. A data frame containing the 2nd and 5th students.

36. A data frame without the 3rd student.

37. A data frame with only the AI hours column.

38. A data frame with only the factor columns[2].

39. A data frame containing only the students with at least one hour of AI use.

40. A data frame with students who study in either Moffitt or Doe.

41. A data frame with a fifth column called `ai_min` with the AI use in minutes.

42. A data frame with a fifth (logical) column called `active_day` where the number of steps exceeds 8,000.

[2] There is a clever way to do this that avoids having to type the names of the columns that are factors. See the help file for `select()`, particularly about predicate functions.

43. The same data frame sorted in descending order of number of hours of AI.

44. The same data frame sorted according to the day of the week. Within rows with the same day of the week, sort in descending order of the number of steps.

45. A data frame with the overall mean hours of AI use. Name that statistic `mean_ai`.

46. A data frame with the minimum steps walked (`min_steps`), the maximum steps walked (`max_steps`), and the standard deviation of steps walked (`sd_steps`).

47. Skim the documentation for the `tibble` package (https://tibble.tidyverse.org/) then write down three ways in which tibbles behave differently than data frames.

48. If you're interested in reading more about the design philosophy behind the Tidyverse, read *The Tidy Tools Manifesto* by Hadley Wickham, https://cran.r-project.org/web/packages/tidyverse/vignettes/manifesto.html, (it's just two pages).