

# **Stats 102A - Homework 3 Instructions**

Homework questions and instructions copyright Miles Chen, Do not post, share, or distribute without permission.

## **Homework 3 Requirements**

You will submit two files.

The files you submit will be:

1. `102a_hw_03_output_First_Last.qmd` Take the provided Quarto file and make the necessary edits so that it generates the requested output.
2. `102a_hw_03_output_First_Last.pdf` Your output PDF file. This is the primary file that will be graded. Make sure all requested output is visible in the output file. **Be sure to properly tag pages on Gradescope. If submission is not properly tagged, the problem will be graded as if the output is missing.**

There is no script file to submit.

## **Academic Integrity**

At the top of your Quarto file, be sure to include the academic integrity statement after modifying it with your name.

## **Reading:**

Read the following:

- a. tidy data: <https://r4ds.had.co.nz/tidy-data.html>
- b. data transformation: <https://r4ds.had.co.nz/transform.html>
- c. regular expressions tutorial <http://regexone.com/>

## Part 1 - dplyr exercises

Use `dplyr` to answer the following questions about the `vehicles` data set.

- a. How many unique vehicle makers (variable `make`) are included in the data set?
- b. How many vehicles made in 2014 are represented in the data set?
- c. For the year 2014, what was the average city mpg (gas mileage) for all **Compact Cars**? What was the average city mpg for **Midsize Cars** in 2014?
- d. For the year 2014, compare makers of midsize cars. Find the average city mpg of midsize cars for each manufacturer. For example, in 2014, Acura has 5 midsize cars with an average city mpg of 20.6, while Audi has 12 midsize cars with an average city mpg of 19.08. Produce a table showing the city mpg for 2014 midsize cars for the 27 manufacturers represented in the table. Arrange the results in descending order, so that the manufacturer with the highest average mpg will be listed first. You only need to include manufacturer and average city mpg. The other information is to make sure you are subsetting the data correctly.
- e. Finally, for the years 1994, 1999, 2004, 2009, and 2014, find the average city mpg of midsize cars for each manufacturer for each year. Use `tidyR` to transform the resulting output so each manufacturer has one row, and five columns (a column for each year). Print out all the rows of the resulting tibble. You can use `print(tibble, n = 40)` to print 40 rows of a tibble.

Make sure your output is visible for each question.

## Part 2 - more dplyr

I have uploaded a data set called `dr4.Rdata`. It contains the dates that fictional users visited a fictional website. The website is able to track if the same user visited the site more than once. For the particular date range, the site had 395 visitors, and 130 of them visited more than once. Some of them (13 people) visited the site 5 times.

Using `dplyr`, find the average time between repeated visits to the site.

You will want to find the total average.

Be careful when calculating this.

For example, the first user to visit the site more than once (row 2, ,YPELGRZNOQUTNPOH) visited on 6-29, 7-27, 8-3, and 8-11. The time difference for the repeated visits are: 28 days, 7 days, and 8 days, respectively, for an average of 14.33 days.

The next user with repeated visits is row 3 (SNTCUXUDIHCCSPJA). This person visited on 6-15 and 8-17, a difference of 63 days.

If your data set had only these two rows, the average time between visits would be  $(28 + 7 + 8 + 63) / 4 = 26.5$  days. It is not  $(14.33 + 63) / 2 = 38.66$  days.

When I first attempted this, I used `filter()`, `mutate()`, `rowwise()`, `ungroup()`, and `summarise()`. Upon further review, I realized that it is entirely possible to complete this task using only `filter()` and `mutate()` commands. I do not care what combination of commands you use. I do care that you get the correct final result.

*Make sure your final output shows the desired average number of days between visits.*

## Part 3 - rvest

You will use the package `rvest` and `polite` to scrape data from the website baseball-reference.com.

Open a polite session using the `bow()` function. Begin at the teams page <http://www.baseball-reference.com/teams/> and scrape the team names and their respective URLs from the “Active Franchises” table. There are 30 teams.

Using the polite session and a loop, use the `nod()` function to visit each team’s page. Scrape the content of the “Franchise History” table. The table you need to extract has the tag `#franchise_years`. Use `html_node()` and `html_table()` to extract the contents of this table.

Some franchises have changed names and locations over the years. To handle this, add a column to the scraped data called `current_name`. This column should contain the current team name for each row. Example: For a row where the franchise is listed as “1965 Milwaukee Braves,” the `current_name` column should have the value “Atlanta Braves.”

After scraping all 30 teams, combine the data from the individual tables into one large data frame.

**Important:** Scraping data for all 30 teams can be time-consuming and might strain the website’s server. To avoid being rate-limited or banned, test your code using only 2 or 3 teams at a time. Once your code works for the smaller test, you can expand it to scrape data for all 30 teams.

After scraping the data, print out the dimensions of the resulting data. Print out the first 10 rows of the resulting data using `print(baseball, n = 10)` (do not use `head()`).

For reference, when I ran my code, my table had 2864 rows and 22 columns.

## Part 4 - dplyr to summarize the baseball data

The Baseball Reference website uses non-breaking space characters (e.g., in names like “Atlanta Braves”).

I’ve written some commands for you in the qmd file that will replace all instances of the non-breaking space and replace it with a standard space character in the baseball table. I’ve done this part for you. Simply run the code in the section **Some light text clean up** of the qmd file. The output of the last command (`all.equal()`) should return `TRUE`.

Once you have created your table, use the data it contains to calculate some summary statistics.

For each franchise, filter the data set to only include data from the years 2001 to 2025 (inclusive). If the franchise changed team names during this period, include the previous team’s data as well. (e.g. the data for the Washington Nationals will also include data for the 2001-2004 Montreal Expos)

Then calculate the following summary statistics for each team across the 25 seasons:

- *total wins (TW)*
- *total losses (TL)*
- *total runs scored (TR)*
- *total runs allowed (TRA)*
- *total win percentage (wins / (wins + losses))*

Sort the resulting table (should have a total of 30 rows and 6 columns) in descending order by total win percentage. Be sure to print all rows and columns of the resulting summary table.

All requested columns and all 30 teams must appear in the final output to receive full credit.

**Verification:** At the top of my table, I had the NY Yankees, with a total win count of 2293 Total Wins, 1652 Total Losses, and a Total Win Percentage of 0.581.

## Part 5 - Regular expressions for the Manager column

Using regular expressions, extract the wins and losses for the managers listed in the managers column. Do not use each season's number of wins or losses. You must extract the information from the managers column using regular expressions. That column has the information written in the form “F.LastName (82-80)”. You will need to use capture groups in your regular expression to separate the different pieces of information.

Be careful as some of the rows contain information for more than one manager. Combine all of the manager information to get a total wins and loss value for each of the managers. Many managers have managed more than one team. Be sure to combine all of the win-loss information for the same manager. You may assume that entries that share the same first initial and last name are the same person.

Create a summary table with one line for each manager. The table should contain the following columns, and should be sorted descending by total number of games. For total win percentage, it should be calculated in the same way as in Part 4.

- *Manager's name (First initial and Last Name)*
- *Total number of games managed*
- *Total number of wins across career*
- *Total number of losses across career*
- *Total win percentage*

You can independently verify if your information is correct on baseball-reference.com. Each manager has his own page with a total count of wins and losses.

Figuring out the regular expression here is probably the trickiest part. There is also an instance where there are two different people with the same first initial and the same last name. Unfortunately, their information will end up being combined. For this homework assignment, that's okay.

Regarding the regular expression, you will need to use capture groups, and thus `str_match_all()`. We use the `_all` variant because some of the entries will have multiple managers.

All requested columns must appear in the final output to receive full credit. Print at least the first 15 lines of the summary table using `print(baseball, n = 15)` (do not use `head()`).

**Verification:** The first line of my table reads: C.Mack, 7679, 3731, 3948, 0.4858706, for manager, games, wins, losses, win percentage.

Also Watch out for T.La Russa who has a space in his name. He managed the second most number of games with a final record of 2884-2499 If you report his name as T.La, you will not get full credit.