HW 1

Lance Ding

1/21/2023

```
pacman::p_load(knitr, tidyverse)
knitr::opts_chunk$set(echo = TRUE, tidy.opts=list(width.cutoff=80), tidy=TRUE)
```

Instructions

Write this homework acting as if I don't know what I asked you. For example, don't simply list question numbers for the headings. **If you gave this document to someone else who didn't know the assignment, they should be able to understand what you did by reading the headings, code, and accompanying text.**

Look to my HW1 and RMarkdown Organization examples, available on Canvas in the **R Markdown Organization Example Module** for how to write good headings, organize your assignment, and how much narrative text (outside of code chunks) I want. But a good rule of thumb is: explain what YOU'RE doing, NOT what YOUR CODE is doing. I do NOT need to see a repeat, line-by-line narration of what your code does - you can use code comments for that. (In complex blocks you usually want at least a comment every few lines.) I DO want to see an overall summary of what you did in your analysis.

If you're feeling overwhelmed with RMarkdown, the best way to learn it is to look again at the examples I provided. Knit them and **compare the HTML and .Rmd files side by side**. Then you can see what you would write in a .Rmd file to produce various results when knitted.

This HW is worth 10 total points.

Below are a list of tasks I want you to accomplish and place into this document. Otherwise you have free reign to play around with RMarkdown elements and practice organizing your analysis.

1. Change the author and date fields in the header above to your name and the data. (0.5 pts)
2. Make sure to load any packages you may need right at the start. Do *NOT* include the `learnr` package, ever, unless you are writing an interactive Tutorial (which you won't do in this) - this will cause problems. (1 pt)
3. Ensure that no chunks have the `include = FALSE` or `echo = FALSE` option, as I want to be able to see *all* your code and output.
4. Include at least:
    i) Two different levels of headings. An easy way to do this is have one overall heading, and a sub-heading for Qs 5 and 6 below. Make your headings descriptive rather than just "Question 5" or "Question 5.i." Remember, someone reading your analysis may have no idea what question 5.i is! (2 pts)
    ii) Three different kinds of text formatting (e.g. bolding, italicizing, bulleted lists, relevant hyperlink, text-as-code, block quote). Check the RMarkdown cheatsheet for other options, but these are likely the simplest. (1.5 pts)

5. Install and load the `nycflights13` package. This contains a dataset, `flights`.

   Open `flights` and explore it using at least two of the techniques we discussed in Tutorial 1.1: `head()` or `tail()`, `str()`, `summary()`, `skim()`, and `dfSummary()`.

   Around your code chunk(s) and output, include a total of a few sentences of text that:

   i) Explain *briefly* what you're doing.
   ii) Explain what you found, including - what data `flights` contains (try `?flights`!), what each observation/row represents and how many there are, and some key variables you found interesting (you don't need to include all of them!).

   I also want to introduce you here to one additional function from the `knitr` package: `kable()`. `kable()` can really help pretty up your tables in RMarkdown. Compare the default output for, say, `summary()` or `head()` with and without using `kable()`:

```
head(mpg)
```

```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

```
kable(head(mpg))
```

manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact

Doesn't the second look much prettier? Use `kable()` on at least one of the exploratory tables you produce. For more details on all the things you can do with `kable()` (and an extension package, `kableExtra`), see this post. (3.5 total pts)

Note the table may look bad when you knit to PDF to submit to me. That's fine, don't worry about it. You should be able to see how good it looks in HTML, and that's the main thing I want you to understand.

6. Find the mean departure and arrival delay times (for rows where that data isn't missing) using a code chunk. Do planes appear to pick up any time in the air (that is, are their arrival delays any smaller than their departure delays)? About how much? Answer in narrative text near that chunk. (1.5 pts)

To submit this assignment:

Ideally, knit straight to PDF by changing `html_document` to `pdf_document` in line 5 above. This should work as long as you properly installed LaTeX in Tutorial 0.1. Otherwise:

1. Knit to HTML. An HTML document should open automatically in another RStudio window.
2. Click "Open in Browser" in that HTML document. It should open as a webpage in your default browser (e.g. Chrome).
3. Click Ctrl+P/Command+P, but instead of printing a hard copy on your printer click "Save as PDF."
4. Save and upload that document to Canvas.

A note on PDF formatting: you may notice that long lines of code "fly off the side of the page" when you knit to PDF. To fix this:

If you're on a Windows machine:

- Install the `formatR` package
- Change your `opts_chunk$set` code line to the following: `knitr::opts_chunk$set(echo = TRUE, tidy.opts=list(width.cutoff=80), tidy=TRUE)`

That should force your code to always wrap rather than fly off the edge of the page of a PDF. Note this does not fix issues of, say, plot titles that are too long getting cut off. But it should fix all the errors with your code not wrapping. Happy PDFing!

If you're on a Mac: I don't have an easy solution for you. Try and keep your lines of code under about 80 characters. Feel free to use more vertical lines of code to accomplish this. But don't waste large amounts of time formatting. I'll ask you for clarification if something critical is missing.

Another less important note on PDF formatting: you may notice that some of your tables look "scrunched" (squeezed horizontally) due to the number of variables they're trying to display. To fix this would take a lot more work than I want you to put in, so please just ignore it and know I won't deduct points. The main thing I want you to notice is how nice these tables look when you knit to HTML.

——BEGIN ANSWER BELOW———

Lance: Homework 1

Question 5: Exploring the `flights` data from `nycflights13`

Install (if not installed) and load the relevant packages (`nycflights13`). `Tidyverse` was loaded at the top of this document so I will not load it again.

```
pacman::p_load(nycflights13)
```

We are going to be exploring the `flights` dataset from the `nycflights13` package for this assignment, so first we will take a look at the raw dataset. Since `kable()` produces a cleaner output than the default `head()` function, we will use `kable()` to generate our preview.

```
kable(head(flights, 5))
```

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	flight	tailnum	origin	dest	air_time	distance	hour	minute	time_hour
2013	1	1	517	515	2	830	819	11	UA	1545	N14228	EWR	IAH	227	1400	5	15	2013-01-01 05:00:00
2013	1	1	533	529	4	850	830	20	UA	1714	N24211	LGA	IAH	227	1416	5	29	2013-01-01 05:00:00
2013	1	1	542	540	2	923	850	33	AA	1141	N619AA	JFK	MIA	160	1089	5	40	2013-01-01 05:00:00
2013	1	1	544	545	-1	1004	1022	-18	B6	725	N804JB	JFK	BQN	183	1576	5	45	2013-01-01 05:00:00
2013	1	1	554	600	-6	812	837	-25	DL	461	N668DN	LGA	ATL	116	762	6	0	2013-01-01 06:00:00

Also, knowing the amount of data in our dataset as well as the dataset's shape will tell us some useful information. To do that, we will use the `glimpse()` function.

```
glimpse(flights)
```

```
## Rows: 336,776
## Columns: 19
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2~
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, ~
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, ~
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1~
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849,~
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851,~
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1~
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "~
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4~
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394~
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA",~
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD",~
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 1~
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, ~
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6~
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0~
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 0~
```

We now know that our dataset contains **336776 rows** and **19 columns**, and we also know the the datatype of each column. As indicated by the column names, we have data on the times, delays, destinations and airline information of 336776 flights that departed NYC. If more clarification is needed, we can employ the help of `?flights`, which brings up the official documentation of the `flights` dataset.

Now that we know what the data looks like and what it means, we can take a look at a summary of the dataset to gain some deeper insight. Specifically, we want to take a look at the **summary statistics** of each column of the dataset, which includes:

- Minimum
- 1st quartile
- Median
- Mean
- 3rd quartile
- Maximum

```
summary(flights)
```

```
##       year          month             day           dep_time    sched_dep_time
##  Min.   :2013   Min.   : 1.000   Min.   : 1.00   Min.   :   1   Min.   : 106
##  1st Qu.:2013   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.: 907   1st Qu.: 906
##  Median :2013   Median : 7.000   Median :16.00   Median :1401   Median :1359
##  Mean   :2013   Mean   : 6.549   Mean   :15.71   Mean   :1349   Mean   :1344
##  3rd Qu.:2013   3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:1744   3rd Qu.:1729
##  Max.   :2013   Max.   :12.000   Max.   :31.00   Max.   :2400   Max.   :2359
##                                                  NA's   :8255
##    dep_delay          arr_time    sched_arr_time   arr_delay
##  Min.   : -43.00   Min.   :   1   Min.   :   1   Min.   : -86.000
##  1st Qu.:  -5.00   1st Qu.:1104   1st Qu.:1124   1st Qu.: -17.000
##  Median :  -2.00   Median :1535   Median :1556   Median :  -5.000
##  Mean   :  12.64   Mean   :1502   Mean   :1536   Mean   :   6.895
##  3rd Qu.:  11.00   3rd Qu.:1940   3rd Qu.:1945   3rd Qu.:  14.000
##  Max.   :1301.00   Max.   :2400   Max.   :2359   Max.   :1272.000
##  NA's   :8255      NA's   :8713                  NA's   :9430
##    carrier              flight       tailnum             origin
##  Length:336776      Min.   :   1   Length:336776      Length:336776
##  Class :character   1st Qu.: 553   Class :character   Class :character
##  Mode  :character   Median :1496   Mode  :character   Mode  :character
##                     Mean   :1972
##                     3rd Qu.:3465
##                     Max.   :8500
##
##      dest              air_time        distance         hour
##  Length:336776      Min.   : 20.0   Min.   :  17   Min.   : 1.00
##  Class :character   1st Qu.: 82.0   1st Qu.: 502   1st Qu.: 9.00
##  Mode  :character   Median :129.0   Median : 872   Median :13.00
##                     Mean   :150.7   Mean   :1040   Mean   :13.18
##                     3rd Qu.:192.0   3rd Qu.:1389   3rd Qu.:17.00
##                     Max.   :695.0   Max.   :4983   Max.   :23.00
##                     NA's   :9430
##      minute        time_hour
##  Min.   : 0.00   Min.   :2013-01-01 05:00:00.00
##  1st Qu.: 8.00   1st Qu.:2013-04-04 13:00:00.00
##  Median :29.00   Median :2013-07-03 10:00:00.00
##  Mean   :26.23   Mean   :2013-07-03 05:22:54.64
##  3rd Qu.:44.00   3rd Qu.:2013-10-01 07:00:00.00
##  Max.   :59.00   Max.   :2013-12-31 23:00:00.00
##
```

Some things that I found interesting:

- `dep_delay`, `arr_delay`

  - I'm sure there's some interesting relationship between these and some of the other variables like `time_hour` and `origin`
  - Both `dep_delay` and `arr_delay` have medians that are lower than their means, meaning that they are both right skewed.

- `time_hour`
  - The documentation mentions how the data from the `origin` column and the data from this column can be used to join flights data to weather data, which could allow for more accurate delay analysis

Question 6: Do flights pick up time in the air?

We use the `mean()` function to find the means of our desired columns, namely the `dep_delay` and `arr_delay` columns. To keep the page concise, we will round our values to **2 decimal places** of accuracy.

```
paste("Mean of dep_delay:", round(mean(flights$dep_delay, na.rm = TRUE), 2))
```

```
## [1] "Mean of dep_delay: 12.64"
```

```
paste("Mean of arr_delay:", round(mean(flights$arr_delay, na.rm = TRUE), 2))
```

```
## [1] "Mean of arr_delay: 6.9"
```

From these numbers, we can see that planes on average (*mean*) spend less time in the air than we expect them to. To calculate how much less time, we can do these calculations:

Time in air = Arrival time - Departure time
Expected time in air+Δt = (Expected arrival time + Arrival delay)- (Expected departure time + Departure delay)
Δt = Arrival delay - Departure delay
$mean_{\Delta t} = mean_{arr\_delay} - mean_{dep\_delay}$
$\bar{\Delta t} \approx 6.895 - 12.639 = -5.744$ mins

From these calculations we can see that planes on average (mean) pick up about 5.744 minutes while in the air, which I did not expect. However, when I thought back to my recent flights, this seems to hold.