PSTAT 10 Homework 2

Due 7/6/22 11:59pm

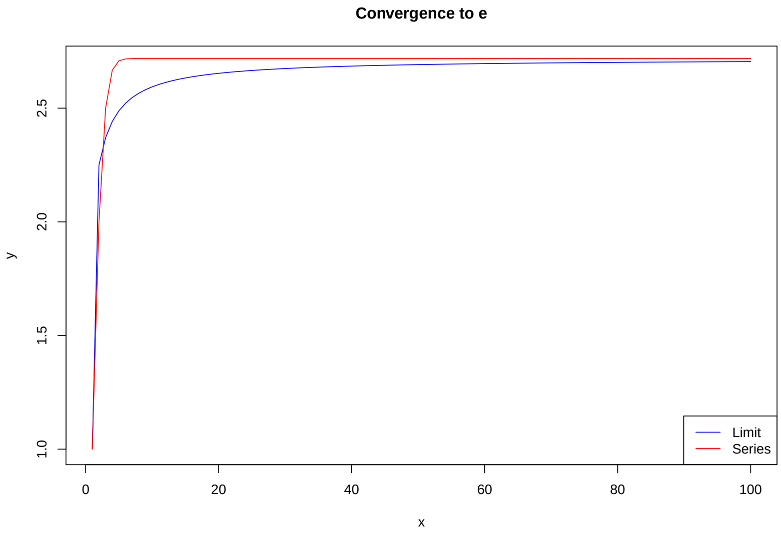
Problem 1: Convergence to e

The number $e \approx 2.718$ can be expressed in many different ways. One way is as a *limit* and another is as an *infinite series*:

$$e = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n \qquad \text{limit representation}$$

$$e = \sum_{k=0}^{\infty}\frac{1}{k!} \qquad \text{series representation}$$

Create two vectors containing these values at the points $x = 1, 2, \ldots, 100$. Then create the following plot in base R to assess the convergence of these two representations.



Hint: One way is to start with the following:

```
x <- 1:100
e_limit <- vector(length = length(x))
e_limit[1] <- 1
e_series <- vector(length = length(x))
e_series[1] <- 1
```

Then you can use a `for` loop to fill in the values.

Problem 2: nycflights13

For this problem, we will work with the `flights` data set in the `nycflights13` package. Install and load the `nycflights13` package. Also make sure you load `tidyverse` in case you need it.

```
library(nycflights13)
library(tidyverse)
```

1. Provide a brief description of the data set and a few of the variables (use `?flights`). Is `flights` a tibble?
2. Extract a tibble containing American Airlines (AA) flights to LAX that departed before 1030. Return only the columns `month`, `day`, `dep_time`, `dest`, and `carrier`. How many flights total fit these criteria?
3. Extract a tibble containing all flights on Christmas day; 12/25/2013. What is the **total number of miles** traveled across all flights on this day?
4. The `air_time` variable gives the duration of the flight in minutes. Create a tibble containing flights on Christmas day and only the variables `month`, `day`, `origin`, `dest`, and `air_time_hour` where the last variable gives the duration of the flight in *hours*.

```
## # A tibble: 922 x 5
##    month   day origin dest  air_time_hour
##    <int> <int> <chr>  <chr>         <dbl>
##  1     1    25 JFK    RIC           0.933
##  2     1    25 JFK    SYR           0.75
##  3     1    25 LGA    CLT           1.37
##  4     1    25 EWR    ALB           0.5
##  5     1    25 JFK    PHL           0.45
##  6     1    25 EWR    CLT           1.32
##  7     1    25 EWR    IAH           3.32
##  8     1    25 LGA    IAH           3.47
##  9     1    25 JFK    MIA           2.4
## 10     1    25 JFK    BQN           3.08
## # ... with 912 more rows
```

Problem 3: mtcars

For this problem we will make some plots with the `mtcars` data set in the `datasets` library.

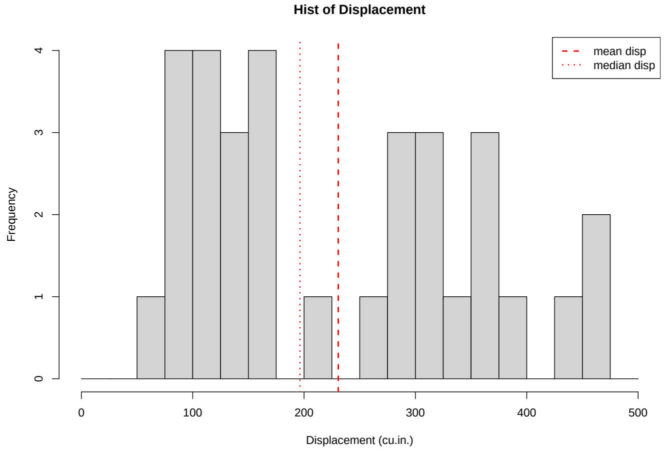
```
library(datasets)
```

If you're not happy with the size and scale of your plots, these can be changed via RMarkdown options. For example, my barplot in part 4 of this problem uses the following options:

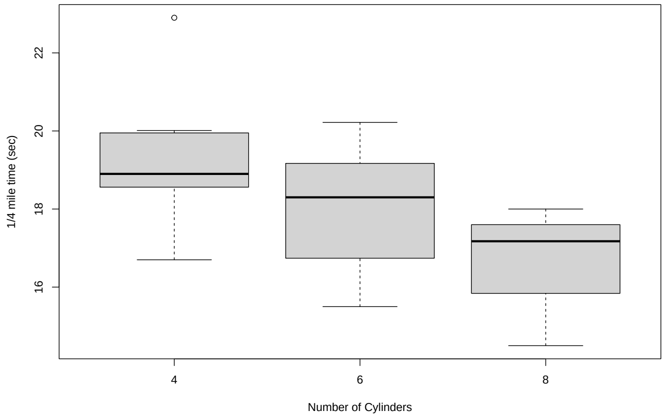
`{r, fig.align = "center", fig.dim=c(4.5, 4.5), out.width="40%", out.height="40%"}`

`fig.dim` specifies the dimensions of the figure while the `out` parameters are used to scale the figure.

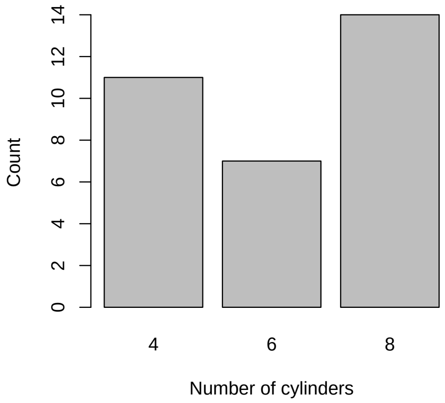
1. Provide a brief description of this data set and some of the variables. Is it a tibble?
2. Create the following histogram of horsepower along with vertical lines indicating the mean and median horsepower. *Hint*: I set the `breaks` to go from 0 to 500 in increments of 25.



3. Create the following boxplot of the 1/4 mile time against the number of cylinders. There is one outlier. Which car is the outlier? To answer this, note that the names of the cars are row names in the data set.



4. Create the following barplot of counts of cars with different numbers of cylinders.



Problem 4: Search insert position

Write a function `search_insert_position(v, target)` which takes a sorted numerical vector `v`, a numerical target `target`. If `target` is present in `v`, return the index of `target`. Otherwise, return the index in `v` of target where it would be if it were inserted in order.

The input `v` is guaranteed to be sorted and to contain unique values (i.e. no duplicates).

```
x <- c(1, 3, 5, 6)
search_insert_position(x, 5)
```

```
## [1] 3
```

```
search_insert_position(x, 2)
```

```
## [1] 2
```

```
search_insert_position(x, 7)
```

```
## [1] 5
```

Hint: My solution considers various cases separately. I used the functions `%in%` that tests membership, `which` to find TRUE indices of a logical vector, and `all` which tests if all entries of a logical vector are TRUE. Pay particular attention to the final test case above.

Testing membership with `%in%`:

```
"cat" %in% c("dog", "cow", "cat", "owl")
```

```
## [1] TRUE
```

```
12 %in% c(3, 6, 1, 0)
```

```
## [1] FALSE
```

Using `all`:

```
all(c(T, T, T))
```

```
## [1] TRUE
```

```
all(c(T, T, F, T))
```

```
## [1] FALSE
```