# Homework Solutions

CME/STAT 195, Fall 2018

*Assigned: October 6, 2018*

*Due: October 19, 2018 at 11:59pm*

## Instructions

This assignment is due on Friday October 19, 2018 at 11:59pm. You must submit your homework as report including a write up and all relevant code, and associated outputs it generated. You should learn how to use R Markdown to generate such a report and render it either as an HTML, PDF or Word Document. You can refer to chapter 3 of R Markdown: The Definitive Guide to review details on generating reports using R Markdown. If you generate your report with R Markdown, submit both the ".Rmd" file AND a rendered document.

Remarks:

- This homework covers the material of lectures 1-6, so you are encouraged to start working on it early and continue gradually as we advance into the class.
- If you generate random numbers, set a seed and report it, so that all your work can be perfectly reproduced.
- Do not print an entire `data.frame` or a whole vector with more than 20 entries in your homework, since it would produce an unnecessarily large file, when you render your document.
- Clearly label your final answer to the questions asked instead of only producing output of your code. Your answers should be as concise as possible and your code should be easily understandable.
- You are welcome to work with other students, but you must write your own code, and prepare your own writeup separately. If you choose to collaborate, indicate who you worked with on your submission.
- You are free to use the web or any books to look up the documentation of relevant R functions and debug errors, but you cannot look for the solution to the specific homework problems.
- If you have any questions regarding the homework, please ask on canvas or at office hours.

## Exercise 1: R Basics [20 pt]

**Material: Lecture 1.**

### a. Arithmetic operations [5pt]

Compute the following using R:

- $4.84 \log_{10}(51!)$, where $x!$ is factorial of $x$
- $4.02 \sqrt[3]{7^2 + e^7}$
- $20 \cos(2\pi + 0.25) + 32 \sin\left(\frac{3\pi}{4}\right)$
- $\left\lfloor \frac{4.011\pi}{3} \binom{5}{2} \right\rfloor$, where $\lfloor x \rfloor$ means rounding to the largest integer not greater than x. where $\binom{x}{y}$ is the notation for combination
- $8.1 \sum_{i=1}^{100} \frac{1}{i}$

```r
4.84 * log10(factorial(51))
```

```
## [1] 320.3627
```

```r
4.02 * (7^2 + exp(7))^{1/3}
```

```
## [1] 42.06374
```

```r
20 * cos(2*pi + 0.25) + 32 * sin(3 * pi / 4)
```

```
## [1] 42.00567
```

```r
8.1 *sum(1/seq_len(100))
```

```
## [1] 42.01776
```

## b. Matrix operations [5pt]

Generate a matrix $A$ with 15 rows and 5 columns with entries being random uniform numbers on an interval $[0, 1]$. Then generate a matrix $B$ with 5 rows and 7 columns where entries drawn from a Gaussian distribution with mean 0 and variance 10. Use `set.seed()` function with a chosen seed (record the seed) for reproducibility. Type in `?set.seed` in the R console to learn more about the function. With the two matrices compute:

- $AB$ (a matrix product)
- multiply the 3rd row of $A$ by the 4th column of $B$ and compute the sum of entries in the resulting vector, then check that agrees with a corresponding term in the matrix product you computed in the previous part.
- obtain a vector which is a product of matrix multiplication between matrix $A$ and the 4th column of $B$.

```r
set.seed(123)
(A <- matrix(runif(15*5), nrow = 15))
```

```
##            [,1]       [,2]       [,3]       [,4]         [,5]
##  [1,] 0.2875775 0.89982497 0.96302423 0.13880606 0.6651151946
##  [2,] 0.7883051 0.24608773 0.90229905 0.23303410 0.0948406609
##  [3,] 0.4089769 0.04205953 0.69070528 0.46596245 0.3839696378
##  [4,] 0.8830174 0.32792072 0.79546742 0.26597264 0.2743836446
##  [5,] 0.9404673 0.95450365 0.02461368 0.85782772 0.8146400389
##  [6,] 0.0455565 0.88953932 0.47779597 0.04583117 0.4485163414
##  [7,] 0.5281055 0.69280341 0.75845954 0.44220007 0.8100643530
##  [8,] 0.8924190 0.64050681 0.21640794 0.79892485 0.8123895095
##  [9,] 0.5514350 0.99426978 0.31818101 0.12189926 0.7943423211
## [10,] 0.4566147 0.65570580 0.23162579 0.56094798 0.4398316876
## [11,] 0.9568333 0.70853047 0.14280002 0.20653139 0.7544751586
## [12,] 0.4533342 0.54406602 0.41454634 0.12753165 0.6292211316
## [13,] 0.6775706 0.59414202 0.41372433 0.75330786 0.7101824014
## [14,] 0.5726334 0.28915974 0.36884545 0.89504536 0.0006247733
## [15,] 0.1029247 0.14711365 0.15244475 0.37446278 0.4753165741
```

```r
(B <- matrix(rnorm(5*7), nrow = 5))
```

```
##            [,1]       [,2]        [,3]       [,4]       [,5]       [,6]
## [1,] -0.7717918 -0.1639310 -0.88643672  1.2283928 -1.0700682  2.1499193
## [2,]  0.2865486  1.2429188 -1.31893760  0.2760235  1.6858872 -1.3343536
## [3,] -1.2205120 -0.9343851  0.02884391 -1.0489755 -0.2416898  0.4958705
## [4,]  0.4345504  0.3937087 -0.43212979 -0.5208693 -0.4682005  1.2339762
## [5,]  0.8001769  0.4036315  1.68987252  1.6232025 -0.7729782  0.6343621
##              [,7]
## [1,]  0.4120223
## [2,]  0.7935853
## [3,] -0.1524106
## [4,] -0.2288958
## [5,] -0.9007918
```

```
A %*% B
```

```
##              [,1]       [,2]        [,3]        [,4]        [,5]
##  [1,] -0.54696100  0.4945416 -0.34997723  0.59875898  0.39741414
##  [2,] -1.46200377 -0.5364271 -0.93776350  0.12235002 -0.82915700
##  [3,] -0.63687921 -0.3217164  0.04942006  0.17001573 -1.04862599
##  [4,] -1.22328391 -0.2649823 -0.84356295  0.64762342 -0.92092895
##  [5,]  0.54225092  1.6757487 -1.08594063  2.26841910 -0.43446187
##  [6,]  0.01538873  0.8507910 -0.46171788  0.50445955  0.96728449
##  [7,] -0.29442164  0.5668977 -0.18220196  1.12891554 -0.41363278
##  [8,]  0.22787454  1.0900441 -0.60202400  1.94856665 -0.92944698
##  [9,]  0.15955556  1.2167101 -0.50135350  1.84394138  0.33816994
## [10,]  0.14848131  0.9220881 -0.76205696  0.92067796 -0.04176089
## [11,] -0.01627499  0.9762041 -0.59284231  2.33823460 -0.54377876
## [12,] -0.14102988  0.5187522 -0.09929077  1.22712445 -0.21414073
## [13,]  0.03797335  0.8240534 -0.49773536  1.32272823 -0.72503793
## [14,] -0.41983350  0.2735253 -1.26406796 -0.06886378 -0.63395587
## [15,]  0.31971841  0.3628176  0.36053506  0.58361683 -0.44169676
##             [,6]        [,7]
##  [1,]  0.4883263  0.05489868
##  [2,]  2.1615697  0.24379870
##  [3,]  1.9842078 -0.35591894
##  [4,]  2.3575653  0.19477566
##  [5,]  2.3358044  0.21104795
##  [6,] -0.5110159  0.23736406
##  [7,]  1.6465757 -0.17912450
##  [8,]  2.6724801 -0.07165387
##  [9,]  0.6709314  0.22430799
## [10,]  1.1928069  0.14859630
## [11,]  1.9159599  0.20785261
## [12,]  1.0107418 -0.04062336
## [13,]  2.2491572 -0.12453488
## [14,]  2.1330349  0.20375974
## [15,]  0.8641720 -0.37795392
```

```
sum(A[3, ] * B[, 4])
```

```
## [1] 0.1700157
```

```
A %*% B[, 4]
```

```
##              [,1]
##  [1,]  0.59875898
##  [2,]  0.12235002
##  [3,]  0.17001573
##  [4,]  0.64762342
##  [5,]  2.26841910
##  [6,]  0.50445955
##  [7,]  1.12891554
##  [8,]  1.94856665
##  [9,]  1.84394138
## [10,]  0.92067796
## [11,]  2.33823460
## [12,]  1.22712445
## [13,]  1.32272823
## [14,] -0.06886378
## [15,]  0.58361683
```

## c. Factors [5pt]

In this part of the exercise we use a built-in data set, `sleep`, on student's sleep. This data stores information on the increase in hours of sleep, type of drug administered, and the patient ID, in respective columns. You can learn more about this data set from the page, accessed by calling `?sleep`.

The column `ID` for patient identity is a factor with labels from 1 to 10. Rename the ID labels to letters of the alphabet in the reverse order, with label "A" assigned to patient 10, "B" to patient 9, "C" to patient 8, . . . , and "J" to patient 1.

```
sleep <- sleep %>%
    mutate(ID_NEW = factor(ID, levels = seq(10, 1),
                           labels = LETTERS[seq_len(10)]))
head(sleep)
```

```
##   extra group ID ID_NEW
## 1   0.7     1  1      J
## 2  -1.6     1  2      I
## 3  -0.2     1  3      H
## 4  -1.2     1  4      G
## 5  -0.1     1  5      F
## 6   3.4     1  6      E
```

## d. Tibbles [5pt]

Create a tibble, 'birthdays', which stores information on the birthdays of 5 people either real of fictional. The data table should have columns:

1. 'first': first name
2. 'last': last name
3. 'birthday': the person's birthday in format YYYY-MM-DD ("%Y-%m-%d")
4. 'city': city where the person lives

Convert the birthdays to date objects using `as.Date()` function. Compute the difference (in days) between your birthday and the birthday of each of the people and append that information as a new column 'bday_diff' of the data-frame.

```
birthdays <- tibble(
    first = c("Harry", "Ron", "Hermione", "Ginny", "Draco"),
    last = c("Potter", "Weasley", "Granger", "Weasley", "Malfoy"),
    birthday = c("1980-07-31", "1980-03-01", "1979-09-19", "1981-08-11",
                 "1980-06-05"),
    city = c("Hogwarts", "Hogwarts", "Hogwarts", "Hogwarts", "Hogwarts")
    ) %>%
    mutate(
        birthday = as.Date(birthday),
        bday_diff = birthday - as.Date("1926-12-31"))
birthdays
```

```
## # A tibble: 5 x 5
##   first    last    birthday   city     bday_diff
##   <chr>    <chr>   <date>     <chr>    <time>
## 1 Harry    Potter  1980-07-31 Hogwarts 19571 days
## 2 Ron      Weasley 1980-03-01 Hogwarts 19419 days
## 3 Hermione Granger 1979-09-19 Hogwarts 19255 days
## 4 Ginny    Weasley 1981-08-11 Hogwarts 19947 days
## 5 Draco    Malfoy  1980-06-05 Hogwarts 19515 days
```

## Exercise 2: Programming [20pt]

**Material: Lecture 2.**

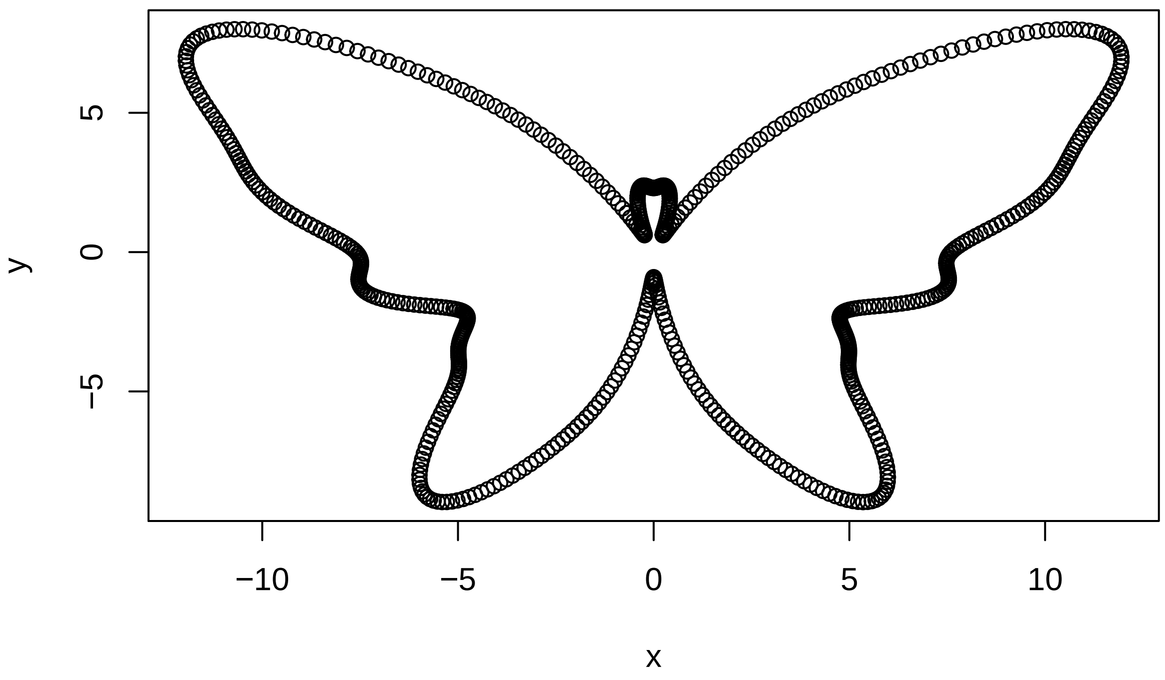
### a. Parametric function [5pt].

- Write a function in are that evaluates the following:

$$\begin{aligned} f(\theta) = 7 - 0.5\sin(\theta) + 2.5\sin(3\theta) + 2\sin(5\theta) - 1.7\sin(7\theta) + \\ + 3\cos(2\theta) - 2\cos(4\theta) - 0.4\cos(16\theta) \end{aligned}$$

- Generate a vector, `theta`, equal to a sequence from 0 to $2\pi$ with increments of 0.01
- Compute a vector $x = f(\theta) \cdot \cos(\theta)$ and $y = f(\theta) \cdot \sin(\theta)$ for $\theta$ you just created.
- Plot a scatter plot of (x, y) with two vectors computed.

```
f <- function(theta) {
    7 - 0.5 * sin(theta) + 2.5 * sin(3 * theta) + 2 * sin(5 * theta) -
        1.7 * sin(7 * theta) + 3 * cos(2 * theta) - 2 * cos(4 * theta) -
        0.4 * cos(16 * theta)
}
theta <- seq(0, 2*pi, by = 0.01)
x <- f(theta) * cos(theta)
y <- f(theta) * sin(theta)
plot(x, y)
```



### b. Multiple arguments [10pt]

Write a function `time_diff()` that takes two dates as inputs and returns the difference between them in units of "hours", "days", "weeks", "months", or "years", defined by an optional argument 'units', set by default to "days". Use the function to compute time left to your next birthday separately in units of: months, days, and hours.

```r
time_diff <- function(d1, d2, units = "days") {
    time_diff <- as.numeric(as.Date(d1) - as.Date(d2))
    switch(
        units,
        "hours" = time_diff * 24,
        "days" = time_diff,
        "weeks" = time_diff / 7,
        "months" = round(time_diff / (365.25/12)),
        "years" = round(time_diff / 365.25)
    )
}

t_h <- time_diff(as.Date("2019-08-23"), Sys.Date(), "hours")
t_days <- time_diff(as.Date("2019-08-23"), Sys.Date())
t_months <- time_diff(as.Date("2019-08-23"), Sys.Date(), "months")


cat("There are", t_h , "hours left to my next birthday.\n")
```

```
## There are 7224 hours left to my next birthday.
```

```r
cat("There are", t_days , "days left to my next birthday.\n")
```

```
## There are 301 days left to my next birthday.
```

```r
cat("There are", t_months , "months left to my next birthday.\n")
```

```
## There are 10 months left to my next birthday.
```

There are 11 months

### c. Control flow: Fibonacci numbers [5pt].

Fibonacci sequence starts with numbers 1 and 2, and each subsequent term is generated by adding the previous two terms. The first 10 terms of the Fibonacci sequence are thus: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, . . . . Find the total sum of **even numbers** in the Fibonacci sequence, each not exceeding one million.

```r
prev_fib <- 1; fib <- 2;
even_fib_sum <- fib
while(fib <= 1e6) {
    new_fib <- prev_fib + fib
    prev_fib <- fib
    fib <- new_fib
    if (fib %% 2 == 0) {
        even_fib_sum <- even_fib_sum + fib
    }
}
even_fib_sum
```

```
## [1] 1089154
```

The sum of even numbers in the Fibonacci sequence not greater than 1M is 1,089,154.

## Exercise 3: Data Import/Export and transformation [20pt]

**Material: Lecture 3 and 5.**

### a. Import data [5pt]

Visit the following URL: https://raw.githubusercontent.com/cme195/cme195.github.io/master/assets/data/share-of-people-who-say-they-are-happy.txt

Observe the structure and format of the data. Then, use a function from `readr` package to read the data in the URL into R. Then, find the country with the highest share of happy people in 2014.

```r
url <- "https://raw.githubusercontent.com/cme195/cme195.github.io/master/assets/data/share-of-people-wh
happy <- read_delim(url, ";")
```

```
## Parsed with column specification:
## cols(
##   Entity = col_character(),
##   Code = col_character(),
##   Year = col_integer(),
##   ShareOfHappyPeople = col_double()
## )
```

## b. Export data [5pt]

Filter observations from the data set on happiness that correspond to years after 2000. Export the subset of the data as a tab-delimited text file to a chosen location on your computer.

```
write_tsv(happy %>% filter(Year > 2000), "../path/to/chosen/destination/file.tsv")
```

## c. `dplyr` functions [10pt]

In this exercise we use the package, `nycflights13`, storing datasets on flights and airports in the city of New York in 2013. Install the package with `install.packages("nycflights13")` if you have not done so already, then load it with `library(nycflights13)`.

The dataset 'fights' is a tibble with 336,776 observations! To learn about the details about this dataset, type `?flights` in your R console.

Use `dplyr` functions (and the `%>%` operator) to compute, for each combination of departure airport,'origin', destination airport, 'dest', and 'carrier': the average 'dep_delay', the average 'air_time' and the average ratio 'dep_delay'/'air_time'

For each 'carrier' report the route ('origin'-'dest') with the highest mean ratio of departure delay over air time. Now, you know which flights not to take with a given carrier.

**Note**: Since, the dataset contains missing values, when computing the averages, remember to exclude the missing values (use 'na.rm = TRUE' in `mean()` function).

```
library(nycflights13)
df <- flights %>%
    group_by(dest, origin, carrier) %>%
    summarise(
        mean_dep_delay = mean(dep_delay, na.rm = TRUE),
        mean_air_time = mean(air_time, na.rm = TRUE),
        mean_ratio  = mean(dep_delay/air_time, na.rm = TRUE)
    )

df %>%
    group_by(carrier) %>%
    top_n(1, mean_ratio)
```

```
## # A tibble: 16 x 6
## # Groups:   carrier [16]
##    dest  origin carrier mean_dep_delay mean_air_time mean_ratio
##    <chr> <chr>  <chr>            <dbl>         <dbl>      <dbl>
##  1 ATL   JFK    EV              124           118       1.05
##  2 BOS   JFK    B6               13.5          38.5     0.347
##  3 BWI   EWR    WN               10.4          37.6     0.286
##  4 CAK   LGA    FL               20.8          64.0     0.326
##  5 DEN   LGA    F9               20.2         230.      0.0892
##  6 DTW   EWR    OO               61            84.5     0.689
##  7 HNL   JFK    HA                4.90        623.      0.00776
##  8 IAD   LGA    YV               21.0          48.5     0.436
##  9 ORD   JFK    AA               24.7         122.      0.201
## 10 ORF   JFK    MQ               21.5          52.6     0.405
## 11 PHL   JFK    9E               18.0          30.8     0.573
## 12 PHL   JFK    US                3.79         31.9     0.119
## 13 PIT   LGA    DL               22.1          57.6     0.403
## 14 RDU   EWR    UA                60            62       0.968
## 15 SEA   EWR    AS                 5.80        326.      0.0180
## 16 SFO   JFK    VX                19.0         350.      0.0548
```

### d. Joining/merging datasets [5pt]

The `nycflights13` package also includes datasets other than `flights`. In this exercise you will combine data tables together.

- First, drop columns that contain delay (ending with '*delay'), and scheduling (starting with 'sched*') information, and save as a new tibble `flights2`.
- Add a column with the full name of carriers operating the flights to `flights2` by merging it with the `airline` tibble. Which column is(are) used for joining?
- Another dataset available in the package is `weather`, storing data on weather at different airports at specific days and times. Use this dataset to merge the weather information to the data in the previous step. Which column is(are) used for joining?
- There, is also a data-frame `planes` included in the package. `planes` share columns 'year' and 'tailnum' in `planes` with `flights2` data-frame, but column 'year' in `planes` means a different thing (year produced) than in `flights2`(year of flight). Use only the column 'tailnum' to merge `flights2` and `planes`. Note that the 'suffix' argument can be used to set different names to distinguish between year of production and year of of the flight.
- Now, use the data-frame `airports` to merge `flights2` with the information **on the origin airport**. Note that the column 'faa' is the airport identifier in the dataset `airports`. You must use the `by =` argument in the join function and specify which columns from `flights2` you are matching to which column in `airports`.

```
flights2 <- select(
    flights, -starts_with("sched_"), -ends_with("_delay"))
```

We use "carrier" to join `flights2` and `airlines`.

```
flights2 <- flights2 %>%
    left_join(airlines)
```

```
## Joining, by = "carrier"
```

We use "year", "month", "day", "origin", "hour", and "time_hour" to join `flights2` and `airlines`.

```
flights2 <- flights2 %>%
    left_join(weather)
```

```
## Joining, by = c("year", "month", "day", "origin", "hour", "time_hour")
```

```
flights2 <- flights2 %>%
    left_join(planes, by = c("tailnum"), suffix = c(".flight", ".production"))
```

```
flights2 <- flights2 %>%
    left_join(airports, by = c("origin" = "faa"))
```

## Exercise 4: Data Visualization [20pt]

**Material: Lecture 3-4.**

The following url contains data on fossil fuel emissions for different countries between 1751 and 2014: "http://cdiac.ess-dive.lbl.gov/ftp/ndp030/CSV-FILES/nation.1751_2014.csv"

## a. Import data with `readr` [5pt]

This dataset is messy, and you will need to fix the warning messages `read_csv()` returns.

- Rows 1-3 in contain information on the dataset itself, and not the variables; so after reading the data in, we need to delete these rows.
- The datasets contains characters "." which needs to be replaced with `NA` for missing data.
- Rename column `Total CO2 emissions from fossil-fuels and cement production (thousand metric tons of C)` to something shorter, e.g. 'Total_CO2'

```
url <- "http://cdiac.ess-dive.lbl.gov/ftp/ndp030/CSV-FILES/nation.1751_2014.csv"
emissions <- read_csv(url)
```

```
## Parsed with column specification:
## cols(
##   Nation = col_character(),
##   Year = col_integer(),
##   `Total CO2 emissions from fossil-fuels and cement production (thousand metric tons of C)` = col_in
##   `Emissions from solid fuel consumption` = col_integer(),
##   `Emissions from liquid fuel consumption` = col_integer(),
##   `Emissions from gas fuel consumption` = col_integer(),
##   `Emissions from cement production` = col_integer(),
##   `Emissions from gas flaring` = col_character(),
##   `Per capita CO2 emissions (metric tons of carbon)` = col_character(),
##   `Emissions from bunker fuels (not included in the totals)` = col_integer()
## )

## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)

## Warning: 615 parsing failures.
## row # A tibble: 5 x 5 col     row col                expected actual  file
## ... ................. ... ..........................................................................
## See problems(...) for more details.
```

```
emissions <- emissions[-(1:3), ]
emissions[emissions == "."] <- NA
emissions <- type_convert(emissions)
```

```
## Parsed with column specification:
## cols(
##   Nation = col_character(),
##   `Emissions from gas flaring` = col_integer(),
##   `Per capita CO2 emissions (metric tons of carbon)` = col_double()
## )
```

```
emissions <- emissions %>%
    rename(Emission_CO2 = `Total CO2 emissions from fossil-fuels and cement production (thousand metric
emissions
```

```
## # A tibble: 17,232 x 10
##    Nation  Year Emission_CO2 `Emissions from~ `Emissions from~
##    <chr>  <int>        <int>            <int>            <int>
##  1 AFGHA~  1949            4                4                0
##  2 AFGHA~  1950          23                6               18
##  3 AFGHA~  1951          25                7               18
##  4 AFGHA~  1952          25                9               17
##  5 AFGHA~  1953          29               10               18
##  6 AFGHA~  1954          29               12               18
##  7 AFGHA~  1955          42               17               25
##  8 AFGHA~  1956          50               17               33
##  9 AFGHA~  1957          80               21               59
## 10 AFGHA~  1958          90               25               65
## # ... with 17,222 more rows, and 5 more variables: `Emissions from gas
## #   fuel consumption` <int>, `Emissions from cement production` <int>,
## #   `Emissions from gas flaring` <int>, `Per capita CO2 emissions (metric
## #   tons of carbon)` <dbl>, `Emissions from bunker fuels (not included in
## #   the totals)` <int>
```

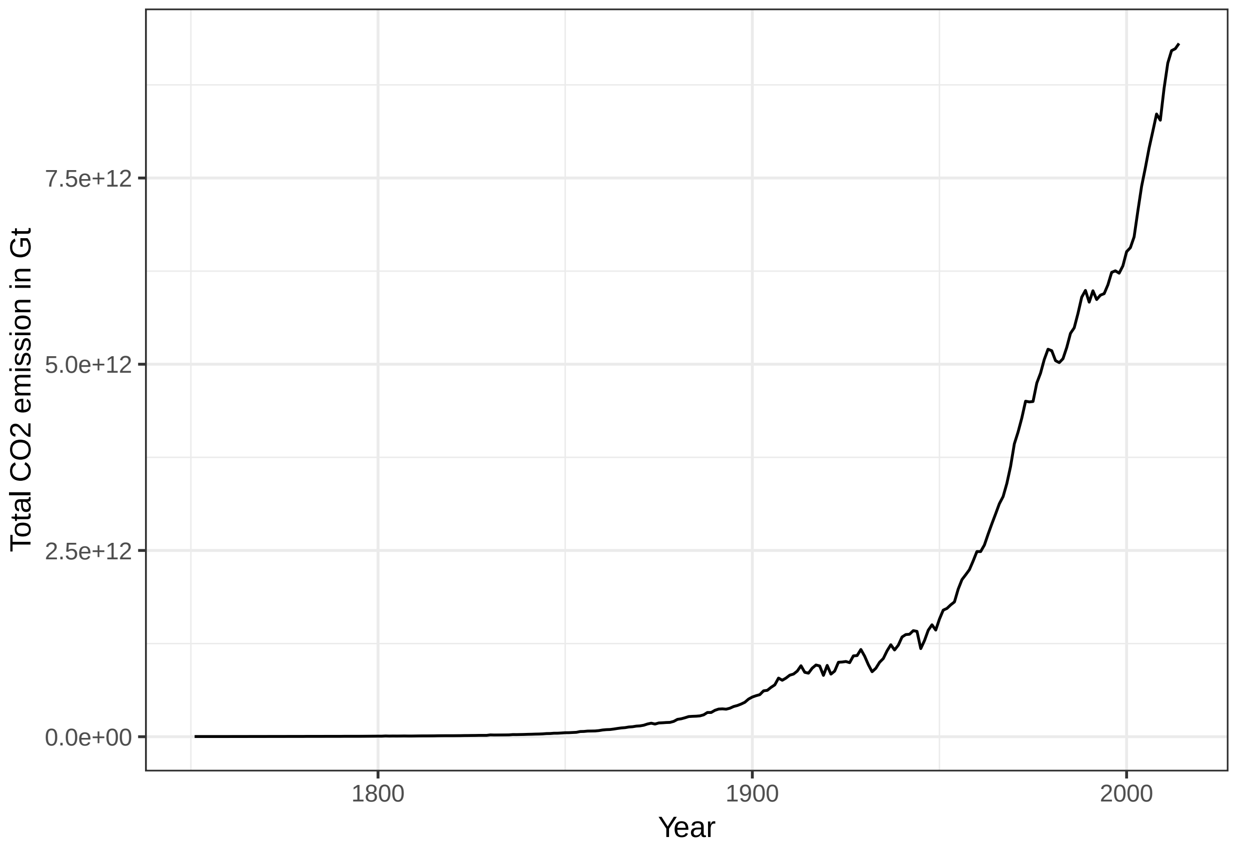
### b. Summarize data [5pt]

Use `dplyr` functions to compute the total yearly $CO_2$ emissions (column `Total.CO2.emissions.from.fossil.fuels.and.ceme` `tons.of.C.`) summed over all countries (the world total $CO_2$ emission). Use the dataset to plot the World's yearly $CO_2$ emission in Gt.

```r
yearlyCO2 <- emissions %>%
    group_by(Year) %>%
    summarise(total_CO2 = sum(`Emission_CO2`))
yearlyCO2
```

```
## # A tibble: 264 x 2
##     Year total_CO2
##    <int>     <int>
##  1  1751      2552
##  2  1752      2553
##  3  1753      2553
##  4  1754      2554
##  5  1755      2555
##  6  1756      2731
##  7  1757      2732
##  8  1758      2733
##  9  1759      2734
## 10  1760      2734
## # ... with 254 more rows
```

```r
ggplot(yearlyCO2) +
    geom_line(aes(x = Year, y = total_CO2/1e-6)) +
    ylab("Total CO2 emission in Gt") +
    theme_bw()
```



### c. Line plots [5pt]

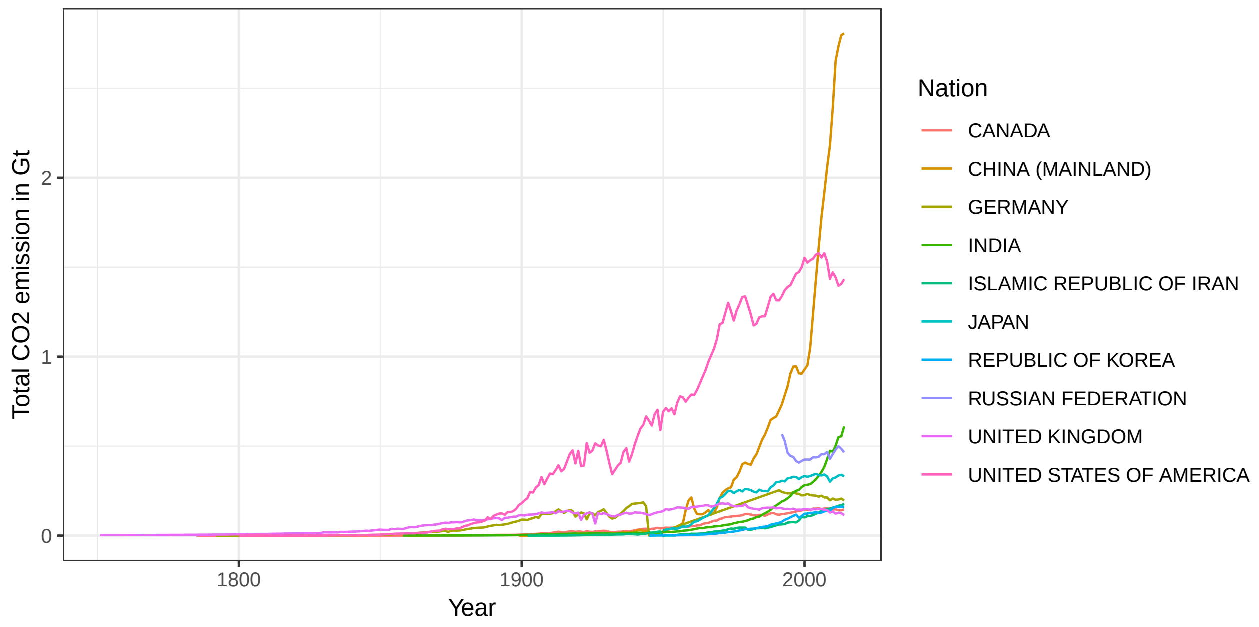
Find the top 10 countries with highest emission after year 2000 (including 2000).

```
top10 <- emissions %>%
    filter(Year >= 2000) %>%
    group_by(Nation) %>%
    summarise(CountryCO2Post200 = sum(Emission_CO2)) %>%
    top_n(10) %>%
    .[["Nation"]]
```

```
## Selecting by CountryCO2Post200
```

Plot the yearly total CO2 emissions of these top 10 countries with a different color for each country. Use billion tonnes (Gt) units, i.e. divide the total emissions by 10^6.

```
ggplot(
    emissions %>%
        filter(Nation %in% top10),
    aes(x = Year, y = Emission_CO2/1e6)) +
    geom_line(aes(color = Nation)) +
    ylab("Total CO2 emission in Gt") +
    theme_bw()
```



## d. Stacked plots [5pt]

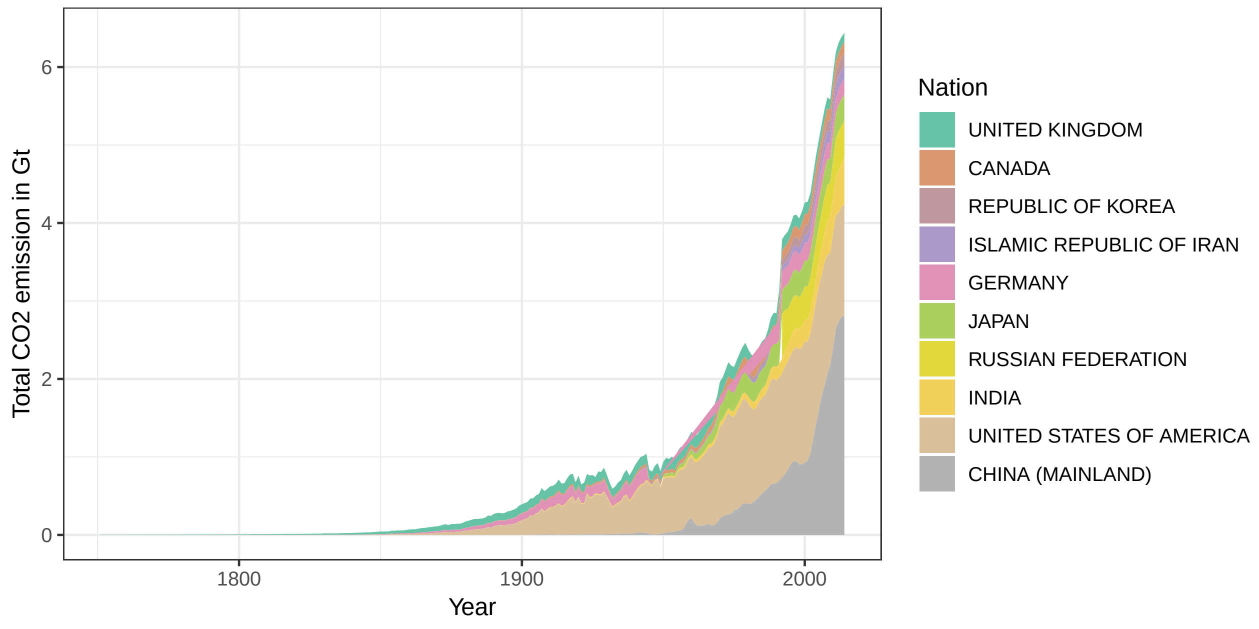
Use `geom_area()` to generate a plot similar to the one you generated above but, with emission levels stacked on top of each other (summing to the total for the ten countries) with areas colored by countries.

```
df <- emissions %>% filter(Nation %in% top10)
ordered_countries <- df %>%
    filter(Year == max(Year)) %>%
    arrange(Emission_CO2) %>% .[["Nation"]]
df <- df %>%
    mutate(Nation = factor(Nation, levels = ordered_countries))
```

```
library(RColorBrewer)
colors10 <- colorRampPalette(brewer.pal(9, "Set2"))(10)
```

```
## Warning in brewer.pal(9, "Set2"): n too large, allowed maximum for palette Set2 is 8
## Returning the palette you asked for with that many colors
```

```
ggplot(
    df,
    aes(x = Year, y = Emission_CO2/1e6)) +
    geom_area(aes(fill = Nation), position = "stack") +
    ylab("Total CO2 emission in Gt") +
    scale_fill_manual(values = colors10) +
    theme_bw()
```



## Exercise 5: Linear Models [20pt]

**Material: Lecture 4 and 6.**

In this exercise we will use a dataset containing information on sales of a product and the amount spent on advertising using different media channels. The data are available from: http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv.

### a. Import and plot the data [5pt]

Read the data from "http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv".

Then, generate a scatterplot of sales against the amount of TV advertising. Color the points by the mount of 'radio' advertising. Then, add a linear fit line.

```
url <- "http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv"
sales <- read_csv(url)
```

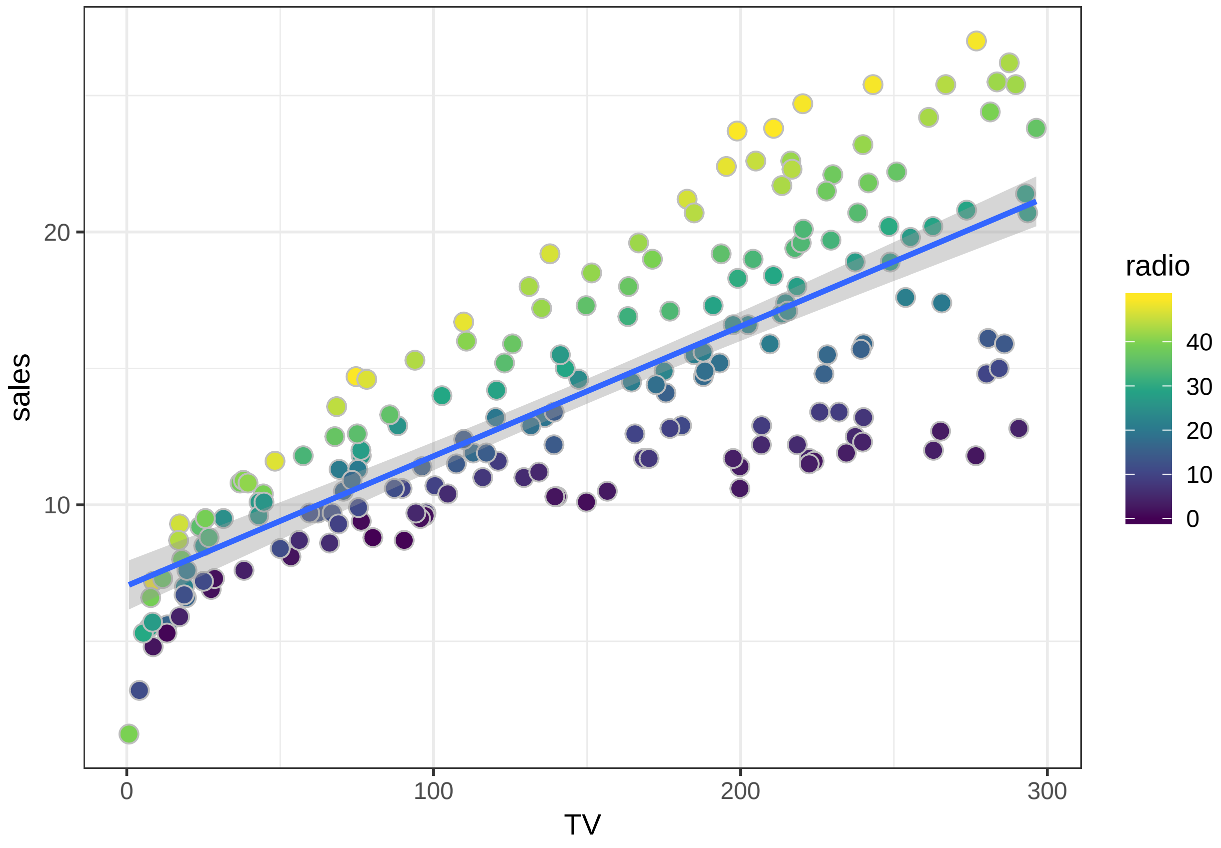
```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   TV = col_double(),
##   radio = col_double(),
##   newspaper = col_double(),
##   sales = col_double()
## )
```

```
sales
```

```
## # A tibble: 200 x 5
##       X1    TV radio newspaper sales
##    <int> <dbl> <dbl>     <dbl> <dbl>
##   1      1 230.   37.8      69.2  22.1
##   2      2  44.5  39.3      45.1  10.4
##   3      3  17.2  45.9      69.3   9.3
##   4      4 152.   41.3      58.5  18.5
##   5      5 181.   10.8      58.4  12.9
##   6      6   8.7  48.9      75     7.2
##   7      7  57.5  32.8      23.5  11.8
##   8      8 120.   19.6      11.6  13.2
##   9      9   8.6   2.1       1     4.8
##  10     10 200.    2.6      21.2  10.6
## # ... with 190 more rows
```

```r
ggplot(sales, aes(x = TV, y = sales)) +
    geom_point(aes(fill = radio), color = "grey", pch = 21, size = 3) +
    geom_smooth(method = "lm") +
    scale_fill_viridis_c() +
    theme_bw()
```



## b. Simple Linear Regression [5pt]

The dataset has 200 rows. Divide it into a train set with 150 observations and a test set with 50 observations, i.e. use `sample()` without replacement to randomly choose row indices of the advertising dataset to include in the train set. The remaining indices should be used for the test set.

Fit a linear model to the training set, where the sales values are predicted by the amount of TV advertising. Print the summary of the fitted model. Then, predict the sales values for the test set and evaluate the test model accuracy in terms of root mean squared error (MSE), which measures the average level of error between the prediction and the true response.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

```r
set.seed(123)
idx <- sample(1:200, size = 150)
train <- sales[idx, ]
test <- sales[-idx, ]

fit <- lm(sales ~ TV, train)
pred_test <- predict(fit, test)

RMSE <- sqrt(mean((pred_test - test$sales)^2))

cat("The RMSE for the simple regression model is:", RMSE, ".\n")
```

```
## The RMSE for the simple regression model is: 3.43658 .
```

## c. Multiple linear regression [5pt]

Fit a multiple linear regression model including all the variables ‘TV’, ‘radio’, ‘newspaper’ to model the ‘sales’ in the training set. Then, compute the predicted sales for the test set with the new model and evaluate the RMSE.
Did the error decrease from the one corresponding to the previous model?

```r
fit.mult <- lm(sales ~ TV + radio + newspaper, train)
pred_test <- predict(fit.mult, test)

RMSE <- sqrt(mean((pred_test - test$sales)^2))
cat("The RMSE for the multiple regression model is:", RMSE, ".\n")
```

```
## The RMSE for the multiple regression model is: 1.909814 .
```

Yes, the RMSE decrease after including more predictors in the model.

## d. Evaluate the model [5pt]

Look at the summary output for the multiple regression model and note which of the coefficient in the model is significant. Are all of them significant? If not refit the model including only the features found significant. Which of the models should you choose?

```r
summary(fit.mult)
```

```
##
## Call:
## lm(formula = sales ~ TV + radio + newspaper, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3581 -0.8916  0.2041  1.2089  2.6565
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.273848   0.339641   9.639   <2e-16 ***
## TV           0.044485   0.001544  28.816   <2e-16 ***
## radio        0.190370   0.009643  19.741   <2e-16 ***
## newspaper   -0.005491   0.006310  -0.870    0.386
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.612 on 146 degrees of freedom
## Multiple R-squared:  0.9038, Adjusted R-squared:  0.9018
## F-statistic: 457.1 on 3 and 146 DF,  p-value: < 2.2e-16
```

It seems like the 'newspaper' predictor is not significant, so we will remove it.

```
fit.mult2 <- lm(sales ~ TV + radio, train)
pred_test <- predict(fit.mult2, test)

RMSE <- sqrt(mean((pred_test - test$sales)^2))
cat("The RMSE for the new multiple regression model is:", RMSE, ".\n")
```

```
## The RMSE for the new multiple regression model is: 1.890502 .
```

The second multiple regression model has a slightly better test error. We should not use more predictors than necessary, so 'newspaper' should be discarded from the model.

```
sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.5 LTS
##
## Matrix products: default
## BLAS: /usr/lib/libblas/libblas.so.3.6.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] RColorBrewer_1.1-2 nycflights13_1.0.0 bindrcpp_0.2.2
##  [4] forcats_0.3.0      stringr_1.3.1      dplyr_0.7.7
##  [7] purrr_0.2.5        readr_1.1.1        tidyr_0.8.1
## [10] tibble_1.4.2       ggplot2_3.0.0      tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.5  haven_1.1.2       lattice_0.20-35
##  [4] colorspace_1.3-2  viridisLite_0.3.0 htmltools_0.3.6
##  [7] yaml_2.2.0        utf8_1.1.4        rlang_0.2.2
## [10] pillar_1.3.0      glue_1.3.0        withr_2.1.2
## [13] modelr_0.1.2      readxl_1.1.0      bindr_0.1.1
## [16] plyr_1.8.4        munsell_0.5.0     gtable_0.2.0
## [19] cellranger_1.1.0  rvest_0.3.2       evaluate_0.11
## [22] labeling_0.3      knitr_1.20        curl_3.2
## [25] fansi_0.3.0       broom_0.5.0       Rcpp_0.12.19
## [28] scales_1.0.0      backports_1.1.2   jsonlite_1.5
## [31] hms_0.4.2         digest_0.6.18     stringi_1.2.4
## [34] grid_3.5.1        rprojroot_1.3-2   cli_1.0.1
## [37] tools_3.5.1       magrittr_1.5      lazyeval_0.2.1
## [40] crayon_1.3.4      pkgconfig_2.0.2   xml2_1.2.0
## [43] lubridate_1.7.4   assertthat_0.2.0  rmarkdown_1.10
## [46] httr_1.3.1        rstudioapi_0.8    R6_2.3.0
## [49] nlme_3.1-137      compiler_3.5.1
```