MM916 Week 4 Question Sheet

Question 1

Write a function that takes a single vector as an argument and returns the minimum and maximum values in an appropriate format.

```
minMax <- function(x){
  minimum <- min(x)
  maximum <- max(x)
  list(minimum=minimum, maximum=maximum)
}
```

Question 2

Think about the function `is_odd()` that we looked at in lecture (or see activity 4.5). How might we edit this function to create a new function called `is_even()` that returns `TRUE` if the value passed in is even and `FALSE` if the value passed in is odd?

```
# The only thing that needs to be edited in the is_odd() function is the way that
# TRUE and FALSE are returned. We can do this using ! to denote not - this will
# change TRUE to FALSE and FALSE to TRUE. The function becomes:`

is_even <- function(x){
  # Find the remainder of x when divided by 2
  value <- x%%2
  # Make this a logical value. 1 = TRUE in R, so TRUE indicates that x is odd
  value <- as.logical(value)
  # Use ! (which means NOT) to flip it so that TRUE now means x is even
  value <- !value

  return(value)
}
```

Question 3

A temperature in Fahrenheit (F) can be calculated from a temperature in celsius (C) via the following equation:

$$F = \frac{9C}{5} + 32$$

a) Write a function that takes a vector of temperatures in celsius and converts to farenheit.

```
Cel2Far <- function(Cel){
   Far <- 9*Cel/5 + 32

   return(Far)
}
```

b) Write a function that takes a vector of temperatures in fahrenheit and converts to celsius.

```r
Far2Cel <- function(Far){
   Cel <- 5*(Far - 32)/9

   return(Cel)
}
```

c) Test the functions by verifying that 0°C = 32°F and 100°C = 212°F.

```r
# Should be 32
Cel2Far(0)
```

```
## [1] 32
```

```r
# Should be 0
Far2Cel(32)
```

```
## [1] 0
```

```r
# Should be 212
Cel2Far(100)
```

```
## [1] 212
```

```r
# Should be 100
Far2Cel(212)
```

```
## [1] 100
```

d) What other tests might be useful here?

```r
# Another thing that might be a useful test is that one function is the inverse
# of the other by running the result of one function through the other:`

# Should give 23.5
Cel2Far(Far2Cel(23.5))
```

```
## [1] 23.5
```

```r
# Should give 38.6
Far2Cel(Cel2Far(38.6))
```

```
## [1] 38.6
```

Question 4

The skewness of a variable can be calculated as follows:

$$\text{Skewness} = \frac{\frac{1}{n-2}\left(\sum_{i=1}^{n}(x_i - \bar{x})^3\right)}{s^3}$$

in the above, n is the number of observations, $\bar{x}$ is the sample mean and s is the sample standard deviation.

Create a function the takes a vector `x` as and argument and returns the skewness.

a) Write out pseudocode that breaks the calculation into steps.

```r
# The pseudocode might look something like:
# 1. Take x as input
# 2. Calculate the length of x
# 3. Calculate the mean of x
# 4. Calculate the standard deviation of x
```

```
# 5. Plug these values into the formula to calculate the skewness
# 6. Return it`
```

b) Write the function in R

```
  # 1. Take x as input
  skewness <- function(x){

  # 2. Calculate the length of x
  n <- length(x)

  # 3. Calculate the mean of x
  xbar <- mean(x)

  # 4. Calculate the standard deviation of x
  s <- sd(x)

  # 5. Plug these values into the formula to calculate the skewness
  skewness <- (sum((x - xbar)^3)/(n-2))/(s^3)

  # 6. Return it`
  return(skewness)
}
```

c) As a test, paste the following code into R and pass the object into your skewness function. It should give a value of 2.399649.

```
x <- c(0.2128, 0.0301, 0.0922, 0.0838, 0.0831, 0.1962, 0.1001, 0.0809, 0.0678, 0.0431, 0.0415, 0.1246, 0

skewness(x)
```

```
## [1] 2.399649
```

(continues next page)

Question 5

If a, b and c are the lengths of the sides of a triangle, then the area is given by $\sqrt{s(s-a)(s-b)(s-c)}$, where $s = \frac{1}{2}(a+b+c)$.

a) Write a function called `tri_area` which takes as input a, b and c and returns the area of the triangle.

```
#1. Take as input a, b and c
tri_area <- function(a,b,c){

#2. Calculate s
s <- 0.5*(a+b+c)

# 3. Calculate the area
area <- sqrt(s*(s-a)*(s-b)*(s-c))

# 4. Return the area
return(area)
}
```

b) Think about what some appropriate unit tests might be. What special cases need to be addressed?

```
# basic case: try a 3-4-5 right triangle, since it's easy to calculate the area by hand (3*4/2 = 6)
tri_area(3,4,5)
```

```
## [1] 6
```

```
# what if one of the sides is 0? should return 0
tri_area(0,4,4)
```

```
## [1] 0
```

```
# what if the combination of side lengths is impossible, like 1, 1, 100?
tri_area(1,1,100)
```

```
## Warning in sqrt(s * (s - a) * (s - b) * (s - c)): NaNs produced
```

```
## [1] NaN
```

```
# what if a side length is negative? this should probably give an error or warning
tri_area(3,4,-5)
```

```
## [1] 6
```

c) Rewrite your function using `if`, `warning`, and `stop` to deal with any special cases you identified that give NA or nonsensical answers.

```
tri_area <- function(a,b,c){
  if (a<0 | b<0 | c<0) {
    stop('all side lengths must be >= 0.')
  }
  s <- 0.5*(a+b+c)
  area <- sqrt(s*(s-a)*(s-b)*(s-c))
  if (is.na(area)) {
    warning('impossible triangle specified: returning NA.')
  }
  return(area)
}
```

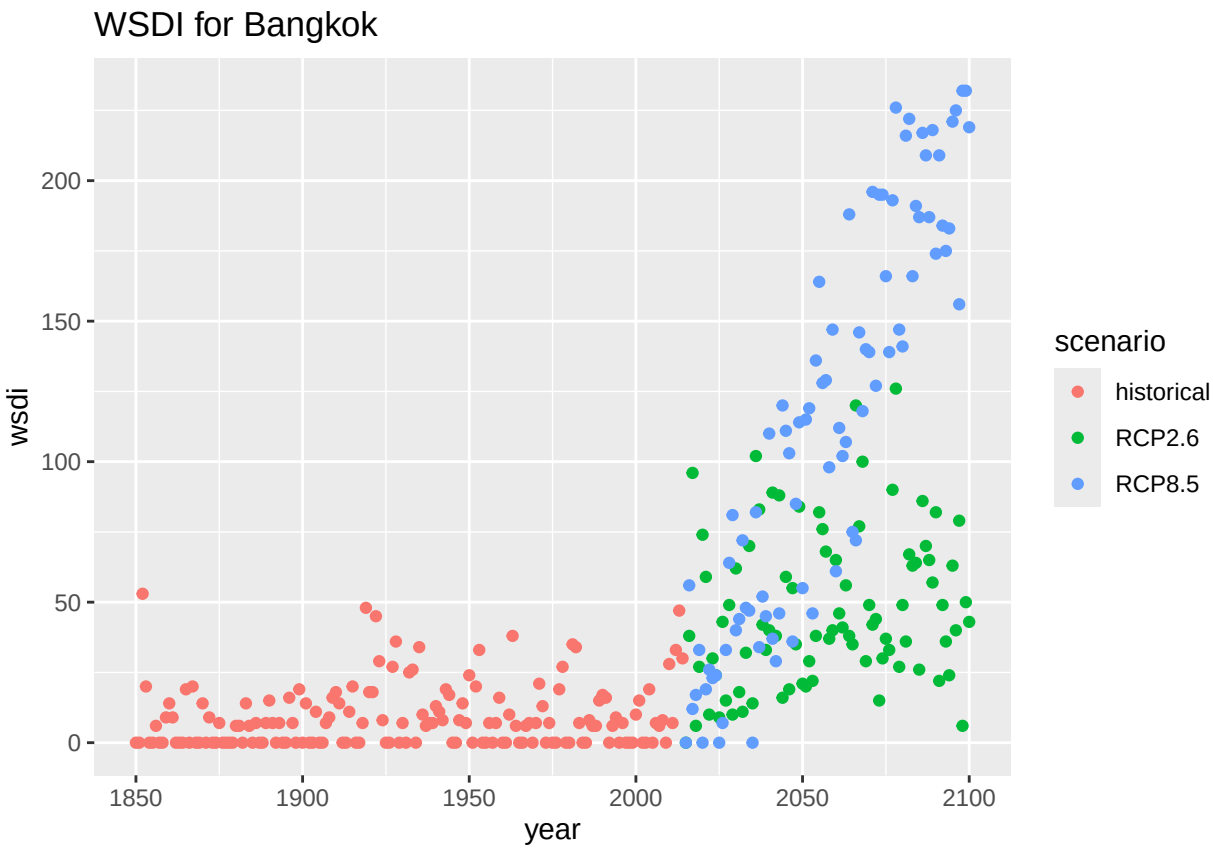
Question 6

The file `heatwave_index.RData` contains a data frame (`heatwaves`) giving a climate index called WSDI, or Warm Spell Duration Index, for world cities with populations over 1 million. This is a measure of how many days per year are part of an extended heatwave, compared to what was a normal temperature for that city in the 20th century. This dataset is based on output from a global climate model, run for three different scenarios:

- *RCP2.6*, an optimistic scenario
- *RCP8.5*, a more pessimistic scenario (but by no means the worst-case scenario)
- *historical*, an approximate reconstruction of past climate that can be used as a baseline.

a) Load the data into R (since it's an `.RData` file, use `load()` not a `read` function). To get a sense of the dataset, make a scatter plot of `wsdi` vs. `year` for Bangkok, color-coded by scenario.

```
load('heatwave_index.RData')
heatwaves |> filter(city=="Bangkok") |>
  ggplot() + geom_point(aes(year,wsdi,color=scenario)) +
  labs(title = "WSDI for Bangkok")
```



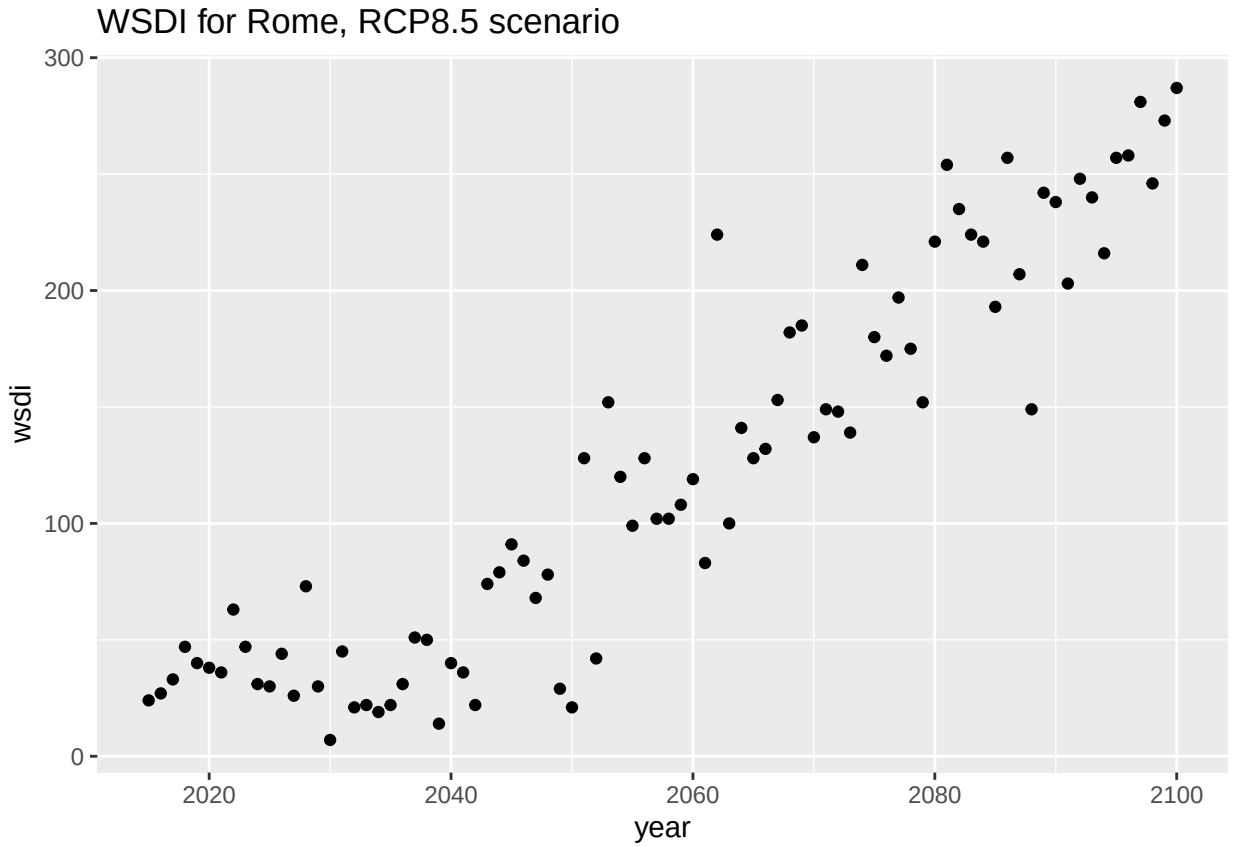
b) Now write a function called `heatwavesRCP85` that takes a city name as input and makes a time series plot similar to the one in part a) but for RCP8.5 only, for that city. Give the plot an appropriate title (you might want to use the `paste` or `paste0` function). Test it using the city of your choice.

```
load('heatwave_index.RData')

heatwavesRCP85 <- function(cityName, data) {
  data |> filter(scenario=="RCP8.5" & city==cityName) |>
    ggplot() + geom_point(aes(year,wsdi)) +
    labs(title = paste0("WSDI for ", cityName, ", RCP8.5 scenario")) -> gg
```

```
  return(gg)
}

heatwavesRCP85("Rome",heatwaves)
```



c) Modify the function so that the user can use either a city or a country name as input. If a country, the function shoulds make a single time-series plot based on the average across all cities in that country.

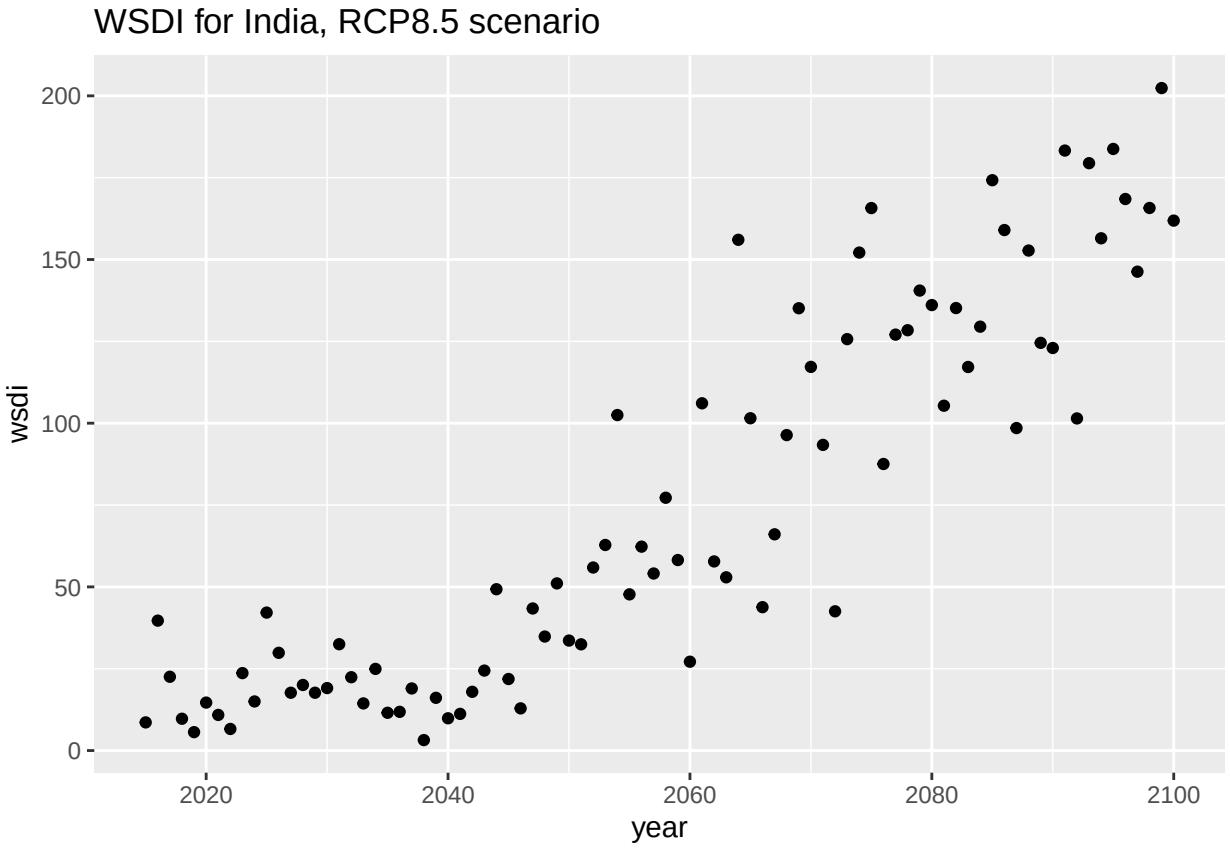
```
load('heatwave_index.RData')

heatwavesRCP85 <- function(name, data) {
   # subset the data to a single city or country
  if (name %in% data$city) {
    selectedData <- data |> filter(scenario=="RCP8.5" & city==name)
  } else if (name %in% data$country) {
    selectedData <- data |> filter(scenario=="RCP8.5" & country==name) |>
      group_by(year) |>
      summarise(wsdi=mean(wsdi,na.rm=TRUE))
  } else {
    stop(paste(name, ' not found.'))
  }

  # plot the data
  selectedData |>
    ggplot() + geom_point(aes(year,wsdi)) +
    labs(title = paste0("WSDI for ", name, ", RCP8.5 scenario")) -> gg
  return(gg)
```

```
}

heatwavesRCP85("India",heatwaves)
```



Question 7 (a challenge!)

The dataset `earth-impact-craters.csv` contains information on 190 craters left behind by meteorite and comet impacts. Most of the columns are in a convenient form for analysis, but `age_millions_of_years_ago` contains ranges in a variety of text formats.

Write custom functions and a tidyverse workflow that clean this up by adding two new columns to the dataset, `min_age` and `max_age` (both measured in millions of years). For example, when the age is given as `290 \pm 20`, `max_age` should be 310 and `min_age` should be 270.

Make a figure that gives an overview of the age data.

```
min_age <- function(age) {
  # extracts the minimum age from one value of `age_millions_of_years`

  if (startsWith(age,'<')) {          # age given as "< value"
    min_age = 0
  } else if (startsWith(age,'>')) {   # age given as "> value"
    min_age = as.numeric(str_remove(age,'<'))
  } else if (startsWith(age,'~')) {   # age given as "~ value"
    min_age = as.numeric(str_remove(age,'~')) * 0.8   # assume +/- 20% uncertainty
  } else if (str_detect(age,'±')) {   # age given as "value ± value"
    vals = str_split(age,'±')
    centralVal = as.numeric(vals[[1]][1])
```

```
    uncertainty = as.numeric(vals[[1]][2])
    min_age = centralVal - uncertainty
  } else if (str_detect(age,'-')) {   # age given as "value - value"
    vals = str_split(age,'-')
    min_age = as.numeric(vals[[1]][1])
  } else {
    min_age = as.numeric(age)         # all other formats, including single values
  }
  # there are still a few complex cases unhandled, such as '~443-470'
  return(min_age)
}

max_age <- function(age) {
  # extracts the maximum age from one value of `age_millions_of_years`

  if (startsWith(age,'<')) {          # age given as "< value"
    max_age = as.numeric(str_remove(age,'<'))
  } else if (startsWith(age,'>')) {   # age given as "> value"
    max_age = 0
  } else if (startsWith(age,'~')) {   # age given as "~ value"
    max_age = as.numeric(str_remove(age,'~')) * 1.2   # assume +/- 20% uncertainty
  } else if (str_detect(age,'±')) {   # age given as "value ± value"
    vals = str_split(age,'±')
    centralVal = as.numeric(vals[[1]][1])
    uncertainty = as.numeric(vals[[1]][2])
    max_age = centralVal + uncertainty
  } else if (str_detect(age,'-')) {   # age given as "value - value"
    vals = str_split(age,'-')
    max_age = as.numeric(vals[[1]][2])
  } else {
    max_age = as.numeric(age)         # all other formats, including single values
  }
  return(max_age)
}

craters <- read_csv('earth-impact-craters.csv')
```

```
## Rows: 189 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (8): crater_name, state, country, target_rock, age_millions_years_ago, b...
## dbl (3): diameter_km, latitude, longitude
## lgl (2): exposed, drilled
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
craters %>% rowwise() %>%
  mutate(min_age = min_age(`age_millions_years_ago`),
         max_age = max_age(`age_millions_years_ago`)) %>%
  ungroup() -> craters2
```

```
## Warning: There were 12 warnings in `mutate()`.
## The first warning was:
```

```
## i In argument: `min_age = min_age(age_millions_years_ago)`.
## i In row 5.
## Caused by warning in `min_age()`:
## ! NAs introduced by coercion
## i Run `dplyr::last_dplyr_warnings()` to see the 11 remaining warnings.
```

```r
# the simplest overview is just making a histogram of each of the new columns:
#     craters2 %>% ggplot() + geom_histogram(aes(min_age))
#     craters2 %>% ggplot() + geom_histogram(aes(max_age))
# but this loses the association of individual min-max values.
# sort by min_age
craters2 %>% arrange(min_age) %>%
# add an ID number column in this sort order
  mutate(id = 1:n()) %>%
# now put min_age and max_age back together as a single column 'age'. A single ID
# will correspond to a pair of age values
  pivot_longer(cols = c('min_age','max_age'),
               names_to = 'age_type',
               values_to = 'age') %>%
# line plot grouping by ID, so that a line segment connects each min-max pair but the
# separate craters are left separate. (To better understand group= in line plots,
# try aes(age,id,group=age_type).)
  ggplot() +
  geom_line(aes(age,id,group=id))
```

```
## Warning: Removed 14 rows containing missing values or values outside the scale range
## (`geom_line()`).
```