Activity_6

Joshua Holt and Georgia Vasey

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   4.0.0     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```
library(here)
```

```
here() starts at /Users/josh1/Desktop/GitAndGithub-DataSci-172026/ds4eeb_Lab4/Activity 6
```

1. Functions

```
# Create a new function called add_together
# x and y will be the two arguments to the function
add_together  <- function(x, y){

  # Add x and y together, store as the object "output"
  output <- x + y

  # Print out whatever is stored in "output"
```

```r
  return(output)

}
```

Q1.1)

```r
add_together(x=3, y=5)
```

```
[1] 8
```

Q1.2)

```r
add_together(x=3, y="five")
```

```
Error in x + y : non-numeric argument to binary operator
```

The error is explaining that you cannot input letters or other characters that are not numbers into the add_together function.

Q1.3)

```r
#create new function, main_time, with x, y, and z will be the arguments
main_time <- function(x, y, z){
  output2 <- (x-y)^2/z
  return(output2)
}
main_time(x=5, y=2, z=9)
```

```
[1] 1
```

Q1.4)

```
#Defining Vector
bison <- c(1000, 800, 1200, 1400)

#Creating function to find mean
deviation <- function(x){
  output3 <- x-mean(x)
  return(output3)

}

deviation(x=bison)
```

```
[1] -100 -300  100  300
```

2. Iteration

Q2.1)

```
?iris
```

The ?iris function brings us to the help page of iris which tells us that the unit of measurement is centimeters.

Q2.2)

```
iris %>%
  group_by(Species) %>%
  summarize(across(.cols = everything(),
                   .fns = median))
```

```
# A tibble: 3 x 5
  Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
  <fct>             <dbl>       <dbl>        <dbl>       <dbl>
1 setosa              5           3.4         1.5          0.2
2 versicolor          5.9         2.8         4.35         1.3
3 virginica           6.5         3           5.55         2
```

Q2.3)

```r
#reading/defining data in csv
cereal <- read_csv("data/cereal.csv")
```

```
Rows: 77 Columns: 16
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (3): name, mfr, type
dbl (13): calories, protein, fat, sodium, fiber, carbo, sugars, potass, vita...

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.
```

```r
cereal |>
  group_by(mfr) |>
  summarize(across(.cols = where(is.numeric),
                   .fns = mean))
```

```
# A tibble: 7 x 14
  mfr     calories protein   fat sodium fiber carbo sugars potass vitamins shelf
  <chr>      <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>    <dbl> <dbl>
1 Americ~    100      4    1        0   0      16     3       95      25     2
2 Genera~    111.     2.32 1.36   200.  1.27   14.7   7.95    85.2    35.2   2.14
3 Kellog~    109.     2.65 0.609  175.  2.74   15.1   7.57   103.     34.8   2.35
4 Nabisco     86.7    2.83 0.167   37.5 4      16     1.83   121.      8.33  1.67
5 Post       109.     2.44 0.889  146.  2.78   13.2   8.78   114.     25     2.44
6 Quaker~     95      2.62 1.75    92.5 1.34   10     5.25    74.4    12.5   2.38
7 Ralsto~    115      2.5  1.25   198.  1.88   17.6   6.12    89.2    25     2
# i 3 more variables: weight <dbl>, cups <dbl>, rating <dbl>
```

Q2.4)

```r
for(i in 1:10){
  print(i^2)
}
```

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
[1] 49
[1] 64
[1] 81
[1] 100
```

Q2.5)

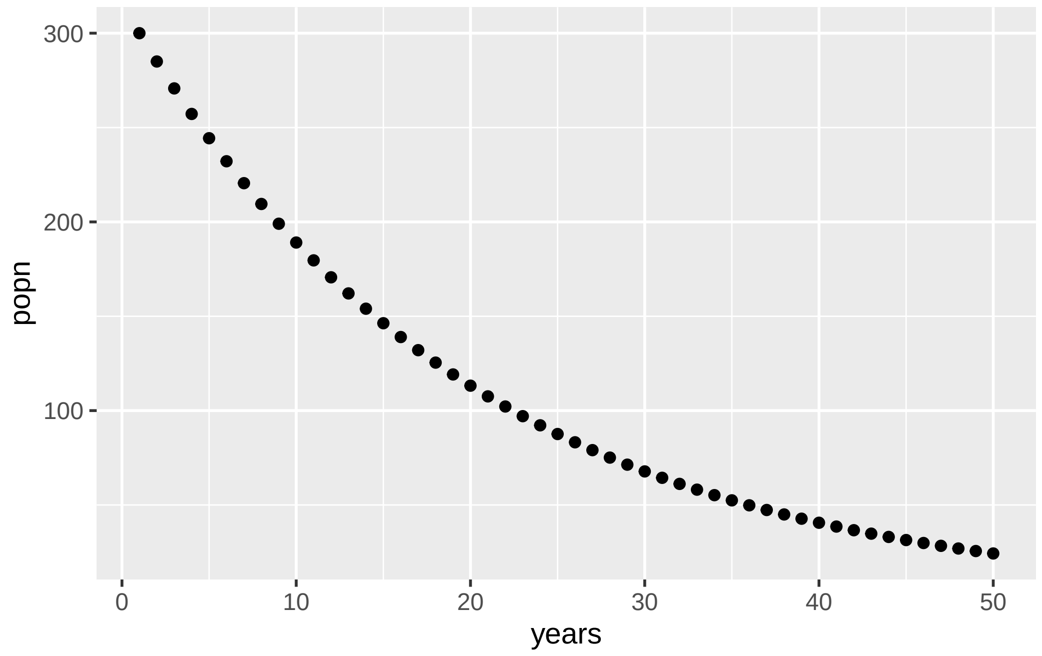
Q2.5a

```
N0 = 300  #initial population size

years = 50  #number of years into the future

N = vector(length = years)  # create an empty vector to store pop. sizes

N[1] = N0  #initial population size should be the first N

lambda = 0.95  #growth rate

# For every year t in 2 through 20 (remember, "years" also equals 20), apply the following e
for (t in 2:50) {
  N[t] = N[t - 1] * lambda # Apply the equation
}

# Store the data output as a dataframe for plotting
popn_data <- tibble(years = 1:years, # Make the years column = 1, 2, 3, ..., 50
                    popn = N) # Make the population column the corresponding population vect
```

Q2.5b

```
# Now plot the data with years on the x axis and population on the y
popn_data %>%
```

```r
  ggplot(aes(x = years, y = popn)) +
  geom_point()
```



The population is declining somewhat exponentially over 50 years. It has declined to under 30 individuals when the current data in the end.

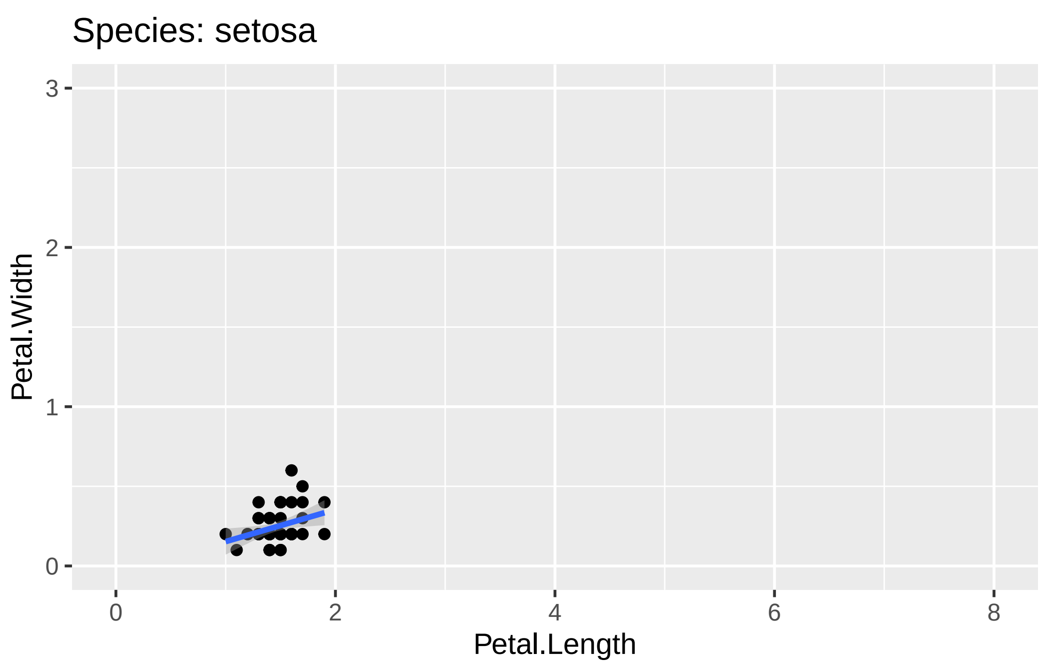
Q2.6)

I prefer the across() function because it gives column names of the ones you want to pull and and you can easily edit the function. The for() loop feels intricate and clunky. I'm more likely to make mistakes.

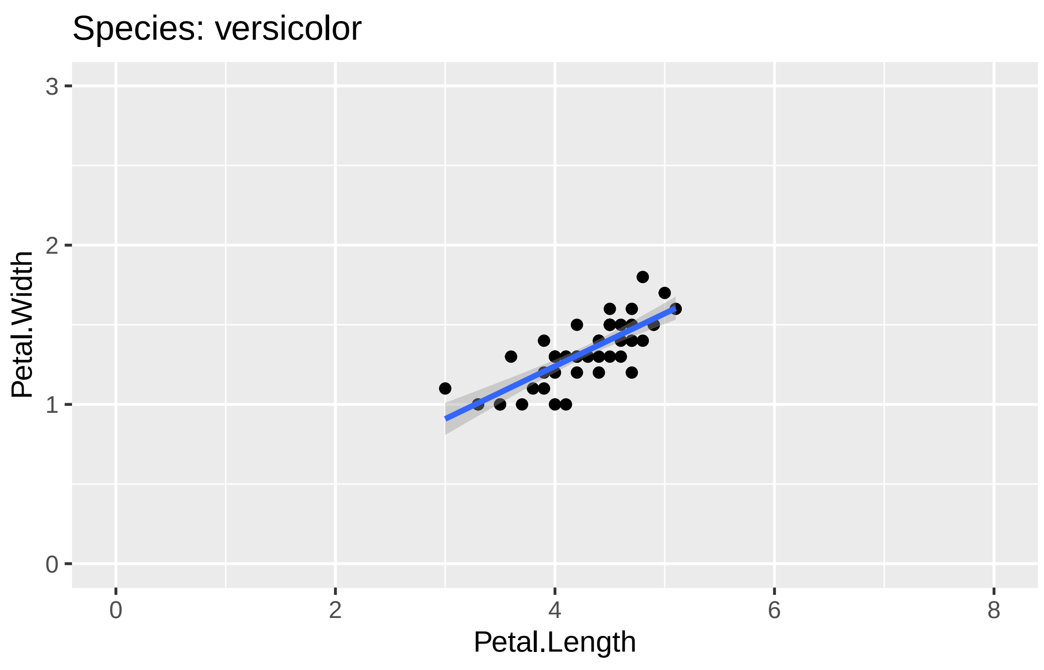
Q2.7)

```r
# Store a vector of unique species names from the Species column of Iris
spp_names <- unique(iris$Species)

# Create a vector that starts with 1 to the length of the spp_names, which is 3, and use i a
for (i in 1:length(spp_names)) {
  filt_data <- iris %>%
    # Keep only the rows where Species matches the current species
    filter(Species == spp_names[i])

  # Build a ggplot object using the filtered data for the current species
  plot <- filt_data %>%
    # Initialize the plot: x = Petal.Length, y = Petal.Width
    ggplot(aes(x = Petal.Length,
               y = Petal.Width)) +
    # Add scatterplot points for each observation
    geom_point() +
    # Add a best-fit linear regression line
    geom_smooth(method = "lm") +
    # Set fixed axis limits so all species plots use the same x and y ranges
    lims(x = c(0,8),
         y = c(0,3)) +
    # Add a title that includes the current species name
    ggtitle(paste("Species:", spp_names[i]))

  #Display the plot for this species inside the loop
  print(plot)
}
```

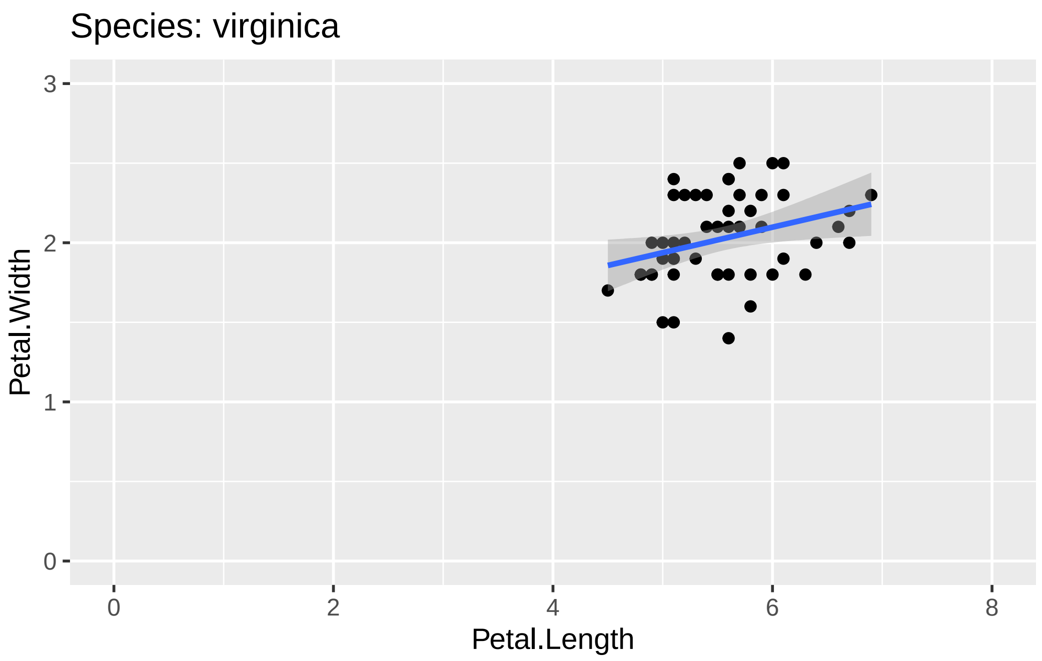
```
`geom_smooth()` using formula = 'y ~ x'
```



```
`geom_smooth()` using formula = 'y ~ x'
```



```
`geom_smooth()` using formula = 'y ~ x'
```


Species: virginica
Petal.Width
Petal.Length
0
1
2
3
0
2
4
6
8