Review

Jai Jot Kaur

11/29/2022

Load or Download a File?

Write a conditional statement that checks if surveys.csv exists in the working directory, if it doesn't then downloads it from https://ndownloader.figshare.com/files/2292172 using download.file(), and finally loads the file into a data frame and displays the first few rows using the head() function. The url needs to be in quotes since it is character data.

```
survey.list <- list.files("/Users/jaijotkaur/Desktop/BIO197/data_science_research/data-raw")
any(survey.list)
```

```
## Warning in any(survey.list): coercing argument of type 'character' to logical
```

```
## [1] NA
```

```
library(stringr)
str_detect(list.files("/Users/jaijotkaur/Desktop/BIO197/data_science_research/data-raw"), "surveys.csv")
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

```
is.element("surveys.csv", list.files("/Users/jaijotkaur/Desktop/BIO197/data_science_research/data-raw"))
```

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

```
??str_detect
getwd()
```

```
## [1] "/Users/jaijotkaur/Desktop/BIO197/data_science_research/documents"
```

```
"a" == c("a", "b", "c")
```

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

```
?list.files

if (is.element("surveys.csv", survey.list)) {
  print("The file surveys.csv is already downloaded")
  } else {
    print("The file is not downloaded")
  }
```

```
## [1] "The file surveys.csv is already downloaded"
```

```
# downloads it from https://ndownloader.figshare.com/files/2292172 using download.file()
    download.file("https://ndownloader.figshare.com/files/2292172", "../data-raw/surveys-download.csv")
# loads the file into a data frame
    surveydataframe <- read.csv("../data-raw/surveys-download.csv")
# displays the first few rows using the head() function
     head(surveydataframe)
```

```
##   record_id month day year plot_id species_id sex hindfoot_length weight
## 1         1     7  16 1977       2         NL   M              32     NA
## 2         2     7  16 1977       3         NL   M              33     NA
## 3         3     7  16 1977       2         DM   F              37     NA
## 4         4     7  16 1977       7         DM   M              36     NA
## 5         5     7  16 1977       3         DM   M              35     NA
## 6         6     7  16 1977       1         PF   M              14     NA
```

Make a version of this conditional statement that is a function, where the name of the file is the first argument and the link for downloading the file is the second argument. The function should return the resulting data frame. Add some documentation to the top of the function describing what it does. Call this function using "species.csv" as the file name and https://ndownloader.figshare.com/files/3299483 as the link. Print the first few rows of the resulting data frame using head().

```
# This function tests if a file is in the data raw directory and if not, it downloads it and reads it a

reading_csv <- function(file_name, link) {

# 1. Test if file is in the data-raw folder
  # file_name <- "species.csv"
  test <- !is.element(file_name, list.files(path = "../data-raw"))
  download.file(link, "../data-raw/file")

# 2. If test is FALSE, download the file
  if(test) {

    # Option 1: save it with a random name:
    # download.file(url = file_link, destfile = "../data-raw/temporary.csv")
    # result <-  read.csv(file = "../data-raw/temporary.csv")

    # Option 2: save it eith the name given in file name:
    destination_file <- stringr:: str_c("../data-raw", file_name)
    download.file(url = link, destfile = destination_file)
    result <- read.csv(file = destination_file)
   return(result)
  }

}
```

```
reading_csv(file_name = "species.csv", link = "https://ndownloader.figshare.com/files/3299483")
```

Multi-file Analysis

If individual_collar_data.zip is not already in your working directory download the zip file using download.file()

```
list_of_files <- list.files(path = "../data-raw/.")
files_present <- is.element("individual_collar_data.zip", list_of_files)

file_name <- "individual_collar_data.zip"


if(!files_present) {
  download.file("http://www.datacarpentry.org/semester-biology/data/individual_collar_data.zip", "../da
}

library(stringr)
str_c("../data-raw/", file_name)
```

```
## [1] "../data-raw/individual_collar_data.zip"
```

Unzip it using unzip()

```
# dir.create(path = "../data-raw/individual_collar_data")
unzip("../data-raw/individual_collar_data.zip", exdir = "../data-raw/individual_collar_data")
```

Obtain a list of all of the files with file names matching the pattern "collar-data-.*.txt" (using list.files())

```
collar_data_files <- list.files("/Users/jaijotkaur/Desktop/BIO197/data_science_research/data-raw", "col
```

Use a loop to load each of these files into R and make a line plot (using geom_path()) for each file with long on the x axis and lat on the y axis. Graphs, like other types of output, won't display inside a loop unless you explicitly display them, so you need put your ggplot() command inside a print() statement. Include the name of the file in the graph as the graph title using labs()

```
# load file
read.csv(file = "../data-raw/individual_collar_data/collar-data-J10-2016-02-26.txt")
```

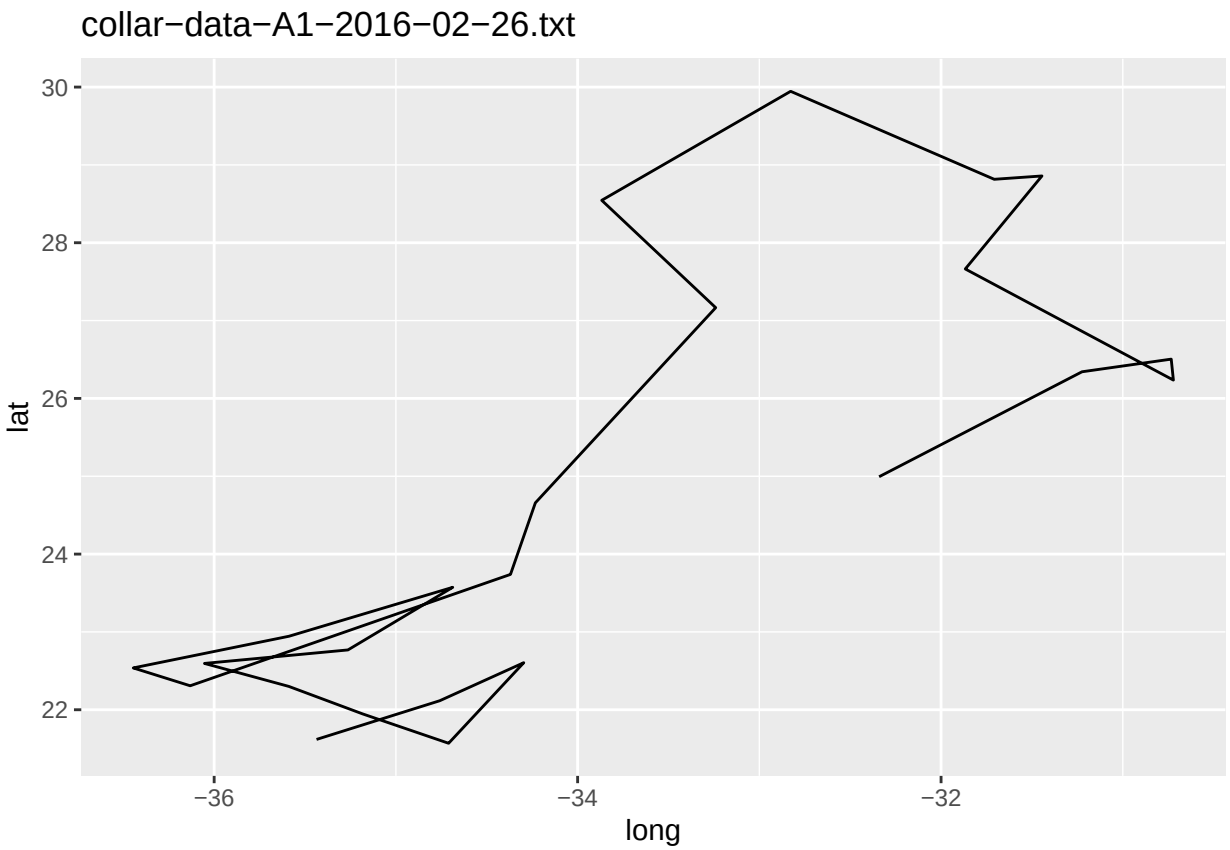
```
##     X       date collar                time      lat      long
## 1   1 2016-02-26    J10 2016-02-26 00:00:00 27.12111 -36.53998
## 2   2 2016-02-26    J10 2016-02-26 01:00:00 26.45791 -36.21363
## 3   3 2016-02-26    J10 2016-02-26 02:00:00 27.80325 -37.70316
## 4   4 2016-02-26    J10 2016-02-26 03:00:00 26.37995 -37.39610
## 5   5 2016-02-26    J10 2016-02-26 04:00:00 26.75686 -37.00541
## 6   6 2016-02-26    J10 2016-02-26 05:00:00 26.33757 -36.40543
## 7   7 2016-02-26    J10 2016-02-26 06:00:00 26.22944 -37.20925
## 8   8 2016-02-26    J10 2016-02-26 07:00:00 26.64954 -39.20656
## 9   9 2016-02-26    J10 2016-02-26 08:00:00 26.18014 -39.16648
## 10 10 2016-02-26    J10 2016-02-26 09:00:00 25.19237 -38.65090
## 11 11 2016-02-26    J10 2016-02-26 10:00:00 25.50803 -40.07132
## 12 12 2016-02-26    J10 2016-02-26 11:00:00 26.59800 -40.26366
## 13 13 2016-02-26    J10 2016-02-26 12:00:00 27.22986 -41.98210
## 14 14 2016-02-26    J10 2016-02-26 13:00:00 26.16071 -40.54453
```

```
## 15 15 2016-02-26    J10 2016-02-26 14:00:00 26.36398 -41.62235
## 16 16 2016-02-26    J10 2016-02-26 15:00:00 24.99279 -42.79979
## 17 17 2016-02-26    J10 2016-02-26 16:00:00 27.33811 -41.89331
## 18 18 2016-02-26    J10 2016-02-26 17:00:00 26.49579 -41.44969
## 19 19 2016-02-26    J10 2016-02-26 18:00:00 24.71200 -42.47906
## 20 20 2016-02-26    J10 2016-02-26 19:00:00 25.29583 -42.04073
## 21 21 2016-02-26    J10 2016-02-26 20:00:00 26.30357 -44.02356
## 22 22 2016-02-26    J10 2016-02-26 21:00:00 26.55108 -44.05999
## 23 23 2016-02-26    J10 2016-02-26 22:00:00 26.89237 -44.86087
## 24 24 2016-02-26    J10 2016-02-26 23:00:00 26.63837 -45.17076
```

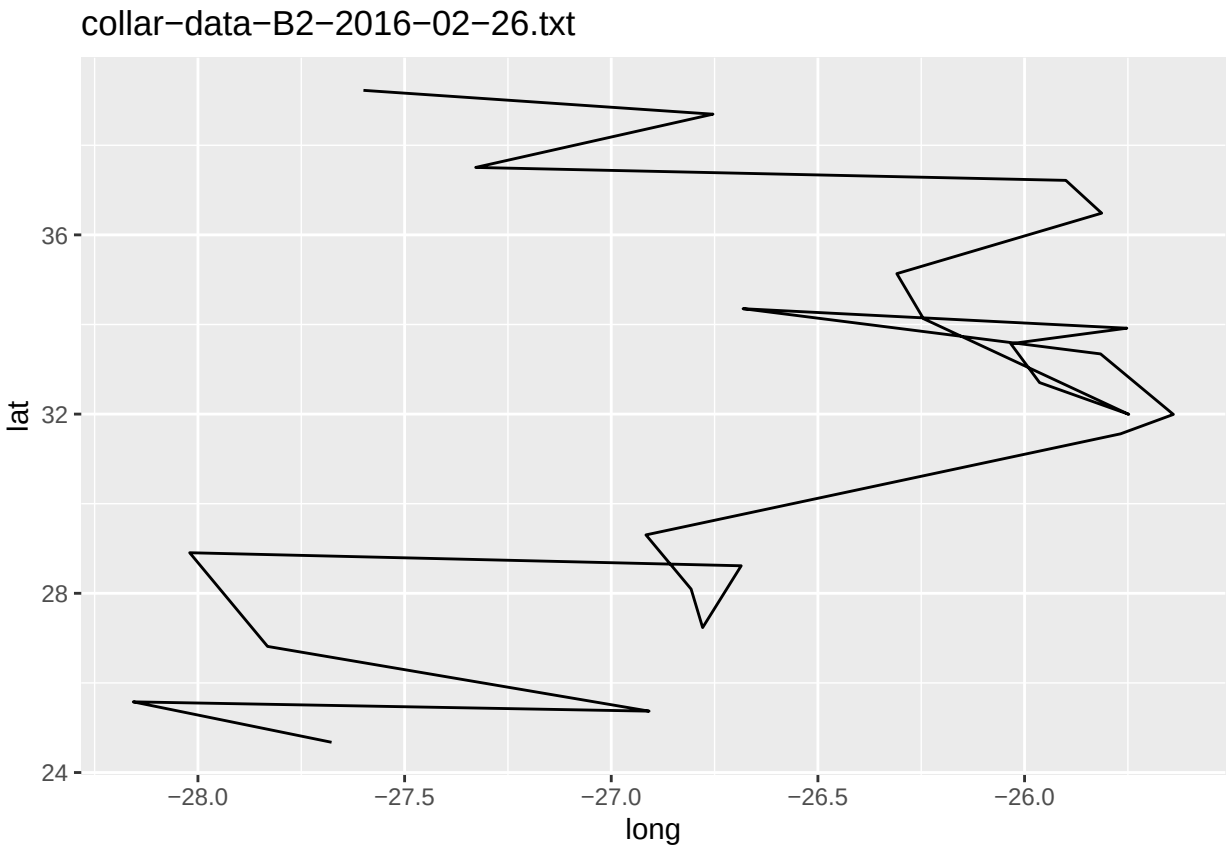
```r
library(ggplot2)

for (i in collar_data_files) {
  print(i)
  collar_data_table <- read.csv(file = i)
  # print(head(collar_data_table))
  ggplot(collar_data_table, aes(x = long, y = lat)) +
    labs(title = i) +
    geom_path() -> collar_data_graph
  print(collar_data_graph)
}
```

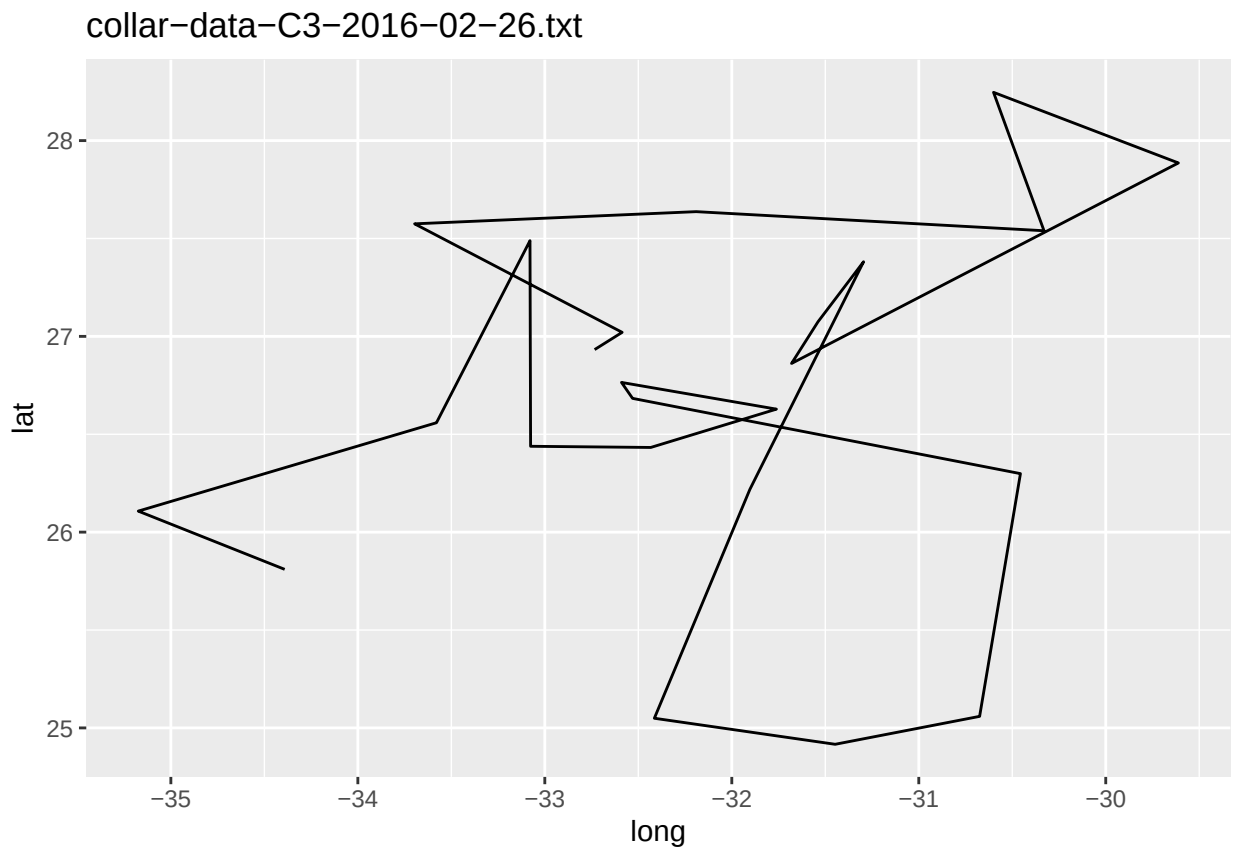
```
## [1] "collar-data-A1-2016-02-26.txt"
```



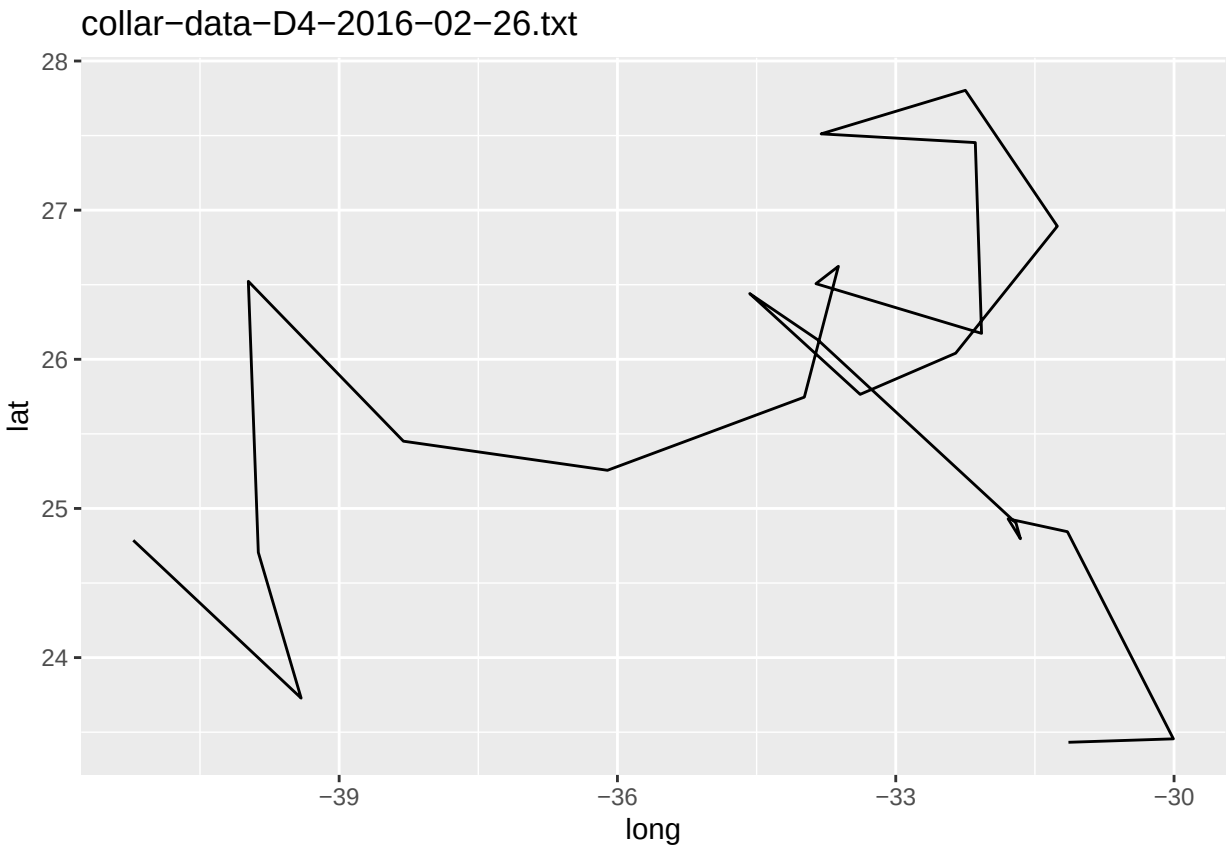

```
## [1] "collar-data-B2-2016-02-26.txt"
```



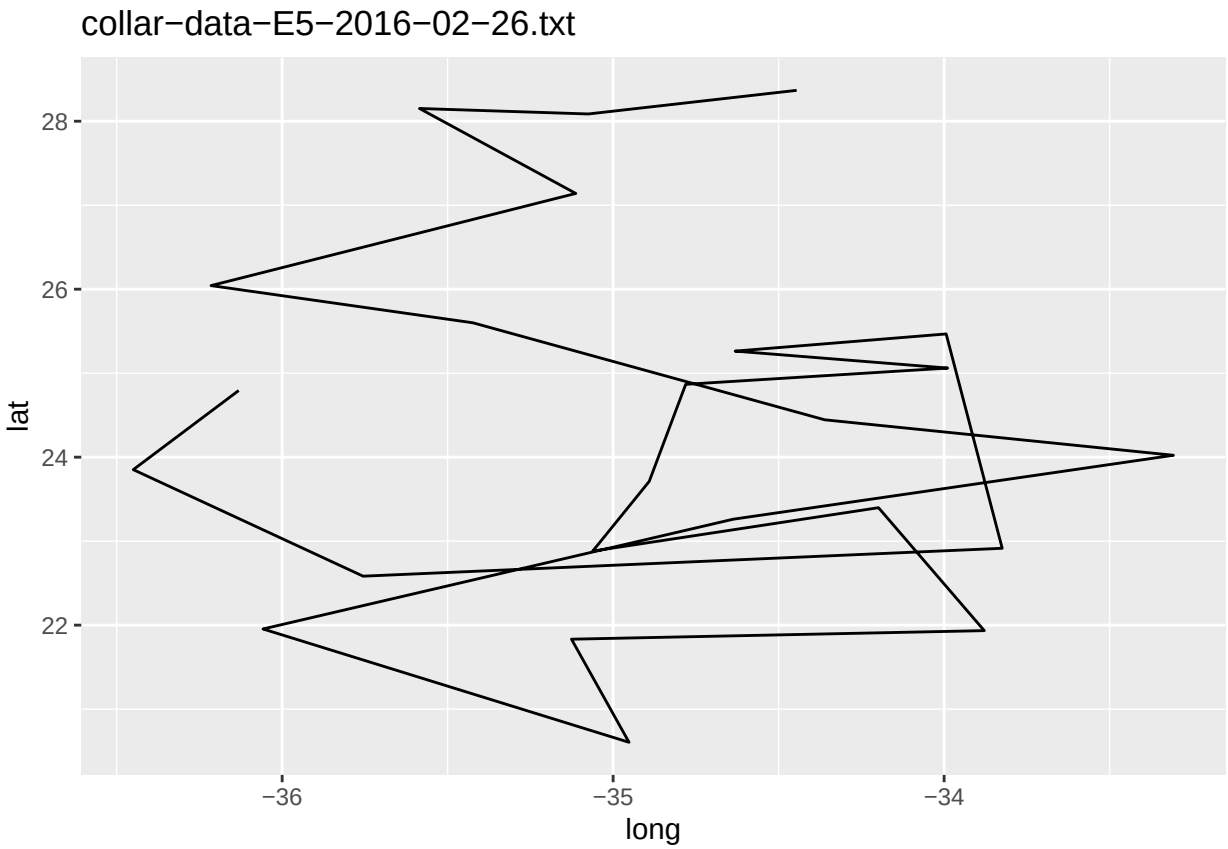

```
## [1] "collar-data-C3-2016-02-26.txt"
```



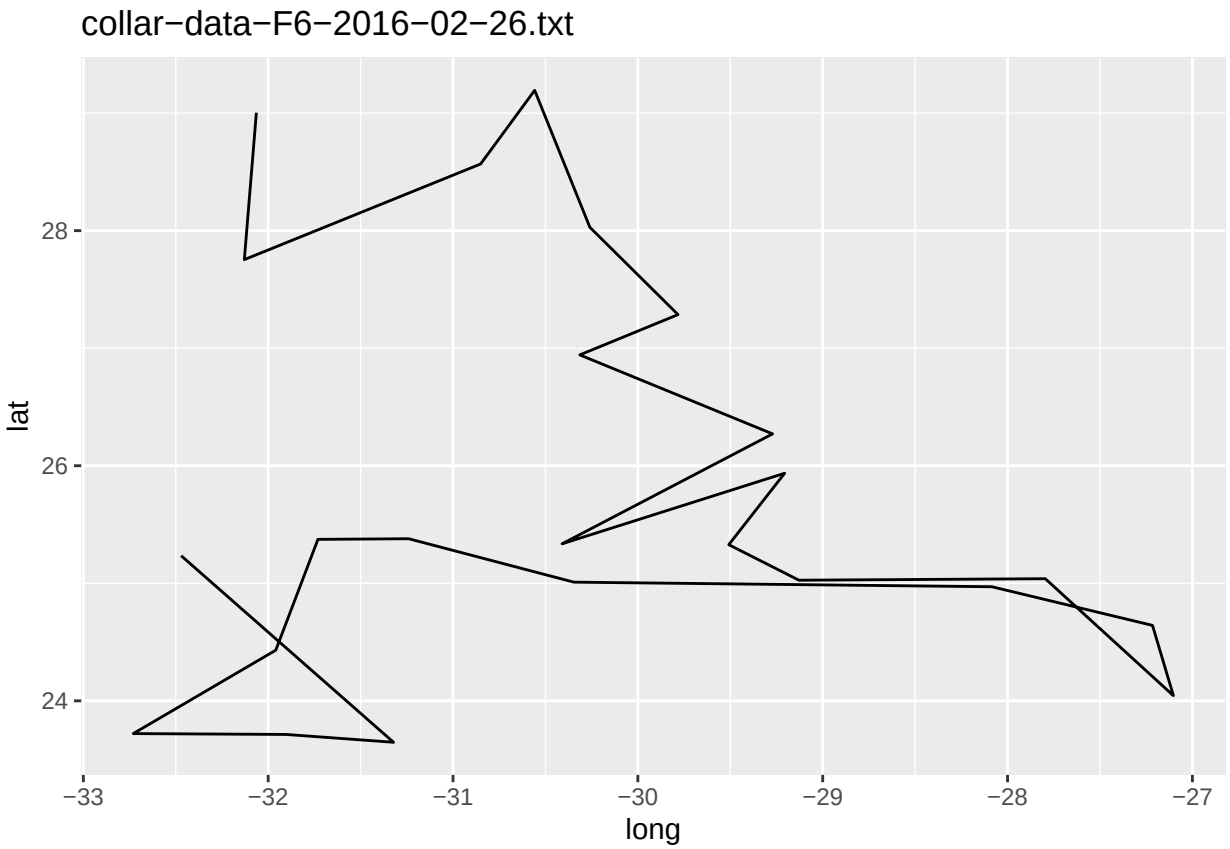

```
## [1] "collar-data-D4-2016-02-26.txt"
```



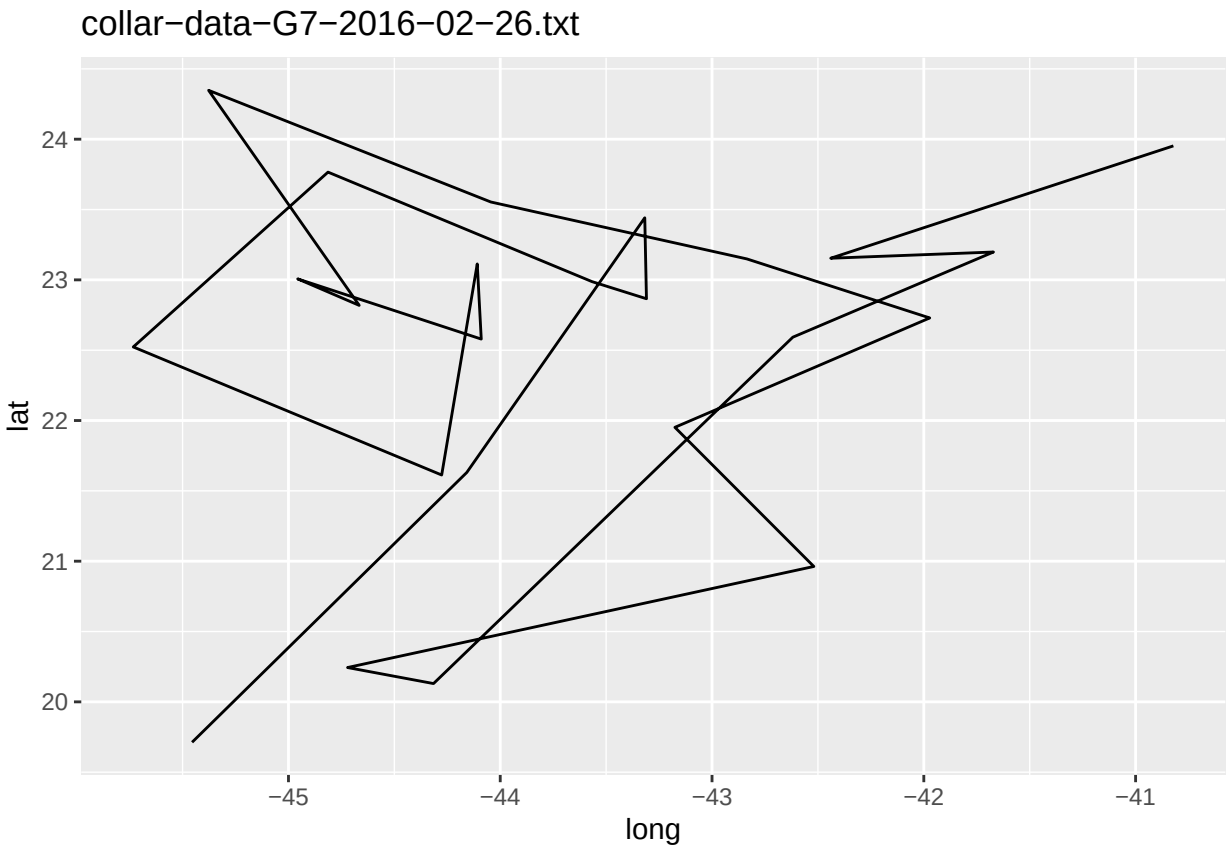

```
## [1] "collar-data-E5-2016-02-26.txt"
```



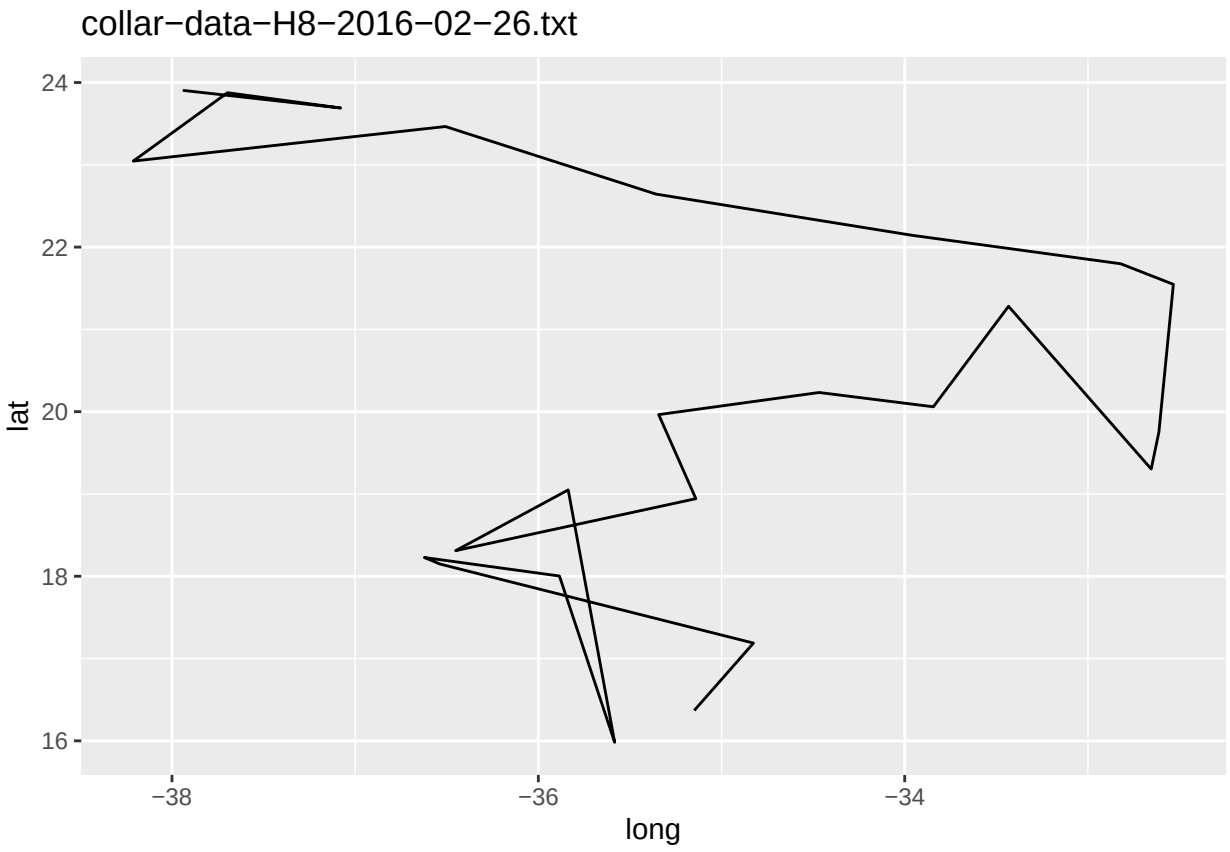

```
## [1] "collar-data-F6-2016-02-26.txt"
```



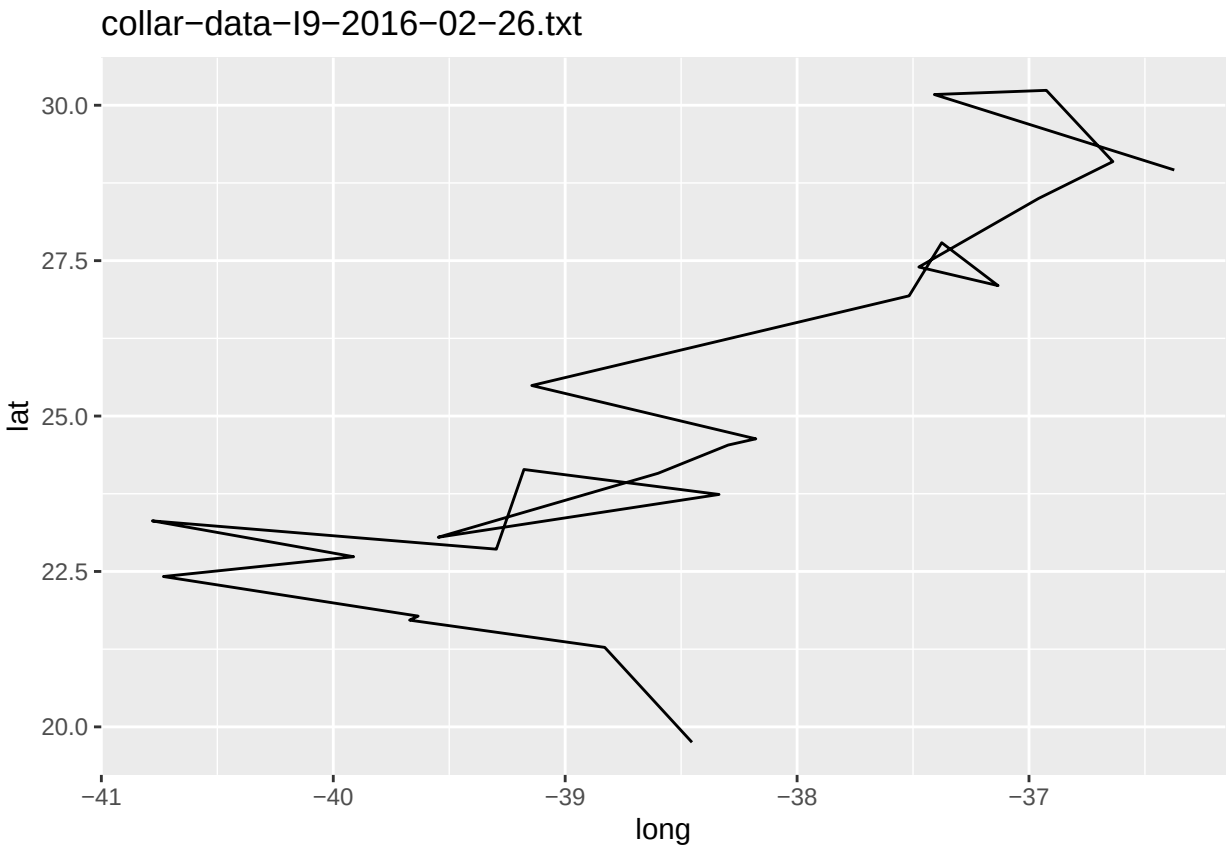

```
## [1] "collar-data-G7-2016-02-26.txt"
```



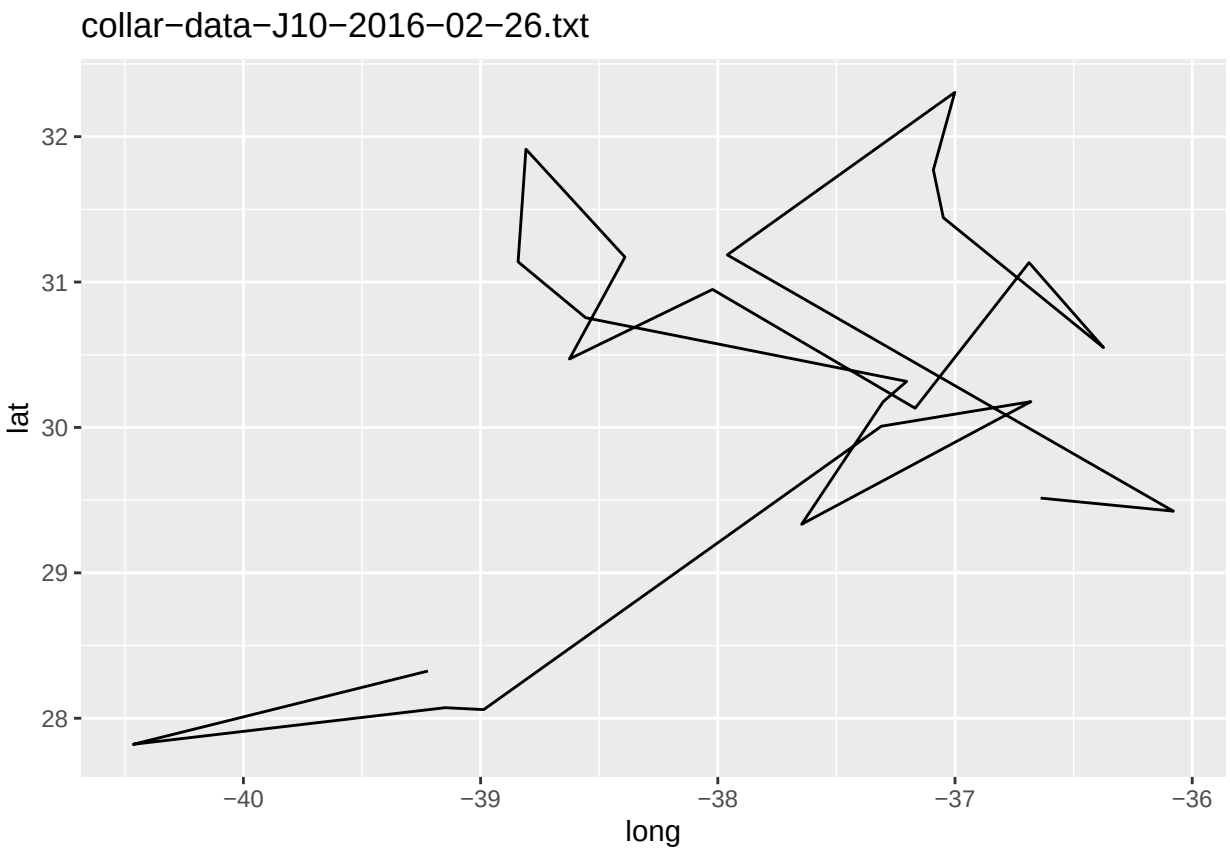

```
## [1] "collar-data-H8-2016-02-26.txt"
```




```
## [1] "collar-data-I9-2016-02-26.txt"
```




```
## [1] "collar-data-J10-2016-02-26.txt"
```



Add code to the loop to calculate the minimum and maximum latitude in the file, and store these values, along with the name of the file, in a data frame. Show the data frame as output.

Solution 1, using empty vectors

```
all_min <- vector()
all_max <- vector()
length(all_min)
```

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

```
length(all_max)
```

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

```
file_name <- vector()
?read.csv
for (i in collar_data_files) {
  print(i)
  collar_data_table <- read.csv(str_c("../data-raw/individual_collar_data/", i))
  file_name <- c(file_name, i)
  min_lat <- min(collar_data_table[, "lat"])
  # min_lat <- min(collar_data_table$lat)
  all_min <- c(all_min, min_lat)
  print(all_min)
```

```
  max_lat <- max(collar_data_table[, "lat"])
  # max_lat <- max(collar_data_table$lat)
  all_max <- c(all_max, max_lat)
  print(all_max)
}
```

```
## [1] "collar-data-A1-2016-02-26.txt"
## [1] 25.2108
## [1] 31.76912
## [1] "collar-data-B2-2016-02-26.txt"
## [1] 25.21080 26.70509
## [1] 31.76912 30.89907
## [1] "collar-data-C3-2016-02-26.txt"
## [1] 25.21080 26.70509 28.86998
## [1] 31.76912 30.89907 33.44421
## [1] "collar-data-D4-2016-02-26.txt"
## [1] 25.21080 26.70509 28.86998 21.34315
## [1] 31.76912 30.89907 33.44421 24.66598
## [1] "collar-data-E5-2016-02-26.txt"
## [1] 25.21080 26.70509 28.86998 21.34315 21.85565
## [1] 31.76912 30.89907 33.44421 24.66598 27.54663
## [1] "collar-data-F6-2016-02-26.txt"
## [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788
## [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623
## [1] "collar-data-G7-2016-02-26.txt"
## [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788 27.67120
## [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623 31.63272
## [1] "collar-data-H8-2016-02-26.txt"
## [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788 27.67120 19.70875
## [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623 31.63272 23.25601
## [1] "collar-data-I9-2016-02-26.txt"
## [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788 27.67120 19.70875
## [9] 25.70252
## [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623 31.63272 23.25601
## [9] 28.49172
## [1] "collar-data-J10-2016-02-26.txt"
##  [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788 27.67120 19.70875
##  [9] 25.70252 24.71200
##  [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623 31.63272 23.25601
##  [9] 28.49172 27.80325
```

```
minmax_dataframe <- data.frame(file_name, all_min, all_max)
minmax_dataframe
```

```
##                         file_name  all_min  all_max
## 1   collar-data-A1-2016-02-26.txt 25.21080 31.76912
## 2   collar-data-B2-2016-02-26.txt 26.70509 30.89907
## 3   collar-data-C3-2016-02-26.txt 28.86998 33.44421
## 4   collar-data-D4-2016-02-26.txt 21.34315 24.66598
## 5   collar-data-E5-2016-02-26.txt 21.85565 27.54663
## 6   collar-data-F6-2016-02-26.txt 17.90788 25.23623
## 7   collar-data-G7-2016-02-26.txt 27.67120 31.63272
## 8   collar-data-H8-2016-02-26.txt 19.70875 23.25601
```

```
## 9   collar-data-I9-2016-02-26.txt 25.70252 28.49172
## 10 collar-data-J10-2016-02-26.txt 24.71200 27.80325
```

Solution 2, using vectors of a predetermined length

```
#  create vectors of a certain length

all_min_lat <- vector(mode = "numeric", length = length(collar_data_files))
all_max_lat <- all_min_lat
all_file_name <- all_min_lat
length(all_min)
```

```
## [1] 10
```

```
length(all_max)
```

```
## [1] 10
```

```
all_file_name
```

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

```
for (i in 1:length(collar_data_files)) {
  file_name_and_path <- str_c("../data-raw/individual_collar_data/", collar_data_files[i])
  all_file_name[i] <- file_name_and_path
  print(file_name_and_path)
  collar_data_table <- read.csv(file = file_name_and_path)
  min_lat <- min(collar_data_table$lat)
  all_min_lat[i] <- min_lat
  max_lat <- max(collar_data_table$lat)
  all_max_lat[i] <- max_lat

  print(all_max_lat)
  print(all_min_lat)
}
```

```
## [1] "../data-raw/individual_collar_data/collar-data-A1-2016-02-26.txt"
##  [1] 31.76912  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
##  [9]  0.00000  0.00000
##  [1] 25.2108  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
## [10]  0.0000
## [1] "../data-raw/individual_collar_data/collar-data-B2-2016-02-26.txt"
##  [1] 31.76912 30.89907  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
##  [9]  0.00000  0.00000
##  [1] 25.21080 26.70509  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
##  [9]  0.00000  0.00000
## [1] "../data-raw/individual_collar_data/collar-data-C3-2016-02-26.txt"
##  [1] 31.76912 30.89907 33.44421  0.00000  0.00000  0.00000  0.00000  0.00000
##  [9]  0.00000  0.00000
##  [1] 25.21080 26.70509 28.86998  0.00000  0.00000  0.00000  0.00000  0.00000
##  [9]  0.00000  0.00000
```

```
## [1] "../data-raw/individual_collar_data/collar-data-D4-2016-02-26.txt"
##  [1] 31.76912 30.89907 33.44421 24.66598  0.00000  0.00000  0.00000  0.00000
##  [9]  0.00000  0.00000
##  [1] 25.21080 26.70509 28.86998 21.34315  0.00000  0.00000  0.00000  0.00000
##  [9]  0.00000  0.00000
## [1] "../data-raw/individual_collar_data/collar-data-E5-2016-02-26.txt"
##  [1] 31.76912 30.89907 33.44421 24.66598 27.54663  0.00000  0.00000  0.00000
##  [9]  0.00000  0.00000
##  [1] 25.21080 26.70509 28.86998 21.34315 21.85565  0.00000  0.00000  0.00000
##  [9]  0.00000  0.00000
## [1] "../data-raw/individual_collar_data/collar-data-F6-2016-02-26.txt"
##  [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623  0.00000  0.00000
##  [9]  0.00000  0.00000
##  [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788  0.00000  0.00000
##  [9]  0.00000  0.00000
## [1] "../data-raw/individual_collar_data/collar-data-G7-2016-02-26.txt"
##  [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623 31.63272  0.00000
##  [9]  0.00000  0.00000
##  [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788 27.67120  0.00000
##  [9]  0.00000  0.00000
## [1] "../data-raw/individual_collar_data/collar-data-H8-2016-02-26.txt"
##  [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623 31.63272 23.25601
##  [9]  0.00000  0.00000
##  [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788 27.67120 19.70875
##  [9]  0.00000  0.00000
## [1] "../data-raw/individual_collar_data/collar-data-I9-2016-02-26.txt"
##  [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623 31.63272 23.25601
##  [9] 28.49172  0.00000
##  [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788 27.67120 19.70875
##  [9] 25.70252  0.00000
## [1] "../data-raw/individual_collar_data/collar-data-J10-2016-02-26.txt"
##  [1] 31.76912 30.89907 33.44421 24.66598 27.54663 25.23623 31.63272 23.25601
##  [9] 28.49172 27.80325
##  [1] 25.21080 26.70509 28.86998 21.34315 21.85565 17.90788 27.67120 19.70875
##  [9] 25.70252 24.71200
```

```
length(file_name_and_path)
```

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

```
vector_minmaxdataframe <- data.frame(all_file_name, all_min_lat, all_max_lat)
vector_minmaxdataframe
```

```
##                                                  all_file_name
## 1   ../data-raw/individual_collar_data/collar-data-A1-2016-02-26.txt
## 2   ../data-raw/individual_collar_data/collar-data-B2-2016-02-26.txt
## 3   ../data-raw/individual_collar_data/collar-data-C3-2016-02-26.txt
## 4   ../data-raw/individual_collar_data/collar-data-D4-2016-02-26.txt
## 5   ../data-raw/individual_collar_data/collar-data-E5-2016-02-26.txt
## 6   ../data-raw/individual_collar_data/collar-data-F6-2016-02-26.txt
## 7   ../data-raw/individual_collar_data/collar-data-G7-2016-02-26.txt
## 8   ../data-raw/individual_collar_data/collar-data-H8-2016-02-26.txt
## 9   ../data-raw/individual_collar_data/collar-data-I9-2016-02-26.txt
```

```
## 10 ../data-raw/individual_collar_data/collar-data-J10-2016-02-26.txt
##    all_min_lat all_max_lat
## 1     25.21080    31.76912
## 2     26.70509    30.89907
## 3     28.86998    33.44421
## 4     21.34315    24.66598
## 5     21.85565    27.54663
## 6     17.90788    25.23623
## 7     27.67120    31.63272
## 8     19.70875    23.25601
## 9     25.70252    28.49172
## 10    24.71200    27.80325
```

Extra. If you're interested in seeing another application of for loops, check out the code used to simulate the data for this exercise using for loops.

```
individuals = paste(c('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'), c(1:10), sep = "")
for (individual in individuals) {
    lat = vector("numeric", 24)
    long = vector("numeric", 24)
    lat[1] = rnorm(1, mean = 26, sd = 2)
    long[1] = rnorm(1, mean = -35, sd = 3)
    for (i in 2:24) {
        lat[i] = lat[i - 1] + rnorm(1, mean = 0, sd = 1)
        long[i] = long[i - 1] + rnorm(1, mean = 0, sd = 1)
    }
    times = seq(from=as.POSIXct("2016-02-26 00:00", tz="UTC"),
                to=as.POSIXct("2016-02-26 23:00", tz="UTC"),
                by="hour")
    df = data.frame(date = "2016-02-26",
                    collar = individual,
                    time = times,
                    lat = lat,
                    long = long)
    write.csv(df, paste("collar-data-", individual, "-2016-02-26.txt", sep = ""))
}
zip("data/individual_collar_data.zip", list.files(pattern = "collar-data-[A-Z][0-9]+-.*"))
```

A function for the UHURU data set

Explain what each line of code in the body of the function is doing. Add the explanations to your Rmd file as comments, before each line of code.

```
# this creates a function named report_rsquared which sonsists of the arguments data, species, and form
report_rsquared <- function(data, species, formula){
  # this creates a subset named subset which filters the data to only include the species ANT
  subset <- dplyr::filter(data, ANT == species)
  # this creates a vector knoene as test which allows the linear models to be fit to data frames
  test <- lm(formula, data = subset)
  # this rounds the value for r squared from the summary of trst by 3 decimal places to acquire a value
  rsquared <- round(summary(test)$r.squared, 3)
  # this function creates a data frame which specifies the values for species and the r squared value
```

```
  output <- data.frame(species = species, r2 = rsquared)
  # this function returns the object for the output
  return(output)
}
```

Execute the function using the UHURU data and specifying species = "CM" and formula = "AXIS1~CIRC".

```
report_rsquared <- function(data, species = "CM", formula= "AXIS1~CIRC"){
  subset <- dplyr::filter(data, ANT == species)
  test <- lm(formula, data = subset)
  rsquared <- round(summary(test)$r.squared, 3)
  output <- data.frame(species = species, r2 = rsquared)
  return(output)
}
```

Modify the function so that it also determines if() the rsquared is significant based on a given threshold. The modified function should return() the species, rsquared and a significance value of "S" for a relationship with an rsquared > threshold or "NS" for an rsquared < threshold.

```
report_rsquared <- function(threshold, rsquared, data, species = "CM", formula= "AXIS1~CIRC"){
  if (rsquared > threshold) {
    list <- c(species, rsquared, significance == "S")
    return(list)
  } else if (rsquared < threshold) {
    return("NS")
  }
  subset <- dplyr::filter(data, ANT == species)
  test <- lm(formula, data = subset)
  rsquared <- round(summary(test)$r.squared, 3)
  output <- data.frame(species = species, r2 = rsquared)
  return(output)
}
```

Execute your modified function for species of "CM", "CS", and "TP" given a threshold = 0.667. `{r.eval = FALSE} report_rsquared(species = "CM", threshold = 0.667) # Error in report_rsquared(species = "CM", threshold = 0.667) : # argument "rsquared" is missing, with no default` # DNA or RNA Iteration

Write a function, dna_or_rna(sequence), that determines if a sequence of base pairs is DNA, RNA, or if it is not possible to tell given the sequence provided. Since all the function will know about the material is the sequence the only way to tell the difference between DNA and RNA is that RNA has the base Uracil ("u") instead of the base Thymine ("t"). Have the function return one of three outputs: "DNA", "RNA", or "UNKNOWN". "'{r.eval = FALSE} sequences = c("ttgaatgccttacaactgatcattacacaggcggcatgaagcaaaaatatactgtgaaccaatgcaggcg", "gauuauuccccacaaagggaguggguauuaggagcugcaucauuuacaagagcagaauguuucaaaugcau", "gaaagcaagaaaaggcaggcgaggaagggaagaaggggggaaacc", "guuuccuacaguauuugaugagaaugagaguuuacuccuggaagauaauauuuagaauguuuacaacugcaccugaucagguggauaaggaagaugaagacu", "gauaaggaagaugaagacuuucaggaaucuaauaaaaaugcacuccaugaauggauucauguauugggaaucagccggguc") dna_or_rna <- function(sequences) sequences = c("ttgaatgccttacaactgatcattacacaggcggcatgaagcaaaaatatactgtgaaccaatgcaggcg", "gauuauuccccacaaagggaguggguauuaggagcugcaucauuuacaagagcagaauguuucaaaugcau", "gaaagcaagaaaaggcaggcgaggaagggaagaaggggggaaacc", "guuuccuacaguauuugaugagaaugagaguuuacuccuggaagauaauauuuagaauguuuacaacugcaccugaucagguggauaaggaagaugaagacu", "gauaaggaagaugaagacuuucaggaaucuaauaaaaaugcacuccaugaauggauucauguauugggaaucagccggguc")

Error in dna_or_rna(sequence) <- function(sequences) sequences = c("ttgaatgccttacaactgatcattacacaggcggcatgaagcaaaaatatactgtgaaccaatgcagg :

could not find function "dna_or_rna<-" if (is.element("t", sequences)) { return(DNA) } else if (is.element("u", sequences)) { return(RNA) } else { return("UNKNOWN") } # Error: no function to return from, jumping to top level

sapply(sequences, dna_or_rna) "'