Week13_Assignment

Ellen Bledsoe

2025-05-17

Assignment

Purpose

The goal of this assignment is to practice writing and using for loops for iteration.

Task

Write R code to successfully answer each question below.

Criteria for Success

- Code is within the provided code chunks or new code chunks are created where necessary
- Code chunks run without errors
- Code chunks have brief comments indicating which code is answering which part of the question
- Code will be assessed as follows:
  - Produces the correct answer using the requested approach: 100%
  - Generally uses the right approach, but a minor mistake results in an incorrect answer: 90%
  - Attempts to solve the problem and makes some progress using the core concept, but returns the wrong answer and does not demonstrate comfort with the core concept: 50%
  - Answer demonstrates a lack of understanding of the core concept: 0%
- Any questions requiring written answers are answered with sufficient detail

Due Date

April 15 at midnight MST

Assignment Exercises

Set Up

Be sure to load the `tidyverse` to use later in the assignment.

```r
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   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- 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 become error
```

1. For Loop Basics (30 pts)

Complete the following tasks.

I recommend copying the commented code and making changes to the copy. That way, if you have made edits that you haven't kept track of, you have the original code to start over as needed.

a. The code below prints the numbers 1 through 5 one line at a time. Modify it to print each of these numbers multiplied by 3.

```r
numbers <- c(1, 2, 3, 4, 5)
# for (number in numbers){
#   print(number)
# }
```

```r
print("1a")
```

```
## [1] "1a"
```

```r
numbers <- c(1, 2, 3, 4, 5)
for (number in numbers){
  print(3 * number)
}
```

```
## [1] 3
## [1] 6
## [1] 9
## [1] 12
## [1] 15
```

b. Write a for loop that loops over the following vector and prints out the mass in kilograms (`mass_kg = 2.2 * mass_lb`)

```r
mass_lbs <- c(2.2, 3.5, 9.6, 1.2)
```

```r
print("1b")
```

```
## [1] "1b"
```

```r
for (mass in mass_lbs) {
  mass_kgs <- 2.2 * mass
  print(mass_kgs)
}
```

```
## [1] 4.84
## [1] 7.7
## [1] 21.12
## [1] 2.64
```

c. Complete the code below so that it prints out the name of each bird one line at a time.

```r
# birds = c("robin", "woodpecker", "blue jay", "sparrow")
# for (i in 1:length(_________)){
#   print(birds[__])
# }
```

```r
birds = c('robin', 'woodpecker', 'blue jay', 'sparrow')
for (i in 1:length(birds)){
  print(birds[i])
}
```

```
## [1] "robin"
## [1] "woodpecker"
## [1] "blue jay"
## [1] "sparrow"
```

d. Complete the code below so that it stores one area for each radius.

```r
radius <- c(1.3, 2.1, 3.5)
# areas <- vector(_____ = "numeric", length = ______)
# for (__ in 1:length(________)){
#   areas[__] <- pi * radius[i] ^ 2
# }
# areas
```

```r
print("1d")
```

```
## [1] "1d"
```

```r
areas <- vector(mode = "numeric", length = length(radius))
for (i in 1:length(areas)){
  areas[i] <- pi * radius[i] ^ 2
}
areas
```

```
## [1]  5.309292 13.854424 38.484510
```

e. Complete the code below to calculate an area for each pair of `lengths` and `widths`, store the areas in a vector, and after they are all calculated print them out:

```r
lengths = c(1.1, 2.2, 1.6)
widths = c(3.5, 2.4, 2.8)
# areas <- vector(length = __________)
# for (i in _____) {
#   areas[__] <- lengths[__] * widths[__]
# }
# areas
```

```r
print("1e")
```

```
## [1] "1e"
```

```r
areas <- vector(length = length(lengths))
for (i in 1:length(lengths)) {
  areas[i] <- lengths[i] * widths[i]
}
areas
```

```
## [1] 3.85 5.28 4.48
```

2. Size Estimates by Name (30 pts)

This is a followup to "Size Estimates by Name" from last week.

Download the `dinosaur_lengths.csv` file from D2L and place it in the correct directory. Read the file into R.

Write a function `mass_from_length()` that uses the equation `mass <- a * length^b` to estimate the size of a dinosaur from its length. This function should take two arguments, `length` and `species`. For each of the following inputs for `species`, so use the associated `a` and `b` values to estimate the species mass using these equations:

- *Stegosauria*: `mass = 10.95 * length ^ 2.64` (Seebacher 2001).
- *Theropoda*: `mass = 0.73 * length ^ 3.63` (Seebacher 2001).
- *Sauropoda*: `mass = 214.44 * length ^ 1.46` (Seebacher 2001).
- For any other value of `species`: `mass = 25.37 * length ^ 2.49`

a. Use this function and a for loop to calculate the estimated mass for each dinosaur, store the masses in a vector, and after all of the calculations are complete show the first few items in the vector using `head()`.
b. Add the results in the vector back to the original data frame. Show the first few rows of the data frame using `head()`.
c. Calculate the mean mass for each `species` using `dplyr` (no `for` loops).

```r
mass_from_length <- function(length, dino_group) {
    if (dino_group == 'Stegosauria') {
        mass <- 10.95 * length ^ 2.64
    } else if (dino_group == 'Theropoda') {
        mass <- 0.73 * length ^ 3.63
    } else if (dino_group == 'Sauropoda') {
        mass <- 214.44 * length ^ 1.46
    } else {
        mass <- 25.37 * length ^ 2.49
    }
    return(mass)
}

dino_data <- read_csv("../data_raw/dinosaur_lengths.csv")
```

```
## Rows: 500 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): species
## dbl (1): lengths
##
## 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
# 2a
print("2a")
```

```
## [1] "2a"
```

```r
lengths <- dino_data$lengths
species <- dino_data$species
masses <- vector(length = length(lengths))
for (i in 1:length(lengths)){
    masses[i] <- mass_from_length(lengths[i], species[i])
}
head(masses)
```

```
## [1] 24341.68 27017.90 67453.38 22114.19 53884.76 52026.34
```

```r
# 2b
print("2b")
```

```
## [1] "2b"
```

```r
dino_data$masses = masses
head(dino_data)
```

```
## # A tibble: 6 x 3
##   species      lengths masses
##   <chr>          <dbl>  <dbl>
## 1 Stegosauria     18.5 24342.
## 2 Ankylosauria    16.4 27018.
## 3 Ankylosauria    23.7 67453.
## 4 Sauropoda       23.9 22114.
## 5 Ankylosauria    21.7 53885.
## 6 Ankylosauria    21.4 52026.
```

```r
# 2c
print("2c")
```

```
## [1] "2c"
```

```r
dino_data %>%
  group_by(species) %>%
  summarize(avg_mass = mean(masses))
```

```
## # A tibble: 4 x 2
##   species      avg_mass
##   <chr>           <dbl>
## 1 Ankylosauria   46819.
## 2 Sauropoda      16104.
## 3 Stegosauria    31924.
## 4 Theropoda      45572.
```

3. Multi-file Analysis (20 pts)

You have satellite collars on a number of different individuals and want to be able to quickly look at all of their recent movements at once. The data is posted daily to a URL that contains one csv file for each individual: zip file

Start your solution by:

- Download the zip file from the link above. Place the zip file in the appropriate sub-directory, if you have them.
- Unzip it using the `unzip()` function
- Obtain a list of all of the files with file names matching the pattern `"collar-data-.*.txt"` (using `list.files()`)

a. 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.

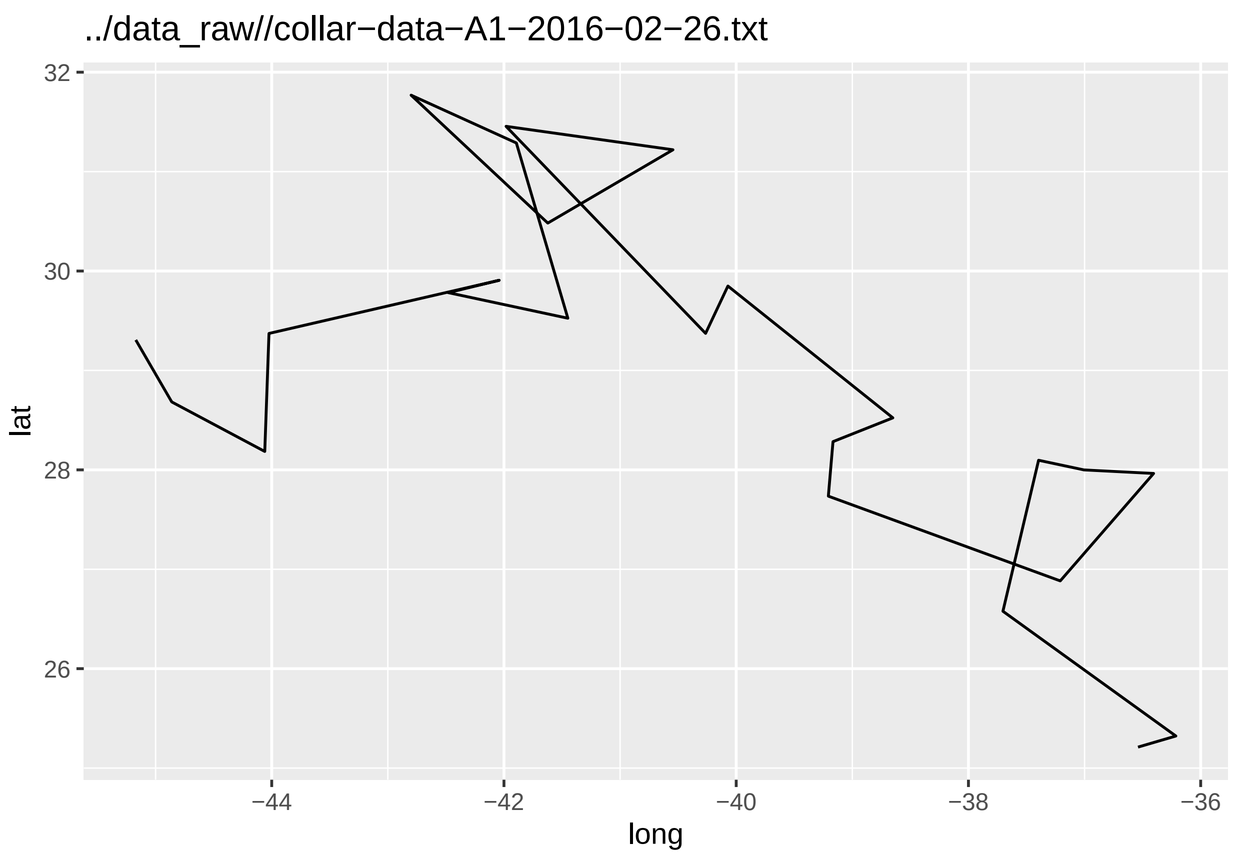
Graphs, like other types of output, won't display inside a loop unless you explicitly display them, so you need put your `ggplot()` command inside a `print()` statement. Do this by saving the output of the ggplot function as an object and then printing that object.

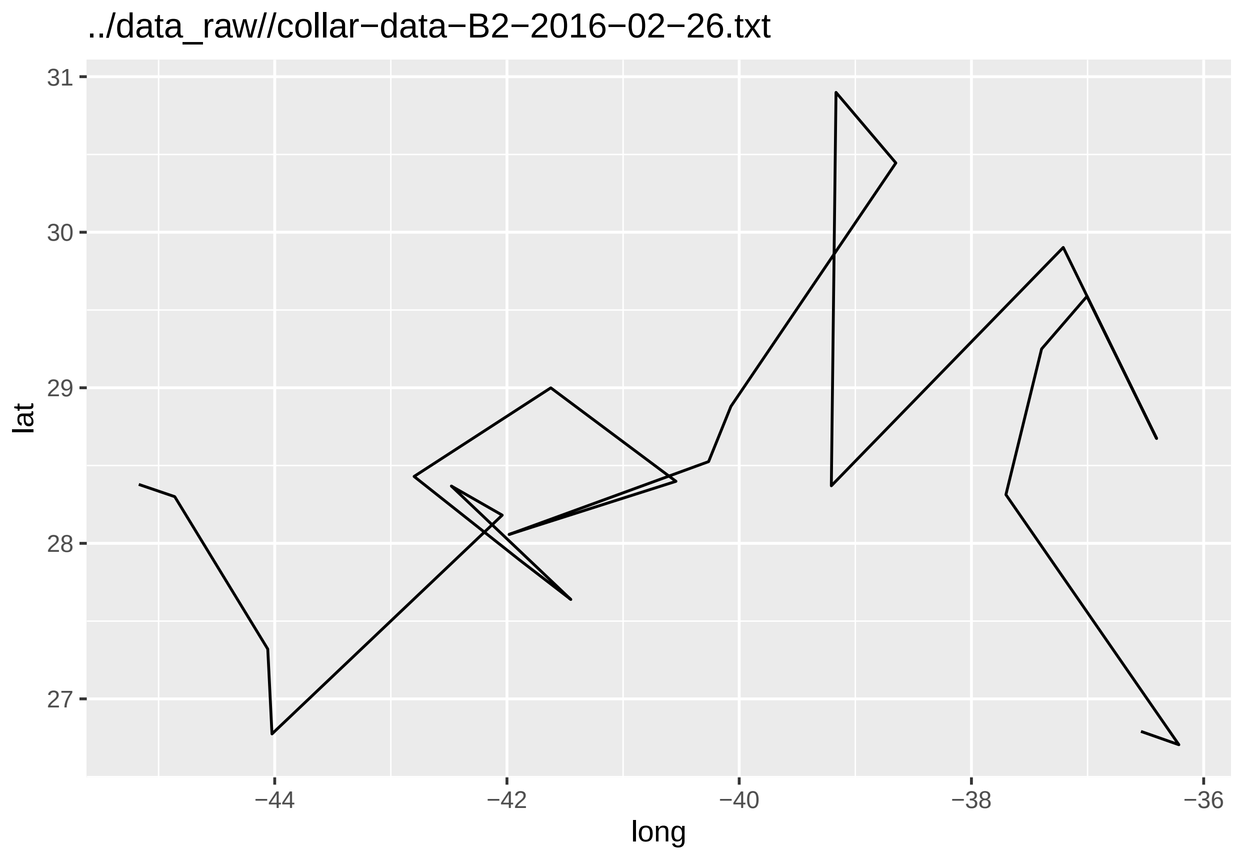
Include the name of the file in the graph as the graph title using the `labs` function (`labs(title = ...)`).

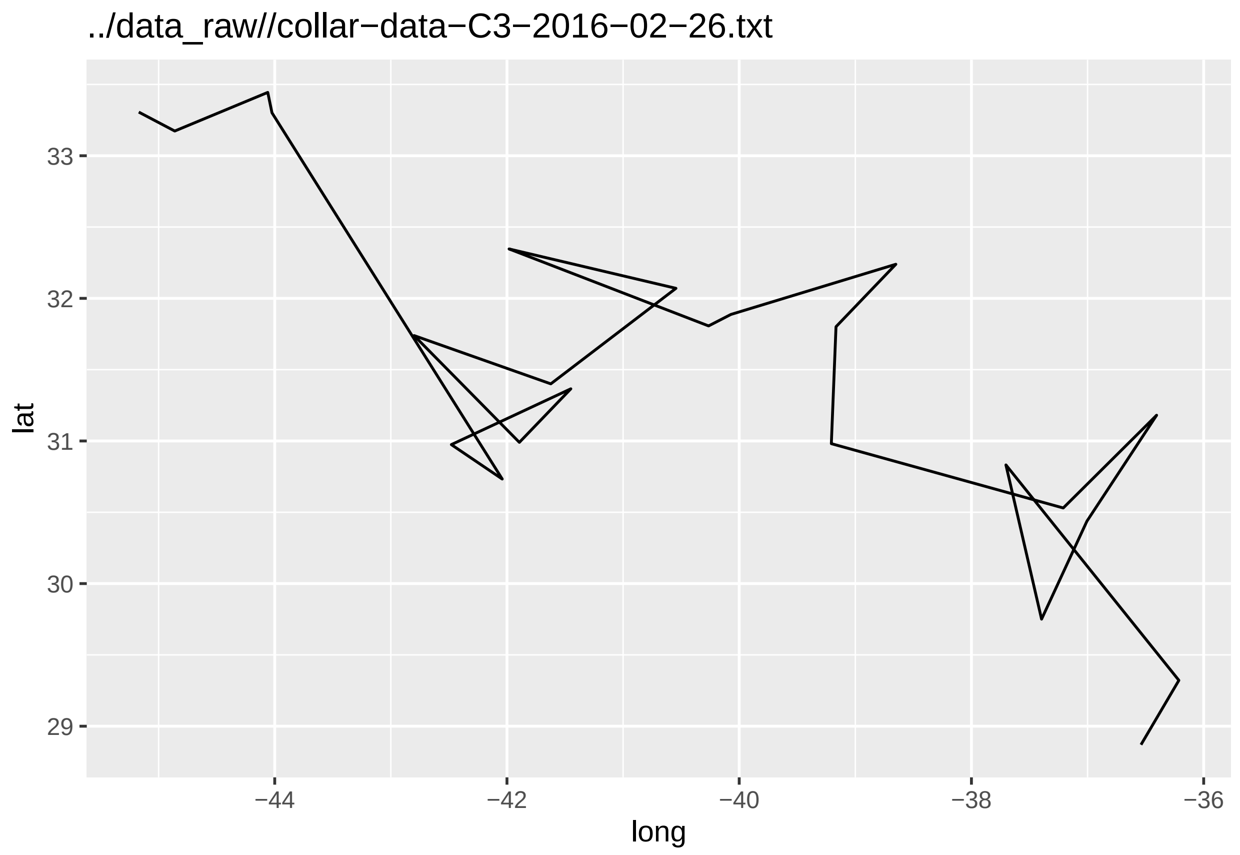
```r
library(ggplot2)

# download.file("http://www.datacarpentry.org/semester-biology/data/individual_collar_data.zip",
#               "../data_raw/individual_collar_data.zip")
# unzip("../data_raw/individual_collar_data.zip",
#       exdir = "../data_raw/")
collar_data_files = list.files(path = "../data_raw/",
                               pattern = "collar-data-.*.txt",
                               full.names = TRUE)

num_data_files = length(collar_data_files)
output = data.frame(file_name = character(num_data_files),
                    max_lat = numeric(num_data_files),
                    min_lat = numeric(num_data_files),
                    observations = numeric(num_data_files),
                    stringsAsFactors = FALSE)

print("3a")
```

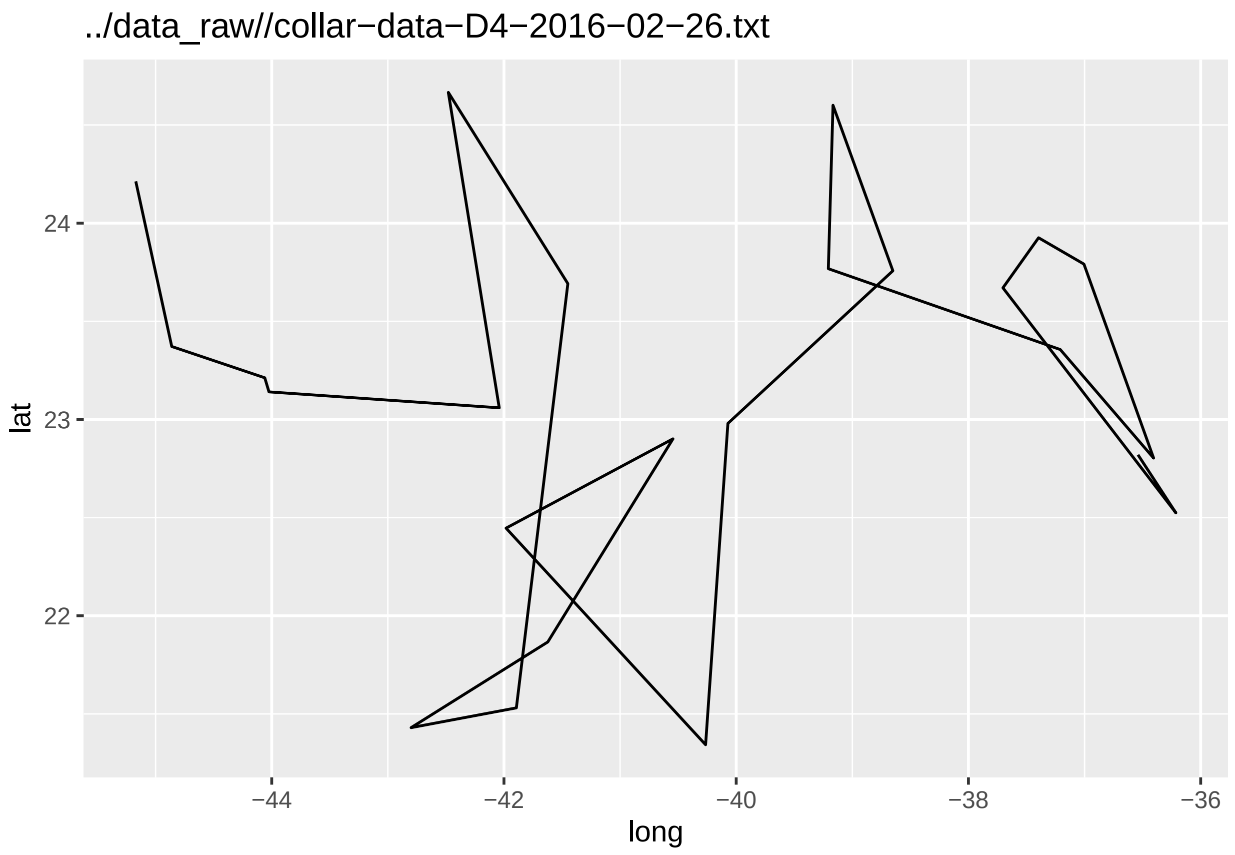
```
## [1] "3a"
```

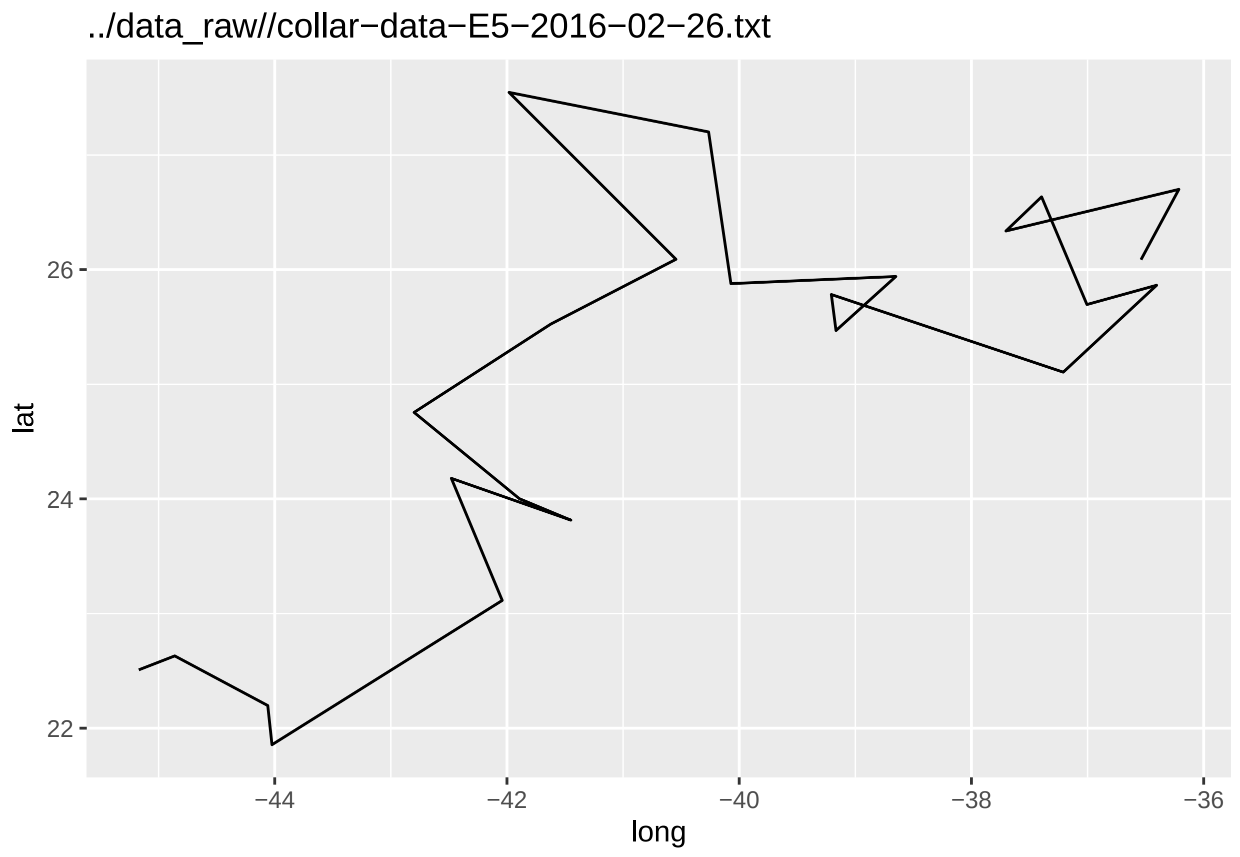
```r
for (i in 1:length(collar_data_files)){
  data = read.csv(collar_data_files[i])
  p <- ggplot(data, aes(x = long, y = lat)) + geom_path() + labs(title = collar_data_files[i])
  print(p)
}
```

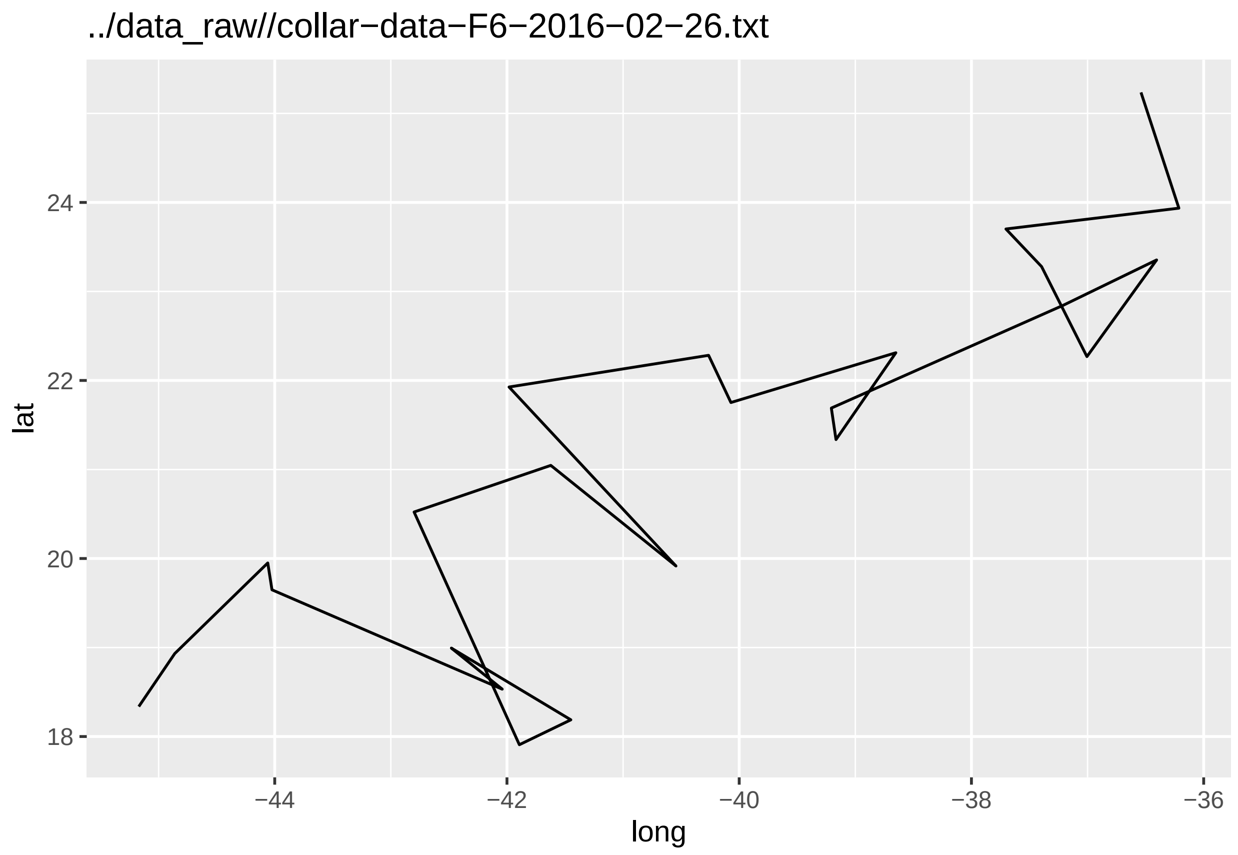


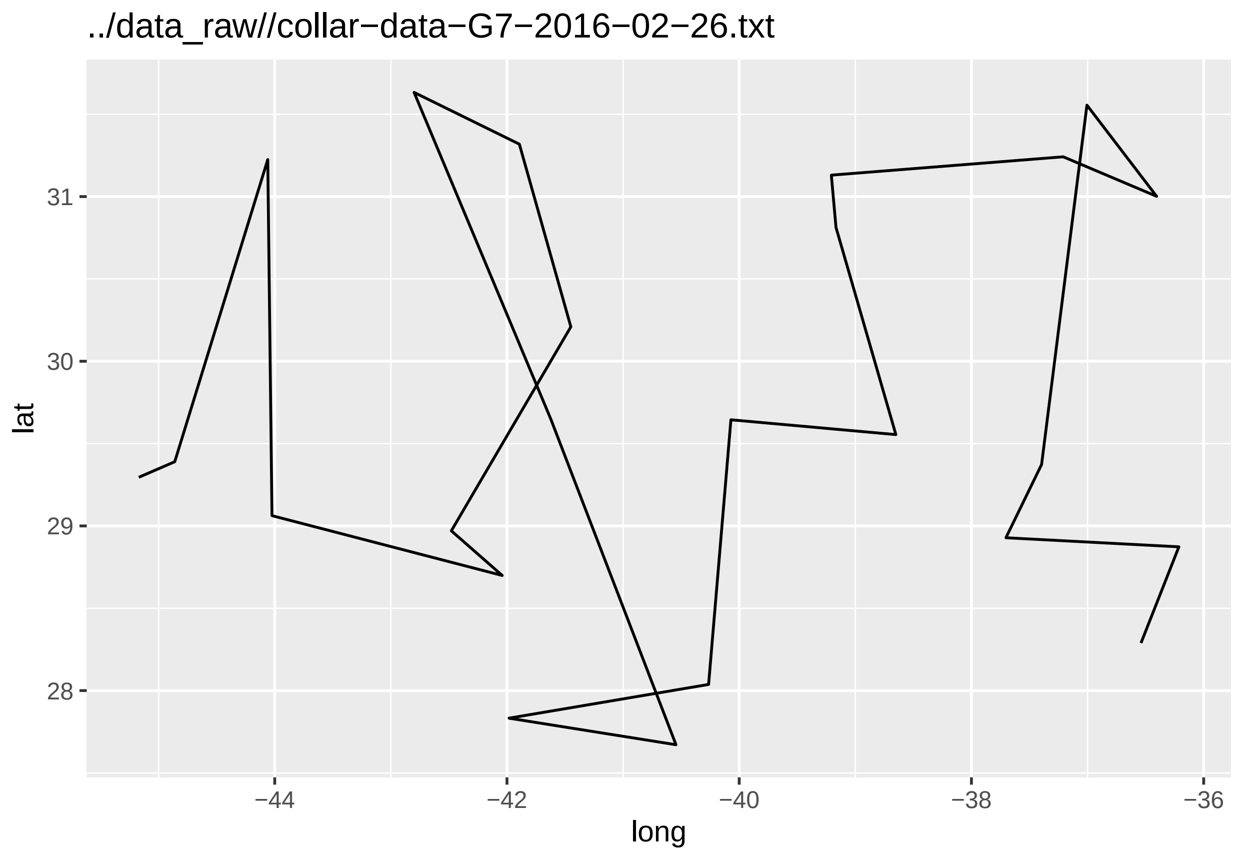

../data_raw//collar−data−B2−2016−02−26.txt
lat
long
31
30
29
28
27
−44
−42
−40
−38
−36

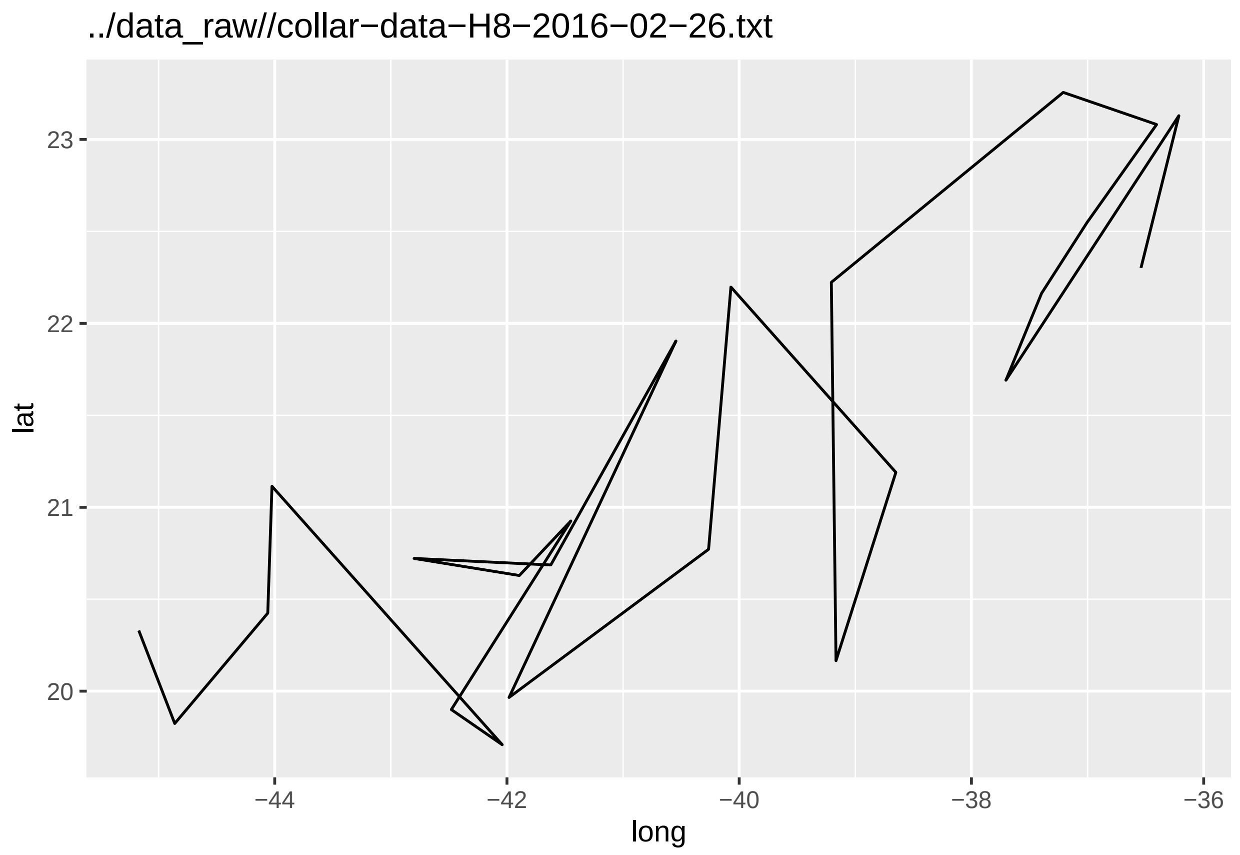

../data_raw//collar−data−C3−2016−02−26.txt
lat
long
33
32
31
30
29
−44
−42
−40
−38
−36


../data_raw//collar−data−D4−2016−02−26.txt
lat
24
23
22
long
−44
−42
−40
−38
−36

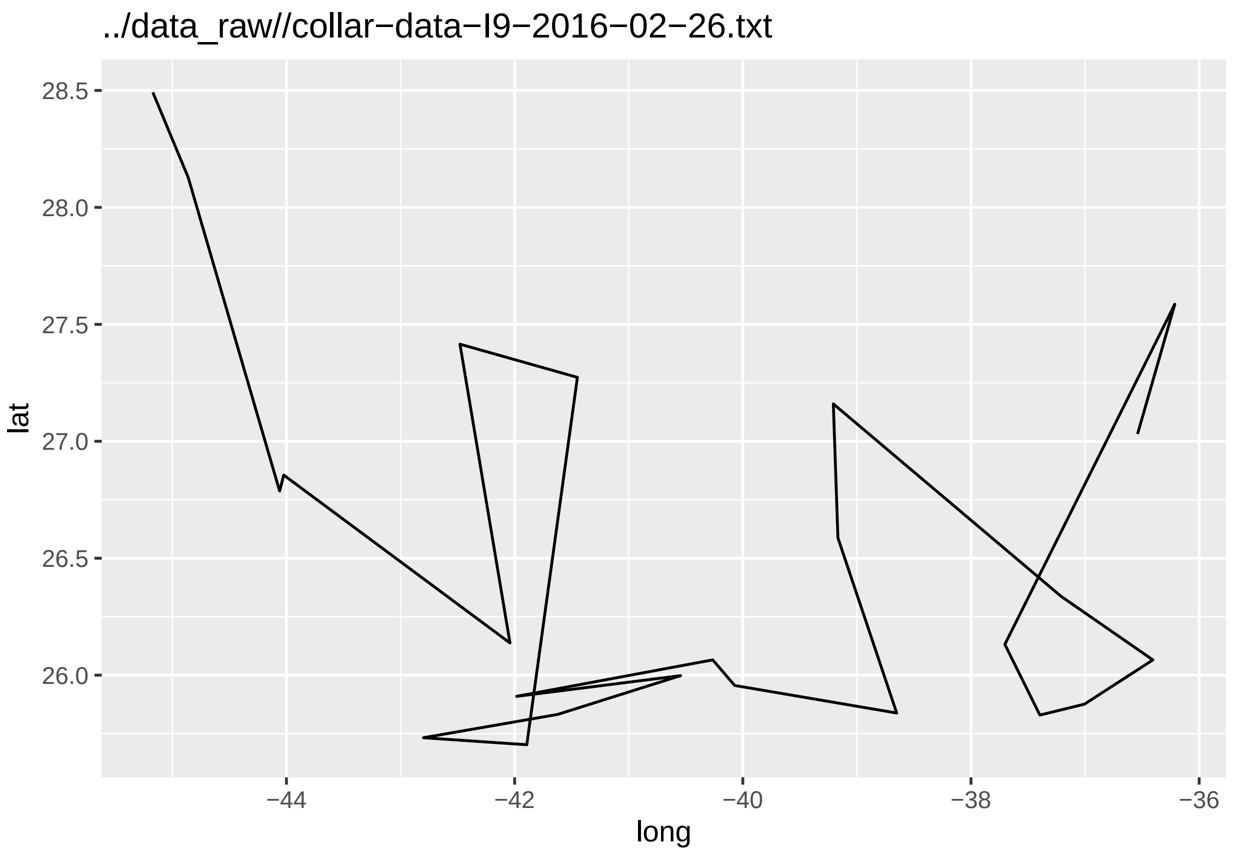

../data_raw//collar−data−E5−2016−02−26.txt
lat
long
22
24
26
−44
−42
−40
−38
−36

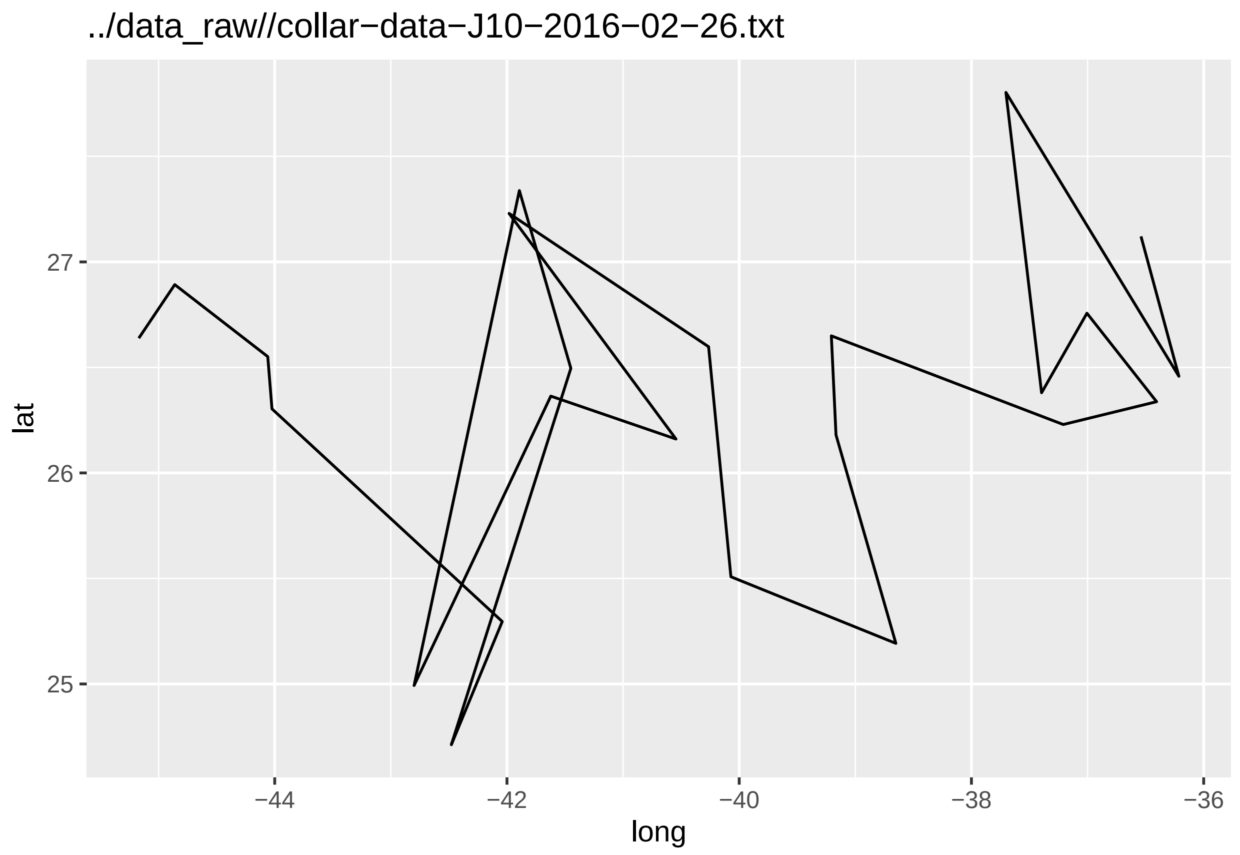

../data_raw//collar−data−F6−2016−02−26.txt
lat
24
22
20
18
long
−44
−42
−40
−38
−36


../data_raw//collar−data−G7−2016−02−26.txt
lat
31
30
29
28
long
−44
−42
−40
−38
−36



../data_raw//collar−data−H8−2016−02−26.txt
lat
long
23
22
21
20
−44
−42
−40
−38
−36

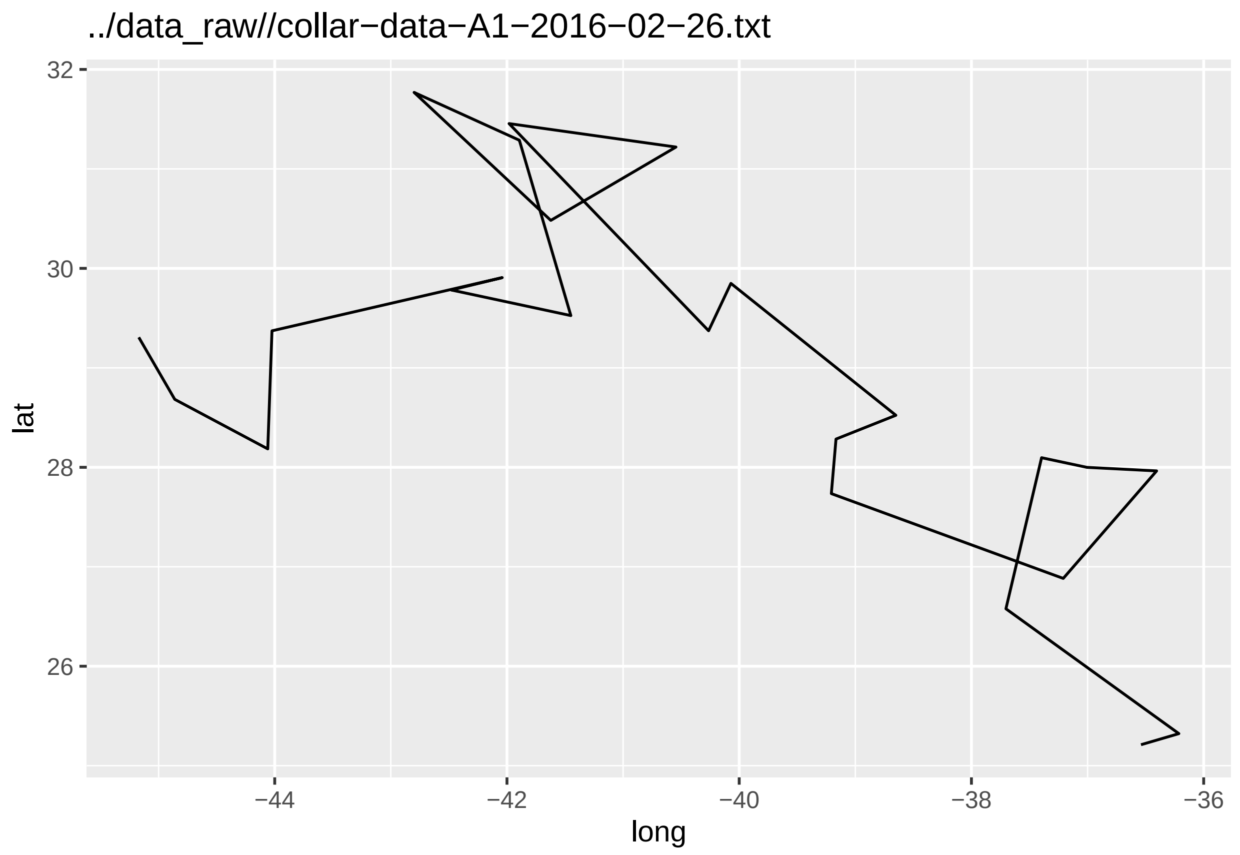

../data_raw//collar−data−I9−2016−02−26.txt
28.5
28.0
27.5
27.0
26.5
26.0
lat
−44
−42
−40
−38
−36
long

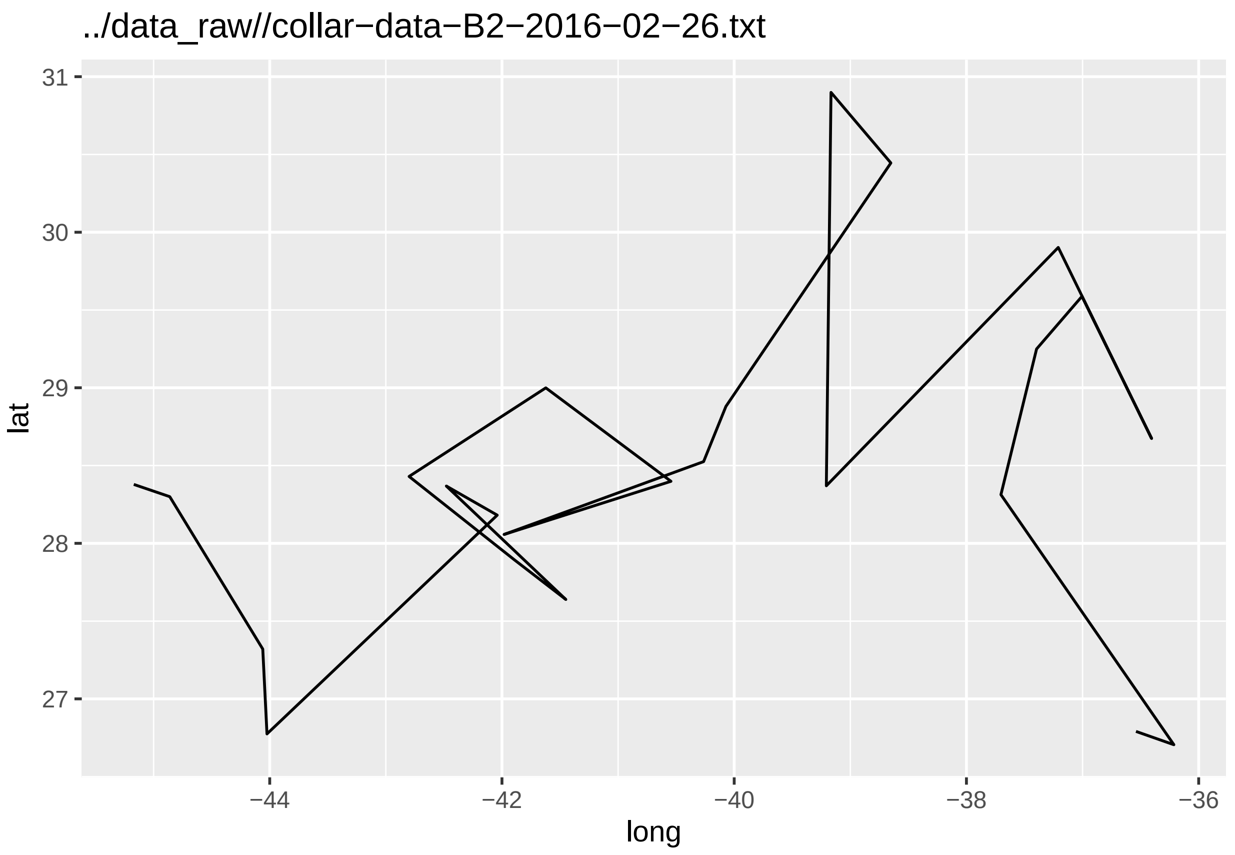


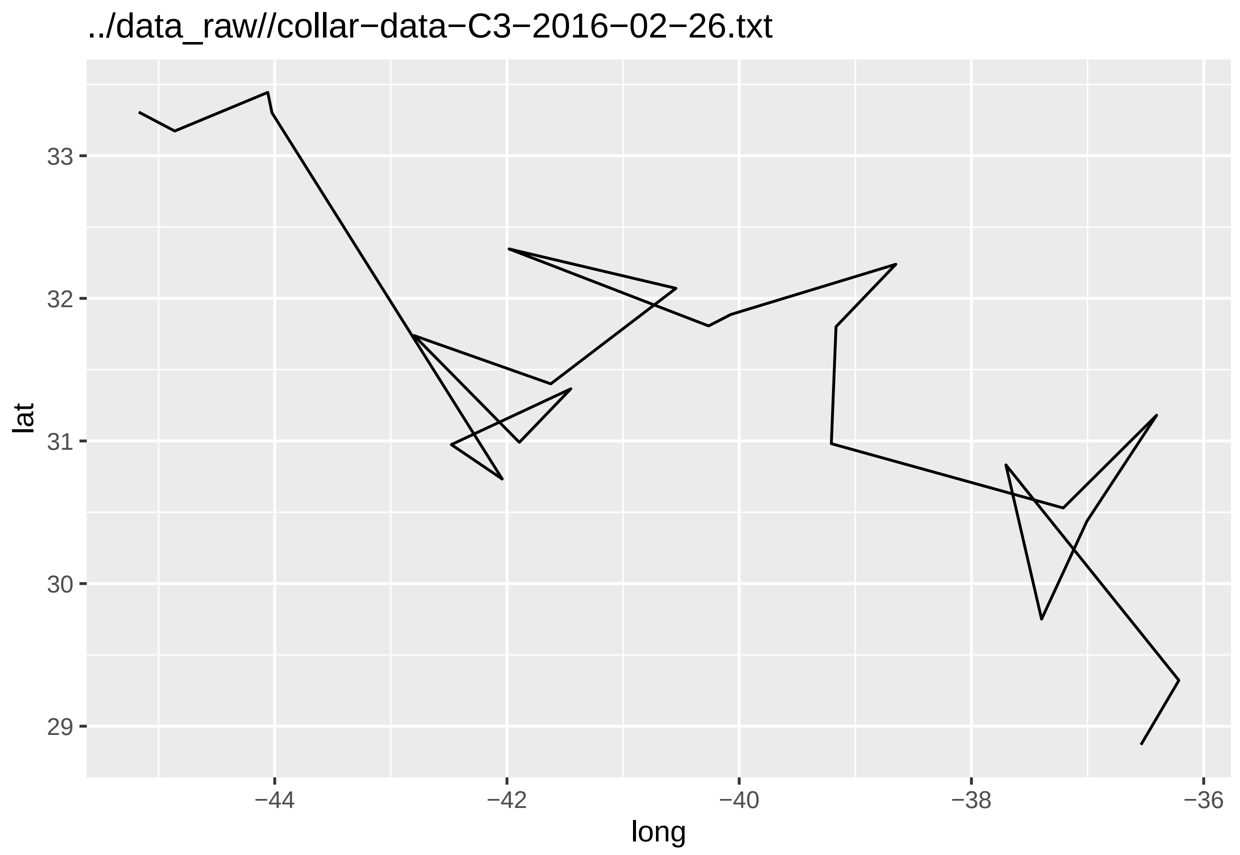
b. Add code to the loop to calculate the minimum and maximum latitude in the file, and store these values, along with the name of the file, in a data frame.

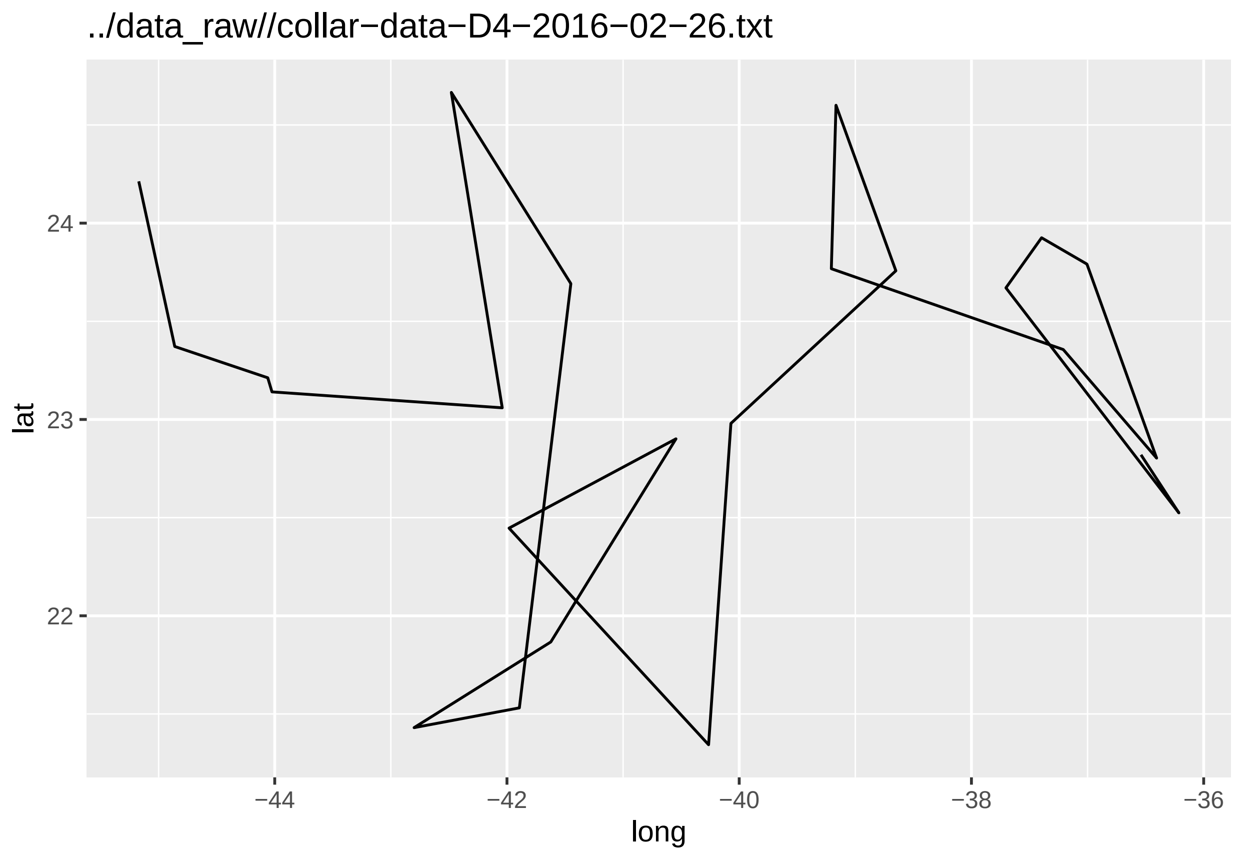
Show the data frame as output.

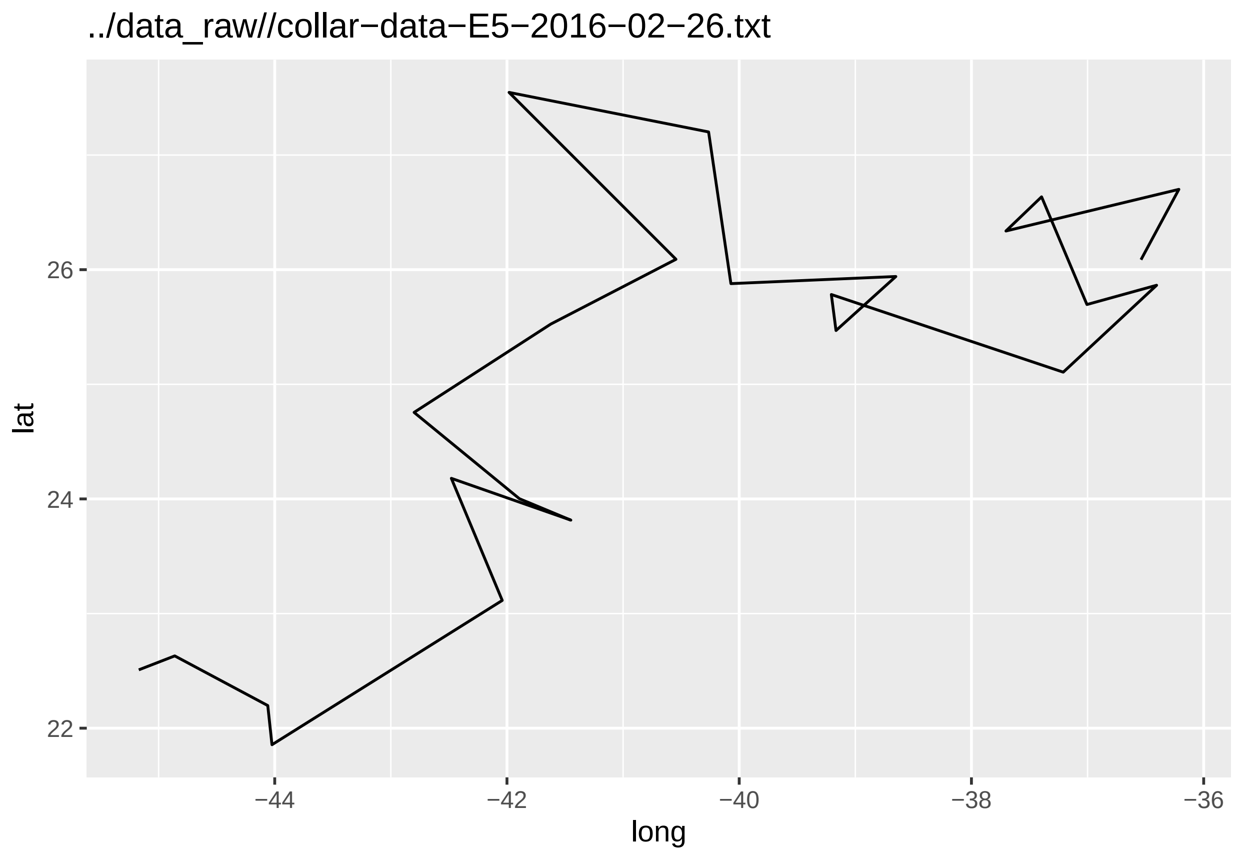
```
for (i in 1:length(collar_data_files)){
  data = read.csv(collar_data_files[i])
  p <- ggplot(data, aes(x = long, y = lat)) + geom_path() + labs(title = collar_data_files[i])
  print(p)
  output$file_name[i] = collar_data_files[i]
  output$max_lat[i] = max(data$lat)
  output$min_lat[i] = min(data$lat)
  output$observations[i] = nrow(data)
}
```

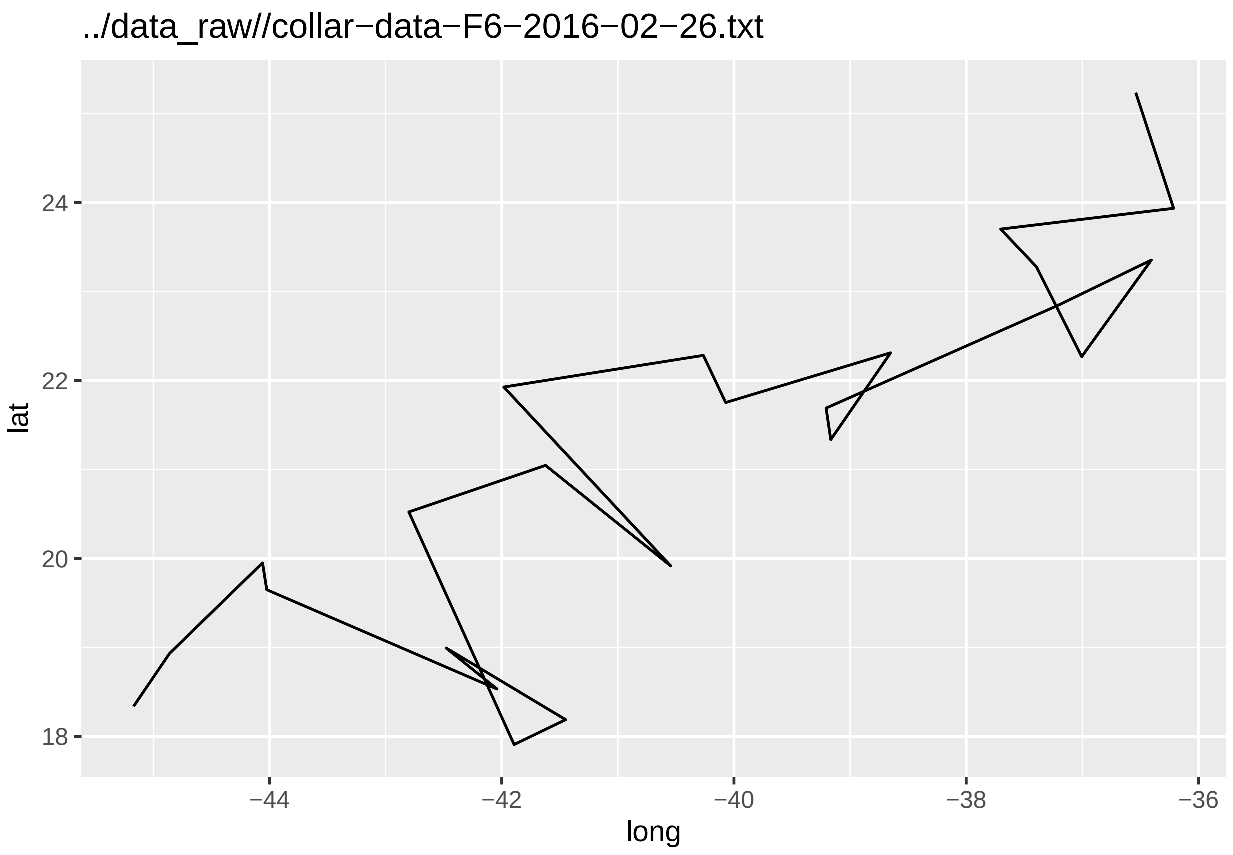

../data_raw//collar−data−A1−2016−02−26.txt
32
30
28
26
lat
−44
−42
−40
−38
−36
long


../data_raw//collar−data−B2−2016−02−26.txt
lat
long
31
30
29
28
27
−44
−42
−40
−38
−36

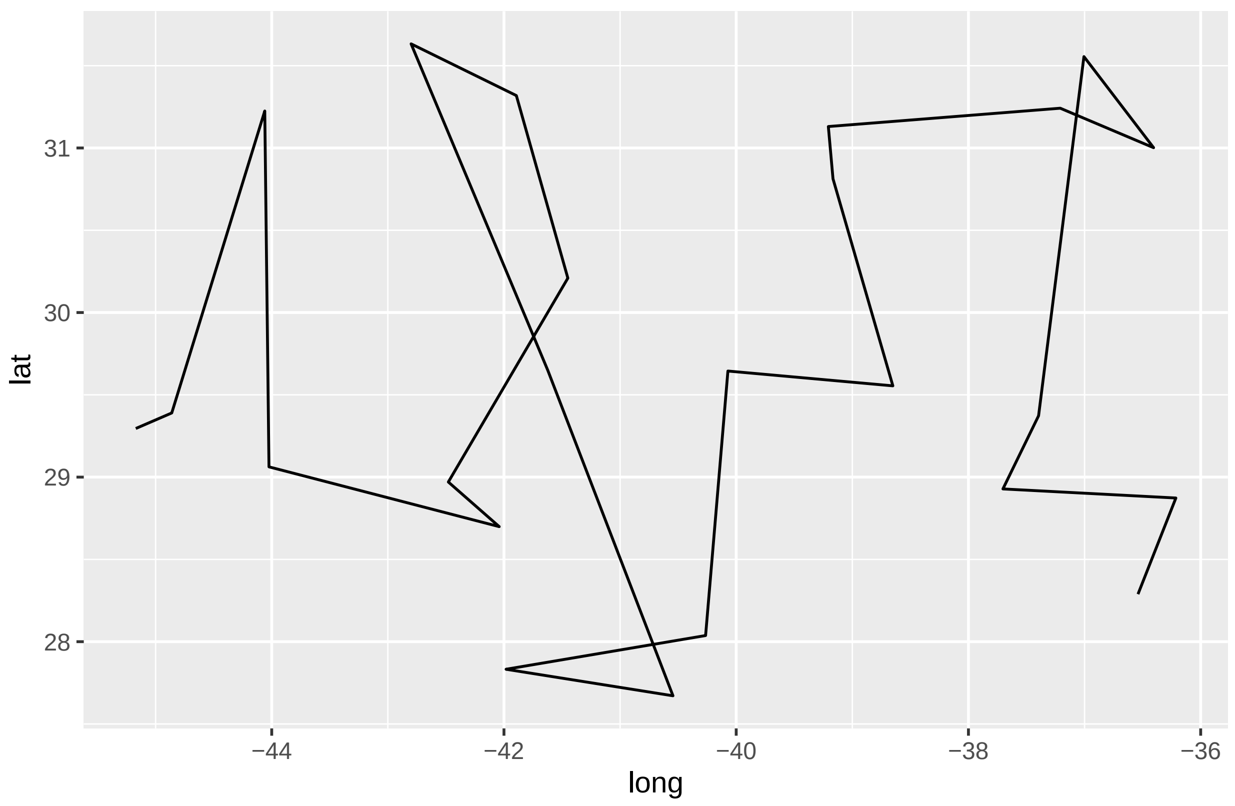

../data_raw//collar−data−C3−2016−02−26.txt
lat
33
32
31
30
29
−44
−42
−40
−38
−36
long

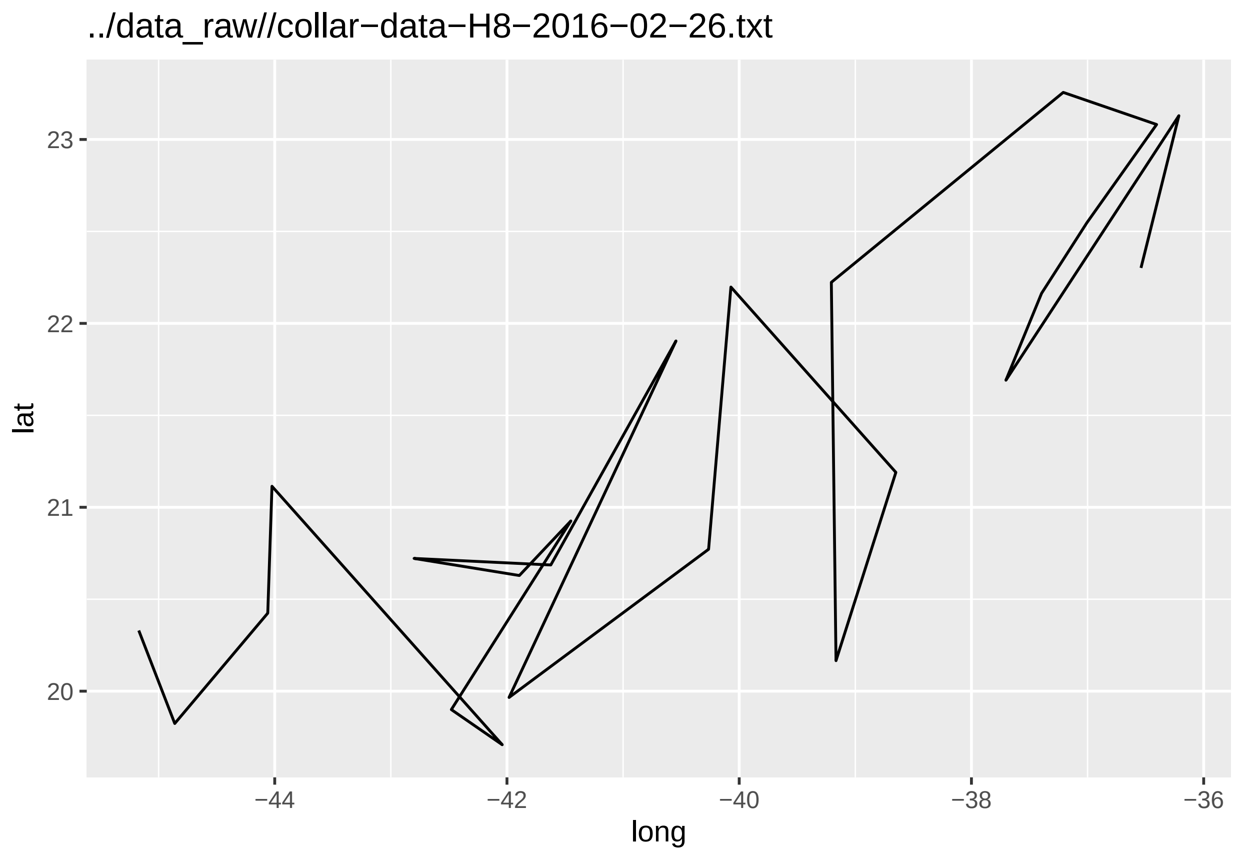

../data_raw//collar−data−D4−2016−02−26.txt
lat
24
23
22
long
−44
−42
−40
−38
−36

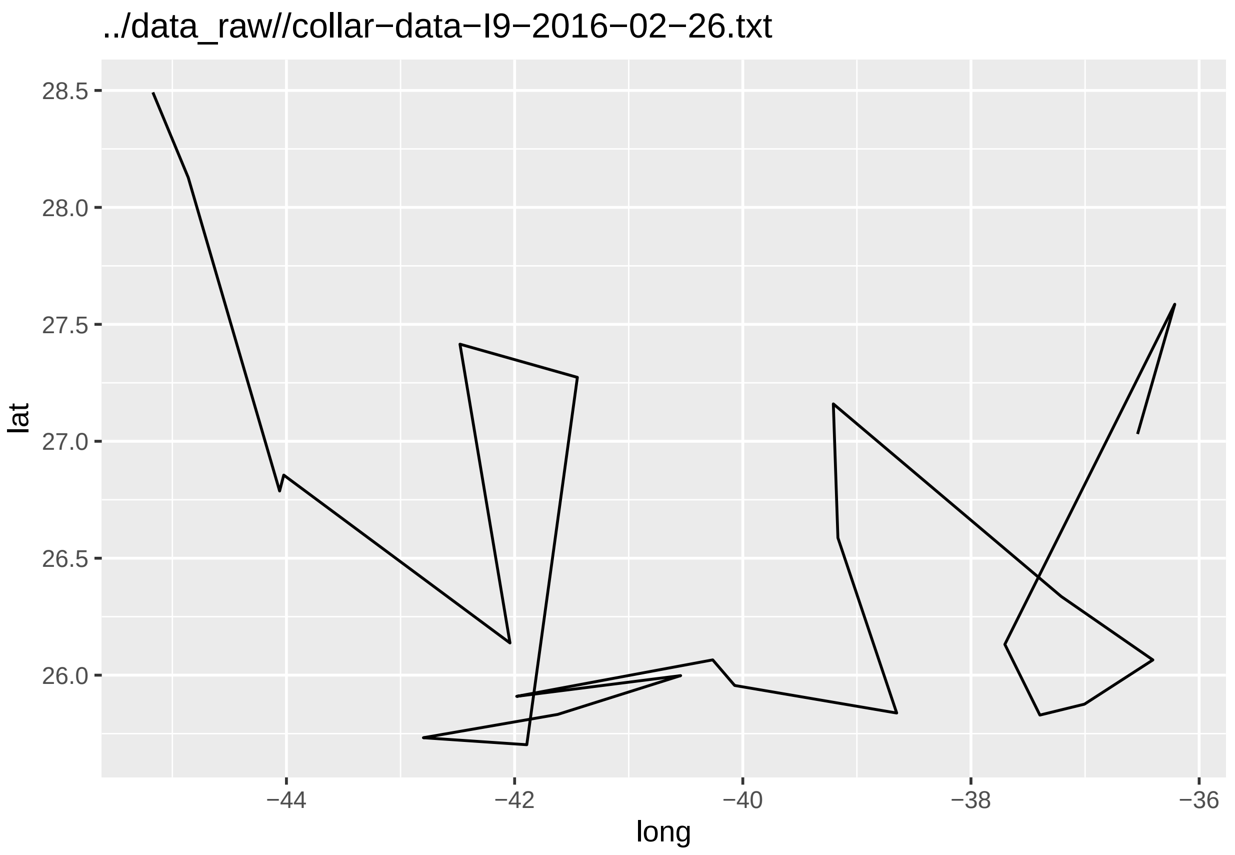

../data_raw//collar−data−E5−2016−02−26.txt
lat
26
24
22
long
−44
−42
−40
−38
−36

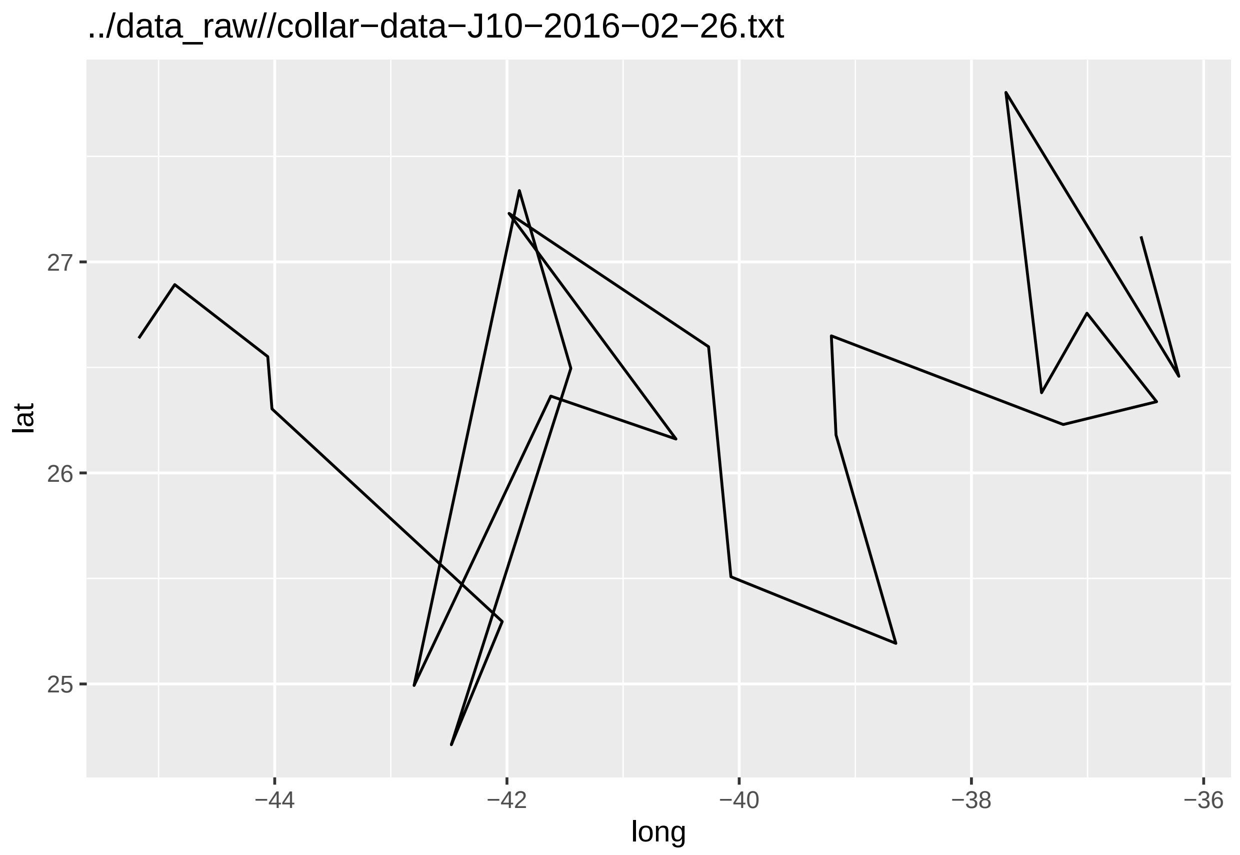

../data_raw//collar−data−F6−2016−02−26.txt
24
22
20
18
lat
−44
−42
−40
−38
−36
long

../data_raw//collar−data−G7−2016−02−26.txt




../data_raw//collar−data−H8−2016−02−26.txt
lat
long
23
22
21
20
−44
−42
−40
−38
−36


../data_raw//collar−data−I9−2016−02−26.txt
lat
28.5
28.0
27.5
27.0
26.5
26.0
long
−44
−42
−40
−38
−36



```
output
```

```
##                                    file_name  max_lat  min_lat observations
## 1   ../data_raw//collar-data-A1-2016-02-26.txt 31.76912 25.21080           24
## 2   ../data_raw//collar-data-B2-2016-02-26.txt 30.89907 26.70509           24
## 3   ../data_raw//collar-data-C3-2016-02-26.txt 33.44421 28.86998           24
## 4   ../data_raw//collar-data-D4-2016-02-26.txt 24.66598 21.34315           24
## 5   ../data_raw//collar-data-E5-2016-02-26.txt 27.54663 21.85565           24
## 6   ../data_raw//collar-data-F6-2016-02-26.txt 25.23623 17.90788           24
## 7   ../data_raw//collar-data-G7-2016-02-26.txt 31.63272 27.67120           24
## 8   ../data_raw//collar-data-H8-2016-02-26.txt 23.25601 19.70875           24
## 9   ../data_raw//collar-data-I9-2016-02-26.txt 28.49172 25.70252           24
## 10 ../data_raw//collar-data-J10-2016-02-26.txt 27.80325 24.71200           24
```

4. DNA or RNA (20 points)

This question has you write a function and then use the function in various forms of iteration.

a. Write a function that determines if a sequence of base pairs is DNA, RNA, or if it is not possible to tell given the sequence provided. RNA has the base Uracil ("u") instead of the base Thymine ("t"), so sequences with u's are RNA, sequences with t's are DNA, and sequences with neither are unknown.

Name the function `dna_or_rna()` and have it take sequence as an argument. Have the function return one of three outputs: "DNA", "RNA", or "UNKNOWN".

HINT: Remember our work with character strings and the `stringr` *package from the* `tidyverse` *(Week 7)?*

```
library(tidyverse)
```

```r
dna_or_rna <- function(sequence){
  # Determines if a character string represents DNA, RNA, or is unknown
  # by searching the character string for the unique base pairs t & u.
  if (str_count(sequence, "t") > 0){
    type = "DNA"
  } else if (str_count(sequence, "u") > 0){
    type = "RNA"
  } else {
    type = "UNKNOWN"
  }
  return(type)
}
```

b. Use the function to determine the sequence type for the following sequences.

```r
seq1 <- "ttgaatgccttacaactgatcattacacaggcggcatgaagcaaaaatatactgtgaaccaatgcaggcg"
seq2 <- "gauuauuccccacaaagggagugggauuaggagcugcaucauuuacaagagcagaauguuucaaaugcau"
seq3 <- "gaaagcaagaaaaggcaggcgaggaagggaagaaggggggaaacc"
```

Before tackling the next questions (c-f), run the following code chunk to create a vector called sequences.

```r
sequences <- c("ttgaatgccttacaactgatcattacacaggcggcatgaagcaaaaatatactgtgaaccaatgcaggcg",
               "gauuauuccccacaaagggagugggauuaggagcugcaucauuuacaagagcagaauguuucaaaugcau",
               "gaaagcaagaaaaggcaggcgaggaagggaagaaggggggaaacc",
               "guuuccuacaguauuugaugagaaugagaguuuacuccuggaagauaauauuagaauguuuacaacugcaccugaucagguggauaag
               "gauaaggaagaugaagacuuucaggaaucuaauaaaaugcacuccaugaauggauucauguaugggaaucagccggguc")
```

c. Use the function you wrote and a `for` loop to create a vector of sequence types for the values in sequences

```r
for (sequence in sequences){
  print(dna_or_rna(sequence))
}
```

```
## [1] "DNA"
## [1] "RNA"
## [1] "UNKNOWN"
## [1] "RNA"
## [1] "RNA"
```

d. Use the function and a `for` loop to create a data frame that includes a column of sequences and a column of their types.

```r
df <- data.frame(sequence = vector(mode = "character", length = length(sequences)),
                 type = vector(mode = "character", length = length(sequences)))

for (i in 1:length(sequences)) {
  df$sequence[i] <- sequences[i]
  df$type[i] <- dna_or_rna(sequences[i])
}

df
```

```
##                                                                                                  seque
## 1                                ttgaatgccttacaactgatcattacacaggcggcatgaagcaaaaatatactgtgaaccaatgcag
## 2                                gauuauuccccacaaagggagugggauuaggagcugcaucauuuacaagagcagaauguuucaaaug
## 3                                                        gaaagcaagaaaaggcaggcgaggaagggaagaaggggggaa
## 4 guuuccuacaguauuugaugagaaugagaguuuacuccuggaagauaauauuagaauguuuacaacugcaccugaucagguggauaaggaagaugaag
## 5                       gauaaggaagaugaagacuuucaggaaucuaauaaaaugcacuccaugaauggauucauguaugggaaucagccgg
##      type
## 1      DNA
## 2      RNA
## 3  UNKNOWN
## 4      RNA
## 5      RNA
```

e. OPTIONAL: Use the function and `sapply` to create a vector of sequence types for the values in sequences

```r
sapply(sequences, dna_or_rna)
```

```
##                                 ttgaatgccttacaactgatcattacacaggcggcatgaagcaaaaatatactgtgaaccaatgcaggcg
##                                                                                                   "DNA
##                                 gauuauuccccacaaagggagugggauuaggagcugcaucauuuacaagagcagaauguuucaaaugcau
##                                                                                                   "RNA
##                                                         gaaagcaagaaaaggcaggcgaggaagggaagaagggggggaaac
##                                                                                               "UNKNOWN
## guuuccuacaguauuugaugagaaugagaguuuacuccuggaagauaauauuagaauguuuacaacugcaccugaucagguggauaaggaagaugaagac
##                                                                                                   "RNA
##                       gauaaggaagaugaagacuuucaggaaucuaauaaaaugcacuccaugaauggauucauguaugggaaucagccgggu
##                                                                                                   "RNA
```

f. OPTIONAL: Use the function, and `mutate()` to create a data frame that includes a column of sequences and a column of their types. First, run the following line of code to turn the sequence vector into a dataframe.

```r
seq_df <- as.data.frame(sequences)
seq_df |>
  rowwise() |>
  mutate(type = dna_or_rna(sequences))
```

```
## # A tibble: 5 x 2
## # Rowwise:
##   sequences                                                           type
##   <chr>                                                               <chr>
## 1 ttgaatgccttacaactgatcattacacaggcggcatgaagcaaaaatatactgtgaaccaatgcaggcg   DNA
## 2 gauuauuccccacaaagggagugggauuaggagcugcaucauuuacaagagcagaauguuucaaaugcau   RNA
## 3 gaaagcaagaaaaggcaggcgaggaagggaagaagggggggaaacc                           UNKN~
## 4 guuuccuacaguauuugaugagaaugagaguuuacuccuggaagauaauauuagaauguuuacaacugcac~ RNA
## 5 gauaaggaagaugaagacuuucaggaaucuaauaaaaugcacuccaugaauggauucauguaugggaauca~ RNA
```