IMT 573: Problem Set 4 - Scrape data and get it in shape

Srushti Chaukhande

Due: Tuesday, October 29, 2024 by 10:00AM PT

Collaborators:

Instructions: Before beginning this assignment, please ensure you have access to R and RStudio.

1. Download the `pedi.Rmd` file from Canvas. Open `problem_set4.Rmd` in RStudio and supply your solutions to the assignment by editing `problem_set4.Rmd`.
2. Replace the "Insert Your Name Here" text in the `author:` field with your own full name. Any collaborators must be listed on the top of your assignment. Collaboration shouldn't be confused with group project work (where each person does a part of the project). Working on problem sets should be your individual contribution.
3. Be sure to include well-documented (e.g. commented) code chucks, figures, and clearly written text chunk explanations as necessary. Any figures should be clearly labeled and appropriately referenced within the text. Be sure that each visualization adds value to your written explanation; avoid redundancy – you do not need four different visualizations of the same pattern.
4. All materials and resources that you use (with the exception of lecture slides) must be appropriately referenced within your assignment. In particular, note that Stack Overflow is licenses as Creative Commons (CC-BY-SA). This means you have to attribute any code you refer from SO.
5. Partial credit will be awarded for each question for which a serious attempt at finding an answer has been shown. But please **DO NOT** submit pages and pages of hard-to-read code and attempts that is impossible to grade. That is, avoid redundancy. Remember that one of the key goals of a data scientist is to produce coherent reports that others can easily follow. Students are *strongly* encouraged to attempt each question and to document their reasoning process even if they cannot find the correct answer. If you would like to include R code to show this process, but it does not run without errors you can do so with the `eval=FALSE` option as follows:

```
a + b # these object dont' exist
# if you run this on its own it with give an error
```

6. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly when you click `Knit PDF`, rename the knitted PDF file to `ps1_ourLastName_YourFirstName.pdf`, and submit the PDF file on Canvas.
7. Collaboration is often fun and useful, but each student must turn in an individual write-up in their own words as well as code/work that is their own. Regardless of whether you work with others, what you turn in must be your own work; this includes code and interpretation of results. The names of all collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.

Setup In this problem set you will need, at minimum, the following R packages. Install missing packages, if required.

Instructions

1. Scrape the web

This question asks you to extract for tables of data from the internet. In particular, you are going to fetch the location of all tall mountains from wikipedia, and plot these on the world map!

Briefly, your tasks are the following:

- Load the wikipedia list of mountains by elevation
- Find all mountains taller than 6800 meters
- Follow the link to the wikipedia page of that mountain, and extract the geographic coordinates (longitude/latitude).
- Plot these locations on the world map.

There is a brief introduction to rvest and webscraping in R at http://faculty.washington.edu/otoomet/machinelearning-R/web-scraping.html. Below is the more detailed task lists. The tasks are listed in order that you may find useful to follow when developing your code, but you may also combine different tasks in a different order.

1.1 Parse the list of mountains (30pt)

This is a suggested approach for you to follow. You may do the tasks slightly differently.

1. (4pt) Load the wikipedia list of mountains by height (https://en.wikipedia.org/wiki/List_of_mountains_by_elevation) and parse it with rvest library.
2. (4pt) Find all the tables there in the html.
3. (7pt) Find the table headers, and determine which columns are mountain names, heights, and where are the links to the individual mountain pages. Note: if you analyze the page in your browser, you'll probably get javascript-aware version. Down- loading in R gives you the non-javascript version, the table headers differ somewhat between these two versions.
4. (13pt) Create a data frame that contains names and heights of the mountains above 6800m, and the links to the corresponding wikipedia pages. You'll add longitude and latitude for each mountain in this data frame later. Hint: you should have 100+ mountains with valid links.
5. (2pt) Print a small sample of your data frame to see that it looks reasonable.

Solution 1.1

1. **Solution:**

Insert Response

```
#Load the wikipedia list of mountains
url <- "https://en.wikipedia.org/wiki/List_of_mountains_by_elevation"
webpage <- read_html(url)
```

2. **Solution:**

Insert Response

```
# Find all the tables in HTML
tables <- webpage %>% html_nodes("table")
# Extract all tables from the page
all_tables <- lapply(tables, function(tbl) {
  # Convert each table to a data frame with all columns as character
  tbl_df <- html_table(tbl)
  tbl_df[] <- lapply(tbl_df, as.character)
  return(tbl_df)
})

#Details of all tables in HTML
str(all_tables)
```

```
## List of 9
##  $ : tibble [14 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ Mountain          : chr [1:14] "Mount Everest" "K2" "Kangchenjunga" "Lhotse" ...
##   ..$ Metres            : chr [1:14] "8,848" "8,611" "8,586" "8,516" ...
##   ..$ Feet              : chr [1:14] "29,029" "28,251" "28,169" "27,940" ...
##   ..$ Range             : chr [1:14] "Himalayas" "Karakoram" "Himalayas" "Himalayas" ...
##   ..$ Location and Notes: chr [1:14] "Nepal/China" "Pakistan/China" "Nepal/India" "Nepal -
 Climbers ascend Lhotse Face in climbing Everest" ...
##  $ : tibble [134 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ Mountain          : chr [1:134] "Gasherbrum III" "Gyachung Kang" "Annapurna II" "Gasherbrum IV
##   ..$ Metres            : chr [1:134] "7,952" "7,952" "7,937" "7,932" ...
##   ..$ Feet              : chr [1:134] "26,089" "26,089" "26,040" "26,024" ...
##   ..$ Range             : chr [1:134] "Karakoram" "Himalayas" "Himalayas" "Karakoram" ...
##   ..$ Location and Notes: chr [1:134] "Pakistan" "Nepal (Khumbu)/China" "Nepal" "Pakistan" ...
##  $ : tibble [116 x 4] (S3: tbl_df/tbl/data.frame)
##   ..$ Mountain          : chr [1:116] "Machapuchare" "Laila Peak (Haramosh Valley)" "Kang Guru" "Gasl
##   ..$ Metres            : chr [1:116] "6,993" "6,985" "6,981" "6,979" ...
##   ..$ Feet              : chr [1:116] "22,943" "22,917" "22,904" "22,897" ...
##   ..$ Location and Notes: chr [1:116] "Annapurna Himalaya, Nepal - Officially unclimbed (attempts no
 2005 avalanche kills 18" "Karakoram, Pakistan" ...
##  $ : tibble [79 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ Mountain          : chr [1:79] "Laila Peak" "Mount Logan" "Alpamayo" "Cerro Lípez" ...
##   ..$ Metres            : chr [1:79] "5,971" "5,959" "5,947" "5,929" ...
##   ..$ Feet              : chr [1:79] "19,590" "19,551" "19,511" "19,452" ...
##   ..$ Range             : chr [1:79] "Himalaya" "Saint Elias Mountains" "Andes" "Andes" ...
##   ..$ Location and Notes: chr [1:79] "Pakistan" "Yukon, Canada - Highest in Canada" "Peru" "Bolivia"
##  $ : tibble [263 x 4] (S3: tbl_df/tbl/data.frame)
##   ..$ Mountain          : chr [1:263] "Mount Blackburn" "Pico Bolívar" "Talgar Peak" "Shota Rustaveli
##   ..$ Metres            : chr [1:263] "4,996" "4,981" "4,979" "4,960" ...
##   ..$ Feet              : chr [1:263] "16,391" "16,342" "16,335" "16,273" ...
##   ..$ Location and Notes: chr [1:263] "Wrangell Mtns., Alaska, US (also given 5036 m)" "Sierra Nevada
 Highest in Venezuela" "Tian Shan, Kazakhstan - Highest in northern Tian Shan" "Caucasus Mountains, Svan
##  $ : tibble [347 x 4] (S3: tbl_df/tbl/data.frame)
##   ..$ Mountain          : chr [1:347] "Piz Zupò" "Truchas Peak" "Fletschhorn" "Mount Albert Edward"
##   ..$ Metres            : chr [1:347] "3,995" "3,994" "3,993" "3,990" ...
##   ..$ Feet              : chr [1:347] "13,107" "13,104" "13,100" "13,091" ...
```

```
##   ..$ Location and Notes: chr [1:347] "Bernina Range, Switzerland" "Sangre de Cristo Mountains, New M
##  $ : tibble [306 x 4] (S3: tbl_df/tbl/data.frame)
##   ..$ Mountain          : chr [1:306] "Tre Cime di Lavaredo" "Pizzo Centrale" "Cascade Mountain" "Co
##   ..$ Metres            : chr [1:306] "2,999" "2,999" "2,998" "2,997" ...
##   ..$ Feet              : chr [1:306] "9,839" "9,839" "9,836" "9,833" ...
##   ..$ Location and Notes: chr [1:306] "Province of Belluno, Italy" "Switzerland" "Vermilion Range, Ca
##  $ : tibble [230 x 4] (S3: tbl_df/tbl/data.frame)
##   ..$ Mountain          : chr [1:230] "Serra da Estrela" "Mount Bogong" "Mount Ishizuchi" "Doi Phu Kh
##   ..$ Metres            : chr [1:230] "1,993" "1,986" "1,982" "1,980" ...
##   ..$ Feet              : chr [1:230] "6,539" "6,516" "6,503" "6,496" ...
##   ..$ Location and Notes: chr [1:230] "Portugal" "Australia - Highest in Victoria" "Japan -
 Tallest in Western Japan" "Luang Prabang Range, Thailand" ...
##  $ : tibble [131 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ Mountain          : chr [1:131] "Sgurr Dearg" "Mount Sizer" "Mount Valin" "Hyangnosan" ...
##   ..$ Metres            : chr [1:131] "986" "980" "980" "979" ...
##   ..$ Feet              : chr [1:131] "3,235" "3,215" "3,215" "3,212" ...
##   ..$ Range             : chr [1:131] "Cuillin" "Diablo Range" "Saguenay Lac St-Jean" "" ...
##   ..$ Location and Notes: chr [1:131] "Scotland" "US (California)" "Canada (Québec)" "Gyeongnam Provi
```

3. **Solution:**

Insert Response

```r
# Preview each table's headers.
lapply(all_tables, function(tbl_df) names(tbl_df))
```

```
## [[1]]
## [1] "Mountain"           "Metres"
## [3] "Feet"               "Range"
## [5] "Location and Notes"
##
## [[2]]
## [1] "Mountain"           "Metres"
## [3] "Feet"               "Range"
## [5] "Location and Notes"
##
## [[3]]
## [1] "Mountain"           "Metres"
## [3] "Feet"               "Location and Notes"
##
## [[4]]
## [1] "Mountain"           "Metres"
## [3] "Feet"               "Range"
## [5] "Location and Notes"
##
## [[5]]
## [1] "Mountain"           "Metres"
## [3] "Feet"               "Location and Notes"
##
## [[6]]
## [1] "Mountain"           "Metres"
## [3] "Feet"               "Location and Notes"
##
```

```
## [[7]]
## [1] "Mountain"           "Metres"
## [3] "Feet"               "Location and Notes"
##
## [[8]]
## [1] "Mountain"           "Metres"
## [3] "Feet"               "Location and Notes"
##
## [[9]]
## [1] "Mountain"           "Metres"
## [3] "Feet"               "Range"
## [5] "Location and Notes"
```

```r
# Extract mountain links from the first column of all tables
mountain_links <- tables %>%
  map_dfr(~ {
    links <- .x %>% html_nodes("tr td:first-child a") %>% html_attr("href")
    titles <- .x %>% html_nodes("tr td:first-child a") %>% html_text()

    # Construct full URLs
    links <- ifelse(startsWith(links, "/"), paste0("https://en.wikipedia.org", links)
                    , links)

    # Create a data frame with names and links for each table
    tibble(Mountain = trimws(titles), Page_link = links)  # Trim whitespace
  })
```

- The columns in each table differ slightly as we can see.
- The links are relative so I have made them whole

4. **Solution:**

Insert Response

```r
# Loop over all tables and filter for mountains above 6800 meters
mountains_above_6800 <- map_dfr(tables, ~ {
  # Extract the data frame directly from the HTML node
  tbl_df <- html_table(.x)
  tbl_df[] <- lapply(tbl_df, as.character)  # Ensure all columns are characters

  # Extract mountain names
  titles <- .x %>% html_nodes("tr td:first-child a") %>% html_text()

  # Select columns for Metres and Feet
  filtered_table <- tbl_df %>%
    select(Metres, Feet) %>%
    filter(as.numeric(gsub("[^0-9.]", "", Metres)) > 6800)  # Filter for heights > 6800m

  # Create a data frame with mountain names and corresponding heights
  filtered_table <- tibble(
    Mountain = titles[1:nrow(filtered_table)],  # Ensure we match rows with titles
    Metres = filtered_table$Metres,
    Feet = filtered_table$Feet
```

```
  )


  # Extract corresponding links for filtered mountains
  filtered_links <- mountain_links %>%
    filter(Mountain %in% filtered_table$Mountain)  # Match mountain names

  # Combine filtered mountains with their corresponding links
  filtered_table <- left_join(filtered_table, filtered_links, by = "Mountain")


  return(filtered_table)
})
```

5. **Solution:**

Insert Response

```
# View the resulting data frame
head(mountains_above_6800)
```

```
## # A tibble: 6 x 4
##   Mountain      Metres Feet   Page_link
##   <chr>         <chr>  <chr>  <chr>
## 1 Mount Everest 8,848  29,029 https://en.wikipedia.org/wiki~
## 2 K2            8,611  28,251 https://en.wikipedia.org/wiki~
## 3 Kangchenjunga 8,586  28,169 https://en.wikipedia.org/wiki~
## 4 Lhotse        8,516  27,940 https://en.wikipedia.org/wiki~
## 5 Makalu        8,485  27,838 https://en.wikipedia.org/wiki~
## 6 Cho Oyu       8,188  26,864 https://en.wikipedia.org/wiki~
```

1.2 Scrape the individual mountain data (25pt)

For this step, we recommend to write a function that takes the link as an argument, and scrapes the mountains' data. The coordinates are given in degrees-minutes-seconds east/west, north/south; so you also need a function to transform those to degrees (east/north is positive, west/south negative) to match the plotting below.

However, you do not have to follow our suggestions if you have a better idea.

1. (4pt) Write a function that converts the longitude/latitude string to degrees (positive and negative). Hint: here is code that converts coordinates like 28 35'46"N or 83 49'13"E into degrees (note: you have to ensure you are using the exact same degree-minute-second symbols as in wikipedia):

```
## ddmmssDD is the coordinate string (e.g., "28 35'46\"N" or "83 49'13\"E")
ddmmssDD <- "28 35'46\"N"  # Example coordinate, replace with your own

# Determine direction: west/south are negative
D <- if (grepl("[WS]", ddmmssDD)) -1 else 1

# Use the exact symbols as in Wikipedia for splitting
```

```
# Split by spaces for degrees and minutes, and handle
# the special symbol for minutes ' and seconds "
dms <- strsplit(ddmmssDD, "[ '\"]")[[1]]

# Extract degrees, minutes, and seconds as numeric values
dd <- as.numeric(dms[1])
mm <- as.numeric(dms[2])
ss <- as.numeric(dms[3])

# Convert to decimal degrees
decimal_degrees <- (dd + mm/60 + ss/3600) * D
```

2. (10pt) Write another function that takes link as an argument and loads the mountain's html page and extracts latitude and longitude. Hint: you may run intro trouble with certain links (in particular non-existing wiki pages). In that case you may want to wrap the download code into try-block like this:

```
page <- try(read_html(url), silent=TRUE)
                          # if it works, you get the page
                          # if not, you get "try-error"
                          # silent: do not show errors in markdown
if(inherits(page, "try-error"))
                          # got 404 or another error, return NULL
  return(NULL)
## Process the page here
## ...
```

3. (7pt) loop over the table of mountains you did above, download the mountain data, and extract the coordinates. Store these into the same data frame. Hint: I managed to download data for 152 mountains

4. (4pt) Print a sample of the dataframe and check that it looks good. How many mountains did you get?

Solution 1.2

1. **Solution:**

Insert Response

```
convert_ddmmss_to_degrees <- function(ddmmssDD) {
  # Clean the string to remove extraneous spaces and special characters
  ddmmssDD <- gsub("[^0-9° NSEW]", "", ddmmssDD)

  # Determine direction: west/south are negative
  D <- if (any(grepl("[WS]", ddmmssDD))) -1 else 1

  # Extract degrees, minutes, and seconds using regex
  dms <- strsplit(ddmmssDD, "[° ]")[[1]]

  # Check that we have all three parts: degrees, minutes, and seconds
  if (length(dms) < 3) {
```

```
    warning("Incomplete coordinate format: ", ddmmssDD)
    return(NA)
  }

  # Convert extracted parts to numeric
  dd <- as.numeric(dms[1])
  mm <- as.numeric(dms[2])
  ss <- as.numeric(dms[3])

  # Check for NAs after conversion
  if (any(is.na(c(dd, mm, ss)))) {
    warning("Invalid coordinate format: ", ddmmssDD)
    return(NA)
  }

  # Calculate decimal degrees
  decimal_degrees <- (dd + mm / 60 + ss / 3600) * D
  return(decimal_degrees)
}

# Test examples
convert_ddmmss_to_degrees("76°37 E")    # Should return a valid decimal degree
```

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

2. **Solution:**

Insert Response

```
# Function to extract latitude and longitude from a mountain's Wikipedia page
extract_lat_long <- function(link) {
  # Try to read the HTML page
  page <- try(read_html(link), silent = TRUE)

  # Check if the page was successfully loaded
  if (inherits(page, "try-error")) {
    # Return NULL if the page does not exist
    return(NULL)
  }

  # Extract latitude and longitude using class names
  # Assuming the latitude and longitude are in elements
  # with class names "latitude" and "longitude"
  latitude <- page %>%
    html_nodes(".infobox-data .latitude") %>%
    html_text(trim = TRUE)

  longitude <- page %>%
    html_nodes(".infobox-data .longitude") %>%
    html_text(trim = TRUE)

  # Check if both latitude and longitude were extracted
```

```r
  if (length(latitude) == 0 || length(longitude) == 0) {
    return(NULL)  # Return NULL if extraction fails
  }

  # Return as a list
  return(list(Latitude = latitude, Longitude = longitude))
}

# Example usage with a link from mountains_above_6800
# Replace with an actual link from your data
example_link <- "https://en.wikipedia.org/wiki/K2"
# Replace with a valid link from mountains_above_6800
#example_link <- "https://de.wikipedia.org/wiki/Kampire_Dior"
lat_long <- extract_lat_long(example_link)

# Print the result
print(lat_long)
```

```
## $Latitude
## [1] "35°52 57 N"
##
## $Longitude
## [1] "76°30 48 E"
```

3. **Solution:**

Insert Response

```r
# Initialize Latitude and Longitude columns
mountains_above_6800$Latitude <- NA
mountains_above_6800$Longitude <- NA

# Loop over each row in the mountains_above_6800 data frame
for (i in 1:nrow(mountains_above_6800)) {
  # Extract the link for the current mountain
  link <- mountains_above_6800$Page_link[i]

  # Validate link
  if (is.na(link) || !is.character(link) || nchar(link) == 0) {
    next
  }

  # Call the extract_lat_long function to get coordinates
  coordinates <- extract_lat_long(link)

  # Check coordinates structure
  if (!is.null(coordinates) && all(c("Latitude", "Longitude") %in% names(coordinates))) {
    mountains_above_6800$Latitude[i] <- coordinates$Latitude
    mountains_above_6800$Longitude[i] <- coordinates$Longitude
  } else {
    mountains_above_6800$Latitude[i] <- NA
    mountains_above_6800$Longitude[i] <- NA
```

```
  }
}

# Convert to decimal degrees
mountains_above_6800 <- mountains_above_6800 %>%
  mutate(
    Latitude = sapply(Latitude, convert_ddmmss_to_degrees),
    Longitude = sapply(Longitude, convert_ddmmss_to_degrees)
  )
```

4. **Solution:**

Insert Response

```
#printing sample of data frame
head(mountains_above_6800)
```

```
## # A tibble: 6 x 6
##   Mountain      Metres Feet   Page_link   Latitude Longitude
##   <chr>         <chr>  <chr>  <chr>          <dbl>     <dbl>
## 1 Mount Everest 8,848  29,029 https://en~     28.0      86.9
## 2 K2            8,611  28,251 https://en~     35.9      76.5
## 3 Kangchenjunga 8,586  28,169 https://en~     27.7      88.1
## 4 Lhotse        8,516  27,940 https://en~     28.0      86.9
## 5 Makalu        8,485  27,838 https://en~     27.9      87.1
## 6 Cho Oyu       8,188  26,864 https://en~     28.1      86.7
```

1.3 Plot the mountains (12pt)

Finally, plot the mountains

1. (8pt) Plot all the mountains on a world map. Color those according to their height. Here is an example how to plot data on map:

```
data <- data.frame(city=c("Vientiane", "Kinshasa"),
                   longitude=c(102.5, 15.3),
                   latitude=c(18, -4.3))
world <- map_data("world")
ggplot(world) +
  geom_polygon(aes(long, lat, group=group),
               col="white", fill="gray") +
  geom_point(data=mountains_above_6800, aes(Longitude, Latitude)) +
  coord_quickmap()
```

2. (4pt) Describe what did you get. Where are the tall mountains located? Do all the locations make sense (i.e. you do not have mountains in the middle of sea and such)?

Solution 1.3

1. **Solution:**

Insert Response

```
# Load necessary libraries
library(ggplot2)
library(maps)
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:viridis':
##
##     unemp
```

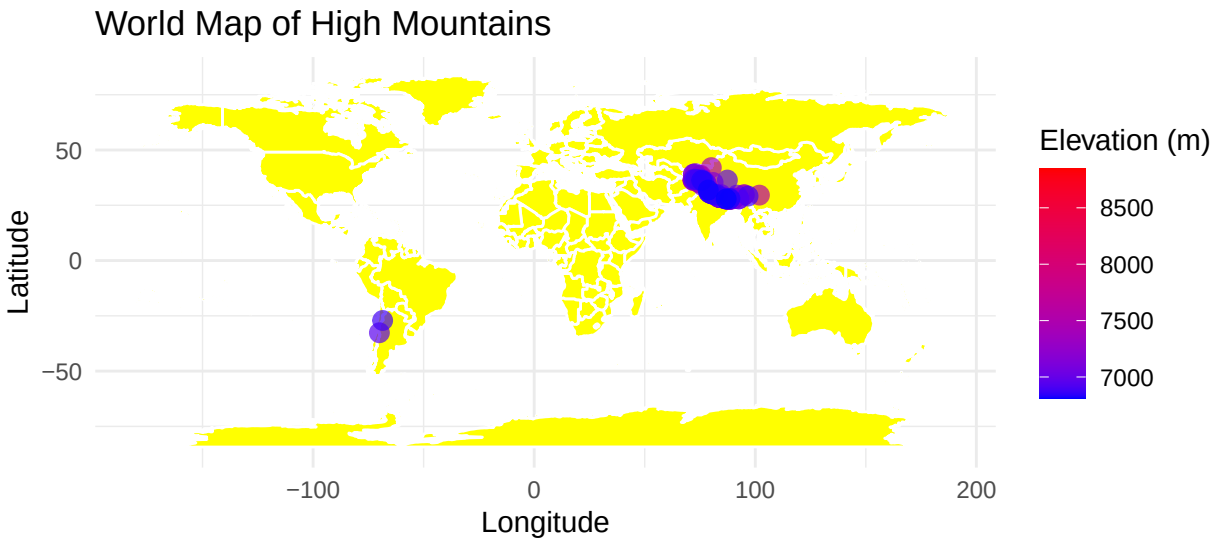
```
## The following object is masked from 'package:purrr':
##
##     map
```

```
# Ensure Longitude and Latitude are numeric
mountains_above_6800$Longitude <- as.numeric(mountains_above_6800$Longitude)
mountains_above_6800$Latitude <- as.numeric(mountains_above_6800$Latitude)

# Clean the Metres column: remove commas and convert to numeric
mountains_above_6800$Metres <- as.numeric(gsub(",", "", mountains_above_6800$Metres))

# Remove rows with NA values in Longitude, Latitude, or Metres
mountains_above_6800 <- na.omit(mountains_above_6800)
# Load world map data
world <- map_data("world")

# Plot mountains with color gradient based on height
ggplot() +
  geom_polygon(data = world, aes(x = long, y = lat, group = group), color = "white", fill = "yellow") +
  geom_point(data = mountains_above_6800,
             aes(x = Longitude, y = Latitude, color = Metres),
             size = 3, alpha = 0.7) +
  scale_color_gradient(low = "blue", high = "red", name = "Elevation (m)") +
  coord_quickmap() +
  labs(title = "World Map of High Mountains",
       x = "Longitude", y = "Latitude") +
  theme_minimal()
```



2. **Solution:**

Insert Response

Most mountains are located above china and India in Himalayas and west of South America. The location does make sense.

Problem 2: Asking Data Science Questions: Crime and Educational Attainment (25pt)

In Problem Set 3, you joined data about crimes and educational attainment. Here you will use this new combined dataset to examine questions around crimes in Seattle and the educational attainment of people living in the areas in which the crime occurred.

(a) (5pt) Develop a Data Science Question Develop your own question to address in this analysis. Your question should be specific and measurable, and it should be able to be addressed through a basic analysis of the crime dataset you compiled in Problem Set 3.

Solution:

Insert Response

My question is : What types of crimes are most prevalent in areas with lower educational attainment?

I chose this questions as according to me it will provide a good idea about crimes where people have less amount of education.

(b) (5pt) Describe and Summarize Briefly summarize the dataset, describing what data exists and its basic properties. Comment on any issues that need to be resolved before you can proceed with your analysis.

Solution:

Insert Response

Dataset Summary The dataset consists of 347,980 records with 44 columns, capturing details on crime incidents in Seattle and the educational attainment of people in areas where these incidents occurred. Key columns include:

Crime Information: Report Number, Occurred Date, Occurred Time, Reported Date, Crime Subcategory, Primary Offense Description. Location Information: Precinct, Sector, Beat, Neighborhood, Latitude, and Longitude. Educational Attainment: Columns representing various education levels, such as no_schooling, high_school_diploma, and bachelors_degree.

Data Quality and Issues Missing Values: Approximately half of the entries contain missing values for the educational attainment fields, likely because not all locations have corresponding census data. These missing values should be addressed, either by filtering or imputing, to ensure consistent analysis.

Inconsistent Data Formats: The Occurred Date column needs to be converted to a date format for time-based analysis, and Occurred Time requires parsing to extract hours.

Thresholds for Analysis: Determining cutoffs (e.g., low vs. high educational attainment) is necessary to make comparisons meaningful. Deciding these thresholds in a way that reflects educational distribution in Seattle would add rigor.

By resolving these issues, we can ensure that our analysis accurately reflects crime patterns in relation to educational attainment across Seattle neighborhoods.

```r
# Load necessary libraries
library(dplyr)
library(ggplot2)
library(lubridate)

# Load the dataset
crime_data <- read.csv("/Users/srushti/Downloads/573 Data Science/crime_data_joined.csv")

# Inspect data structure
str(crime_data)
```

```
## 'data.frame':    347980 obs. of  44 variables:
##  $ Report.Number                          : num  2.01e+13 2.01e+13 2.01e+13 2.01e+13 2.01e+13 ...
##  $ Occurred.Date                          : chr  "2012-04-02" "2012-04-02" "2012-04-02" "2012-04-02"
##  $ Occurred.Time                          : int  2040 2100 1930 2144 2218 2229 2230 2015 2256 2255 ..
##  $ Reported.Date                          : chr  "04/03/2012" "04/02/2012" "04/02/2012" "04/02/2012"
##  $ Reported.Time                          : int  28 2103 2126 2144 2218 2229 2356 2312 2256 2300 ...
##  $ Crime.Subcategory                      : chr  "NARCOTIC" "ROBBERY-COMMERCIAL" "MOTOR VEHICLE THEFT
##  $ Primary.Offense.Description            : chr  "NARC-POSSESS-MARIJU" "ROBBERY-BUSINESS-GUN" "VEH-TH
##  $ Precinct                               : chr  "WEST" "NORTH" "NORTH" "EAST" ...
##  $ Sector                                 : chr  "K" "B" "J" "E" ...
##  $ Beat                                   : chr  "K2" "B2" "J1" "E3" ...
##  $ Neighborhood                           : chr  "PIONEER SQUARE" "BALLARD SOUTH" "BALLARD NORTH" "CA
##  $ Location.1                             : chr  "(47.5998930290529, -122.326813620856)" "(47.6790521
##  $ Latitude                               : num  47.6 47.7 47.7 47.6 47.6 ...
##  $ Longitude                              : num  -122 -122 -122 -122 -122 ...
##  $ census_tract                           : num  5.3e+14 5.3e+14 5.3e+14 5.3e+14 5.3e+14 ...
##  $ census_11                              : num  5.3e+10 5.3e+10 5.3e+10 5.3e+10 5.3e+10 ...
```

```
##  $ GEO.id                                : chr  "1400000US53033009200" NA "1400000US53033004600" "140
##  $ GEO.id2                               : num  5.3e+10 NA 5.3e+10 5.3e+10 NA ...
##  $ GEO.display.label                     : chr  "Census Tract 92, King County, Washington" NA "Census
##  $ total                                 : int  2529 NA 2806 2477 NA NA 5123 NA NA NA ...
##  $ no_schooling                          : int  56 NA 17 100 NA NA 135 NA NA NA ...
##  $ nursery_school                        : int  0 NA 0 0 NA NA 0 NA NA NA ...
##  $ kindergarten                          : int  0 NA 0 44 NA NA 0 NA NA NA ...
##  $ X1st_grade                            : int  0 NA 0 0 NA NA 0 NA NA NA ...
##  $ X2nd_grade                            : int  0 NA 0 0 NA NA 0 NA NA NA ...
##  $ X3rd_grade                            : int  37 NA 0 0 NA NA 79 NA NA NA ...
##  $ X4th_grade                            : int  5 NA 0 0 NA NA 87 NA NA NA ...
##  $ X5th_grade                            : int  17 NA 0 15 NA NA 4 NA NA NA ...
##  $ X6th_grade                            : int  156 NA 0 0 NA NA 43 NA NA NA ...
##  $ X7th_grade                            : int  4 NA 0 0 NA NA 35 NA NA NA ...
##  $ X8th_grade                            : int  100 NA 26 53 NA NA 31 NA NA NA ...
##  $ X9th_grade                            : int  49 NA 0 47 NA NA 88 NA NA NA ...
##  $ X10th_grade                           : int  19 NA 0 0 NA NA 18 NA NA NA ...
##  $ X11th_grade                           : int  14 NA 23 0 NA NA 63 NA NA NA ...
##  $ X12th_grade_no_diploma                : int  63 NA 4 31 NA NA 90 NA NA NA ...
##  $ high_school_diploma                   : int  354 NA 120 104 NA NA 653 NA NA NA ...
##  $ ged_or_alternative_credential         : int  88 NA 4 110 NA NA 165 NA NA NA ...
##  $ some_college_less_than_1_year         : int  134 NA 128 53 NA NA 338 NA NA NA ...
##  $ some_college_1_or_more_years_no_degree: int  503 NA 266 243 NA NA 1289 NA NA NA ...
##  $ associates_degree                     : int  114 NA 106 136 NA NA 501 NA NA NA ...
##  $ bachelors_degree                      : int  536 NA 1175 936 NA NA 1062 NA NA NA ...
##  $ masters_degree                        : int  172 NA 659 365 NA NA 282 NA NA NA ...
##  $ professional_school_degree            : int  71 NA 144 130 NA NA 97 NA NA NA ...
##  $ doctorate_degree                      : int  37 NA 134 110 NA NA 63 NA NA NA ...
```

```r
# Cleaning data: Filter rows with essential educational columns complete
crime_data_clean <- crime_data %>%
  filter(!is.na(total),
         !is.na(no_schooling),
         !is.na(high_school_diploma),
         !is.na(bachelors_degree))

# Fill missing values in 'Precinct' and 'Sector' with the mode
crime_data_clean$Precinct[is.na(crime_data_clean$Precinct)] <-
  as.character(names(sort(table(crime_data_clean$Precinct), decreasing=TRUE))[1])

crime_data_clean$Sector[is.na(crime_data_clean$Sector)] <-
  as.character(names(sort(table(crime_data_clean$Sector), decreasing=TRUE))[1])

# Convert 'Occurred Date' to Date type for time-based analysis
crime_data_clean$Occurred.Date <- ymd(crime_data_clean$Occurred.Date)
```

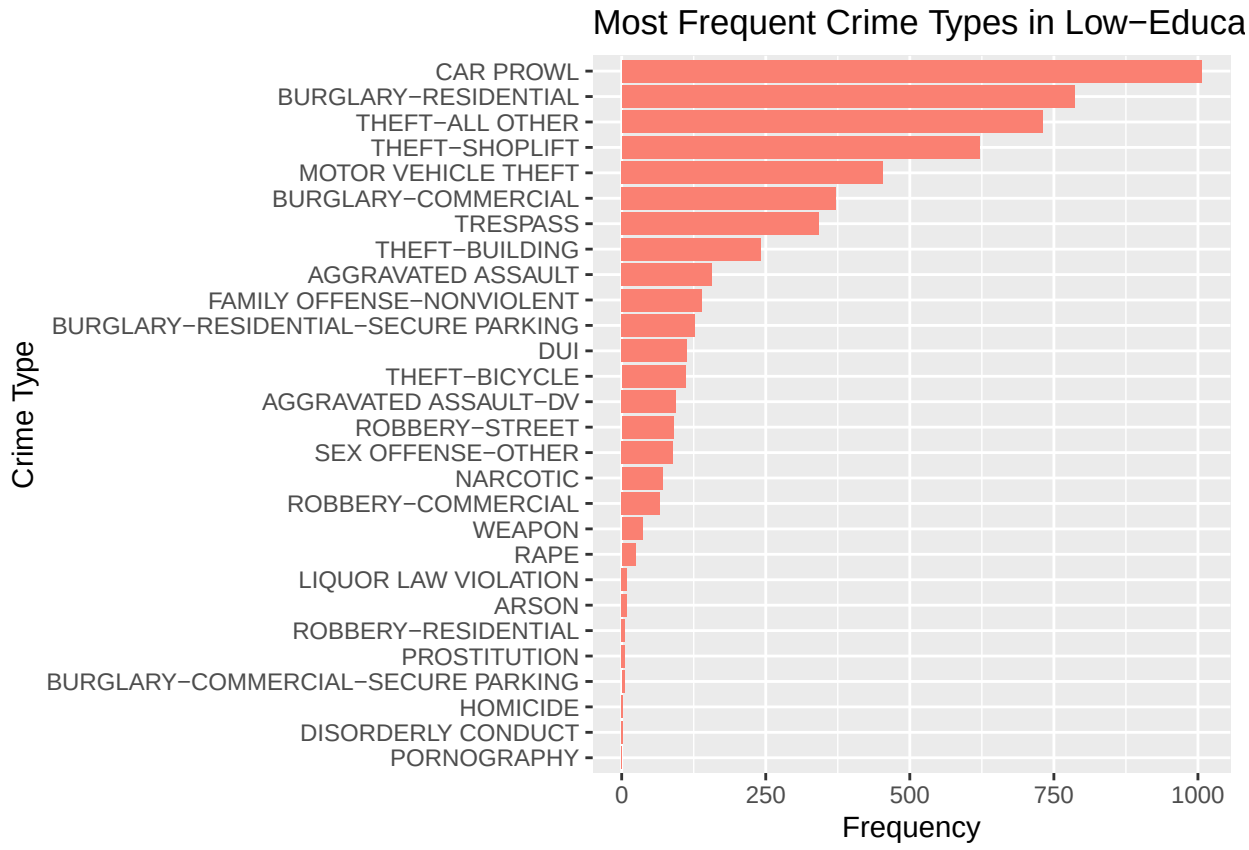
(c) (10pt) Data Analysis Use the dataset to provide empirical evidence that addressed your question from part (a). Discuss your results. Provide at least one visualization to support your narrative.

Solution:

Insert Response

```r
# Question 2: Types of Crimes in Areas with Lower Educational Attainment
# Define threshold for lower educational
#attainment (e.g., below 50% high school diploma rate)
low_education_areas <- crime_data_clean %>%
  filter(high_school_diploma < 50) %>%
  count(Crime.Subcategory) %>%
  arrange(desc(n))

# Visualization for Question 2: Crime Type Frequency in Low-Education Areas
ggplot(low_education_areas, aes(x = reorder(Crime.Subcategory, n), y = n)) +
  geom_bar(stat = "identity", fill = "salmon") +
  coord_flip() +
  labs(
    title = "Most Frequent Crime Types in Low-Education Areas",
    x = "Crime Type",
    y = "Frequency"
  )
```



(d) (5pt) Reflect and Question Comment the questions (and answers) in this analysis. Were you able to answer all of these questions? Are all questions well defined? Is the data good enough to answer all these?

Solution:

Insert Response

- **Question Definition:** This question examines which types of crimes are more frequent in areas with

lower educational attainment (defined here as less than 50% high school diploma attainment). It's specific and measurable, as it categorizes crimes in well-defined areas with low education levels.

- **Data Quality:** The data's crime category and education level breakdown enable this analysis, though the threshold chosen (50%) is arbitrary and could impact results. Defining the threshold with statistical rigor or exploring a range of thresholds could improve robustness.
- **Effectiveness:** The analysis answers the question by generating a frequency chart of crime types within low-education areas, providing a straightforward view of which crimes are most prevalent. This gives valuable insight but lacks depth on underlying factors that might drive crime type in these areas.