Statistical Computing with R - Assignment 2

Colin Yip, Student No: 3953629

Exercise 1

Formatting and output config.

```
library(knitr)
library(formatR)
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 80))
```

Data Preparation

Q1

```
# Data subsetting
listing_data <- read.csv("Airbnb_Ragusa.csv")
```

Q2

```
variable_list <- c("neighbourhood_cleansed", "latitude", "longitude", "price")
subsetted_listing_data <- listing_data[variable_list]
```

Q3

```
# Q3 - price cleanup
price_cleanup_func <- function(x) {
  stripped_x <- gsub("\\$|\\,", "", x)
  as.numeric(stripped_x)
}
# Rewrite price with stripped/converted values
subsetted_listing_data["price"] <- sapply(subsetted_listing_data["price"], price_cleanup_func)
```

Prices per Neighbourhood

Q4

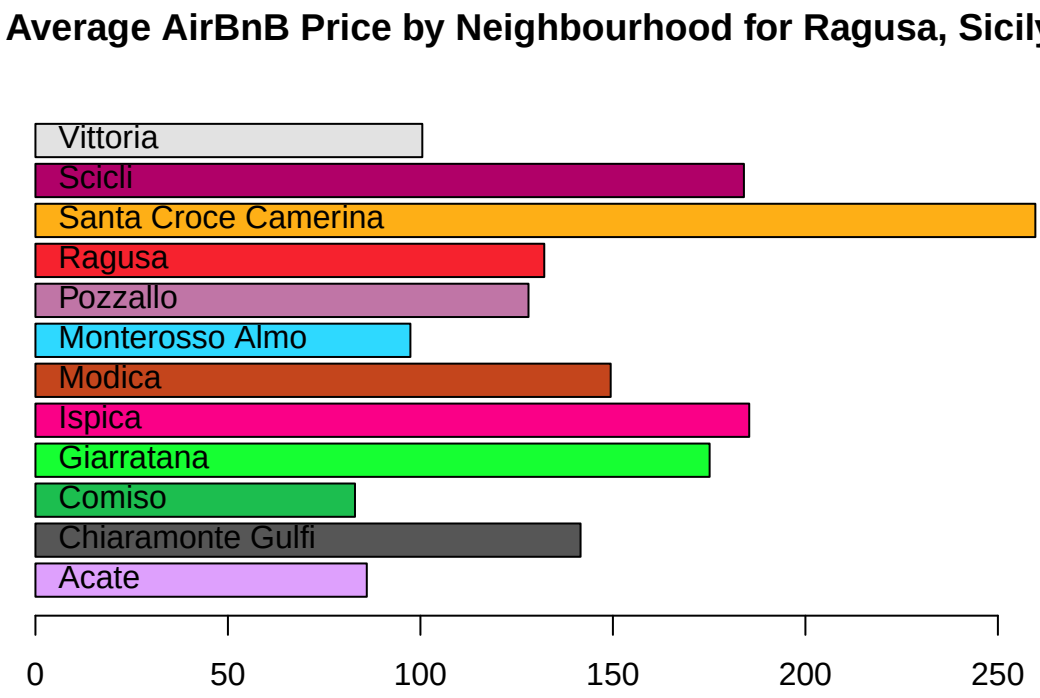
```
# Get mean by neighbourhood
neighbourhood_groups <- split(subsetted_listing_data$price, subsetted_listing_data$neighbourhood_cleansed)
neighbourhood_avgs <- sapply(neighbourhood_groups, mean)
neighbourhood_labels <- names(neighbourhood_avgs)
# Convert calculated averages to numerics
neighbourhood_avg_values <- as.numeric(neighbourhood_avgs)
# Remake dataframe with neighbourhood averages
neighbourhood_avgs_df <- data.frame(
  neighborhood = neighbourhood_labels,
  average = neighbourhood_avg_values
)
knitr::kable(neighbourhood_avgs_df,
             caption = "Airbnb Neighbourhood Average Price")
```

Table 1: Airbnb Neighbourhood Average Price

neighborhood	average
Acate	86.07143
Chiaramonte Gulfi	141.60000
Comiso	83.00000
Giarratana	175.12500
Ispica	185.40506
Modica	149.42586
Monterosso Almo	97.40000
Pozzallo	128.07692
Ragusa	132.16701
Santa Croce Camerina	259.73265
Scicli	184.02929
Vittoria	100.49490

Q5

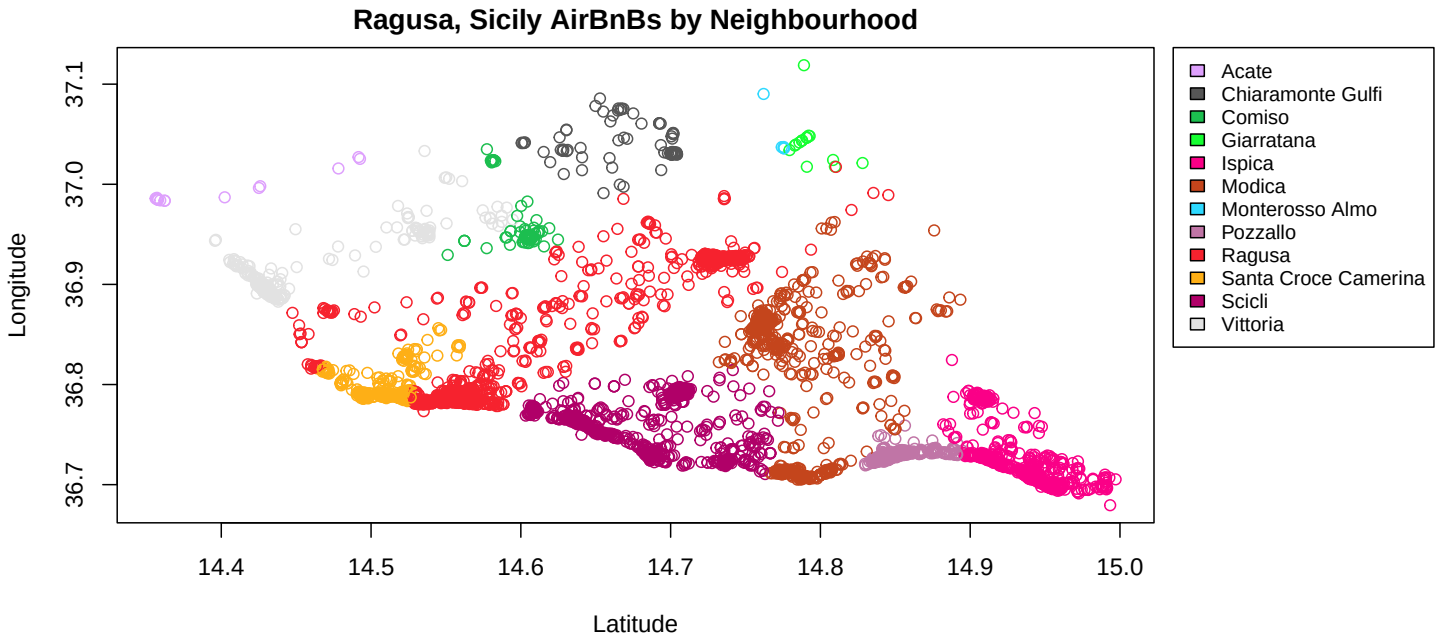
```
# Pull n unique colours from Alphabet palette
palette("Alphabet")
coloursvector <- sample(palette(), 12)
bar_pos <- barplot(neighbourhood_avgs_df$average,
  horiz = T,
  col = coloursvector,
  main = "Average AirBnB Price by Neighbourhood for Ragusa, Sicily"
)
text(1, bar_pos, neighbourhood_avgs_df$neighborhood, pos = 4)
```



AirBnB Locations

Q6 & Q7

```
subsetted_listing_data$col <- coloursvector[as.factor(subsetted_listing_data$neighbourhood)]
neighbourhood_col_groups <- split(subsetted_listing_data$col, subsetted_listing_data$neighbourhood_cleansed)
neighbourhood_col_map <- sapply(neighbourhood_col_groups, unique)
# Re-adjust margins to account for legends and turn off clipping to plot area
par(mar = c(4, 4, 2, 10), xpd = T)
plot(
  x = subsetted_listing_data$longitude,
  y = subsetted_listing_data$latitude,
  col = subsetted_listing_data$col,
  type = "p",
  xlab = "Latitude",
  ylab = "Longitude",
  main = "Ragusa, Sicily AirBnBs by Neighbourhood"
)
legend("topright",
  legend = names(neighbourhood_col_map),
  fill = neighbourhood_col_map,
  inset = c(-0.27, 0),
  cex = 0.8
)
```

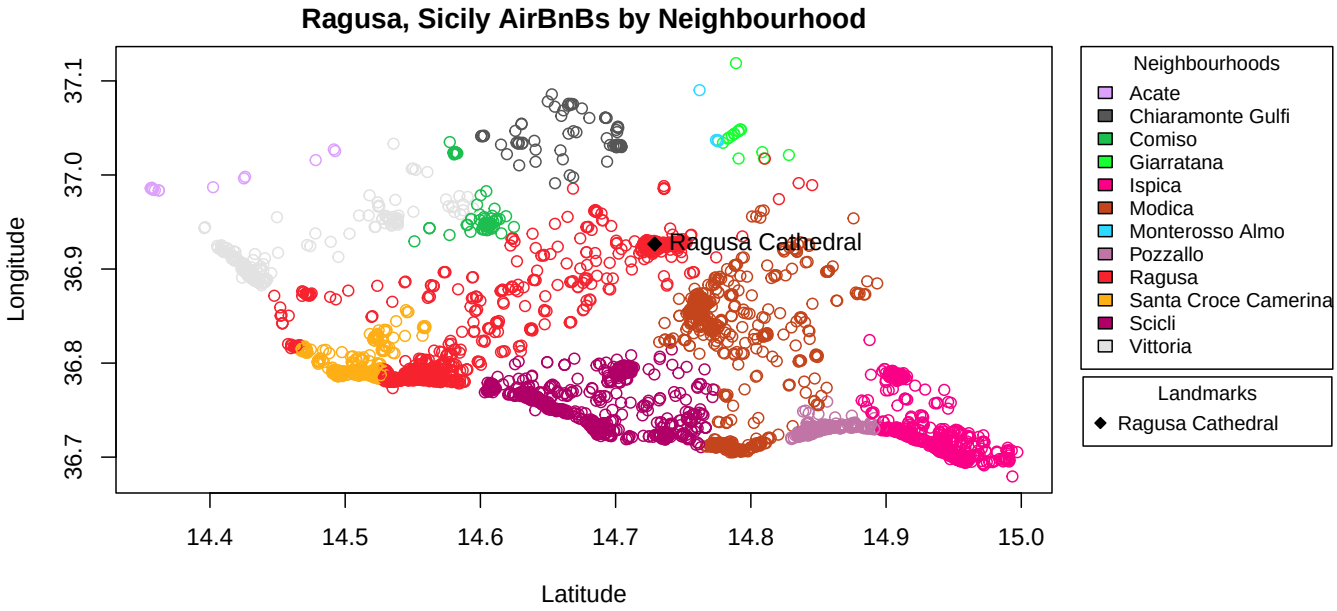


Q9

Yes, the neighbourhoods line up to the same relative positions.

Q10 & 11 & 12

```
cathedral_y <- 36.92655374273344
cathedral_x <- 14.728909384658452
par(mar = c(5, 5, 2, 12.5), xpd = T)
plot(
  x = subsetted_listing_data$longitude,
  y = subsetted_listing_data$latitude,
  col = subsetted_listing_data$col,
  type = "p",
  xlab = "Latitude",
  ylab = "Longitude",
  main = "Ragusa, Sicily AirBnBs by Neighbourhood"
)
points(
  x = cathedral_x,
  y = cathedral_y,
  type = "p",
  col = "black",
  bg = "black",
  pch = 23
)
text(
  x = cathedral_x,
  y = cathedral_y,
  labels = "Ragusa Cathedral",
  offset = c(0.5, 0),
  pos = 4
)
legend("topright",
  legend = names(neighbourhood_col_map),
  fill = neighbourhood_col_map,
  inset = c(-0.3, 0),
  cex = 0.8,
  text.width = 0.144,
  title = "Neighbourhoods"
)
legend("topright",
  legend = c("Ragusa Cathedral"),
  pch = 23,
  pt.bg = "black",
  inset = c(-0.299, 0.74),
  text.width = 0.152,
  cex = 0.8,
  title = "Landmarks"
)
```




```
dev.copy(jpeg,
  width = 775,
  height = 400,
  "ragusa_map.jpg"
)
```

```
## quartz_off_screen
##                 3
```

Exercise 2

Task 1

```
# Download data from git to current working directory
data_url <- "https://raw.githubusercontent.com/1223423/toy_data/main/AAPL_complete.csv"
curr_dir <- getwd()
file_name <- "aapl.csv"
catted_file_path <- paste(curr_dir, file_name, sep = "/")
curl::curl_download(data_url, destfile = catted_file_path)

appl_df <- read.csv(catted_file_path)
appl_df$Date <- as.Date(appl_df$Date, format = "%Y-%m-%d")
```

Task 2

```
# Function to process raw data file with explicit Date column
retrieve_historicals <- function(file_name = "appl.csv") {
  catted_file_path <- paste(curr_dir, file_name, sep = "/")
  reference_df <- read.csv(catted_file_path)
  reference_df$Date <- as.Date(reference_df$Date, format = "%Y-%m-%d")
  reference_df
}

simulate_investment <- function(start_date, end_date, investment_USD) {
  # Evaluate if the investment value is positive
  if (investment_USD <= 0) {
    stop("Value of investment_USD must be > 0")
  }

  # Check for matches against validated date regex
  # No matches indicate invalid date
  regex_pattern <- "\\d{4}\\-(0?[1-9]|1[012])\\-(0?[1-9]|[12][0-9]|3[01])$"
  start_date_grep_val <- grep(regex_pattern, start_date)
  end_date_grep_val <- grep(regex_pattern, end_date)
  if (length(start_date_grep_val) == 0 | length(end_date_grep_val) == 0 ) {
    stop("Invalid date string passed for start_date or end_date")
  }

  # Check if start date is earlier than end date
  start_date_dt <- as.Date(start_date, format = "%Y-%m-%d")
  end_date_dt <- as.Date(end_date, format = "%Y-%m-%d")
  if ((end_date_dt - start_date_dt) <= 0) {
    stop("Start date cannot be before or equal to end date")
  }

  # Check if date falls within data range
  reference_df <- retrieve_historicals()
  if (!(start_date_dt %in% reference_df$Date & end_date_dt %in% reference_df$Date)) {
    stop("start_date or end_date is not within the dates provided by reference data")
  }

  # Subset the large dataframe to required time horizon
  time_horizon_bool <- (appl_df$Date >= start_date_dt) & (appl_df$Date <= end_date_dt)
  time_horizon_limited_df <- reference_df[time_horizon_bool, ]

  # Normalize v_i, i>0, to v_0, and scale to value of s
  v_0 <- time_horizon_limited_df[1, "Open"]
```

```
  s <- floor(investment_USD / v_0)
  remainder_investment <- investment_USD %% v_0
  scaled_daily_open <- time_horizon_limited_df["Open"] / v_0
  daily_value <- scaled_daily_open * v_0 * s + remainder_investment
  daily_profit <- daily_value - investment_USD
  returns_df <- data.frame(
    time_horizon_limited_df$Date,
    daily_value,
    daily_profit
  )
  colnames(returns_df) <- c("Date", "Value", "Profit")

  return(returns_df)
}
```

Task 3

```
knitr::kable(simulate_investment("2011-10-01", "2011-10-11", 10000),
             caption = "Simulated returns between 2011-10-01 and 2011-10-11")
```

Table 2: Simulated returns between 2011-10-01 and 2011-10-11

	Date	Value	Profit
580	2011-10-01	10000.000	0.000
581	2011-10-02	10000.000	0.000
582	2011-10-03	9826.444	-173.556
583	2011-10-04	9677.296	-322.704
584	2011-10-05	9504.748	-495.252
585	2011-10-06	9645.400	-354.600
586	2011-10-07	9708.400	-291.600
587	2011-10-08	9708.400	-291.600
588	2011-10-09	9708.400	-291.600
589	2011-10-10	9793.522	-206.478
590	2011-10-11	10140.148	140.148

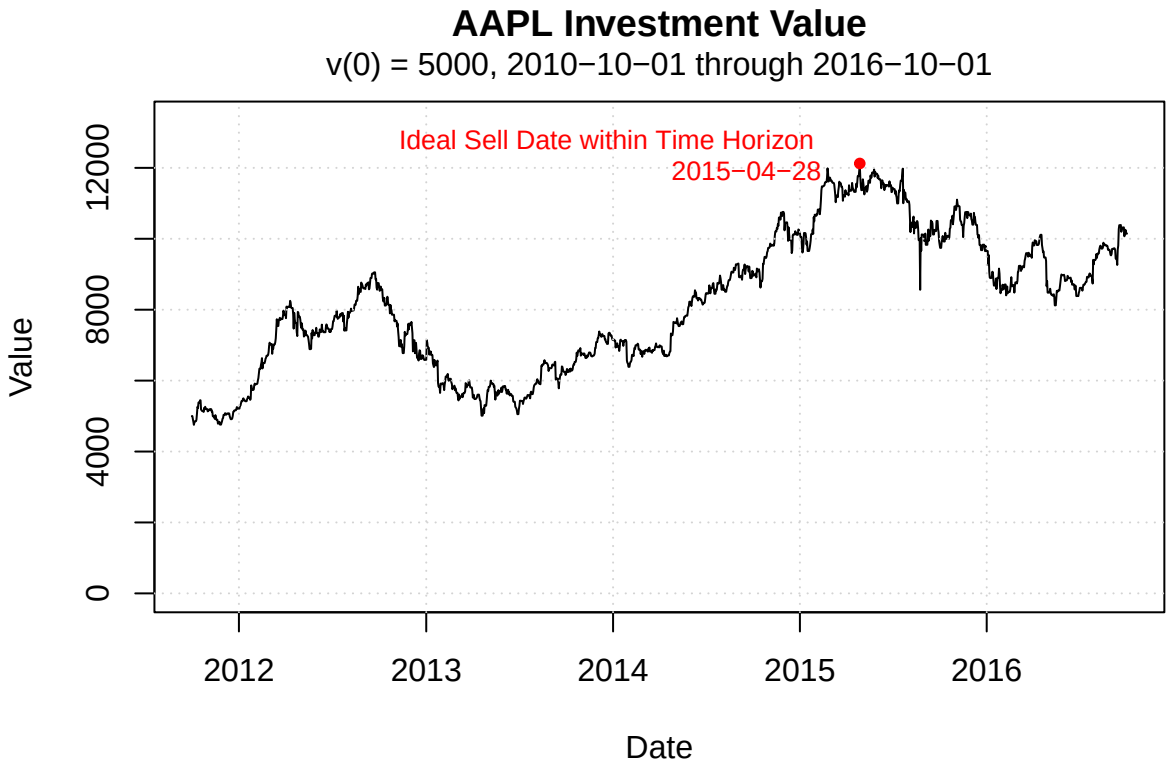
Task 4

```
simulated_data <- simulate_investment("2011-10-01", "2016-10-01", 5000)
plot(simulated_data$Date,
  simulated_data$Value,
  type = "l",
  xlab = "Date",
  ylab = "Value",
  xlim = c(min(simulated_data$Date), max(simulated_data$Date)),
  ylim = c(0, max(simulated_data$Value) * 1.1),
  main = "AAPL Investment Value"
)
mtext(
  side = 3,
  cex = 1,
  line = 0.5,
  "v(0) = 5000, 2010-10-01 through 2016-10-01"
)
grid(
  ny = NULL,
  nx = NA
```

```
)
x_tick_dt <- sapply(
  c(
    "2012-01-01", "2013-01-01", "2014-01-01",
    "2015-01-01", "2016-01-01"
  ),
  as.Date
)
abline(v = x_tick_dt, lty = 3, col = "lightgray")

# Find maximum within time horizon and plot point
max_idx <- which.max(simulated_data$Value)
max_value <- max(simulated_data$Value)
max_dt <- simulated_data[max_idx, "Date"]

points(max_dt, max_value, col = "red", pch = 20)
text(max_dt, max_value,
  paste("Ideal Sell Date within Time Horizon \n ", max_dt),
  col = "red",
  cex = 0.8,
  pos = 2,
  offset = 1
)
```



Task 6

```
# Slice to maximum value row index
simulated_data[max_idx, ]
```

```
##            Date   Value   Profit
## 1885 2015-04-28 12123.7 7123.698
```