DSCI 6607– Fall 2024 Assignment 4

2024-10-26

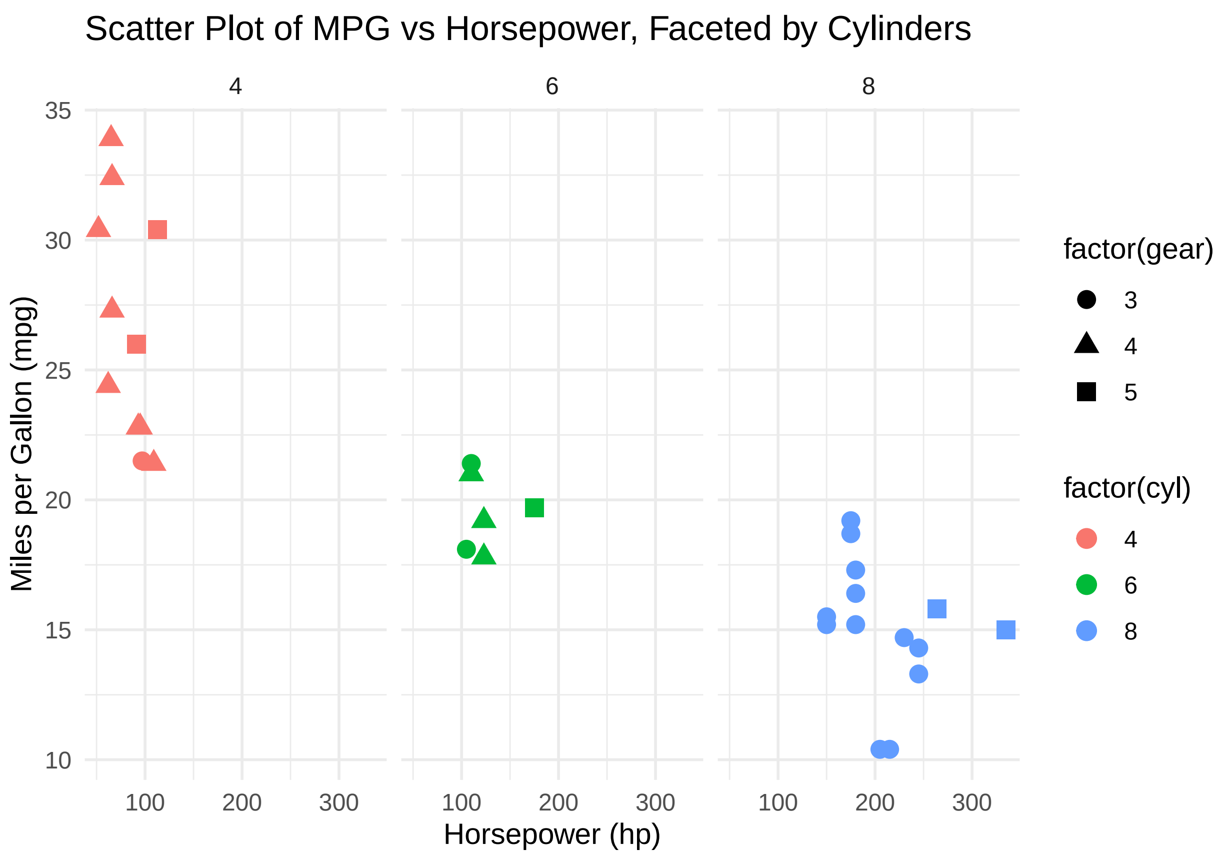
```r
# Abdallah Chidjou
# Citation: (source of help: Lecture note, googling in general, stackoverflow, and chatgpt)
```

Question 1

```r
# Load mtcars (this is already available in R)
data(mtcars)
# View the first few rows of the dataset
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
# View the structure of the dataset to understand the variable types
#str(mtcars)

# Get a summary of the dataset
#summary(mtcars)

# Load the necessary library
library(ggplot2)

# Create the scatter plot
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(color = factor(cyl), shape = factor(gear)), size = 3) +
  facet_wrap(~ cyl) +
  theme_minimal() +
  labs(title = "Scatter Plot of MPG vs Horsepower, Faceted by Cylinders",
       x = "Horsepower (hp)",
       y = "Miles per Gallon (mpg)")
```




```
# a.
# Each panel represents a subset of cars from the dataset
# with the same number of cylinders (cyl = 4, 6, or 8).

# b.
# geom_point(aes(color = factor(cyl), shape = factor(gear)), size = 3)
# done in the plot

# c.
# The variable used for faceting is cyl.

# d.
# Cars with more cylinders (e.g., 8 cylinders) tend to have lower mpg and higher hp,
# indicating they are powerful but less efficient.
# Cars with fewer cylinders (e.g., 4 cylinders) tend to have higher mpg and lower hp,
# indicating they are more fuel-efficient but less powerful.
```

Question 2

```
# a1: Parsing and outputting the date
a1 <- "12/30/14"
parsed_a1 <- strptime(a1, format = "%m/%d/%y")
formatted_a1 <- format(parsed_a1, format = "%b %d, %Y")
print(formatted_a1)
```

```
## [1] "Dec 30, 2014"
```

```r
# a2: Parsing and outputting the date
a2 <- "07-Jan-2017"
parsed_a2 <- strptime(a2, format = "%d-%b-%Y")
formatted_a2 <- format(parsed_a2, format = "%d-%m-%Y")
print(formatted_a2)
```

```
## [1] "07-01-2017"
```

```r
# Create the original vector of date-time strings
convert_date_time <- function(date_vector) {
  # Use sapply to iterate over the vector and parse/format each element
  formatted_dates <- sapply(date_vector, function(x) {
    # Parse the original string to a date-time object
    parsed_date <- strptime(x, format = "%B %d (%Y) - %I:%M%p")
    # Format the parsed date into the desired format
    formatted_date <- format(parsed_date, format = "%m/%d/%Y - %I:%M%p")
    return(formatted_date)
  })
  # Return the formatted dates
  return(formatted_dates)
}
# a3: Parsing and outputting the date
a3 <- c("August 19 (2015) - 3:04PM", "July 1 (2015) - 4:04PM")
formatted_dates <- convert_date_time(a3)
# Print the formatted dates
print(formatted_dates)
```

```
## August 19 (2015) - 3:04PM    July 1 (2015) - 4:04PM
##    "08/19/2015 - 03:04PM"    "07/01/2015 - 04:04PM"
```

```r
# a4: Parsing and outputting the date
a4 <- "January 1, 2010"
parsed_a4 <- strptime(a4, format = "%B %d, %Y")
print(parsed_a4)
```

```
## [1] "2010-01-01 NST"
```

```r
# a5: Parsing and outputting the date
a5 <- "2015-Mar-07"
parsed_a5 <- strptime(a5, format = "%Y-%b-%d")
print(parsed_a5)
```

```
## [1] "2015-03-07 NST"
```

Question 3

```r
# Install and load the required libraries
# install.packages(c("dplyr", "ggrepel"))
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(ggrepel)

# Load the dataset (assuming you have the CSV file named "largest_cities.csv")
cities <- read.csv("largest_cities.csv")


# a. Create a new column `city_density`
cities <- cities %>%
  mutate(city_density = city_pop / city_area)
head(cities)
```

```
##          name country             city_definition population city_pop city_area
## 1       Tokyo   Japan      Metropolis prefecture     37.400   13.515      2191
## 2       Delhi   India National capital territory     28.514   16.753      1484
## 3    Shanghai   China                Municipality     25.582   24.183      6341
## 4   São Paulo  Brazil                Municipality     21.650   12.252      1521
## 5 Mexico City  Mexico                  City-state     21.581    8.919      1485
## 6       Cairo   Egypt          Urban governorate     20.076    9.500      3085
##   metro_pop metro_area urban_pop urban_area        wiki country_code2
## 1    37.274      13452    38.505       8223       Tokyo            JP
## 2    29.000       3483    28.125       2240       Delhi            IN
## 3        NA         NA    22.125       4015    Shanghai            CN
## 4    21.735       7947    20.935       3043   Sao_Paulo            BR
## 5    20.893       7854    20.395        237 Mexico_City            MX
## 6        NA         NA    16.925       1917       Cairo            EG
##   country_code3          country_name_official     continent       lon
## 1           JPN                          Japan          Asia 139.69222
## 2           IND             India, Republic of          Asia  77.23000
## 3           CHN    China, People's Republic of          Asia 121.47472
## 4           BRA Brazil, Federative Republic of South America -46.63333
## 5           MEX  Mexico, United Mexican States North America -99.13333
## 6           EGY        Egypt, Arab Republic of        Africa  31.22889
##        lat koppen_code koppen_main             city num cost_of_living
## 1 35.68972         Cfa   Temperate     Tokyo, Japan  18          86.87
## 2 28.61000         BSh         Dry     Delhi, India 405          28.18
```

```
## 3  31.22861         Cfa   Temperate     Shanghai, China 235          50.07
## 4 -23.55000         Cfb   Temperate   Sao Paulo, Brazil 257          45.52
## 5  19.43333         Cwb   Temperate Mexico City, Mexico 317          38.55
## 6  30.05806         BWh         Dry         Cairo, Egypt 393          30.94
##   cost_rent cost_groceries cost_restaurant local_pp city_density
## 1     38.00          83.42           56.70    89.70  0.006168416
## 2      8.18          26.15           24.76    54.69  0.011289084
## 3     35.67          52.50           36.48    54.40  0.003813752
## 4     16.28          33.29           39.70    31.97  0.008055227
## 5     20.43          33.75           33.11    42.91  0.006006061
## 6      6.48          26.41           25.43    24.53  0.003079417
```

```r
# b. Select name, city_pop, city_area, and city_density columns
selected_data <- cities %>%
  select(name, city_pop, city_area, city_density)

# Display the selected data
# print(selected_data)

# c. Modify city_density by multiplying by 1000
cities <- cities %>%
  mutate(city_density = city_density * 1000)

# Update the `selected_data` after modifying city_density
selected_data <- cities %>%
  select(name, city_pop, city_area, city_density)

# Display the updated selected data
print(selected_data)
```

```
##               name city_pop city_area city_density
## 1            Tokyo   13.515   2191.00   6.16841625
## 2            Delhi   16.753   1484.00  11.28908356
## 3         Shanghai   24.183   6341.00   3.81375177
## 4        São Paulo   12.252   1521.00   8.05522682
## 5      Mexico City    8.919   1485.00   6.00606061
## 6            Cairo    9.500   3085.00   3.07941653
## 7           Mumbai   12.478    603.00  20.69320066
## 8          Beijing   21.707  16411.00   1.32271038
## 9            Dhaka   14.399    338.00  42.60059172
## 10           Osaka    2.725    225.00  12.11111111
## 11   New York City    8.399    786.00  10.68575064
## 12         Karachi   14.910    378.00  39.44444444
## 13    Buenos Aires    3.054    203.00  15.04433498
## 14       Chongqing   30.166  82403.00   0.36607890
## 15        Istanbul   15.029   5196.00   2.89241724
## 16         Kolkata    4.497    205.00  21.93658537
## 17    Metro_Manila    1.780     43.00  41.39534884
## 18           Lagos       NA        NA           NA
## 19  Rio de Janeiro    6.520   1221.00   5.33988534
## 20         Tianjin   15.569   1192.00  13.06124161
## 21        Kinshasa   11.462   9965.00   1.15022579
## 22       Guangzhou   14.498   7434.00   1.95022868
## 23     Los Angeles    3.990   1214.00   3.28665568
```

```
## 24            Moscow   13.200   2511.00  5.25686977
## 25          Shenzhen   12.528    205.00 61.11219512
## 26            Lahore   11.126   1772.00  6.27878104
## 27         Bangalore    8.444    709.00 11.90973202
## 28             Paris    2.148    105.00 20.45714286
## 29            Bogotá    7.963   1587.00  5.01764335
## 30           Jakarta   10.154    664.00 15.29216867
## 31           Chennai    6.727    426.00 15.79107981
## 32              Lima    8.894   2672.00  3.32859281
## 33           Bangkok    5.782   1569.00  3.68514978
## 34             Seoul    9.806    605.00 16.20826446
## 35            Nagoya    2.320    326.00  7.11656442
## 36         Hyderabad    6.993    650.00 10.75846154
## 37            London    8.825   1572.00  5.61386768
## 38            Tehran    9.033    751.00 12.02796272
## 39           Chicago    2.706    589.00  4.59422750
## 40           Chengdu   16.045  14378.00  1.11594102
## 41           Nanjing    7.260   6582.00  1.10300820
## 42             Wuhan   10.893   8494.00  1.28243466
## 43 Ho Chi Minh City    7.431   2061.00  3.60553130
## 44            Luanda    2.166    116.00 18.67241379
## 45         Ahmedabad    5.571    464.00 12.00646552
## 46      Kuala Lumpur    1.768    243.00  7.27572016
## 47             Xi'an    8.989  10135.00  0.88692649
## 48         Hong Kong    7.299   1104.00  6.61141304
## 49          Dongguan    8.342   2465.00  3.38417850
## 50          Hangzhou    9.468  16596.00  0.57049892
## 51            Foshan    7.197   3848.00  1.87032225
## 52          Shenyang    8.294   1298.00  6.38983051
## 53            Riyadh    6.694   1913.00  3.49921589
## 54           Baghdad    8.127   5200.00  1.56288462
## 55          Santiago    0.236     22.00 10.72727273
## 56             Surat    4.467    327.00 13.66055046
## 57            Madrid    3.266    606.00  5.38943894
## 58            Suzhou   10.722   8488.42  1.26313260
## 59              Pune    3.124    276.00 11.31884058
## 60            Harbin   10.636  53068.00  0.20042210
## 61           Houston    2.326   1553.00  1.49774630
## 62            Dallas    1.345    882.00  1.52494331
## 63           Toronto    2.732    630.00  4.33650794
## 64     Dar es Salaam    4.365   1393.00  3.13352477
## 65             Miami    0.471     92.90  5.06996771
## 66    Belo Horizonte    2.503    330.90  7.56421880
## 67         Singapore    5.639    725.70  7.77042855
## 68      Philadelphia    1.526    369.59  4.12889959
## 69           Atlanta    0.420    354.22  1.18570380
## 70           Fukuoka    1.589    343.39  4.62739160
## 71          Khartoum    0.640  22142.00  0.02890434
## 72         Barcelona    1.620    101.40 15.97633136
## 73      Johannesburg       NA        NA          NA
## 74 Saint Petersburg       NA        NA          NA
## 75           Qingdao       NA        NA          NA
## 76            Dalian       NA        NA          NA
## 77 Washington, D.C.    0.702    177.00  3.96610169
```

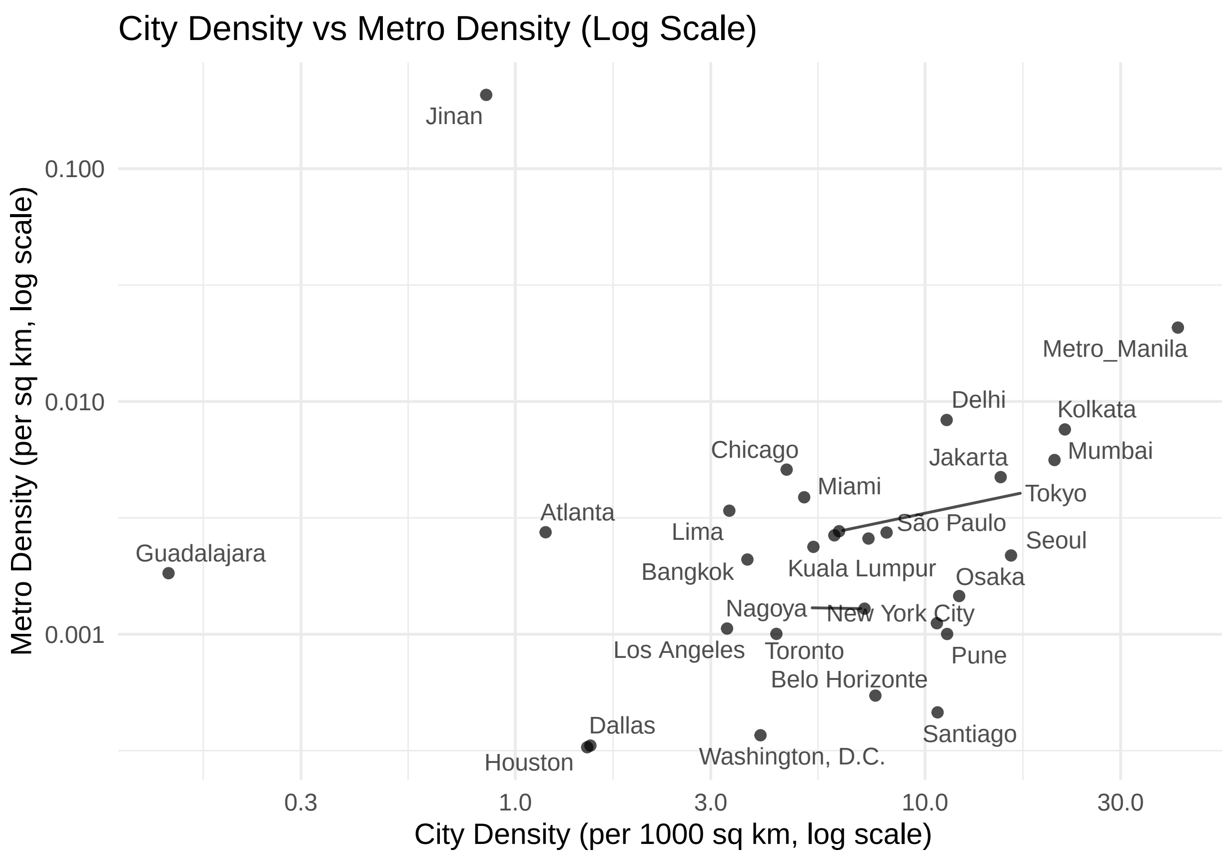
```
## 78          Yangon       NA        NA          NA
## 79      Alexandria       NA        NA          NA
## 80           Jinan    8.700  10244.00  0.84927763
## 81     Guadalajara    1.460  10244.00  0.14252245
```

```r
# d. Calculate the average city density by continent
average_density <- cities %>%
  # Group by continent
  group_by(continent) %>%
  # Calculate average, handling NA values
  summarise(average_city_density = mean(city_density, na.rm = TRUE))

# Print the result
print(average_density)
```

```
## # A tibble: 5 x 2
##   continent     average_city_density
##   <chr>                        <dbl>
## 1 Africa                        5.21
## 2 Asia                         10.6
## 3 Europe                        9.26
## 4 North America                 3.87
## 5 South America                 7.87
```

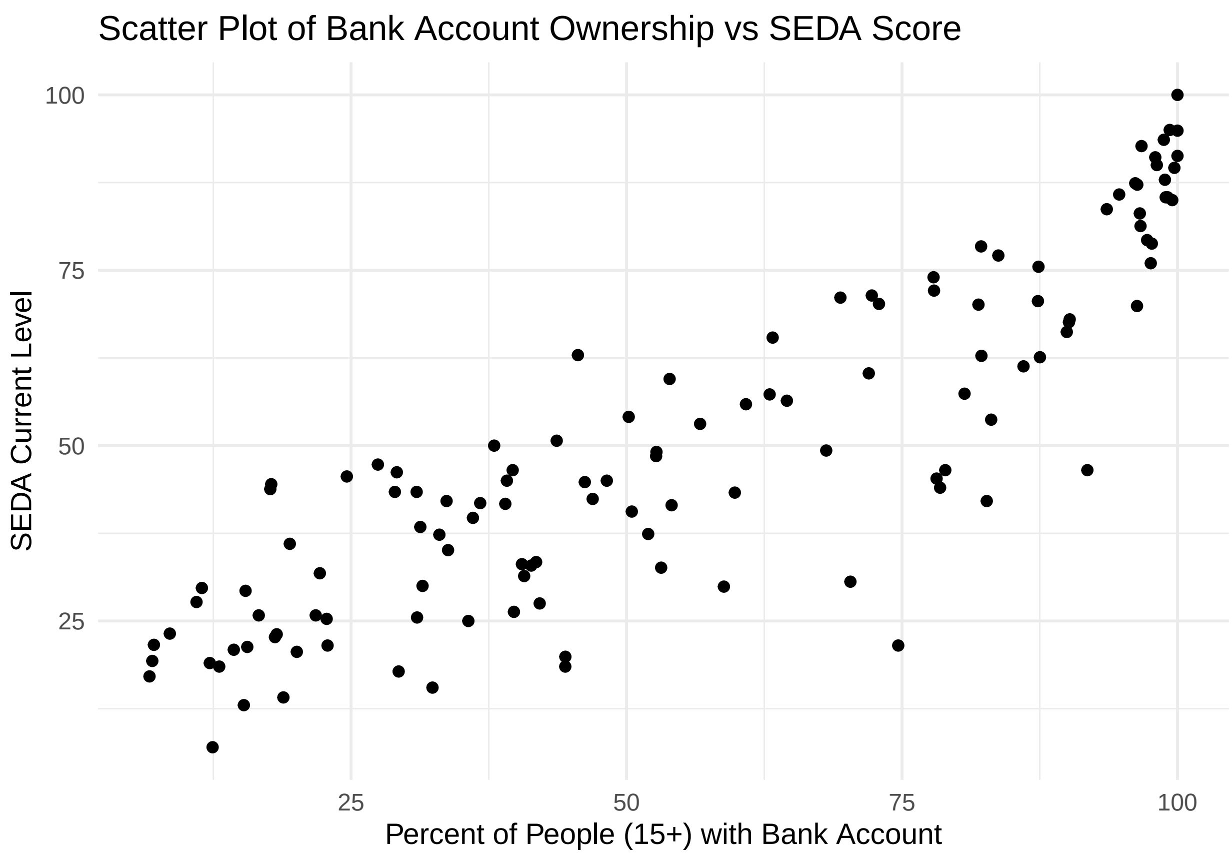
```r
# e. Calculate metro_density if not already available
cities <- cities %>%
  mutate(metro_density = metro_pop / metro_area)

# Filter out rows with missing or zero values for city_density or metro_density to avoid issues with lo
filtered_cities <- cities %>%
  filter(!is.na(city_density), !is.na(metro_density), city_density > 0, metro_density > 0)

# Create the plot
ggplot(filtered_cities, aes(x = city_density, y = metro_density, label = name)) +
  geom_point(alpha = 0.7) +
  geom_text_repel(size = 3, alpha = 0.7) +
  scale_x_log10() +
  scale_y_log10() +
  theme_minimal() +
  labs(title = "City Density vs Metro Density (Log Scale)",
       x = "City Density (per 1000 sq km, log scale)",
       y = "Metro Density (per sq km, log scale)")
```

```
## Warning: ggrepel: 2 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

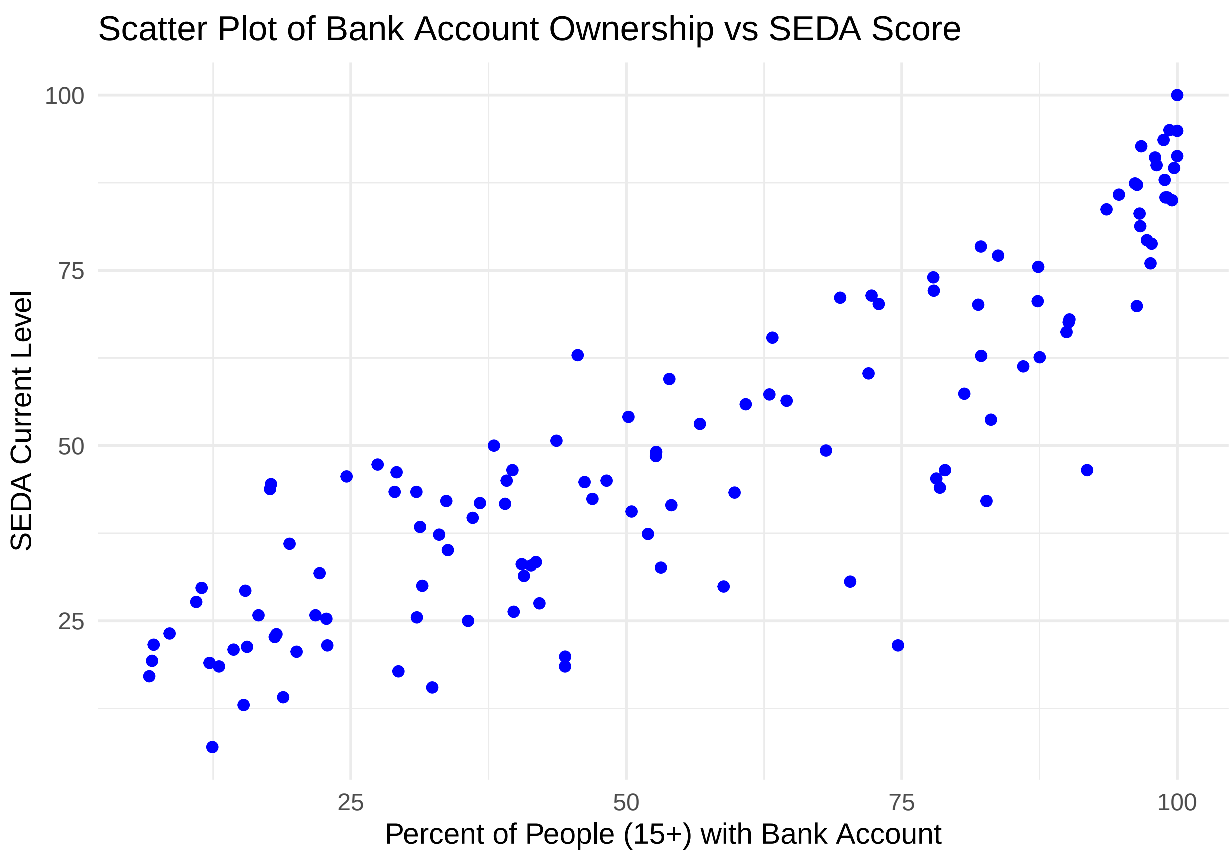


Question 4

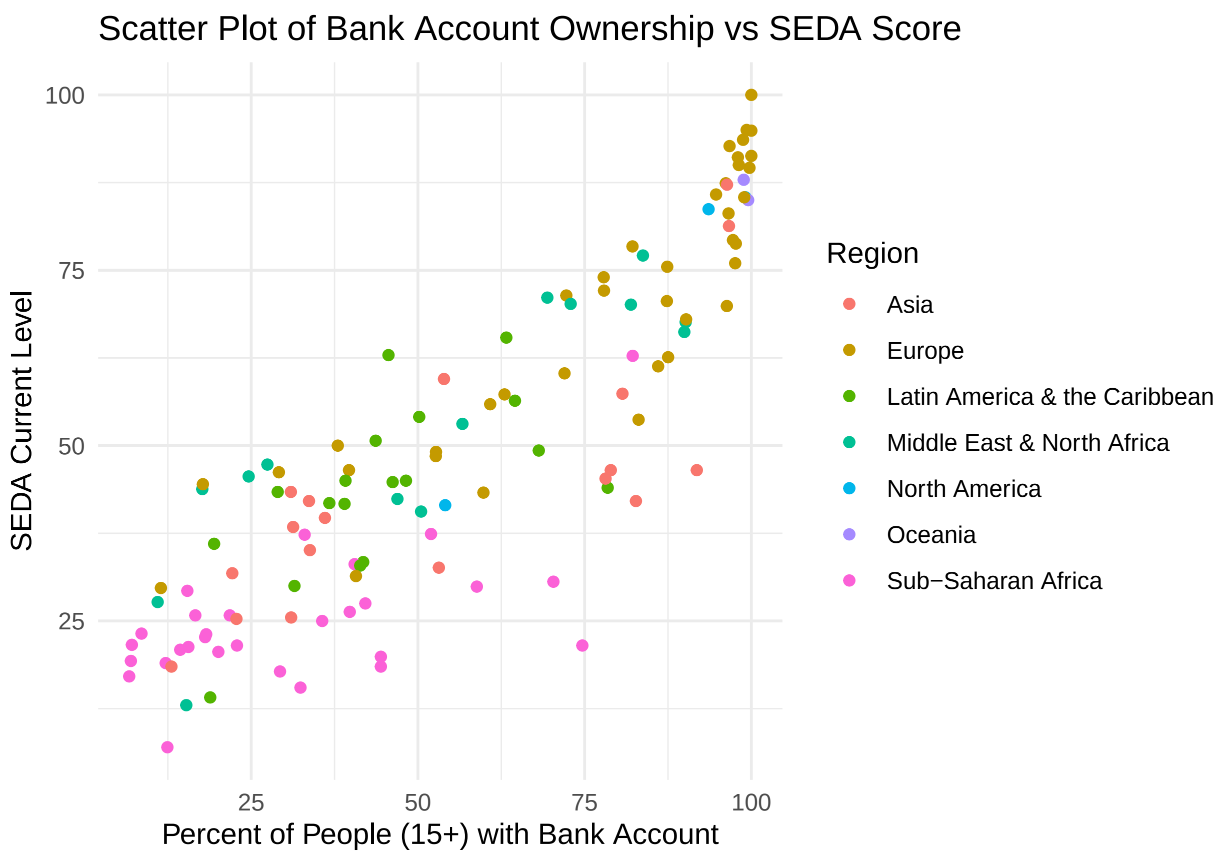
```r
# Load the dataset
economist_data <- read.csv("EconomistData.csv")

# a. Create a scatter plot with the appropriate columns
ggplot(economist_data, aes(x = Percent.of.15plus.with.bank.account, y = SEDA.Current.level)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Scatter Plot of Bank Account Ownership vs SEDA Score",
       x = "Percent of People (15+) with Bank Account",
       y = "SEDA Current Level")
```




```r
# b. Color all points blue.
ggplot(economist_data, aes(x = Percent.of.15plus.with.bank.account, y = SEDA.Current.level)) +
  geom_point(color = "blue") +
  theme_minimal() +
  labs(title = "Scatter Plot of Bank Account Ownership vs SEDA Score",
       x = "Percent of People (15+) with Bank Account",
       y = "SEDA Current Level")
```

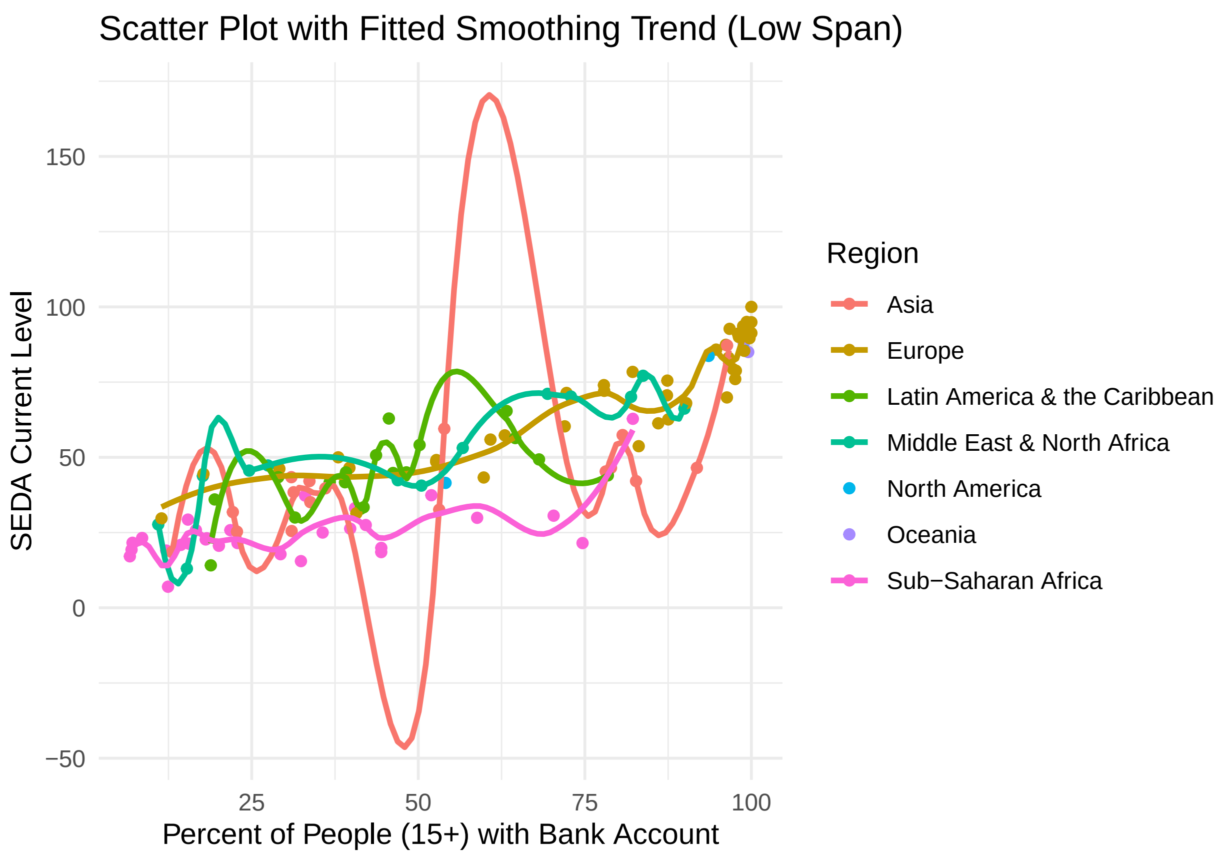



```r
# c. Color points according to the Region variable.
ggplot(economist_data, aes(x = Percent.of.15plus.with.bank.account, y = SEDA.Current.level, color = Regi
  geom_point() +
  theme_minimal() +
  labs(title = "Scatter Plot of Bank Account Ownership vs SEDA Score",
       x = "Percent of People (15+) with Bank Account",
       y = "SEDA Current Level")
```



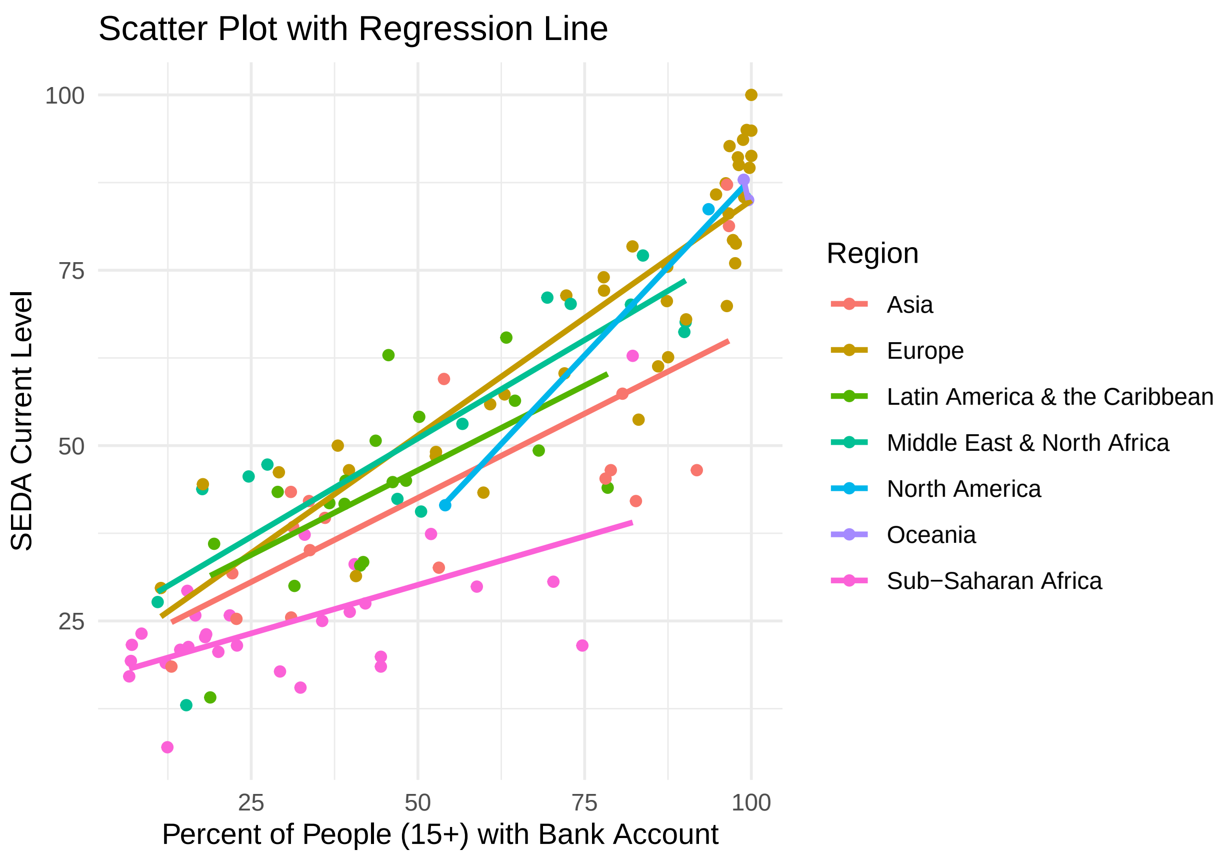

```r
# d. Overlay a fitted smoothing trend on top of the scatter plot.
ggplot(economist_data, aes(x = Percent.of.15plus.with.bank.account, y = SEDA.Current.level, color = Reg
  geom_point() +
  # Adding a fitted smoothing trend with low span
  geom_smooth(span = 0.3, method = "loess", se = FALSE) +
  theme_minimal() +
  labs(title = "Scatter Plot with Fitted Smoothing Trend (Low Span)",
       x = "Percent of People (15+) with Bank Account",
       y = "SEDA Current Level")
```

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

## Warning in sqrt(sum.squares/one.delta): NaNs produced

## Warning: Failed to fit group 5.
## Caused by error in `simpleLoess()`:
## ! span is too small

## Warning: Failed to fit group 6.
## Caused by error in `simpleLoess()`:
## ! span is too small
```



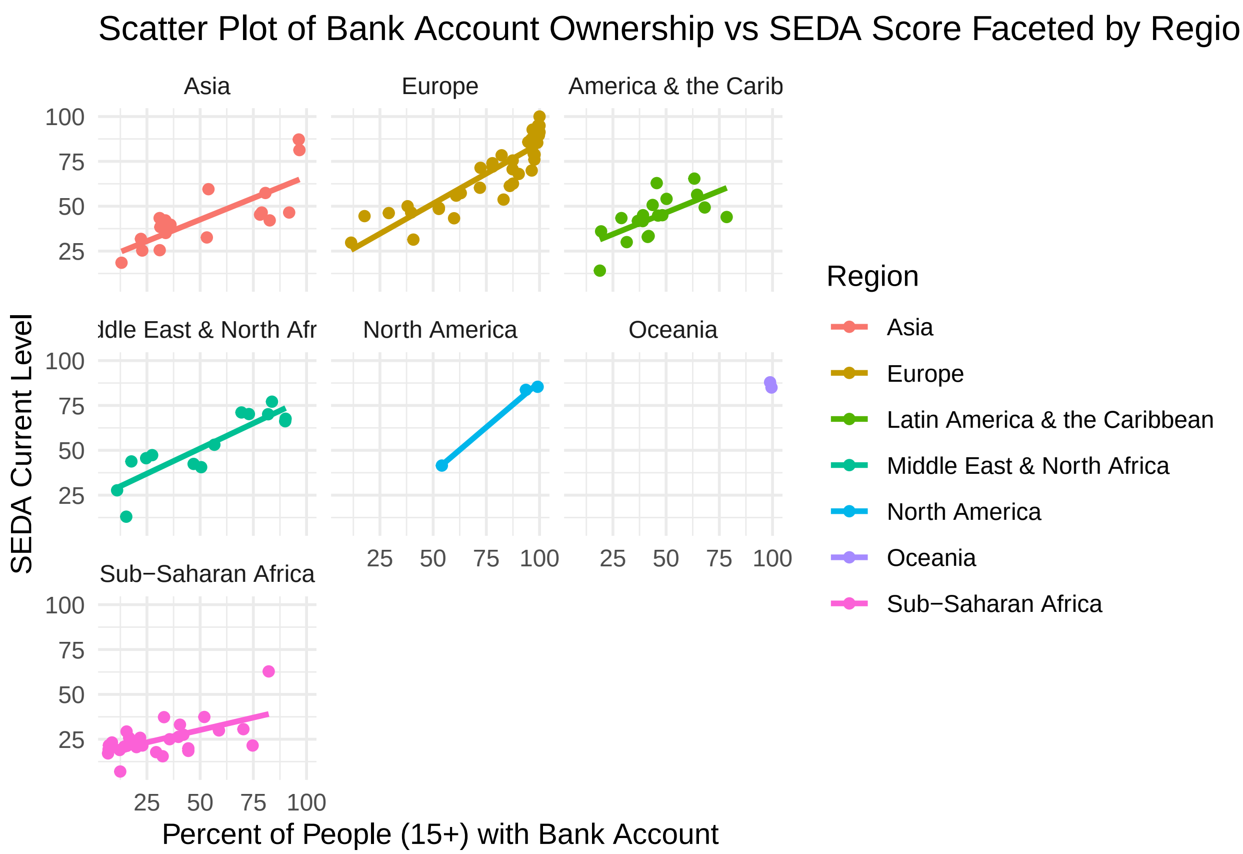

```r
# e. Create a scatter plot with a regression line overlaid
ggplot(economist_data, aes(x = Percent.of.15plus.with.bank.account, y = SEDA.Current.level, color = Regi
  geom_point() +
  # Overlay a regression line (method = "lm" for linear regression)
  geom_smooth(method = "lm", se = FALSE) +
  theme_minimal() +
  labs(title = "Scatter Plot with Regression Line",
       x = "Percent of People (15+) with Bank Account",
       y = "SEDA Current Level")
```

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



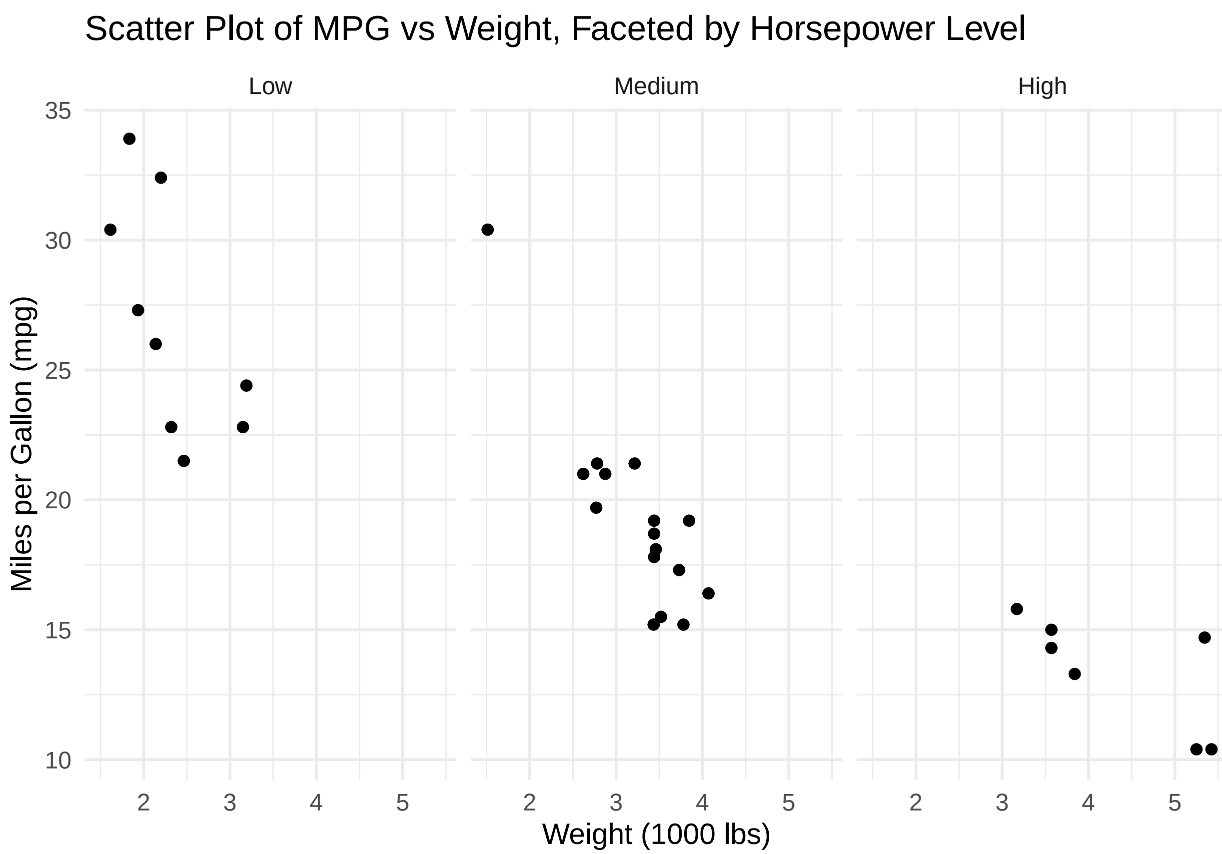

```
# f. Create a scatter plot with a regression line and facet it by Region
ggplot(economist_data, aes(x = Percent.of.15plus.with.bank.account, y = SEDA.Current.level, color = Regi
  geom_point() +
  # Overlay a regression line (method = "lm" for linear regression)
  geom_smooth(method = "lm", se = FALSE) +
  # Facet the plot by Region
  facet_wrap(~ Region) +
  theme_minimal() +
  labs(title = "Scatter Plot of Bank Account Ownership vs SEDA Score Faceted by Region",
       x = "Percent of People (15+) with Bank Account",
       y = "SEDA Current Level")
```

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



Question 5

```r
# Load the ggplot2 library
library(ggplot2)
# Load the mtcars dataset
data(mtcars)


# a. Convert the `hp` (horsepower) variable into a factor with three levels: "Low", "Medium", "High"
mtcars$hp_level <- cut(mtcars$hp,
                       breaks = c(-Inf, 100, 200, Inf),
                       labels = c("Low", "Medium", "High"))

# b. Create the scatter plot of `mpg` vs `wt`, faceted by the new `hp_level` factor
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  facet_wrap(~ hp_level) +
  theme_minimal() +
  labs(title = "Scatter Plot of MPG vs Weight, Faceted by Horsepower Level",
       x = "Weight (1000 lbs)",
       y = "Miles per Gallon (mpg)")
```




```
# c. How does converting hp into categorical groups enhance the interpretability of the plot?

# Converting hp into categories (Low, Medium, High) allows for easier comparison
# between groups rather than interpreting individual values across a continuous scale.

# d. Describe the differences observed in mpg for different hp levels.

# Vehicles in the "Low" horsepower category tend to have higher mpg values,
# indicating they are more fuel-efficient.

# Vehicles with "Medium" horsepower have a moderate mpg value,
# showing a balance between power and efficiency

# Vehicles in the "High" horsepower group generally have lower mpg,
# indicating less fuel efficiency due to their higher power requirements.

# e. What function is used to create categorical levels from continuous variables?

# The cut() function is used to create categorical levels from continuous variables.
# It allows you to specify breakpoints and labels, making it easy to transform numeric
# data into discrete categories.

# f. Can faceting by grouped levels provide more insight than using
# hp as a continuous variable on the x-axis?

# It reduces the complexity by breaking the data into simpler, comparable segments.
```

```r
# helps in focusing on trends within specific categories rather than analyzing each
# individual data point along a continuous scale.
```

Question 6

```r
# Load libraries
library(dplyr)

# Load the dataset from the given URL
url <- "https://raw.githubusercontent.com/Juanets/movie-stats/master/movies.csv"
movies <- read.csv(url)

# a. Find a subset of the movies produced after 2005
movies.sub <- movies %>%
  filter(year > 2005)

# Display first few rows of the subset
head(movies.sub)
```

```
##                                          name rating     genre year
## 1                               The Departed      R     Crime 2006
## 2       The Fast and the Furious: Tokyo Drift  PG-13    Action 2006
## 3 Talladega Nights: the Ballad of Ricky Bobby  PG-13    Comedy 2006
## 4                               The Prestige  PG-13     Drama 2006
## 5                                       Cars      G Animation 2006
## 6                                        300      R    Action 2006
##                         released score   votes          director
## 1  October 6, 2006 (United States)   8.5 1200000   Martin Scorsese
## 2    June 16, 2006 (United States)   6.0  252000        Justin Lin
## 3   August 4, 2006 (United States)   6.6  172000        Adam McKay
## 4 October 20, 2006 (United States)   8.5 1200000 Christopher Nolan
## 5     June 9, 2006 (United States)   7.1  381000     John Lasseter
## 6    March 9, 2007 (United States)   7.6  750000       Zack Snyder
##            writer              star        country   budget     gross
## 1 William Monahan Leonardo DiCaprio  United States 9.00e+07 291465373
## 2    Chris Morgan       Lucas Black  United States 8.50e+07 158964610
## 3    Will Ferrell      Will Ferrell  United States 7.25e+07 163362095
## 4  Jonathan Nolan    Christian Bale United Kingdom 4.00e+07 109676311
## 5   John Lasseter       Owen Wilson  United States 1.20e+08 461991867
## 6     Zack Snyder     Gerard Butler  United States 6.50e+07 456068181
##                  company runtime
## 1           Warner Bros.     151
## 2      Universal Pictures     104
## 3       Columbia Pictures     108
## 4     Touchstone Pictures     130
## 5 Pixar Animation Studios     117
## 6           Warner Bros.     117
```

```r
# b. Keep only the specified columns in `movies.sub`
movies.sub <- movies.sub %>%
  select(name, director, year, country, genre, budget, gross, score)
```

```r
# Display first few rows of the modified subset
head(movies.sub)
```

```
##                                         name          director year
## 1                               The Departed   Martin Scorsese 2006
## 2       The Fast and the Furious: Tokyo Drift        Justin Lin 2006
## 3 Talladega Nights: the Ballad of Ricky Bobby        Adam McKay 2006
## 4                               The Prestige Christopher Nolan 2006
## 5                                       Cars     John Lasseter 2006
## 6                                        300       Zack Snyder 2006
##          country     genre   budget     gross score
## 1  United States     Crime 9.00e+07 291465373   8.5
## 2  United States    Action 8.50e+07 158964610   6.0
## 3  United States    Comedy 7.25e+07 163362095   6.6
## 4 United Kingdom     Drama 4.00e+07 109676311   8.5
## 5  United States Animation 1.20e+08 461991867   7.1
## 6  United States    Action 6.50e+07 456068181   7.6
```

```r
# c. Calculate the profit for each movie in `movies.sub` as a fraction of its budget
movies.sub <- movies.sub %>%
  mutate(
    # Convert `budget` to million dollars and round to 1 decimal place
    budget = round(budget / 1e6, 1),
    # Convert `gross` to million dollars and round to 1 decimal place
    gross = round(gross / 1e6, 1),
    # Calculate profit as a fraction of the budget
    profit_fraction = (gross - budget) / budget
  )

# Display first few rows with profit calculation
head(movies.sub)
```

```
##                                         name          director year
## 1                               The Departed   Martin Scorsese 2006
## 2       The Fast and the Furious: Tokyo Drift        Justin Lin 2006
## 3 Talladega Nights: the Ballad of Ricky Bobby        Adam McKay 2006
## 4                               The Prestige Christopher Nolan 2006
## 5                                       Cars     John Lasseter 2006
## 6                                        300       Zack Snyder 2006
##          country     genre budget gross score profit_fraction
## 1  United States     Crime   90.0 291.5   8.5       2.2388889
## 2  United States    Action   85.0 159.0   6.0       0.8705882
## 3  United States    Comedy   72.5 163.4   6.6       1.2537931
## 4 United Kingdom     Drama   40.0 109.7   8.5       1.7425000
## 5  United States Animation  120.0 462.0   7.1       2.8500000
## 6  United States    Action   65.0 456.1   7.6       6.0169231
```

```r
# d. Count the number of movies produced by each genre and order them in descending order
genre_count <- movies.sub %>%
  group_by(genre) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
```

```r
# Display the genre count
print(genre_count)
```

```
## # A tibble: 16 x 2
##    genre     count
##    <chr>     <int>
##  1 Action      738
##  2 Comedy      629
##  3 Drama       548
##  4 Biography   228
##  5 Animation   189
##  6 Crime       176
##  7 Adventure   151
##  8 Horror      136
##  9 Fantasy      10
## 10 Mystery       5
## 11 Sci-Fi        4
## 12 Thriller      4
## 13 Family        2
## 14 Musical       2
## 15 Romance       2
## 16 Sport         1
```

```r
# e. Group the `movies.sub` dataset by `country` and `genre`, then calculate the required metrics
movies_grouped <- movies.sub %>%
  group_by(country, genre) %>%
  summarise(
    # Count the number of movies in each group
    count = n(),
    # Median of profit as a fraction of budget
    median_profit_fraction = median(profit_fraction, na.rm = TRUE),
    # Mean of the movie score
    mean_score = mean(score, na.rm = TRUE),
    # Variance of the movie score
    variance_score = var(score, na.rm = TRUE),
    # Drop the grouping after summarizing
    .groups = "drop"
  ) %>%
  # Arrange by count in descending order
  arrange(desc(count))

# Display the result
print(movies_grouped)
```

```
## # A tibble: 175 x 6
##    country        genre    count median_profit_fraction mean_score variance_score
##    <chr>          <chr>    <int>                  <dbl>      <dbl>          <dbl>
##  1 United States  Action     527                  1.32        6.33          0.769
##  2 United States  Comedy     502                  1.42        6.14          0.798
##  3 United States  Drama      306                  1.66        6.56          0.757
##  4 United States  Animat~    133                  2.51        6.66          0.847
##  5 United States  Biogra~    122                  0.892       6.99          0.356
```

```
##  6 United States  Crime     111                  0.688      6.58          0.700
##  7 United States  Horror    109                  3.76       5.70          0.807
##  8 United States  Advent~    98                  1.54       6.36          1.14
##  9 United Kingdom Drama      75                  0.829      6.79          0.505
## 10 United Kingdom Action     67                  0.547      6.65          0.560
## # i 165 more rows
```

Question 7

```r
# library(dplyr)
movies_filtered <- movies %>%
  # Filter movies produced after 2001
  filter(year > 2001) %>%
  # Calculate the number of movies for each director and
  # filter out those with fewer than 4 movies
  group_by(director) %>%
  filter(n() >= 4) %>%
  ungroup() %>%
  # Group by genre and director, then calculate the mean score for each director
  group_by(genre, director) %>%
  summarise(mean_score = mean(score, na.rm = TRUE), .groups = "drop") %>%
  ungroup() %>%
  # Step 4: For each genre, select the top two directors with the highest mean scores
  group_by(genre) %>%
  top_n(n = 2, wt = mean_score) %>%
  # Arrange by genre and descending mean score for clarity
  arrange(genre, desc(mean_score)) %>%
  ungroup()
  .groups = "drop"

# Display the result
print(movies_filtered)
```

```
## # A tibble: 25 x 3
##    genre     director           mean_score
##    <chr>     <chr>                   <dbl>
##  1 Action    Park Chan-Wook           8.4
##  2 Action    Christopher Nolan        8.27
##  3 Adventure Christopher Nolan        8.6
##  4 Adventure Quentin Tarantino        8.3
##  5 Animation Andrew Stanton           7.93
##  6 Animation Wes Anderson             7.9
##  7 Biography Roman Polanski           8.5
##  8 Biography Tom McCarthy             8.1
##  9 Comedy    Bong Joon Ho             8.6
## 10 Comedy    Rian Johnson             7.9
## # i 15 more rows
```

Question 8

The continuous random variable X has the following probability density function (PDF), for some positive constant c:

$$f(x) = \frac{3}{(1+x)^3}, \quad 0 \leq x \leq c$$

To determine the constant c such that $f(x)$ is a legitimate PDF, we need to ensure that the total area under the curve of $f(x)$ over the given domain is equal to 1.

Step 1: Set Up the Integral

To find the value of c:

$$\int_0^c f(x)\, dx = 1$$

Substituting $f(x)$:

$$\int_0^c \frac{3}{(1+x)^3}\, dx = 1$$

Step 2: Solve the Integral

To evaluate this integral, we use substitution. Let:

$$u = 1 + x, \quad \text{then } du = dx$$

Rewrite the limits in terms of u:

- When $x = 0$, $u = 1$
- When $x = c$, $u = 1 + c$

The integral becomes:

$$\int_1^{1+c} \frac{3}{u^3}\, du$$

Now, evaluate the antiderivative:

$$\int \frac{3}{u^3}\, du = 3 \int u^{-3}\, du = 3 \cdot \left(\frac{u^{-2}}{-2} \right) = -\frac{3}{2} u^{-2}$$

Step 3: Substitute the Limits

Evaluate the definite integral from $u = 1$ to $u = 1 + c$:

$$\left[-\frac{3}{2} \cdot \frac{1}{u^2} \right]_1^{1+c} = -\frac{3}{2(1+c)^2} + \frac{3}{2}$$

Simplify:

$$\frac{3}{2} - \frac{3}{2(1+c)^2} = 1$$

Step 4: Solve for c

To find c:

1. Move the constant to the other side:

$$\frac{3}{2} - 1 = \frac{3}{2(1+c)^2}$$

$$\frac{1}{2} = \frac{3}{2(1+c)^2}$$

2. Cross multiply to solve for $(1+c)^2$:

$$1 \cdot 2(1+c)^2 = 2 \cdot 3$$

$$2(1+c)^2 = 6$$

3. Divide both sides by 2:

$$(1+c)^2 = 3$$

4. Take the square root of both sides:

$$1 + c = \sqrt{3}$$

5. Solve for c:

$$c = \sqrt{3} - 1$$

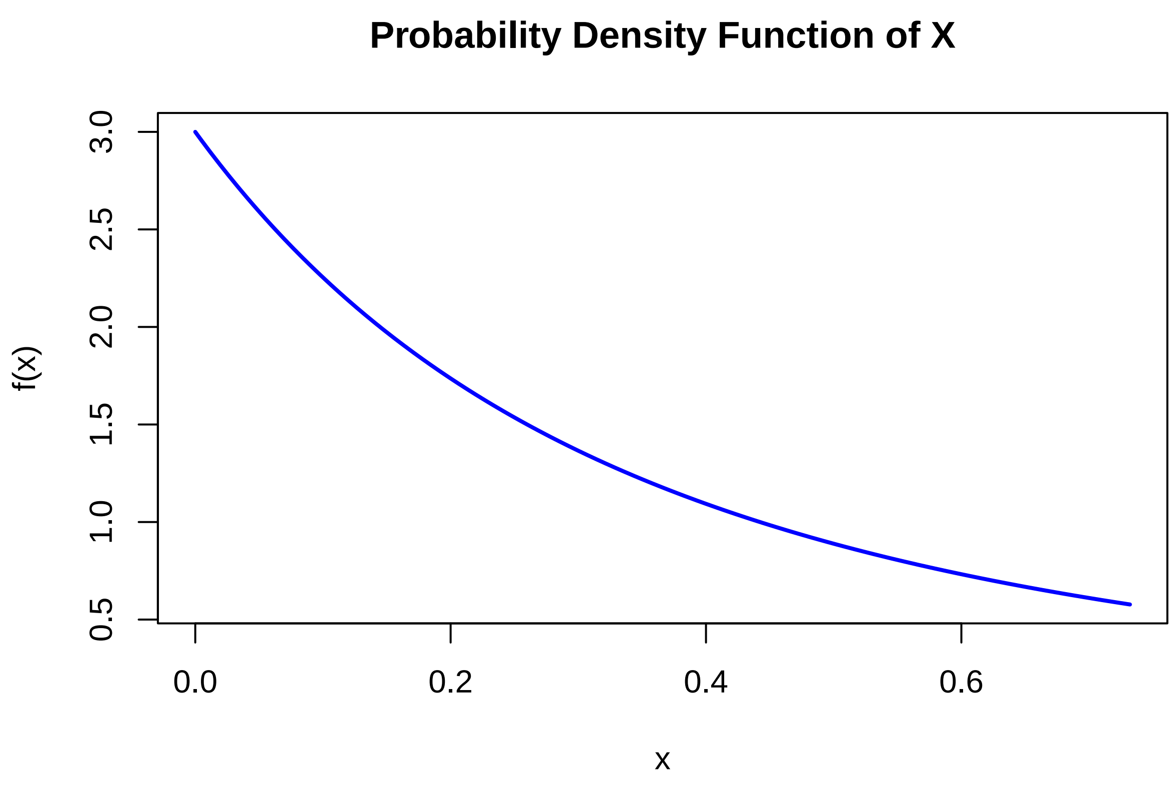
Conclusion

The value of c that makes $f(x)$ a legitimate PDF is:

$$c = \sqrt{3} - 1$$

Now that we have the value of c, we can plot the PDF in R:

```r
# (b)
# Define the value of c
c <- sqrt(3) - 1

# Define the PDF function
f <- function(x) {
  3 / (1 + x)^3
}

# Define the range for plotting
x_vals <- seq(0, c, length.out = 1000)

# Calculate the y-values for each x
y_vals <- f(x_vals)

# Plot the PDF
plot(x_vals, y_vals, type = "l", col = "blue", lwd = 2,
     main = "Probability Density Function of X",
     xlab = "x", ylab = "f(x)")
```



(c) The Expected Value $E(X)$

The expected value $E(X)$ is defined as:

$$E(X) = \int_0^c x \cdot f(x)\,dx = \int_0^c x \cdot \frac{3}{(1+x)^3}\,dx$$

Step 2.1: Set Up and Simplify the Integral

Substituting the value of $f(x)$:

$$E(X) = \int_0^c \frac{3x}{(1+x)^3}\,dx$$

We use substitution again:

- Let $u = 1 + x$, then $du = dx$, and $x = u - 1$.
- When $x = 0$, $u = 1$.
- When $x = c$, $u = 1 + c$.

The integral becomes:

$$E(X) = \int_1^{1+c} \frac{3(u-1)}{u^3}\,du = 3\int_1^{1+c} \left(\frac{u-1}{u^3}\right)\,du$$

Expanding:

$$E(X) = 3\int_1^{1+c} \left(\frac{1}{u^2} - \frac{1}{u^3}\right)\,du$$

Step 2.2: Integrate Each Term

Integrate each term:

$$\int u^{-2}\,du = -\frac{1}{u}$$

$$\int u^{-3}\,du = -\frac{1}{2u^2}$$

Evaluating the definite integral:

$$E(X) = 3\left(\left[-\frac{1}{u}\right]_1^{1+c} - \left[-\frac{1}{2u^2}\right]_1^{1+c}\right)$$

Substituting the limits:

1. **For** $-\frac{1}{u}$:

$$\left[-\frac{1}{u}\right]_1^{1+c} = -\frac{1}{1+c} + \frac{1}{1} = 1 - \frac{1}{1+c}$$

2. **For** $-\frac{1}{2u^2}$:

$$\left[-\frac{1}{2u^2}\right]_1^{1+c} = -\frac{1}{2(1+c)^2} + \frac{1}{2}$$

Putting it all together:

$$E(X) = 3\left(1 - \frac{1}{1+c} + \frac{1}{2} - \frac{1}{2(1+c)^2}\right)$$

Step 2.3: Substitute $c = \sqrt{3} - 1$

Substituting $c = \sqrt{3} - 1$:

1. $1 + c = \sqrt{3}$

$$E(X) = 3\left(1 + \frac{1}{2} - \frac{1}{\sqrt{3}} - \frac{1}{2(\sqrt{3})^2}\right)$$

Simplify:

- $1 + \frac{1}{2} = \frac{3}{2}$
- $\frac{1}{2(\sqrt{3})^2} = \frac{1}{6}$

Combining:

$$E(X) = 3\left(\frac{3}{2} - \frac{1}{\sqrt{3}} - \frac{1}{6}\right)$$

Common denominator for $\frac{3}{2}$ and $\frac{1}{6}$:

$$\frac{3}{2} - \frac{1}{6} = \frac{8}{6} = \frac{4}{3}$$

So:

$$E(X) = 3\left(\frac{4}{3} - \frac{1}{\sqrt{3}}\right)$$

Multiply by 3:

$$E(X) = 4 - \frac{3}{\sqrt{3}}$$

Simplify:

$$E(X) = 4 - \sqrt{3}$$

Final Answer

The expected value $E(X)$ is:

$$E(X) = 4 - \sqrt{3}$$

$$E(X) = 2.26$$

```r
#(d) Define the function for the cumulative distribution (inverse transformation method)

func_inv <- function(u) {
  (2 * (1 - u))^(-1/2) - 1
}

# Simulate 1000 random observations
set.seed(123)  # Set a seed for reproducibility
u_vals <- runif(1000)
simulated_data <- func_inv(u_vals)

# (e) Estimate mean and variance
estimated_mean <- mean(simulated_data)
estimated_variance <- var(simulated_data)

# Print the results
cat("Estimated Mean:", estimated_mean, "\n")
```

```
## Estimated Mean: 0.3725972
```

```r
cat("Estimated Variance:", estimated_variance, "\n")
```

```
## Estimated Variance: 1.944269
```