STA234 HW5

Derin Gezgin

2025-03-31

99/100

Importing the Required Libraries

```r
library(ggplot2)
library(dplyr)
library(tidyverse)
library(gapminder)
library(janitor)
```

Problem 1

We will use the built-in mtcars dataset in R. This dataset contains information about car models from the 1970s.

```r
data(mtcars)
```

Use `str()`, `summary()`, and `head()` to explore the dataset.

```r
str(mtcars)

## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...

summary(mtcars)

##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
```

```
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

```r
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
```

1.(A)

Create a new column called `hp_per_cyl` that calculates the horsepower per cylinder (`hp` divided by `cyl`). Do not print.

```r
mtcars = mtcars %>%
    mutate(hp_per_cyl = hp / cyl)
```

We can simply create the new column using the `mutate` function and taking the ratio of two columns.

1.(B)

Filter the dataset to include only cars with 6 or more cylinders. Do not print data, just create a subset.

```r
mtcars_filtered = mtcars %>%
    filter(cyl >= 6)
```

We can use the `filter()` function to create a subset of `mtcars` with a cylinder value (cyl column in the mtcars dataset) of 6 or more.

1.(C)

Select the columns `mpg`, `cyl`, and `hp_per_cyl` from the filtered dataset and rename them to `Miles_Per_Gallon`, `Cylinders`, and `Horsepower_Per_Cylinder`, respectively. Do not print data, just create a subset.

```
mtcars_selected = mtcars_filtered %>%
    select(mpg, cyl, hp_per_cyl) %>%
    rename(Miles_Per_Gallon = mpg,
           Cylinders = cyl,
           Horsepower_Per_Cylinder = hp_per_cyl)
```

We can first select the columns *mpg, cyl, hp_per_cyl* using the `select` function and we can use the `rename` function to change the column names.

1.(D)

Using `group_by()` and `summarise()`, calculate the average `Miles_Per_Gallon` and average `Horsepower_Per_Cylinder` for each number of cylinders. Present the results in a new dataframe.

```
summary_df = mtcars_selected %>%
    group_by(Cylinders) %>%
    summarise(Avg_Miles_Per_Gallon = mean(Miles_Per_Gallon,
                                          na.rm = TRUE),
              Avg_Horsepower_Per_Cylinder = mean(Horsepower_Per_Cylinder,
                                                 na.rm = TRUE))

summary_df

## # A tibble: 2 × 3
##   Cylinders Avg_Miles_Per_Gallon Avg_Horsepower_Per_Cylinder
##       <dbl>                <dbl>                       <dbl>
## 1         6                 19.7                        20.4
## 2         8                 15.1                        26.2
```

We can use the group_by function to group the rows we have by cylinder count, and create a new summary dataframe of average miles per gallon and horsepower per cylinder using the `summarise` function.

1.(E)

Create a scatter plot showing the relationship between `Horsepower_Per_Cylinder` and `Miles_Per_Gallon`. Use different colors to indicate the number of cylinders. Make sure # of cylinders is a factor, if it is not so by default.

```
is.factor(mtcars_selected$Cylinders)
```

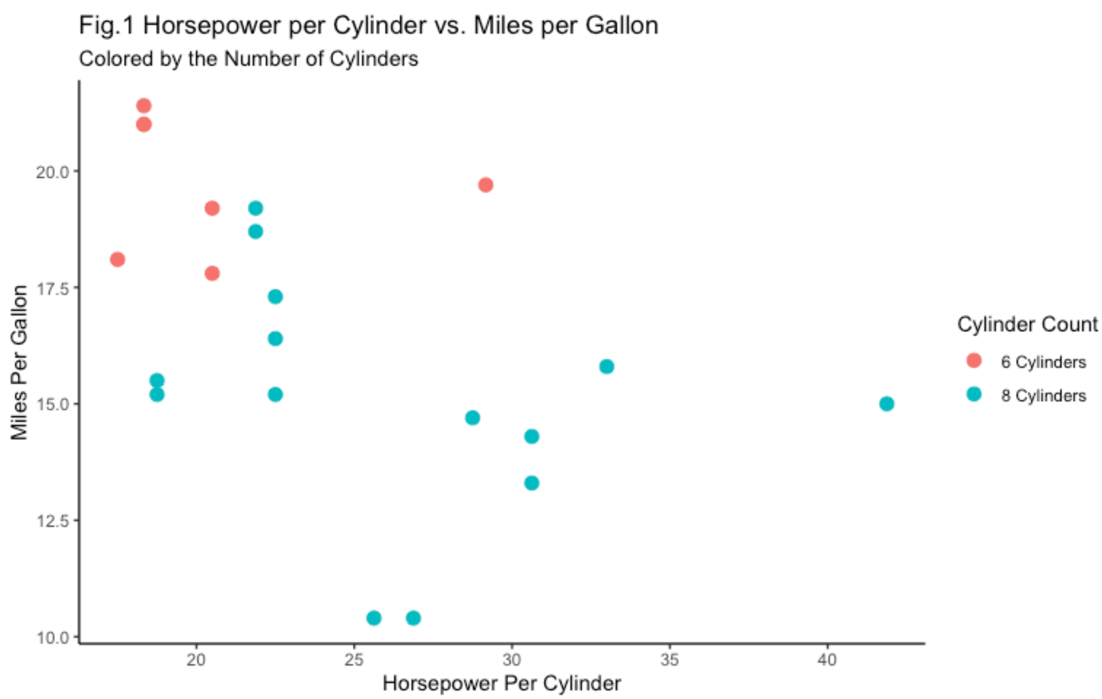
```
## [1] FALSE
```

We can see that the `Cylinders` column is not a factor.

```
mtcars_selected$Cylinders = factor(mtcars_selected$Cylinders,
                                   levels = c(6, 8),
                                   labels = c("6 Cylinders", "8 Cylinders"))
```

We can convert the `Cylinders` column to a factor and also set the labels for specific levels for better explanation of the data.

```
horsePower_mpg_plot = ggplot(mtcars_selected,
                             aes(x = Horsepower_Per_Cylinder,
                                 y = Miles_Per_Gallon)) +
    geom_point(size = 3,
               aes(color = Cylinders)) +
    labs(title = "Fig.1 Horsepower per Cylinder vs. Miles per Gallon",
         subtitle = "Colored by the Number of Cylinders",
         x = "Horsepower Per Cylinder",
         y = "Miles Per Gallon",
         color = "Cylinder Count") +
    theme_classic()

horsePower_mpg_plot
```

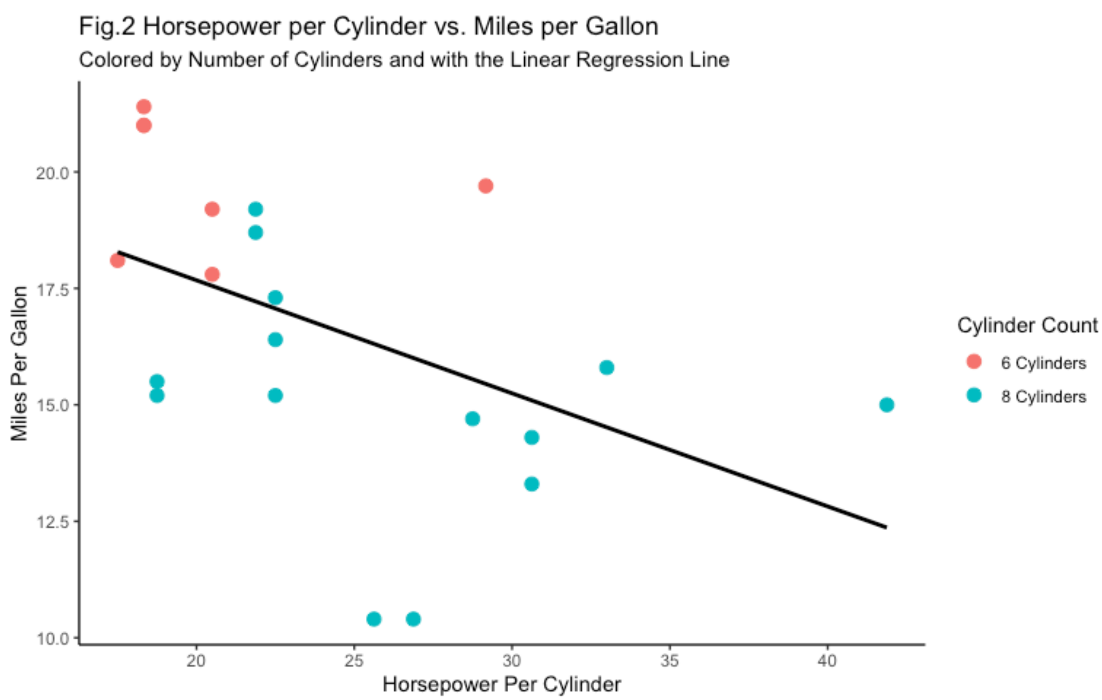


Following this, we can plot the data in a scatterplot using the `geom_point` function.

1.(F)

Enhance your scatter plot by adding a regression line to illustrate the linear trend. Include labels for the axes and a title that convey's plot's message accurately.

```
lm_horsePower_mpg_plot = horsePower_mpg_plot +
    geom_smooth(method = "lm",
                se = FALSE,
                color = "black") +
    labs(title = "Fig.2 Horsepower per Cylinder vs. Miles per Gallon",
         subtitle = "Colored by Number of Cylinders and with the Linear
Regression Line")

lm_horsePower_mpg_plot
```



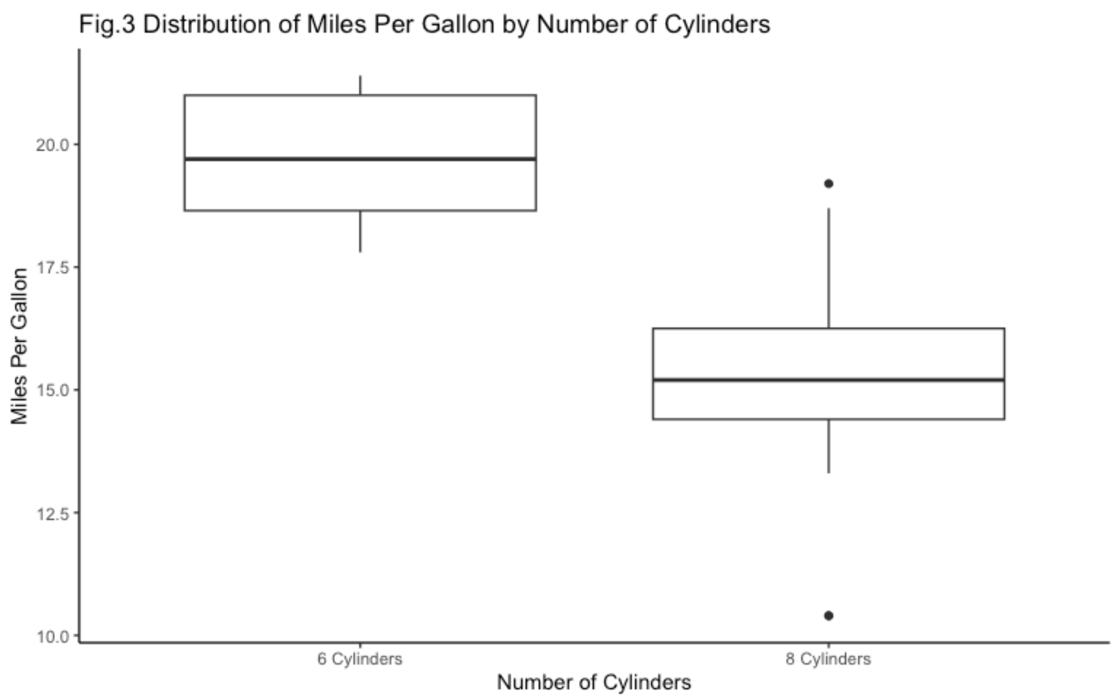
We can easily add the linear regression line using the `geom_smooth` function and the specific `"lm"` method.

1.(G)

Create a boxplot to visualize the distribution of `Miles_Per_Gallon` for different cylinder categories. Make sure you have appropriate labels and title.

```
mpg_boxplot = ggplot(mtcars_selected,
                     aes(x = Cylinders,
                         y = Miles_Per_Gallon))+
    geom_boxplot() +
```

```
    labs(title = "Fig.3 Distribution of Miles Per Gallon by Number of Cylinders",
         x = "Number of Cylinders",
         y = "Miles Per Gallon") +
    theme_classic()

mpg_boxplot
```



We can create the box-plot using the `geom_boxplot` function.

1.(H)

Write a brief interpretation of the results from the graphs. Discuss any relationships and patterns you observe between number of cylinders and its effect on the fuel efficiency.

From the graphs, we can see a somewhat clear relationship between the number of cylinders in a car and its fuel efficiency.

Figure 2 shows us that vehicles with 8 cylinders (blue data points) tend to have a higher horsepower per cylinder but also a lower miles per gallon compared to 6-cylinder vehicles (red data points). We can say that 8-cylinder vehicles might deliver more power but they also have a lower fuel-efficiency. At the same time, the linear regression line shows us that, as the horsepower per cylinder increases, the fuel efficiency is expected to decrease, however, this is not a strong relationship. On the other hand, Figure 3 shows us that the median MPG for 8-cylinder cars is lower in general compared to 6-cylinder cars.

Overall, we can conclude that, according to our data from the `mtcars` dataset, higher cylinder count decreases the fuel efficiency (amount of miles a car can cover with a gallon of fuel).

Problem 2

We will use the gapminder dataset, which's in the `gapminder` package. Install the `gapminder` package.

```
data(gapminder)
```

Read about the data, use `str()`, `summary()`, and `head()` to understand the data.

```
str(gapminder)

## tibble [1,704 × 6] (S3: tbl_df/tbl/data.frame)
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1
...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3
3 ...
##  $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992
1997 ...
##  $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
##  $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460
14880372 12881816 13867957 16317921 22227415 ...
##  $ gdpPercap: num [1:1704] 779 821 853 836 740 ...

summary(gapminder)

##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
##       pop              gdpPercap
##  Min.   :6.001e+04   Min.   :   241.2
##  1st Qu.:2.794e+06   1st Qu.:  1202.1
##  Median :7.024e+06   Median :  3531.8
##  Mean   :2.960e+07   Mean   :  7215.3
##  3rd Qu.:1.959e+07   3rd Qu.:  9325.5
##  Max.   :1.319e+09   Max.   :113523.1
##

head(gapminder)
```

```
## # A tibble: 6 × 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

2.(A)

Create a new column for `gdp_per_cap` that calculates GDP per capita. Do not print.

```
gapminder = gapminder %>%
    mutate(gdp_per_cap = gdpPercap)
```

We can use the `mutate` function to create the new column.

2.(B)

Filter the dataset to include only data from the year 2007. Do not print, just create a subset data.

```
gapminder_2007 = gapminder %>%
    filter(year == 2007)
```

We can filter the data to only have the data from 2007, using the `filter` function.

2.(C)

Select the relevant columns: `country`, `continent`, `lifeExp` (life expectancy), and `gdp_per_cap`, and rename them for clarity. Do not print, just create a subset data.

```
gapminder_2007_selected = gapminder_2007 %>%
    select(country, continent, lifeExp, gdp_per_cap) %>%
    rename(Country = country,
           Continent = continent,
           lifeExpectancy = lifeExp,
           GDPperCapita = gdp_per_cap)
```

We can select the columns using the `select` function. To rename them, we can simply use the `rename` function.

2.(D)

Group the dataset by continent and calculate the average life expectancy and GDP per capita for each continent in 2007. Show the summary.

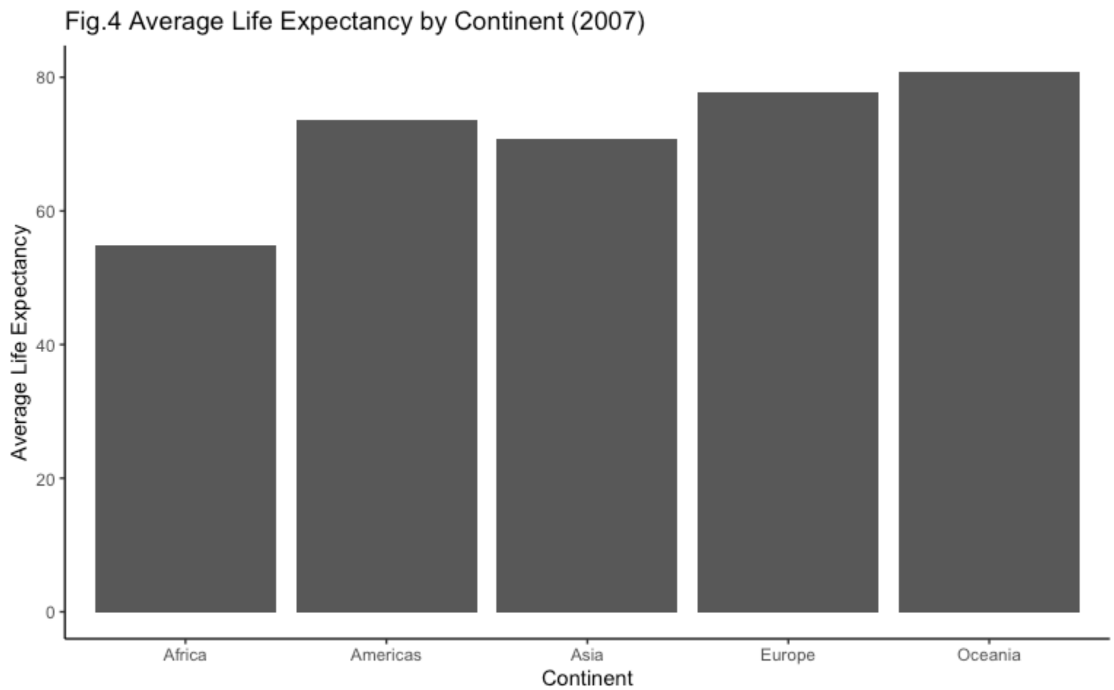
```
continent_summary = gapminder_2007_selected %>%
    group_by(Continent) %>%
    summarise(Avg_Life_Expectancy = mean(lifeExpectancy,
                                         na.rm = TRUE),
              Avg_GDP_Per_Capita = mean(GDPperCapita,
                                        na.rm = TRUE))

continent_summary

## # A tibble: 5 × 3
##   Continent Avg_Life_Expectancy Avg_GDP_Per_Capita
##   <fct>                   <dbl>              <dbl>
## 1 Africa                   54.8              3089.
## 2 Americas                 73.6             11003.
## 3 Asia                     70.7             12473.
## 4 Europe                   77.6             25054.
## 5 Oceania                  80.7             29810.
```

To show this summary, we can first group the data by Continent, and then calculate the averages.

2.(E)

Create a bar plot displaying average life expectancy for each continent.

```
avg_lifeExpectancy_bplot = ggplot(continent_summary,
                                  aes(x = Continent,
                                      y = Avg_Life_Expectancy)) +
    geom_bar(stat = "identity") +
    labs(title = "Fig.4 Average Life Expectancy by Continent (2007)",
         x = "Continent",
         y = "Average Life Expectancy") +
    theme_classic()

avg_lifeExpectancy_bplot
```

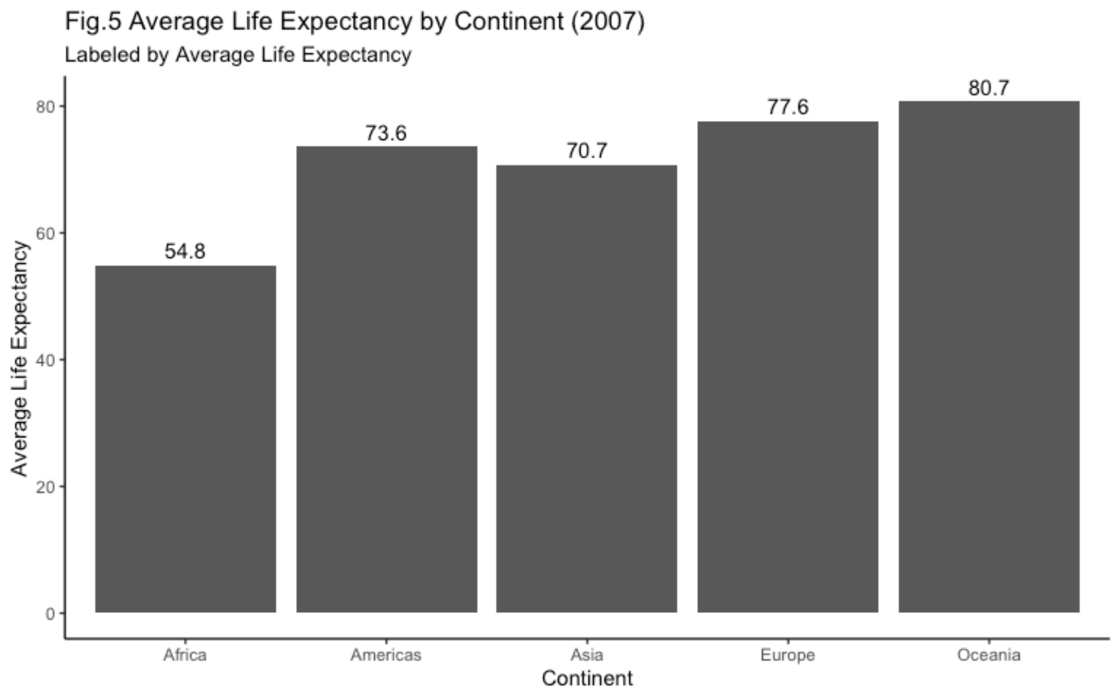


We can use the geom_bar function to create the bar plot of average life expectancy for each continent.

2.(F)

In the above barplot add average life expectancy as text (using `geom_text()`) for each continent at the top of the bar.

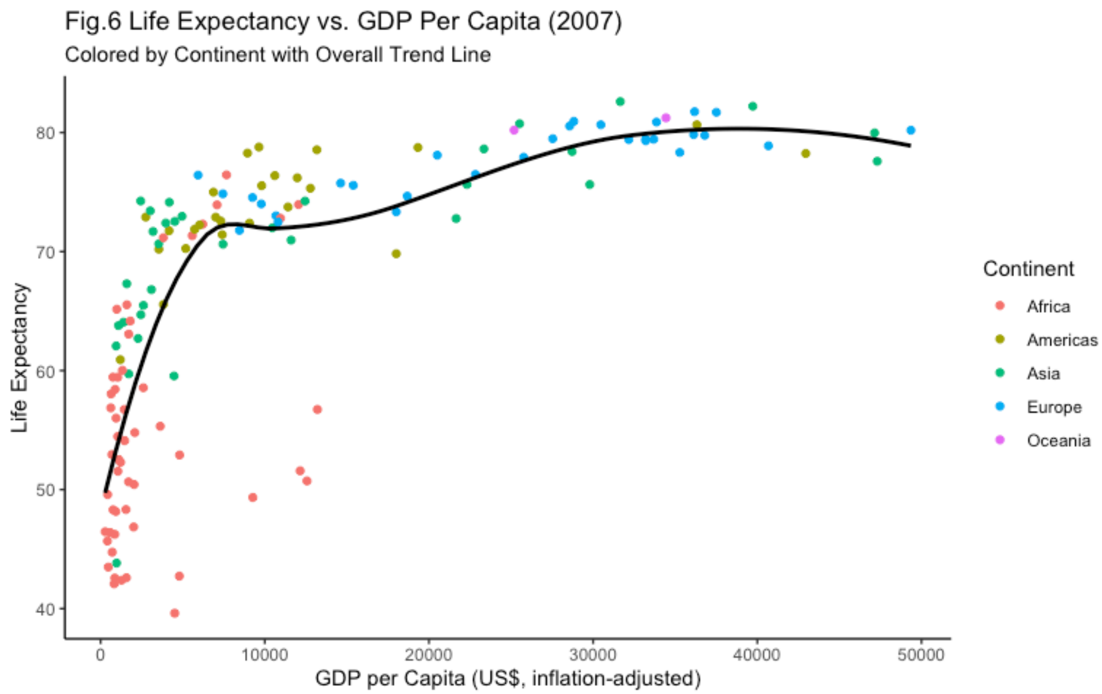
```
avg_bplot_wText = avg_lifeExpectancy_bplot +
    geom_text(aes(label = round(Avg_Life_Expectancy,1),
                  vjust = -0.5,)) +
    labs(title = "Fig.5 Average Life Expectancy by Continent (2007)",
         subtitle = "Labeled by Average Life Expectancy")

avg_bplot_wText
```



2.(G)

Create a scatter plot showing the relationship between life expectancy and GDP per capita for all countries in 2007. Use different colors for different continents. Add a regression line to the scatter plot to illustrate the trend. Write a brief explanation of the relationship between GDP per capita and life expectancy across continents.

```
lifeExpectancy_GDP_plot = ggplot(gapminder_2007_selected,
                                 aes(x = GDPperCapita,
                                     y = lifeExpectancy)) +
    geom_point(aes(color = Continent)) +
    geom_smooth(se = FALSE,
                color = "black") +
    labs(title = "Fig.6 Life Expectancy vs. GDP Per Capita (2007)",
         subtitle = "Colored by Continent with Overall Trend Line",
         x = "GDP per Capita (US$, inflation-adjusted)",
         y = "Life Expectancy",
         color = "Continent") +
    theme_classic()

lifeExpectancy_GDP_plot
```

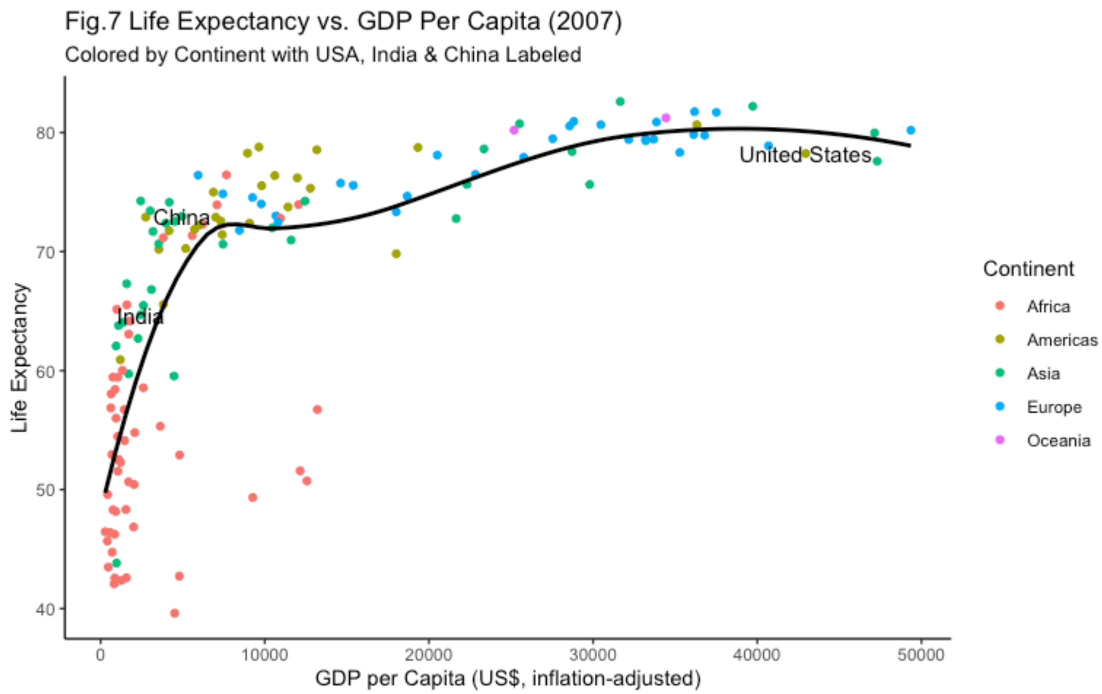


From Figure 6, we can see a very clear positive non-linear relationship between GDP per Capita and Life Expectancy. As the GDP per captita increases, the life expectancy also increase. Between the 0-10000$ range, this increase is really significant while it slows down for the rest of the GDP per Capita range. We can also see that most of the countries in Africa are in the low Life Expectancy, low GDP per Capita range, while the European countries show a high life expectancy (over 70) regardless of the GDP per Capita.

2.(H)

In the above scatter plot, add the `geom_text()` function to label USA, India, China on the plot.

```r
lifeExpectancy_GDP_plot_text = lifeExpectancy_GDP_plot +
    geom_text(data = subset(gapminder_2007_selected,
                            Country %in% c("United States",
                                           "India",
                                           "China")),
              aes(label = Country),
              size = 4) +
    labs(title = "Fig.7 Life Expectancy vs. GDP Per Capita (2007)",
         subtitle = "Colored by Continent with USA, India & China Labeled")

lifeExpectancy_GDP_plot_text
```



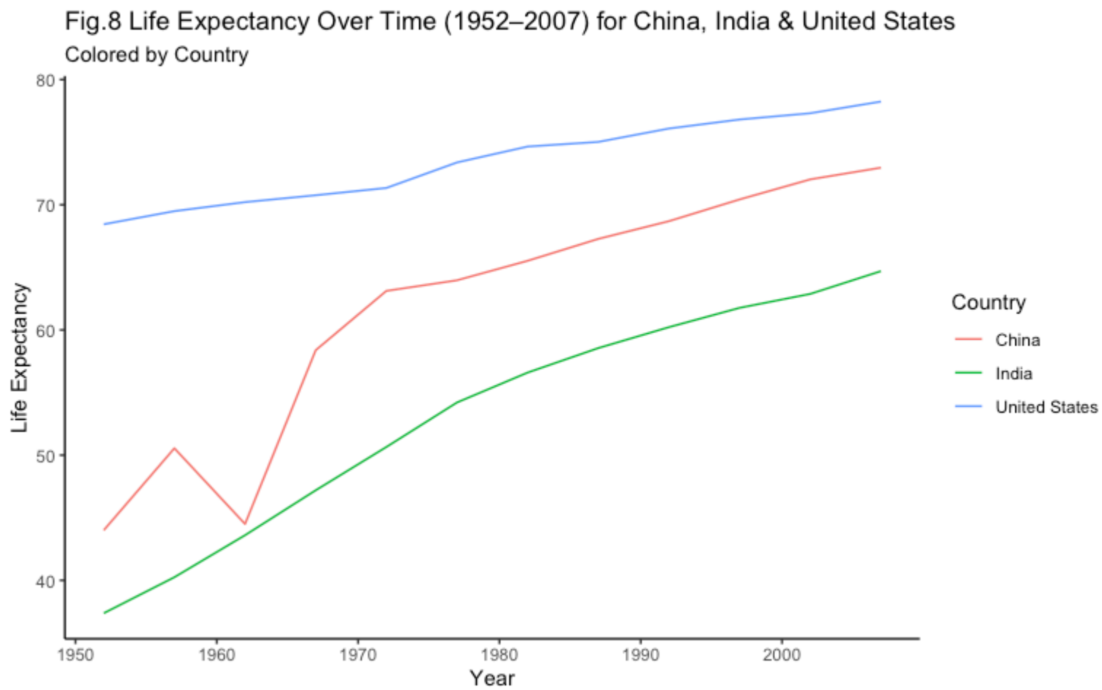
2.(I)

Create a line graph showing the change in life expectancy for US, India and China across the years available. Use facets for different countries, and colors, points for clarity. Try facets with fixed scales and then free scales.

```
gapminder_subset = gapminder %>%
    filter(country %in% c("United States", "India", "China"))
```

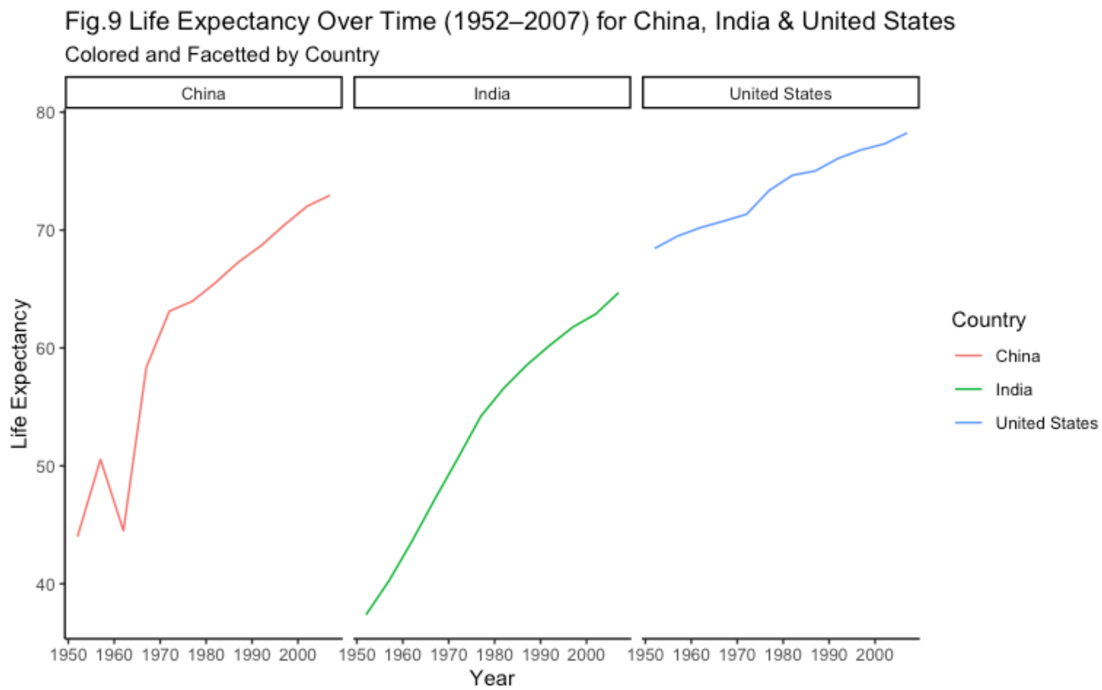
To tackle this question, we should first create a subset of the `gapminder` dataset to select the requested countries.

```
lifeExp_lGraph_base = ggplot(gapminder_subset,
                                aes(x = year,
                                    y = lifeExp,
                                    color = country)) +
    geom_line() +
    labs(title = "Fig.8 Life Expectancy Over Time (1952-2007) for China,
India & United States",
         subtitle = "Colored by Country",
         x = "Year",
         y = "Life Expectancy",
         color = "Country") +
    theme_classic()

lifeExp_lGraph_base
```



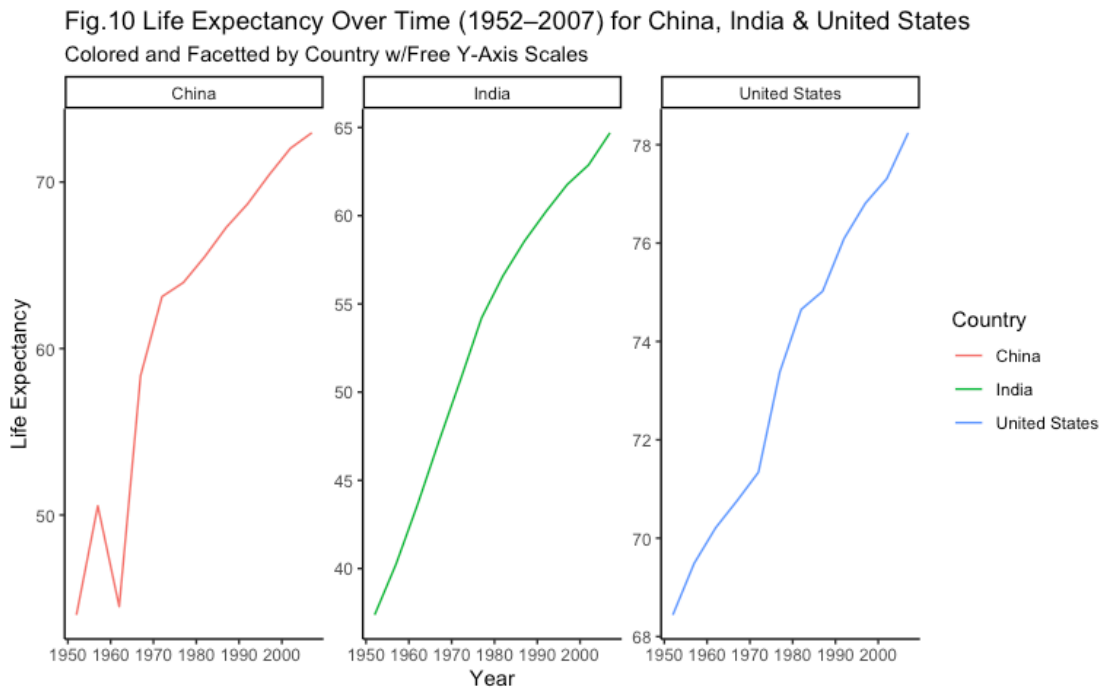
We can use the geom_line function to create a line graph of Life Expectancy over years for China, India, and United States.

```
lifeExp_lGraph_faceted = lifeExp_lGraph_base +
    facet_wrap(~ country) +
    labs(title = "Fig.9 Life Expectancy Over Time (1952–2007) for China,
India & United States",
         subtitle = "Colored and Facetted by Country")

lifeExp_lGraph_faceted
```



We can basically add `facet_wrap` on top of our base plot.

```
lifeExp_lGraph_faceted_freeY = lifeExp_lGraph_base +
    facet_wrap(~ country, scales = "free_y") +
    labs(title = "Fig.10 Life Expectancy Over Time (1952–2007) for China,
India & United States",
         subtitle = "Colored and Facetted by Country w/Free Y-Axis Scales")

lifeExp_lGraph_faceted_freeY
```



To make the y-axis have a free scale, we can add an argument `scales = "free_y"` to the `facet_wrap`.

Add points for clarity!! -1

Problem 3

Using the data on parcel boundaries with address and revenue-related information for properties in Wake County, NC (parcels.csv file from Moodle). Get packages tidyverse and janitor. Data Source.

```
parcel.data = read.csv("parcels.csv")
```

3.(A)

Use function `clean_name()` on the parcels data, and show names before and after using the function. Do not print data, just the names.

```
names(parcel.data)

##  [1] "OBJECTID"          "PIN_NUM"           "CALC_AREA"
##  [4] "REID"              "MAP_NAME"          "OWNER"
##  [7] "ADDR1"             "ADDR2"             "ADDR3"
## [10] "DEED_BOOK"         "DEED_PAGE"         "DEED_DATE"
## [13] "DEED_ACRES"        "BLDG_VAL"          "LAND_VAL"
## [16] "TOTAL_VALUE_ASSD"  "BILLCLASS"
```

```
"BILLING_CLASS_DECODE"
## [19] "PROPDESC"              "HEATEDAREA"          "STNAME"
## [22] "STYPE"                 "STPRE"               "STSUF"
## [25] "STNUM"                 "STMISC"              "SITE_ADDRESS"
## [28] "FULL_STREET_NAME"      "CITY"                "CITY_DECODE"
## [31] "PLANNING_JURISDICTION" "TOWNSHIP"            "TOWNSHIP_DECODE"
## [34] "FIREDIST"              "YEAR_BUILT"          "TOTSALPRICE"
## [37] "SALE_DATE"             "TYPE_AND_USE"        "TYPE_USE_DECODE"
## [40] "DESIGNSTYL"            "DESIGN_STYLE_DECODE" "UNITS"
## [43] "LAND_CLASS"            "LAND_CLASS_DECODE"   "EXEMPTDESC"
## [46] "EXEMPTSTAT"            "OWNERSHIP"           "ACTIVITY"
## [49] "FUNCTION"              "STRUCTURE"           "SITE"
## [52] "TOTSTRUCTS"            "TOTUNITS"            "OLD_PARCEL_NUMBER"
## [55] "ZIPNUM"                "PARCEL_PK"           "LAND_CODE"
## [58] "SHAPEAREA"             "SHAPELEN"
```

```
parcel.data = clean_names(parcel.data)
```

```
names(parcel.data)
```

```
##  [1] "objectid"              "pin_num"             "calc_area"
##  [4] "reid"                  "map_name"            "owner"
##  [7] "addr1"                 "addr2"               "addr3"
## [10] "deed_book"             "deed_page"           "deed_date"
## [13] "deed_acres"            "bldg_val"            "land_val"
## [16] "total_value_assd"      "billclass"
"billing_class_decode"
## [19] "propdesc"              "heatedarea"          "stname"
## [22] "stype"                 "stpre"               "stsuf"
## [25] "stnum"                 "stmisc"              "site_address"
## [28] "full_street_name"      "city"                "city_decode"
## [31] "planning_jurisdiction" "township"            "township_decode"
## [34] "firedist"              "year_built"          "totsalprice"
## [37] "sale_date"             "type_and_use"        "type_use_decode"
## [40] "designstyl"            "design_style_decode" "units"
## [43] "land_class"            "land_class_decode"   "exemptdesc"
## [46] "exemptstat"            "ownership"           "activity"
## [49] "function"              "structure"           "site"
## [52] "totstructs"            "totunits"            "old_parcel_number"
## [55] "zipnum"                "parcel_pk"           "land_code"
## [58] "shapearea"             "shapelen"
```

We can see that the names are in a clean and consistent lower-case format.

3.(B)

Which city has the fewest land parcels in the dataset?

```
fewest_city = parcel.data %>%
    group_by(city_name = city_decode) %>%
    summarise(parcel_count = n()) %>%
```

```
    arrange(parcel_count) %>%
    slice(1)

fewest_city

## # A tibble: 1 × 2
##   city_name parcel_count
##   <chr>            <int>
## 1 CLAYTON              3
```

If we group by the data by the city names, count the parcels per city and sort it in ascending order, we can see that **Clayton** has the fewest land parcels in the dataset with 3 parcels.

3.(C)

Create a tibble that shows the year a parcel was built and the total value, where all parcels are located in Apex and are more than one acre in area. Sort the result in ascending order by year built. Do not print the results.

```
apex_parcels_built = parcel.data %>%
    filter(city_decode == "APEX", calc_area > 1) %>%
    select(year_built, total_value_assd) %>%
    arrange(year_built)
```

We can achieve this by filtering the values of `city_decode` equal to APEX and `calc_area` larger than 1. After this, we can select the year built and the assessed value, and sort it by year in ascending order.

3.(D)

Compute the mean area for each design style.

```
mean_design_area = parcel.data %>%
    group_by(design_style_decode) %>%
    summarise(mean_design_area = mean(calc_area,
                                      na.rm = TRUE))
```

To calculate this, we can group the data by design style and then calculate the mean of the `calc_area` values for each design style.

3.(E)

Which city with at least 1,000 parcels classified as a "Townhouse" had the highest proportion of parcels as "Townhouse"?

```
townhouse_parcels = parcel.data %>%
    group_by(city_decode) %>%
    summarise(total_parcels = n(),
              townhouse_parcels = sum(design_style_decode == "Townhouse"),
              townhouse_proportion = townhouse_parcels / total_parcels) %>%
```

```r
    filter(townhouse_parcels >= 1000) %>%
    arrange(desc(townhouse_proportion)) %>%
    slice(1)

townhouse_parcels
```

```
## # A tibble: 1 × 4
##   city_decode total_parcels townhouse_parcels townhouse_proportion
##   <chr>               <int>             <int>                <dbl>
## 1 MORRISVILLE          7753              2618                0.338
```

To find this, we can first group the data by city names using the `group_by` function. We can then calculate the total parcels, the townhouse parcels and the proportion of the townhouse parcels using the `summarise` function. Finally, we can filter the cities with less than 1000 townhouse parcels, and sort the data in descending order.

We can see that **Morrisville** is the city with the highest proportion of "Townhouse" parcels among the cities with at least 1000 "Townhouse" parcels. It has 7753 total parcels, with 2618 of them being Townhouse parcels, which represents around 33% of the total parcels.

Sources

For 2.(F): How to shift the text on a boxplot? StackOverflow question

For 3.(B) & 3.(E): How to get the first row in `dplyr`? StackOverflow question

For general figures: How to set the figure size in RMarkdown? RMarkdown documentation