PSTAT100 Lab4 Solutions

Lab 4: Smoothing

Introduction

So far, you've encountered a number of visualization techniques for displaying tidy data. In those visualizations, all graphic elements represent the values of a dataset—they are visual displays of actual data.

In general, **smoothing means evening out**. Visualizations of actual data are often irregular—points are distributed widely in scatterplots, line plots are jagged, and bars can be discontinuous. When we look at such visuals, we tend to attempt to look past these irregularities in order to discern patterns. For example, we may want to identify the **overall shape** of a histogram or the general trend in a scatterplot. Smoothing techniques allow us to **even out irregularities in graphical displays of actual data**, helping the eye detect underlying patterns.

For our purposes, smoothing consists of drawing a **line or curve** on top of an existing statistical graphic. From a technical point of view, this amounts to **adding derived geometric objects** to a graphic that have fewer irregularities than the displays of actual data.

In this lab, you'll learn some basic **smoothing techniques**:

- **Kernel density estimation** for histogram smoothing
- **LOESS smoothing** for scatterplots
- **Linear smoothing via regression**

We will implement these techniques using the **ggplot2** package in R.

Sections of the lab:

0. **Introduction to ggplot2 smoothing functions**
1. **Histogram smoothing: Kernel density estimation**
2. **Scatterplot smoothing: LOESS and linear regression smoothing**
3. **Creating polished visualizations**

Goals:

- Get familiar with **ggplot2** for creating smooth visualizations.
- Construct **handmade** histograms using density estimation.
- Implement **kernel density estimation** using `geom_density()`.
- Implement **LOESS smoothing** using `geom_smooth(method = "loess")`.
- Implement **linear regression smoothing** using `geom_smooth(method = "lm")`.

Dataset:

We'll use the same data as last week to stick to a familiar example:

```r
## ensure that the .csv file is saved in the same folder as this .qmd file
data <- read.csv("lab4-data.csv")
head(data)
```

```
  Country.Name Year Life.Expectancy Male.Life.Expectancy Female.Life.Expectancy
1  Afghanistan 2019            63.2                 63.3                   63.2
2  Afghanistan 2015            61.7                 61.0                   62.3
3  Afghanistan 2010            59.9                 59.6                   60.3
4      Albania 2019            78.0                 76.3                   79.9
5      Albania 2015            77.8                 76.1                   79.7
6      Albania 2010            76.2                 74.2                   78.3
  GDP.per.capita region      sub.region Population
1       507.1034   Asia   Southern Asia   38041754
2       578.4664   Asia   Southern Asia   34413603
3       543.3030   Asia   Southern Asia   29185507
4      5353.2449 Europe Southern Europe    2854191
5      3952.8012 Europe Southern Europe    2880703
6      4094.3503 Europe Southern Europe    2913021
```

```r
str(data)
```

```
'data.frame':	620 obs. of  9 variables:
 $ Country.Name          : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Albania" ...
 $ Year                  : int  2019 2015 2010 2019 2015 2010 2000 2019 2015 2010 ...
 $ Life.Expectancy       : num  63.2 61.7 59.9 78 77.8 76.2 73.5 77.1 76.5 75.9 ...
 $ Male.Life.Expectancy  : num  63.3 61 59.6 76.3 76.1 74.2 70.7 76.2 75.6 75 ...
 $ Female.Life.Expectancy: num  63.2 62.3 60.3 79.9 79.7 78.3 76.9 78.1 77.5 76.8 ...
 $ GDP.per.capita        : num  507 578 543 5353 3953 ...
 $ region                : chr  "Asia" "Asia" "Asia" "Europe" ...
 $ sub.region            : chr  "Southern Asia" "Southern Asia" "Southern Asia" "Southern Eu
 $ Population            : num  38041754 34413603 29185507 2854191 2880703 ...
```

Data Transformations in ggplot2

In `ggplot2`, data transformations such as **filtering**, **binning**, and **smoothing** can be performed before or within the plotting commands. These transformations help in pre-processing the dataset and making visualizations clearer.

In this section, we will focus on:

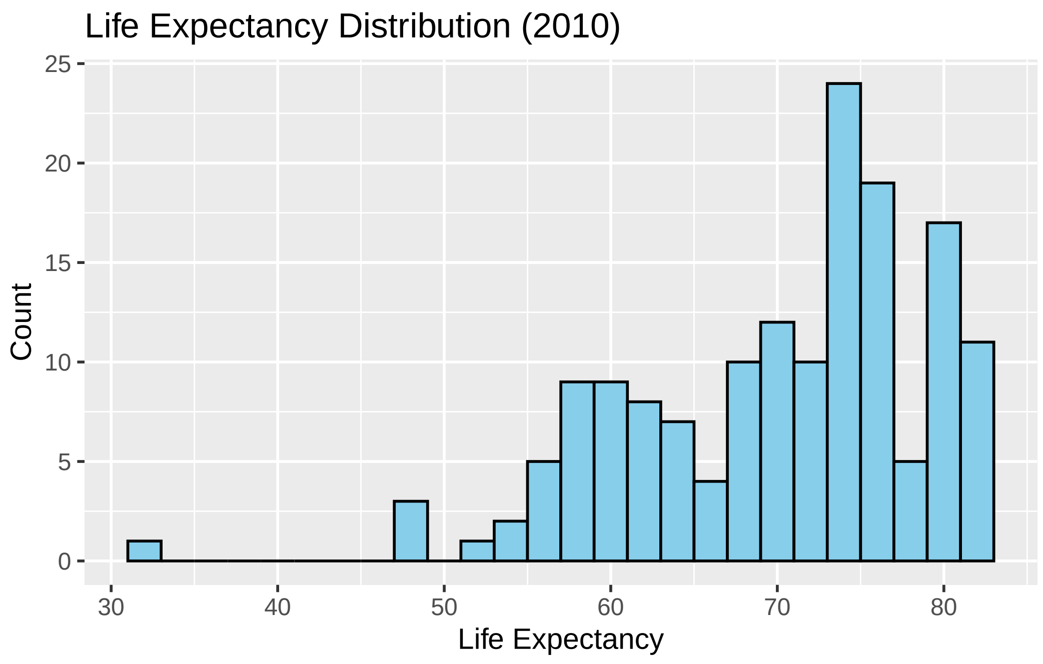
- Filtering data for visualization.
- Creating histograms with **binned data**.
- Using functions in `ggplot2` library to streamline transformations.

Filtering Data

Last week, you saw how to create histograms. As a refresher, to make a histogram of life expectancies across the globe in **2010**, we can **filter** the dataset and then plot it using the following commands:

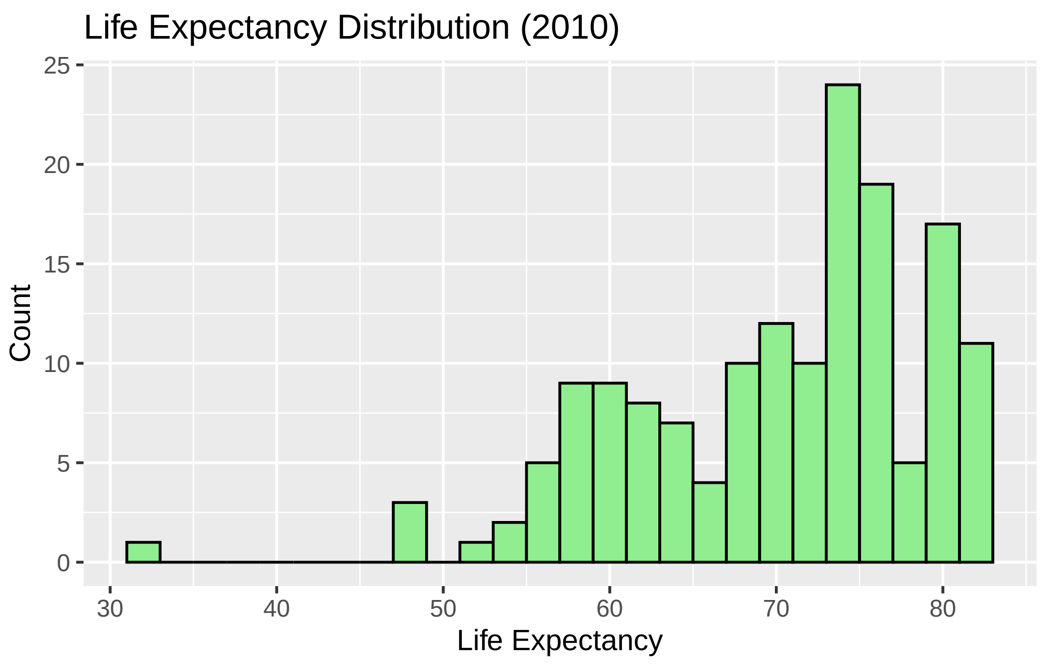
```r
# Load necessary library
library(ggplot2)
library(dplyr)  # For data manipulation

# Filter data for the year 2010
data2010 <- data %>% filter(Year == 2010)
```

```r
# Plot histogram of Life Expectancy in 2010
ggplot(data2010, aes(x = Life.Expectancy)) +
  geom_histogram(binwidth = 2, fill = "skyblue", color = "black") +
  labs(title = "Life Expectancy Distribution (2010)",
       x = "Life Expectancy",
       y = "Count")
```



Alternatively, we can **filter within the ggplot command** to avoid creating an intermediate dataset:

```r
# Filtering within the plotting command
ggplot(data %>% filter(Year == 2010), aes(x = Life.Expectancy)) +
  geom_histogram(binwidth = 2, fill = "lightgreen", color = "black") +
  labs(title = "Life Expectancy Distribution (2010)",
       x = "Life Expectancy",
       y = "Count")
```

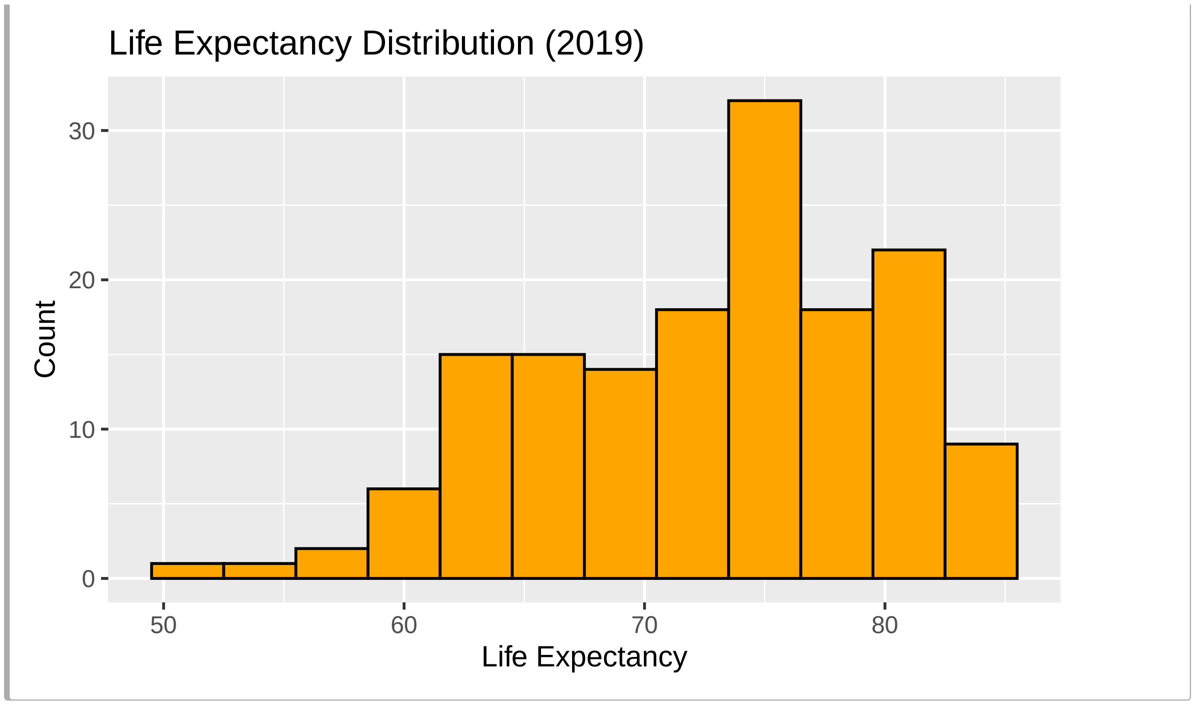


Question 1:

Construct a histogram of life expectancies across the globe **in 2019** by filtering the dataset **within** the ggplot() function. Use a **bin size of 3** years instead of 2.

YOUR ANSWER:

```
ggplot(data %>% filter(Year == 2019), aes(x = Life.Expectancy)) +
  geom_histogram(binwidth = 3, fill = "orange", color = "black") +
  labs(title = "Life Expectancy Distribution (2019)",
       x = "Life Expectancy",
       y = "Count")
```



Counting of Observations in Each Bin

In `ggplot2`, the process of counting how many observations fall into each bin is handled **automatically** when creating histograms. This aggregation step **reduces the number of values** by summarizing raw data into bins.

Instead of manually computing bin counts, we can **extract the aggregated bin data** directly from `ggplot2` using `ggplot_build()`.

Extracting Binned Data with `ggplot_build()`

The codes below give us a dataframe containing the **bin midpoints** (`x`) and **counts** (`count`), which we can use for further analysis.

```r
# Filter data for the year 2010
data2010 <- data %>% filter(Year == 2010)

# Create histogram using stat_bin()
p <- ggplot(data2010, aes(x = Life.Expectancy)) +
  stat_bin(binwidth = 2, aes(y = ..count..))

# Extract binned data
binned_data <- ggplot_build(p)$data[[1]]

# Display extracted bin counts
head(binned_data[,c("x","count")])
```

```
   x count
1 32     1
2 34     0
3 36     0
4 38     0
5 40     0
6 42     0
```

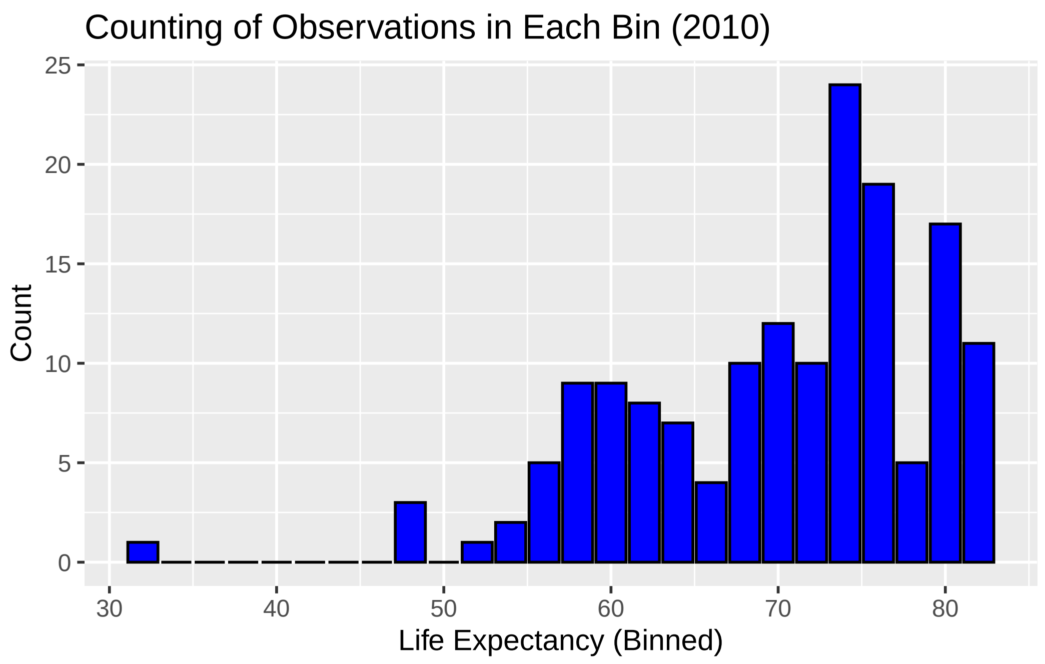
```
tail(binned_data[,c("x","count")])
```

```
    x count
21 72    10
22 74    24
23 76    19
24 78     5
25 80    17
26 82    11
```

Plot the Extracted Aggregated Data

Now that we have extracted the bin counts from `ggplot_build()`, we can plot the **manually** aggregated data using `geom_bar()`.

```
# Plot aggregated data manually
ggplot(binned_data, aes(x = x, y = count)) +
  geom_bar(stat = "identity", fill = "blue", color = "black") +
  labs(title = "Counting of Observations in Each Bin (2010)",
       x = "Life Expectancy (Binned)",
       y = "Count")
```



Question 2:

Modify the code above to **extract** the binned data (counts of observations and midpoints for each bin) and **plot** the aggregated data (using `geom_bar()`) for **2015**, using a **bin size of 3 years**.

YOUR ANSWER:

```
data2019 <- data %>% filter(Year == 2019)

p_2019 <- ggplot(data2019, aes(x = Life.Expectancy)) +
  stat_bin(binwidth = 3, aes(y = ..count..))

binned_data_2019 <- ggplot_build(p_2019)$data[[1]]

# Display extracted bin counts
head(binned_data_2019[,c("x","count")])
```

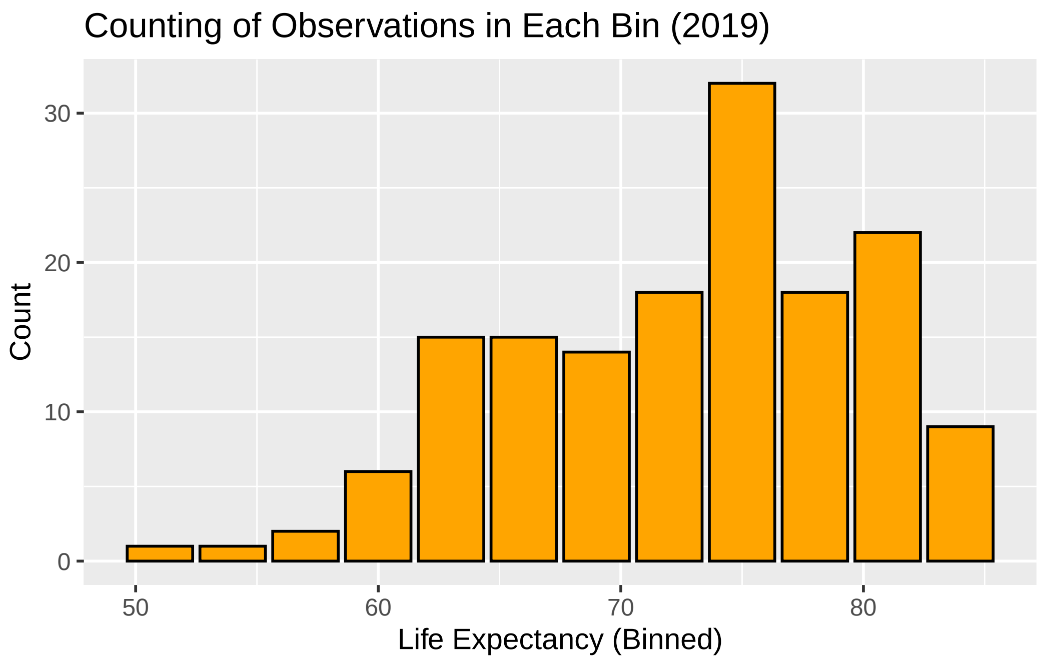
```
   x count
1 51     1
2 54     1
3 57     2
```

```
4 60     6
5 63    15
6 66    15
```

```
tail(binned_data_2019[,c("x","count")])
```

```
    x count
7  69    14
8  72    18
9  75    32
10 78    18
11 81    22
12 84     9
```

```
ggplot(binned_data_2019, aes(x = x, y = count)) +
  geom_bar(stat = "identity", fill = "orange", color = "black") +
  labs(title = "Counting of Observations in Each Bin (2019)",
       x = "Life Expectancy (Binned)",
       y = "Count")
```



Converting Count Scale to Density Scale in ggplot2

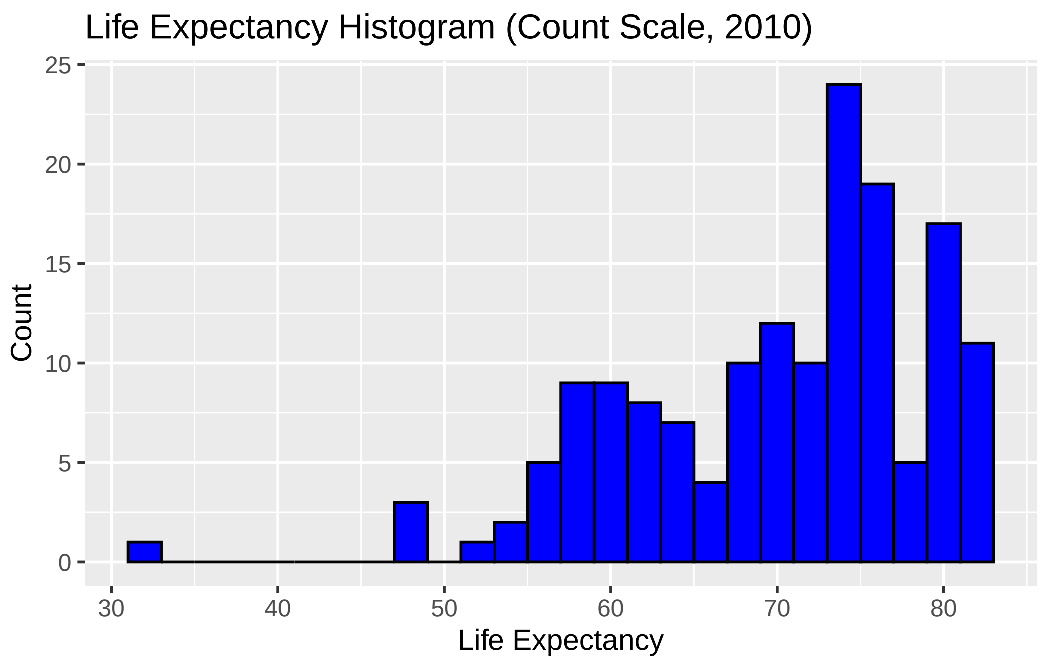
By default, histograms in `ggplot2` are displayed using the **count scale**, where the y-axis represents **the number of observations in each bin**.

However, we can also display histograms using the **density scale**, where the y-axis represents **proportions of the total bar area**. This ensures that the total area under the histogram **sums to 1**, making it easier to compare distributions across different datasets or sample sizes.

Histogram on Count Scale

First, let's plot a **regular** histogram where the y-axis represents the **count** of observations per bin.

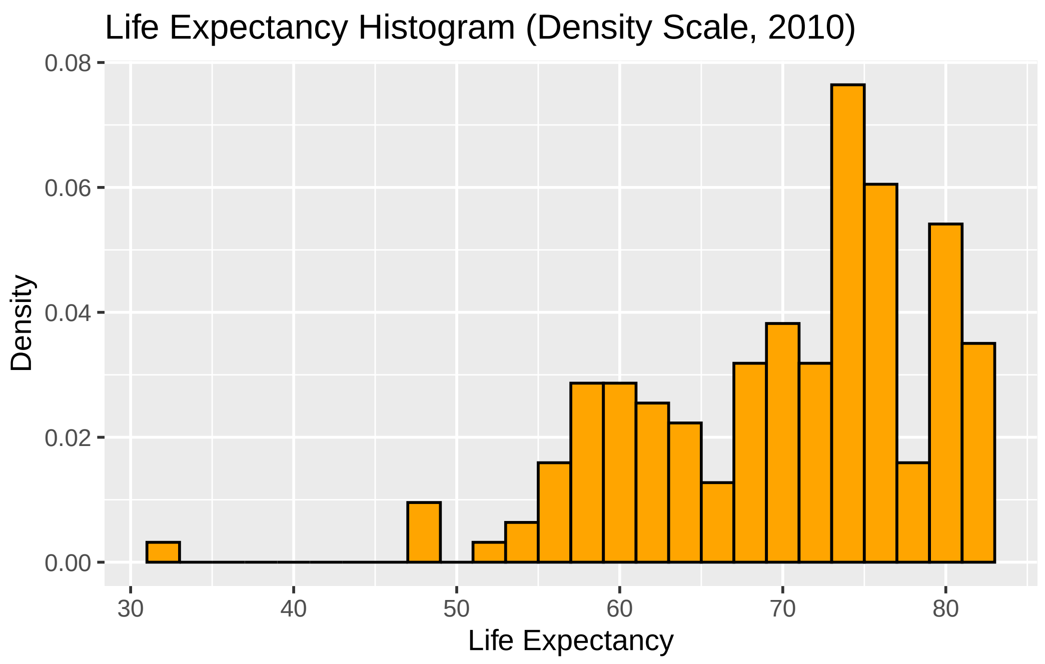
```r
# Filter data for the year 2010
data2010 <- data %>% filter(Year == 2010)

# Histogram using count scale
ggplot(data2010, aes(x = Life.Expectancy)) +
  geom_histogram(binwidth = 2, aes(y = ..count..),
                 fill = "blue", color = "black") +
  labs(title = "Life Expectancy Histogram (Count Scale, 2010)",
       x = "Life Expectancy",
       y = "Count")
```



Histogram on Density Scale

To switch to the density scale, we simply replace `aes(y = ..count..)` with `aes(y = ..density..)` in `geom_histogram()`. This automatically normalizes the histogram, ensuring that the total area sums to 1.

```r
# Histogram using density scale (simpler approach)
ggplot(data2010, aes(x = Life.Expectancy)) +
  geom_histogram(binwidth = 2, aes(y = ..density..),
                 fill = "orange", color = "black") +
  labs(title = "Life Expectancy Histogram (Density Scale, 2010)",
       x = "Life Expectancy",
       y = "Density")
```

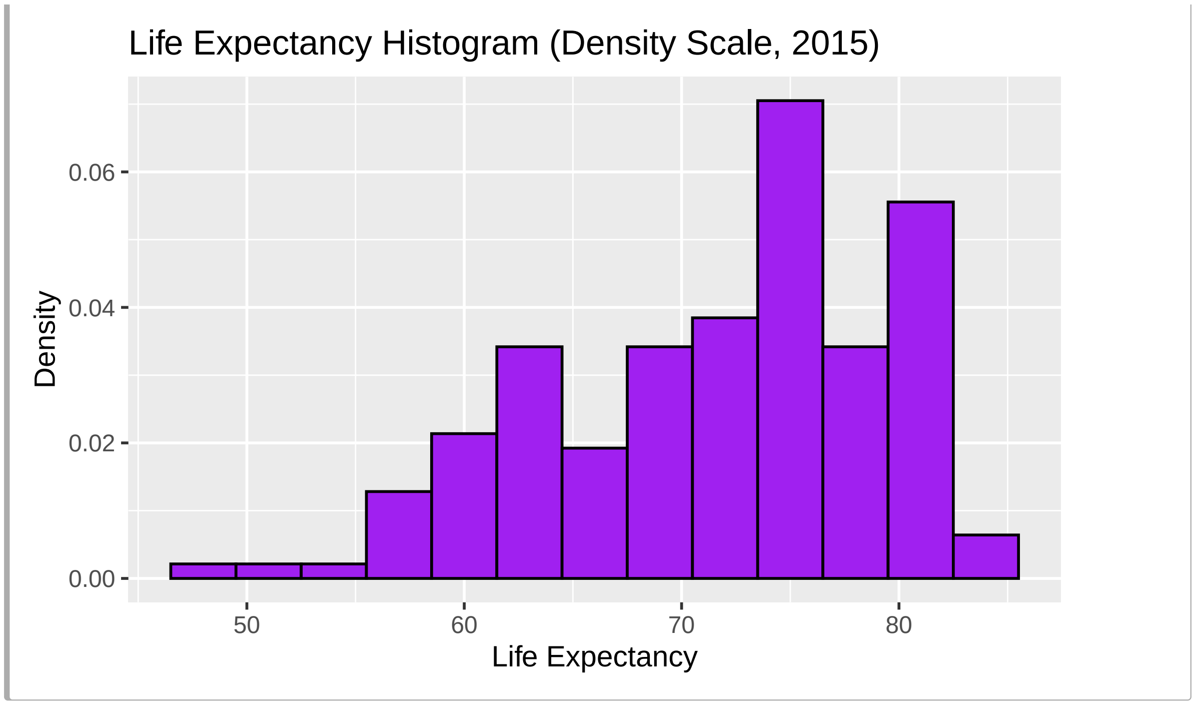


Question 3: Convert to Density Scale for 2015

Modify the code above to convert the histogram for the year **2015** to the **density scale**, using a **bin size of 3 years**.

YOUR ANSWER:

```
data2015 <- data %>% filter(Year == 2015)

ggplot(data2015, aes(x = Life.Expectancy)) +
  geom_histogram(binwidth = 3, aes(y = ..density..),
                 fill = "purple", color = "black") +
  labs(title = "Life Expectancy Histogram (Density Scale, 2015)",
       x = "Life Expectancy",
       y = "Density")
```



Density Estimation in ggplot2

Now that we have explored different transformations, we can apply a **smoothing technique** known as **kernel density estimation**.

Histograms show the **empirical density** by displaying the distribution of values in the sample. A **kernel density estimate** is simply a **smoothed version of the empirical density**. It approximates the **distribution of population values** that the sample comes from.

Why Use Kernel Density Estimation?

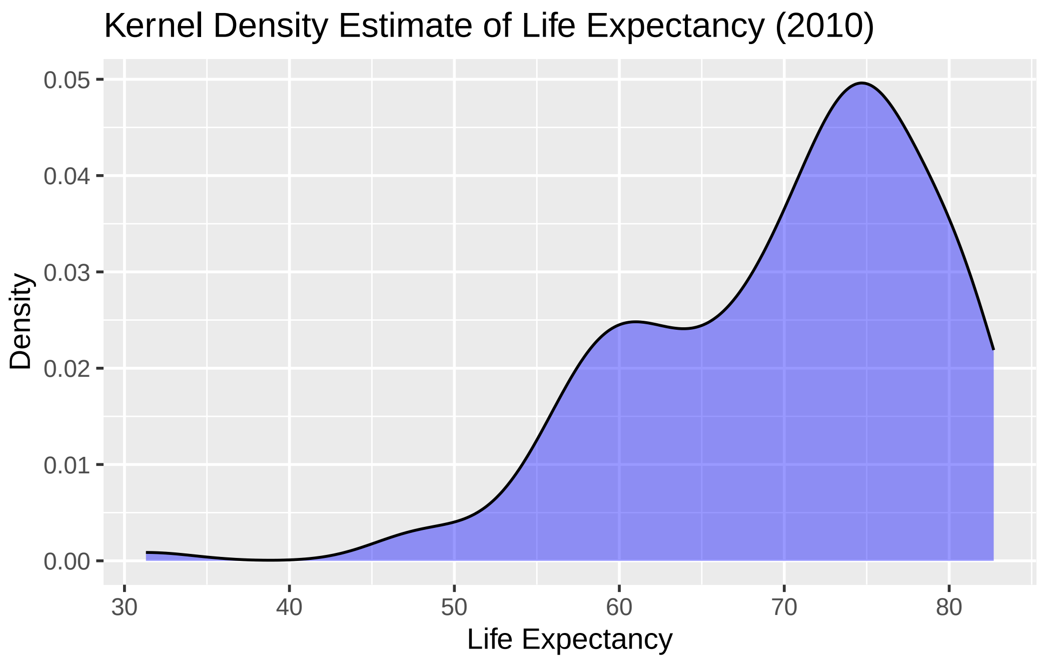
- Helps **visualize** the distribution shape more clearly.
- Allows identification of **multiple peaks** (multi-modality).
- **Smooths out irregularities** in histograms

Instead of manually computing kernel density estimates, we can use the **built-in `geom_density()` function** in `ggplot2`.

Plot Kernel Density Estimate

To generate a density estimate for life expectancies in 2010, we use `geom_density()`.

```r
# Filter data for the year 2010
data2010 <- data %>% filter(Year == 2010)

# Kernel density plot
ggplot(data2010, aes(x = Life.Expectancy)) +
  geom_density(fill = "blue", alpha = 0.4) +
  labs(title = "Kernel Density Estimate of Life Expectancy (2010)",
       x = "Life Expectancy",
       y = "Density")
```

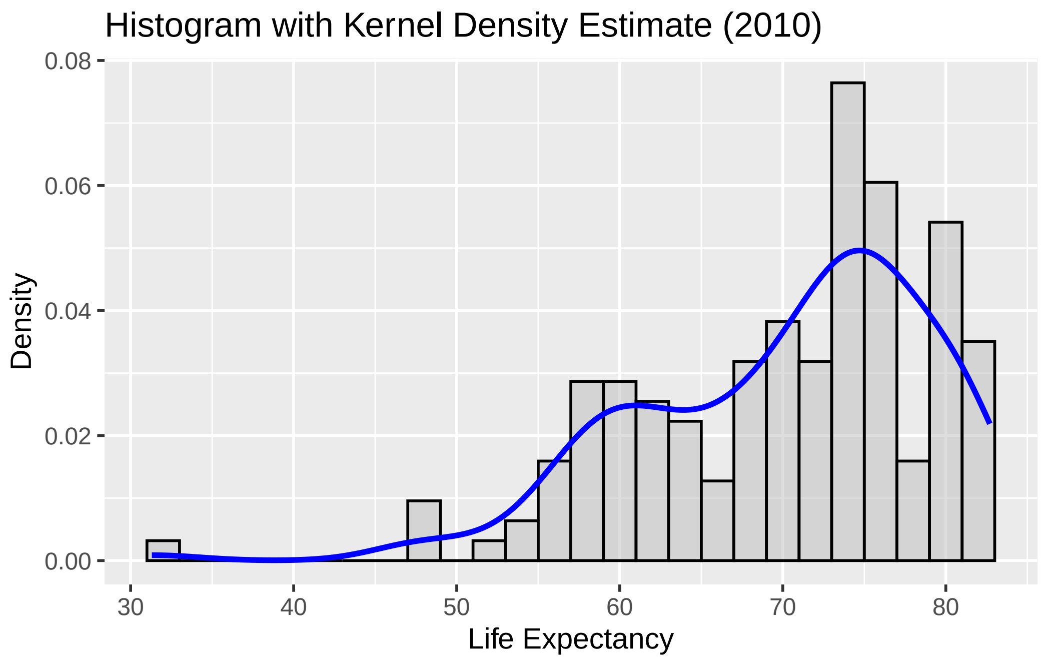


This produces a **smooth curve** representing the estimated probability density of life expectancy values.

Overlay Density Curve on Histogram

We can **overlay** the density estimate on top of the histogram to compare the **empirical distribution (histogram)** and the **smoothed estimate**.

```r
ggplot(data2010, aes(x = Life.Expectancy)) +
  geom_histogram(aes(y = ..density..), binwidth = 2,
                 fill = "gray", color = "black", alpha = 0.5) +
  geom_density(color = "blue", size = 1) +
  labs(title = "Histogram with Kernel Density Estimate (2010)",
       x = "Life Expectancy",
       y = "Density")
```



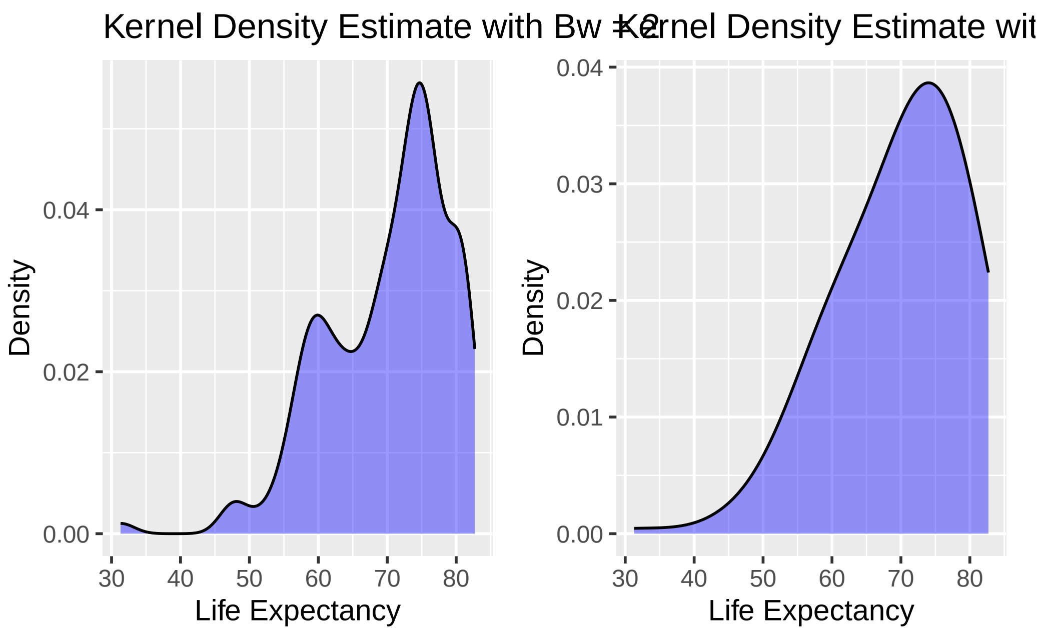
Bandwidth Adjustment

The **bandwidth parameter** controls the amount of **smoothing**.

- **Larger** bandwidth = **Smoother** curve (less detail)
- **Smaller** bandwidth = **Wigglier** curve (More local detail)

We can adjust the bandwidth in `geom_density()` using the `bw` argument

```r
library(patchwork)
p1<- ggplot(data2010, aes(x = Life.Expectancy)) +
  geom_density(fill = "blue", alpha = 0.4, bw = 2) +
  labs(title = "Kernel Density Estimate with Bw = 2",
       x = "Life Expectancy",
       y = "Density")

p2<- ggplot(data2010, aes(x = Life.Expectancy)) +
  geom_density(fill = "blue", alpha = 0.4, bw = 6) +
  labs(title = "Kernel Density Estimate with Bw = 6",
       x = "Life Expectancy",
       y = "Density")

(p1+p2)
```

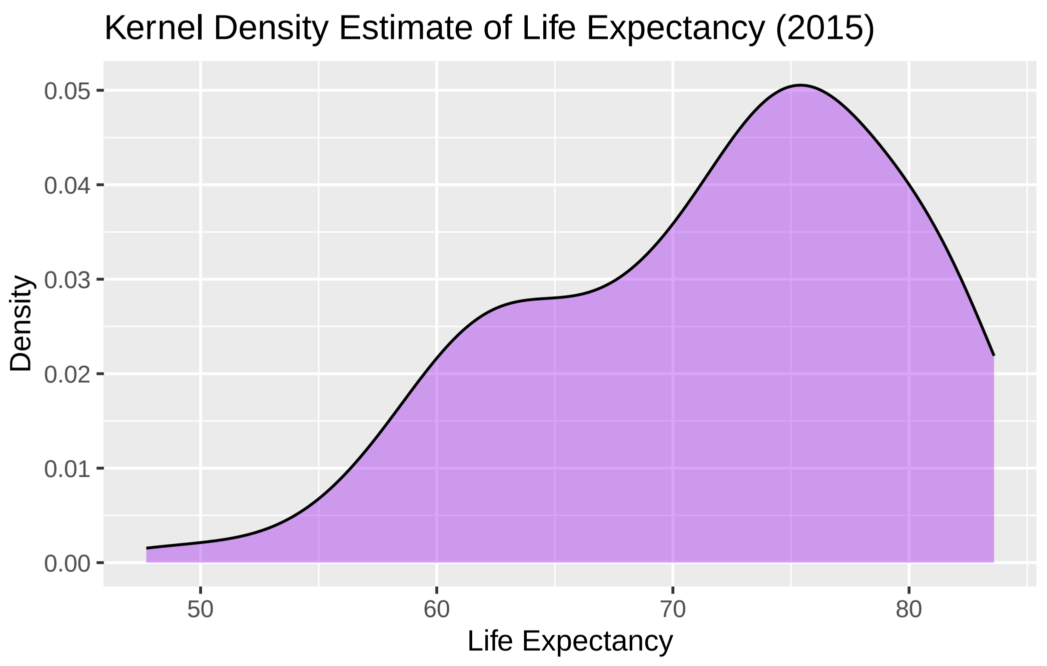


Question 4: Kernel Density Estimation for 2015

Modify the plotting code by **decreasing the bandwidth parameter**.
Try several values, and then choose one that you feel **captures the shape of the distribution well without getting too wiggly**.

YOUR ANSWER:

```r
data2015 <- data %>% filter(Year == 2015)

ggplot(data2015, aes(x = Life.Expectancy)) +
  geom_density(fill = "purple", alpha = 0.4, bw = 3) +
  labs(title = "Kernel Density Estimate of Life Expectancy (2015)",
       x = "Life Expectancy",
       y = "Density")
```



Comparing Distributions

The visual advantage of a **kernel density estimate** is even more apparent when comparing distributions.

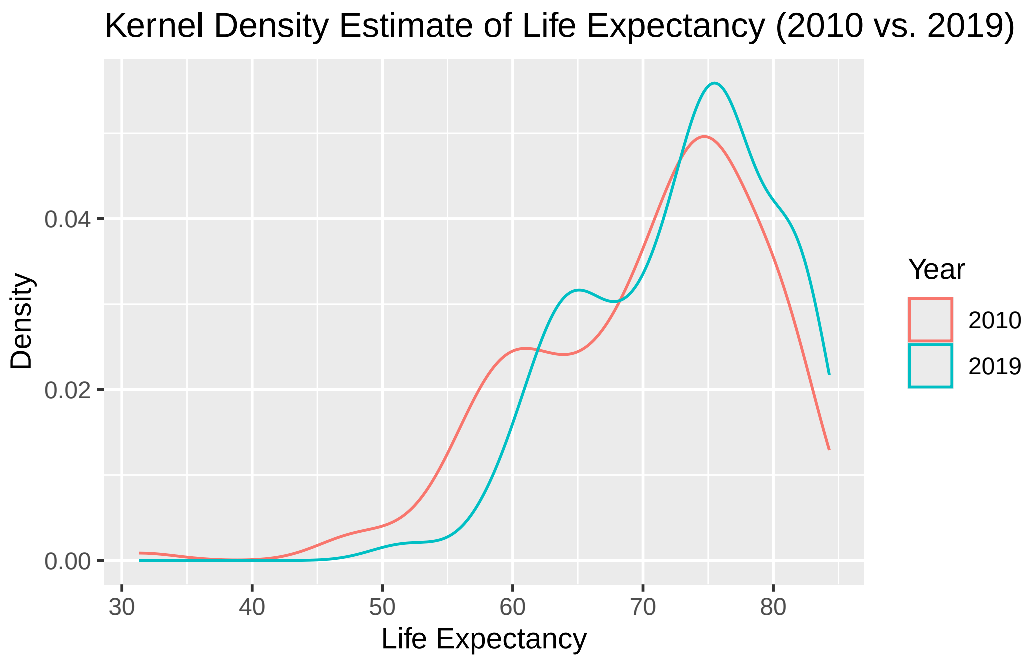
A major task in **exploratory data analysis** is understanding how the distribution of a variable changes over time or across groups. For example, we have already seen that **life expectancy seems to change over time**. We can explore this further by **comparing distributions for different years**.

We can plot **multiple density estimates** on the **same plot** by grouping the data by year.

Compare Kernel Density Estimates for Different Years

We compare the life expectancy distributions for 2010 and 2019 by grouping by Year in `geom_density()`.

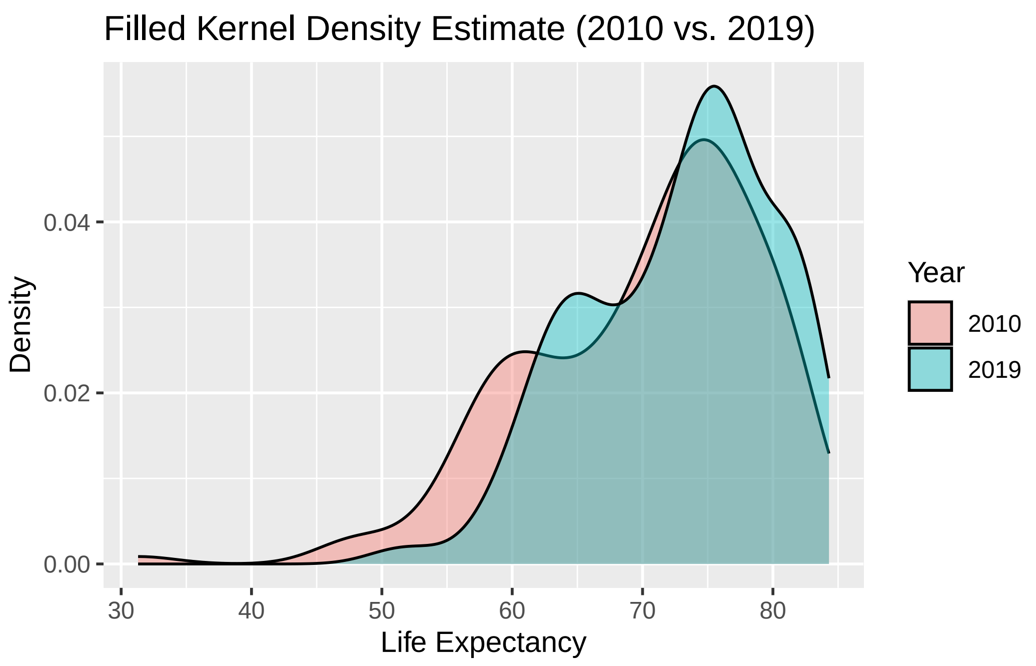
```r
# Filter data for 2010 and 2019
data_filtered <- data %>% filter(Year %in% c(2010, 2019))

# Plot density estimates for both years
ggplot(data_filtered, aes(x = Life.Expectancy, color = as.factor(Year))) +
  geom_density() +
  labs(title = "Kernel Density Estimate of Life Expectancy (2010 vs. 2019)",
       x = "Life Expectancy",
       y = "Density",
       color = "Year")
```



Fill the Area Under the Density Curve

To enhance visibility, we fill the density curves with transparency using `fill = Year` and `alpha` for transparency.

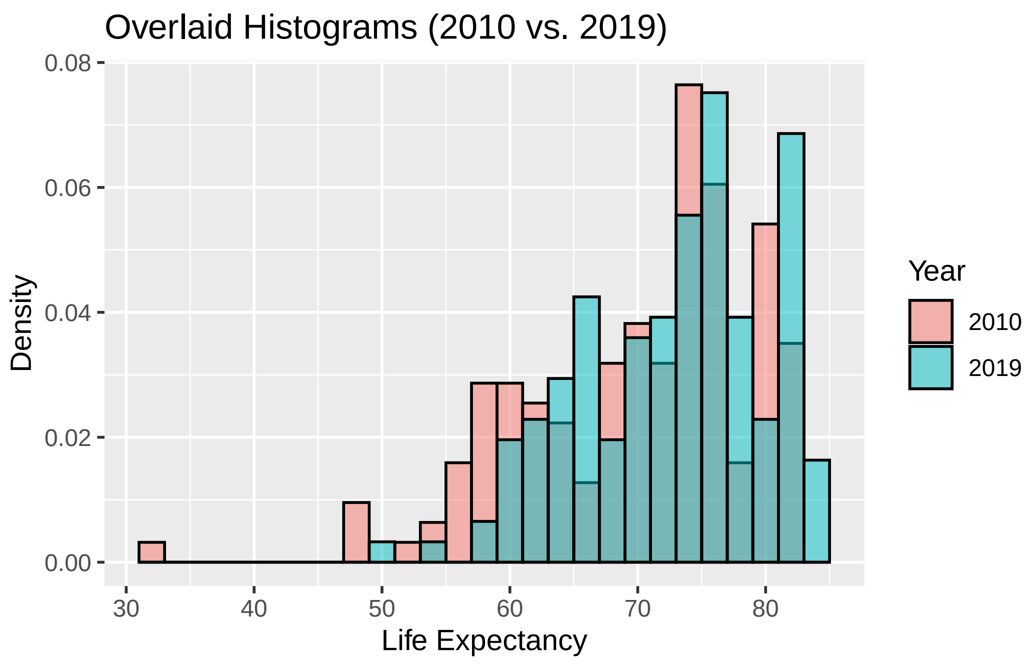
```r
# Density plot with filled areas
ggplot(data_filtered, aes(x = Life.Expectancy, fill = as.factor(Year))) +
  geom_density(alpha = 0.4) +
  labs(title = "Filled Kernel Density Estimate (2010 vs. 2019)",
       x = "Life Expectancy",
       y = "Density",
       fill = "Year")
```



Comparing with Histograms (Less Effective)

Overlaying **histograms** can sometimes be useful, but they can also be harder to interpret compared to density plots.

```
# Overlaid histograms (not recommended due to clutter)
ggplot(data_filtered, aes(x = Life.Expectancy, fill = as.factor(Year))) +
  geom_histogram(aes(y = ..density..), binwidth = 2,
                 alpha = 0.5, position = "identity", color = "black") +
  labs(title = "Overlaid Histograms (2010 vs. 2019)",
       x = "Life Expectancy",
       y = "Density",
       fill = "Year")
```



Question 5: Multiple Density Estimates

Follow the example above to construct a plot showing **separate density estimates of life expectancy** for each **region** in 2010.

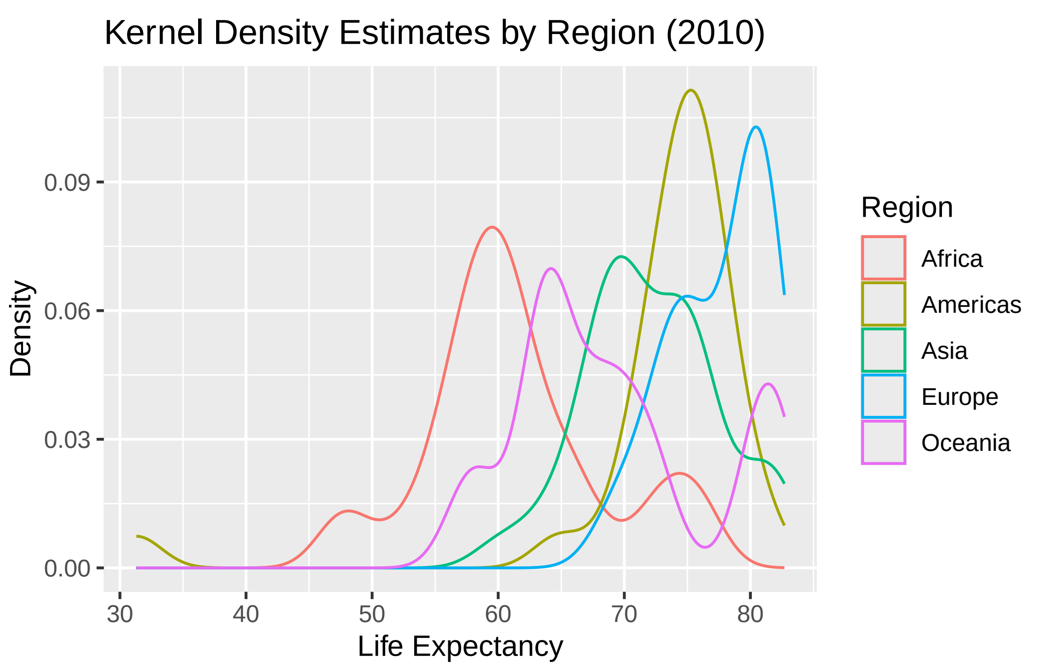
You can choose whether to **fill the area beneath the smooth curves** or not.
Be sure to **adjust the bandwidth** parameter and select a value that makes sense.

Step 1: Compute Density Estimates for Each Region

We filter the dataset for 2010 and plot separate density curves for each region.

YOUR ANSWER:

```r
# Filter data for 2010
data2010 <- data %>% filter(Year == 2010)

# Plot density estimates for different regions
ggplot(data2010, aes(x = Life.Expectancy, color = region)) +
  geom_density(bw = 2) +
  labs(title = "Kernel Density Estimates by Region (2010)",
       x = "Life Expectancy",
       y = "Density",
       color = "Region")
```

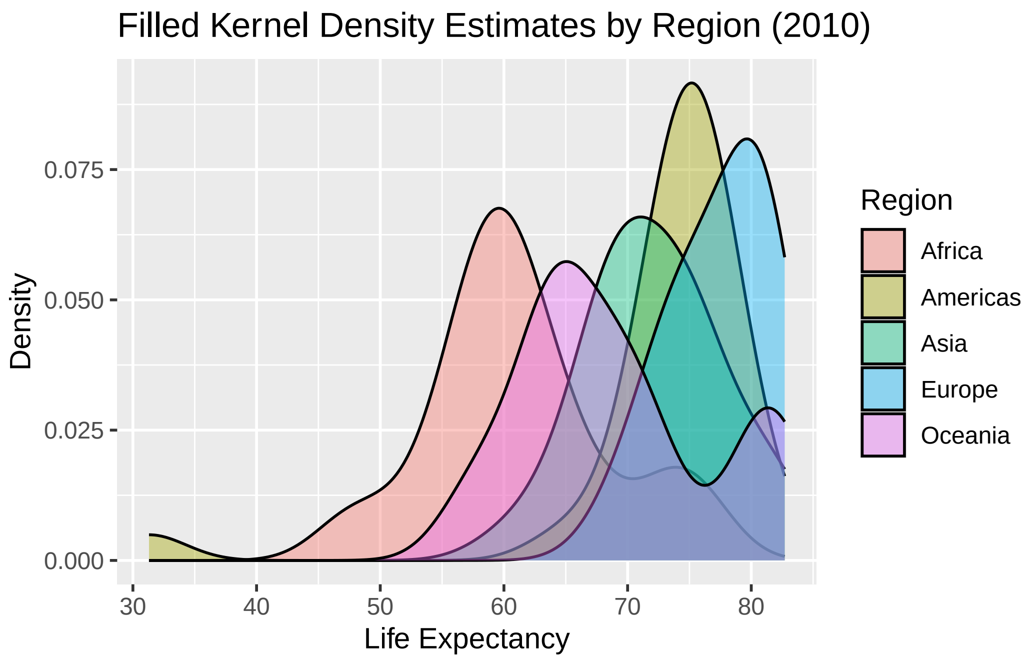


Step 2: Fill the Area Under the Density Curves

To enhance visibility, we can fill the area under the density curves with transparency using `fill = region` and `alpha`.

YOUR ANSWER:

```
# Density plot with filled areas
ggplot(data2010, aes(x = Life.Expectancy, fill = region)) +
  geom_density(alpha = 0.4, bw = 3) +
  labs(title = "Filled Kernel Density Estimates by Region (2010)",
       x = "Life Expectancy",
       y = "Density",
       fill = "Region")
```



Note:
Remember to modify the `bw` parameter in above codes to adjust the smoothness of the density curves.

Question 6: Interpretation

Do the distributions of life expectancies seem to differ by region? If so, what is one difference that you notice? Answer in 1-2 sentences.

YOUR ANSWER:

Yes, the distributions of life expectancies differ by region.
One noticeable difference is that the mode of the distribution of Africa gives the lowest life expectancies, around 50-60 years, while that os Europe has the highest, mostly above

75 years. Additionally, the distribution of life expectancies of Asia displaying more variability. This suggests that life expectancy varies significantly across regions, likely due to differences in healthcare, economic development, and living conditions.

Question 7: Outlier

Notice that **small peak far to the left** in the **Americas** region. That is an **outlier** in life expectancy.

Tasks:

1. **Identify the country** with the **lowest life expectancy** in the Americas in 2010.
2. **Check life expectancies** for that country across different years.
3. **Investigate why life expectancy was so low** in that country for 2010.

Task 1: Find the Country with the Lowest Life Expectancy

Filter for 2010 and Americas, then find the **country** with the **lowest** life expectancy.

YOUR ANSWER:

```r
# Find the country with the lowest life expectancy in the Americas in 2010
lowest_Americas <- data %>%
  filter(Year == 2010, region == "Americas") %>%
  arrange(Life.Expectancy) %>%
  head(1)  # Select the lowest value

# Print the outlier row
lowest_Americas
```

```
  Country.Name Year Life.Expectancy Male.Life.Expectancy Female.Life.Expectancy
1        Haiti 2010            31.3                   28                   35.4
  GDP.per.capita   region                      sub.region Population
1       1172.099 Americas Latin America and the Caribbean    9949322
```

Task 2: Examine Life Expectancies for This Country

Now, check life expectancy **across all years** for the **identified outlier country**.

YOUR ANSWER:

```r
# Get the country name
outlier_country_name <- lowest_Americas$Country.Name

# Filter data for this country across all years
outlier_country <- data %>%
  filter(Country.Name == outlier_country_name)

# Print the life expectancy trend
outlier_country
```

```
  Country.Name Year Life.Expectancy Male.Life.Expectancy Female.Life.Expectancy
1        Haiti 2019            64.1                 63.3                   64.8
2        Haiti 2015            62.6                 62.1                   63.1
3        Haiti 2010            31.3                 28.0                   35.4
4        Haiti 2000            57.0                 57.0                   57.2
  GDP.per.capita   region                      sub.region Population
1       1272.491 Americas Latin America and the Caribbean   11263077
2       1389.120 Americas Latin America and the Caribbean   10695542
3       1172.099 Americas Latin America and the Caribbean    9949322
4        811.534 Americas Latin America and the Caribbean    8463806
```

Task 3: Interpretation

What happened in 2010?

Can you explain why life expectancy was so low for this country in 2010? Hint: If you're unsure, search for major events in this country in 2010 (e.g., conflicts, disasters, epidemics).

YOUR ANSWER:

The country with the lowest life expectancy in the **Americas in 2010** was **Haiti**.
Its life expectancy dropped sharply due to the **devastating earthquake in January 2010**, which caused a major humanitarian crisis, widespread destruction, and a high death toll.

Scatterplot Smoothing

In this section, you'll explore two techniques for **smoothing scatterplots**:

1. **LOESS (Locally Weighted Scatterplot Smoothing)** – Produces a **curved** trend line.

2. **Linear Regression Smoothing** – Produces a **straight-line** trend.

Recall that in lab 3, we illustrated a variable transformation to reduce **skewness in GDP per capita**, That is, we will apply a **log transformation** to `GDP.per.capita`.

```r
data_mod1 <- data %>%
  mutate(log_GDP_per_capita = log(GDP.per.capita))
```

LOESS or Local Polynomial Regression Fitting

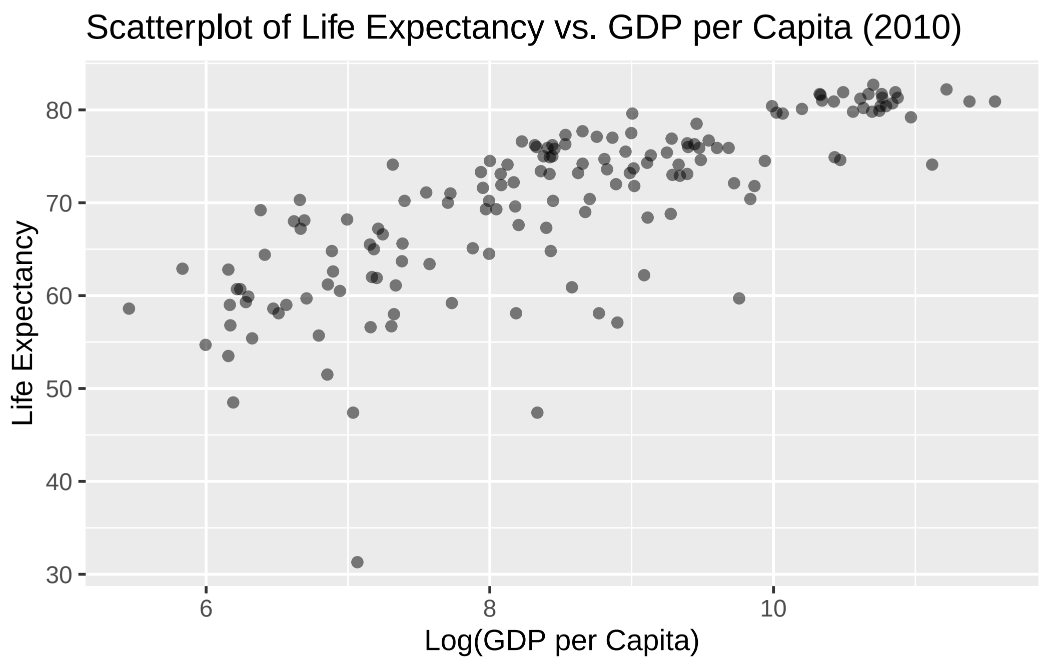
Locally weighted scatterplot smoothing (LOESS) is a flexible technique for visualizing trends in scatterplots. It is conceptually **similar to kernel density estimation**, but applied to **scatterplots** instead of distributions.

To illustrate, let's examine the relationship between **GDP per capita** and **life expectancy** for the year 2010.

Step 1: Scatterplot of GDP per Capita vs. Life Expectancy

First, we create a basic **scatterplot**.

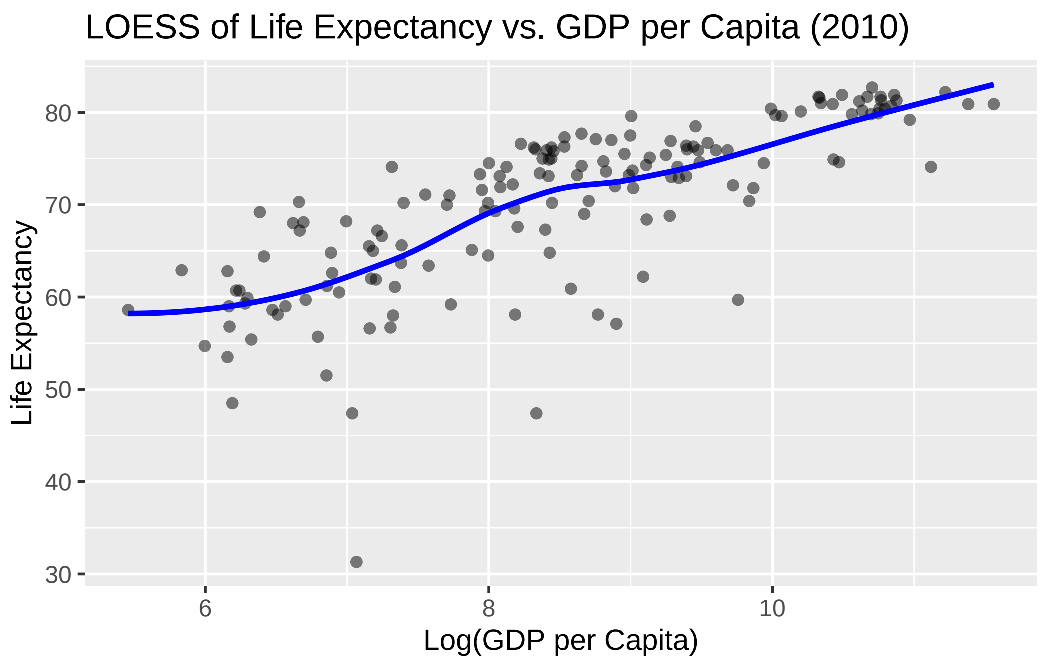
```r
# Filter data for 2010
data_2010 <- data_mod1 %>% filter(Year == 2010)

# Scatterplot
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy)) +
  geom_point(alpha = 0.5) +
  labs(title = "Scatterplot of Life Expectancy vs. GDP per Capita (2010)",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy")
```



Step 2: Add LOESS

To add a **LOESS curve**, we use `geom_smooth(method = "loess")`.

```
# Scatterplot with LOESS smoothing
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "loess", color = "blue", se = FALSE) +
  labs(title = "LOESS of Life Expectancy vs. GDP per Capita (2010)",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy")
```

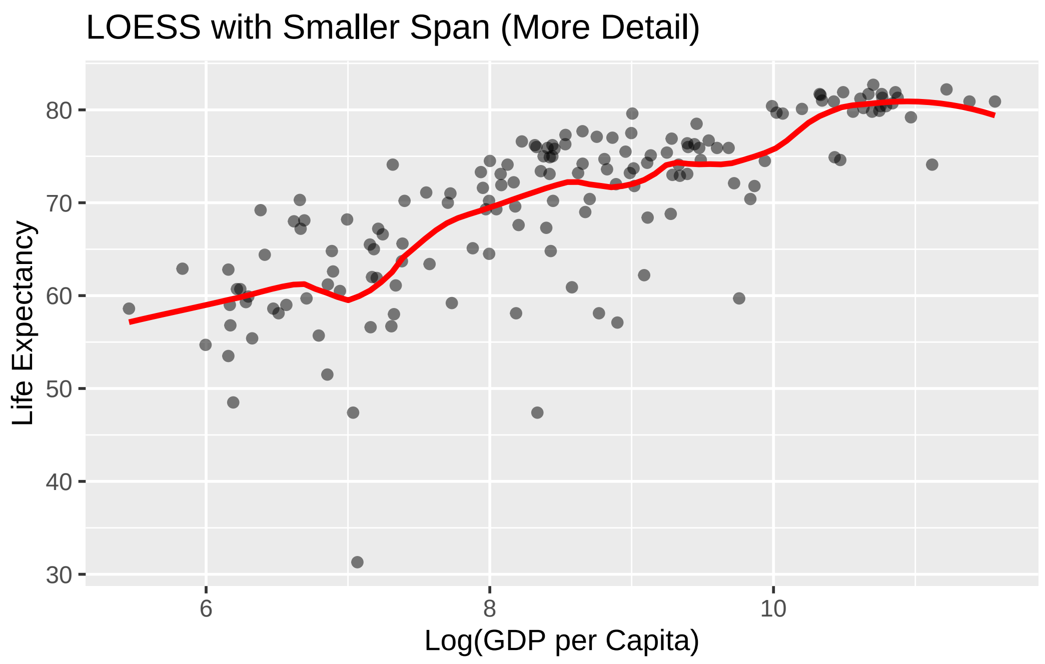


Step 3: Adjusting LOESS Span

The **span parameter** controls the degree of smoothing:

- **Larger** span → **Smoother** curve (more general trend)
- **Smaller** span → **Wigglier** curve (more localized trends)

```
# Adjusting LOESS span
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "loess", color = "red", se = FALSE,
              span = 0.3) + # More detailed fit
  labs(title = "LOESS with Smaller Span (More Detail)",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy")
```



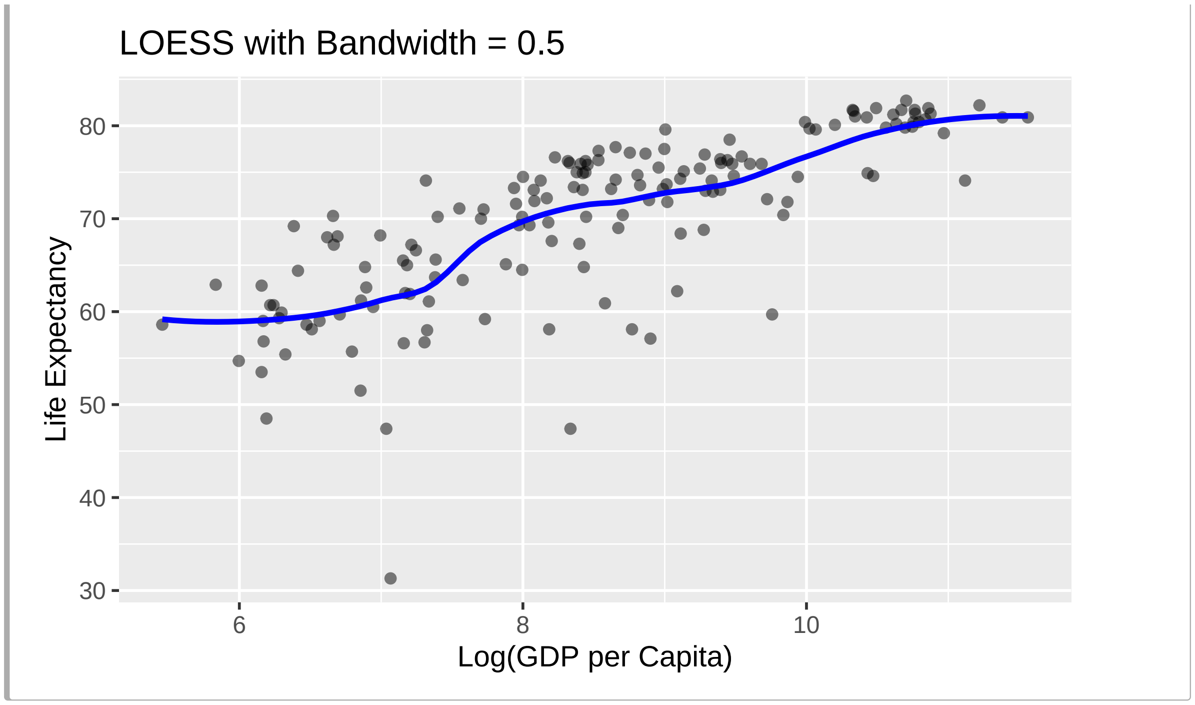
Question 8: LOESS Bandwidth Selection

Tinker with the **bandwidth parameter** (`span` in ggplot2) to see its effect on the **smoothness of the LOESS curve**.
Choose a value that produces a trend line that **best represents the data**

Modify the `span` parameter in `geom_smooth()` to change the smoothness of the LOESS curve.

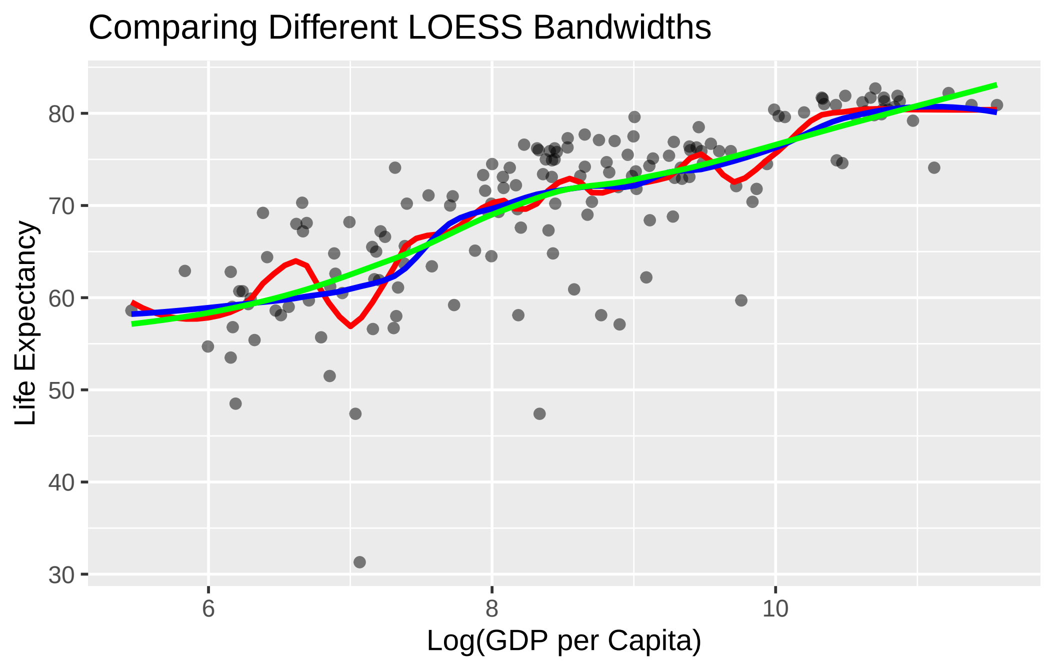
YOUR ANSWER:

```r
# Filter data for 2010
data_2010 <- data_mod1 %>% filter(Year == 2010)

# Scatterplot with LOESS (adjust span)
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "loess", span = 0.5, color = "blue", se = FALSE) +
  labs(title = "LOESS with Bandwidth = 0.5",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy")
```



Try different `span` values to see how the LOESS curve changes. Draw LOESS curves with different `span` values in the same scatterplot.

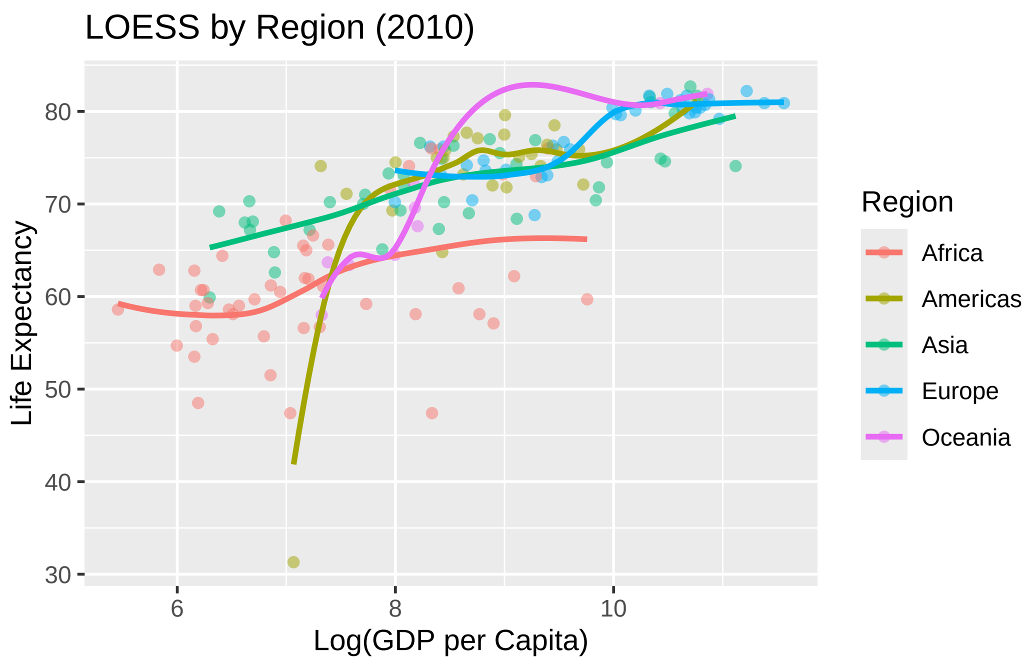
```r
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "loess", span = 0.2, color = "red", se = FALSE) +  # More localized fi
  geom_smooth(method = "loess", span = 0.4, color = "blue", se = FALSE) +  # Medium localized
  geom_smooth(method = "loess", span = 0.8, color = "green", se = FALSE) + # Smoother fit
  labs(title = "Comparing Different LOESS Bandwidths",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy")
```



Groupwise LOESS (Separate Curves by Region)

To compute LOESS trends for each region separately, set `group = region`.

```r
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy,
                      color = region)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "loess", span = 0.7, se = FALSE) +
  labs(title = "LOESS by Region (2010)",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy",
       color = "Region")
```



Regression Smoothing

Later in the course, you will learn about **linear regression**, but here we introduce it as a **visualization technique**.

- Like LOESS, **linear regression** smooths scatterplot data.
- The key difference: **Linear regression produces a straight line**, while **LOESS allows curvature**.

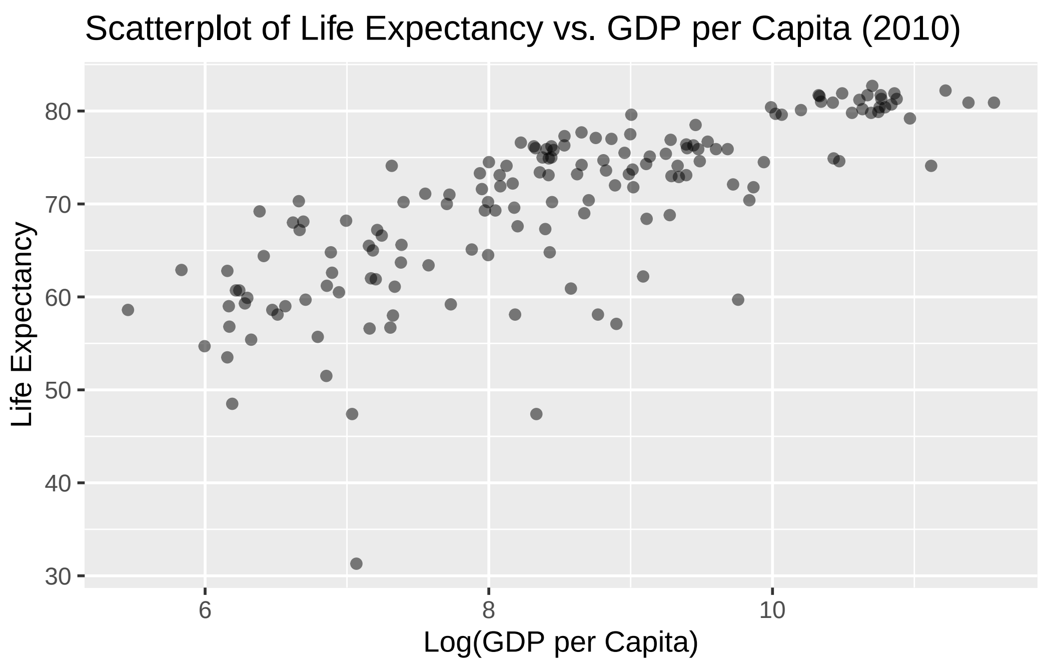
In cases where **LOESS curves are almost linear**, a **regression line** may be the **cleaner choice**.

Step 1: Scatterplot of GDP per Capita vs. Life Expectancy

First, we create a basic scatterplot.

```
# Filter data for 2010
data_2010 <- data_mod1 %>% filter(Year == 2010)

# Scatterplot
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy)) +
```

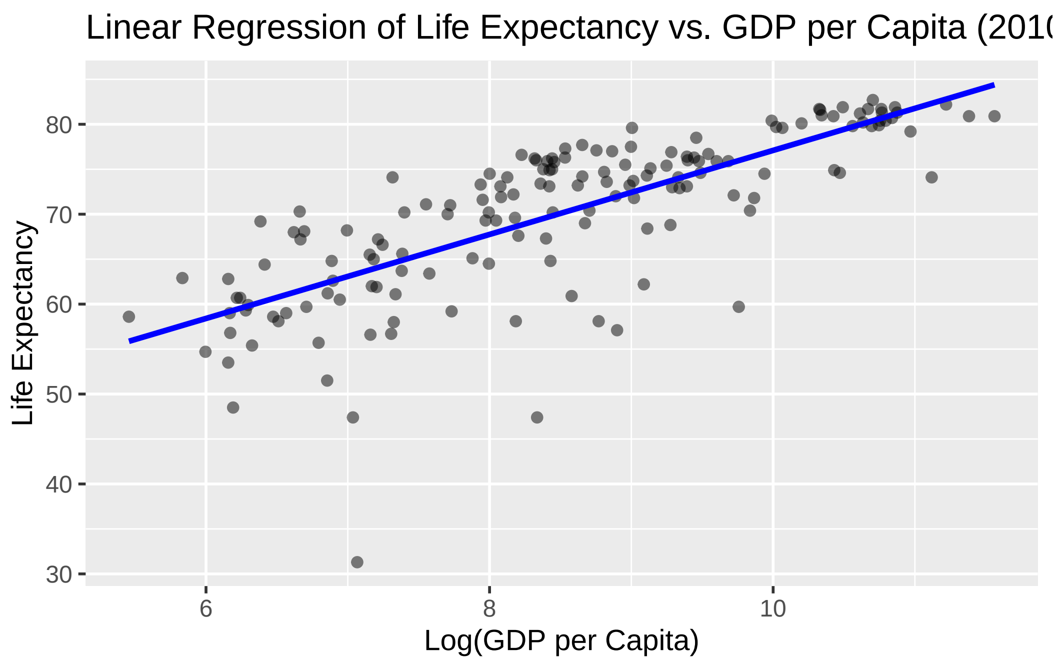
```r
  geom_point(alpha = 0.5) +
  labs(title = "Scatterplot of Life Expectancy vs. GDP per Capita (2010)",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy")
```



Step 2: Adding a Linear Regression Line

To add a linear regression trend, we use `geom_smooth(method = "lm")`.

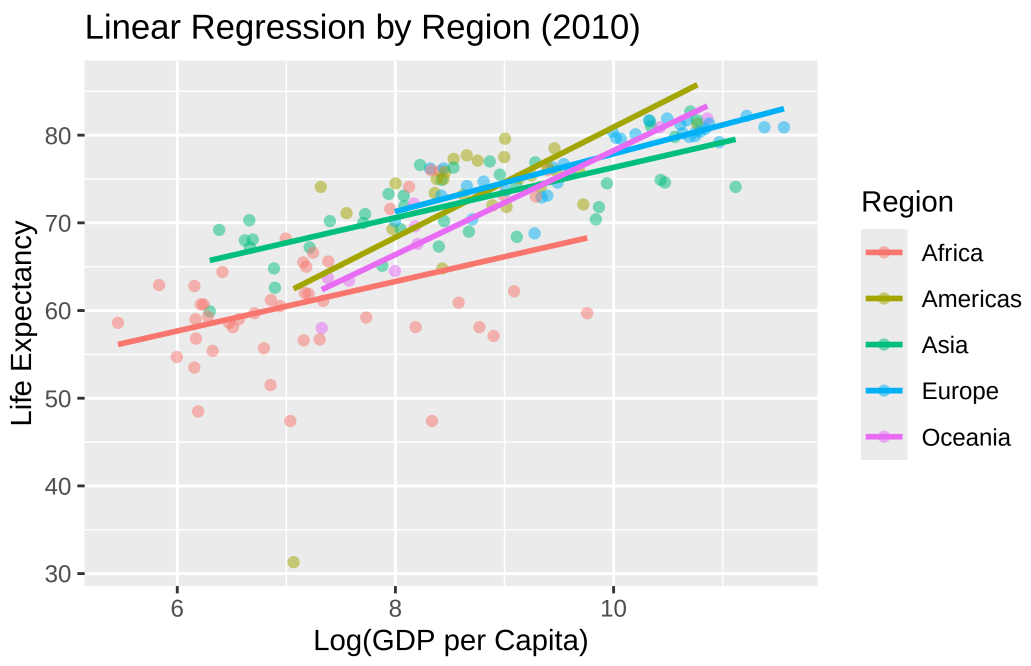
```r
# Scatterplot with linear regression smoothing
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", color = "blue", se = FALSE) +
  labs(title = "Linear Regression of Life Expectancy vs. GDP per Capita (2010)",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy")
```



Step 3: Groupwise Regression by Region

To compute separate regression lines for each region, we set `color = region`.

```
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy,
                      color = region)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Linear Regression by Region (2010)",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy",
       color = "Region")
```

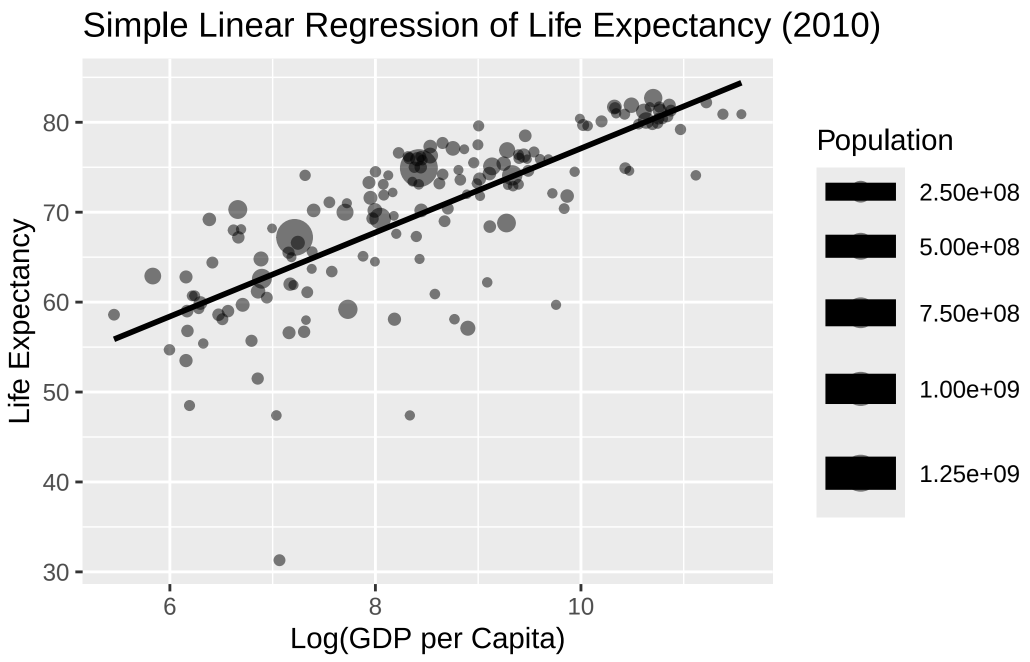


Question 9: Simple Regression Line

Modify the code to construct a scatterplot of life expectancy vs. log GDP per capita (2010)

- The points should be **sized by population**.
- **Remove color grouping** by region.
- Overlay a single **linear regression line**.

```r
ggplot(data_2010, aes(x = log_GDP_per_capita, y = Life.Expectancy,
                      size = Population)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", color = "black", se = FALSE) +
  labs(title = "Simple Linear Regression of Life Expectancy (2010)",
       x = "Log(GDP per Capita)",
       y = "Life Expectancy",
       size = "Population")
```



Submission

1. Rename and save the notebook.
2. Restart the kernel and run all cells. (**CAUTION**: if your notebook is not saved, you will lose your work.)
3. Carefully look through your notebook and verify that all computations execute correctly. You should see **no errors**; if there are any errors, make sure to correct them before you submit the notebook.
4. Download the notebook as an `.qmd` file. This is your backup copy.
5. Export the notebook as PDF and upload to Canvas.