Data Visualization 02

Dr Saufi

2024-05-30

Contents

1 Overview of Graphics Packages in R

There are several graphics packages in R designed for different purposes:

1.1 Popular General Graphics Packages

1. `graphics`: A base R package for basic plotting.
2. `ggplot2`: A user-contributed package by Hadley Wickham, based on the grammar of graphics.
3. `lattice`: Another user-contributed package for advanced plotting.

1.2 Specialized Graphics Packages

1. `survminer::ggsurvplot`: For survival analysis plots.
2. `sjPlot`: For data visualization tailored to social sciences.

1.3 Introduction to ggplot2

`ggplot2` is highlighted for its elegance, ease of use, and versatility: - Implements the grammar of graphics concept, making the learning process faster and facilitating the creation of complex graphics. - It combines the best features of base and lattice graphics without their drawbacks.

1.4 Key Components of ggplot2

- Start with: `ggplot()`
- Specify data: `data = X`
- Define variables: `aes(x = , y = )`
- Choose graph type: `geom_histogram()`, `geom_point()`, etc.

1.5 Questions to Ask Before Making Graphs

Before creating a graph, consider:

- Which variable(s) to plot?
- The type of variable(s) (factor or numerical)
- The number of variables to plot together (single, two, or three variables)

1.6 Preparation for Data Analysis

1.6.1 Set a New Project Directory

Starting a new project in RStudio is recommended for clean and organized analysis:

1. Steps to create a new project
2. Go to File -> Click New Project -> New Directory -> New Project -> Create New Project

1.6.2 Read Data

Common data formats include CSV, Excel, SPSS, Stata, and SAS files. Here's an example of reading a CSV file:

```
# Read a CSV file into R
mydata <- read.csv('HousingData.csv')
summary(mydata)
```

```
      CRIM                ZN              INDUS            CHAS        
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
 1st Qu.: 0.08190   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
 Median : 0.25372   Median :  0.00   Median : 9.69   Median :0.00000  
 Mean   : 3.61187   Mean   : 11.21   Mean   :11.08   Mean   :0.06996  
 3rd Qu.: 3.56026   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
 Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
 NA's   :20         NA's   :20       NA's   :20      NA's   :20       
      NOX               RM              AGE              DIS        
 Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
 1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.17   1st Qu.: 2.100  
 Median :0.5380   Median :6.208   Median : 76.80   Median : 3.207  
 Mean   :0.5547   Mean   :6.285   Mean   : 68.52   Mean   : 3.795  
 3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 93.97   3rd Qu.: 5.188  
 Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
                                  NA's   :20                       
      RAD              TAX            PTRATIO             B        
 Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
 1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
 Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
 Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
 3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
 Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  

     LSTAT            MEDV      
 Min.   : 1.730   Min.   : 5.00  
 1st Qu.: 7.125   1st Qu.:17.02  
 Median :11.430   Median :21.20  
 Mean   :12.715   Mean   :22.53  
 3rd Qu.:16.955   3rd Qu.:25.00  
 Max.   :37.970   Max.   :50.00  
 NA's   :20                      
```

Packages for reading different data formats include `haven`:

- SAS: `read_sas()` and `read_xpt()`
- SPSS: `read_sav()` and `read_por()`
- Stata: `read_dta()`

1.6.3 Database Connections

Data from databases like MySQL, SQLite, Postgresql, and MariaDB are becoming increasingly important.

1.6.4 Summary of ggplot2 Usage with Examples

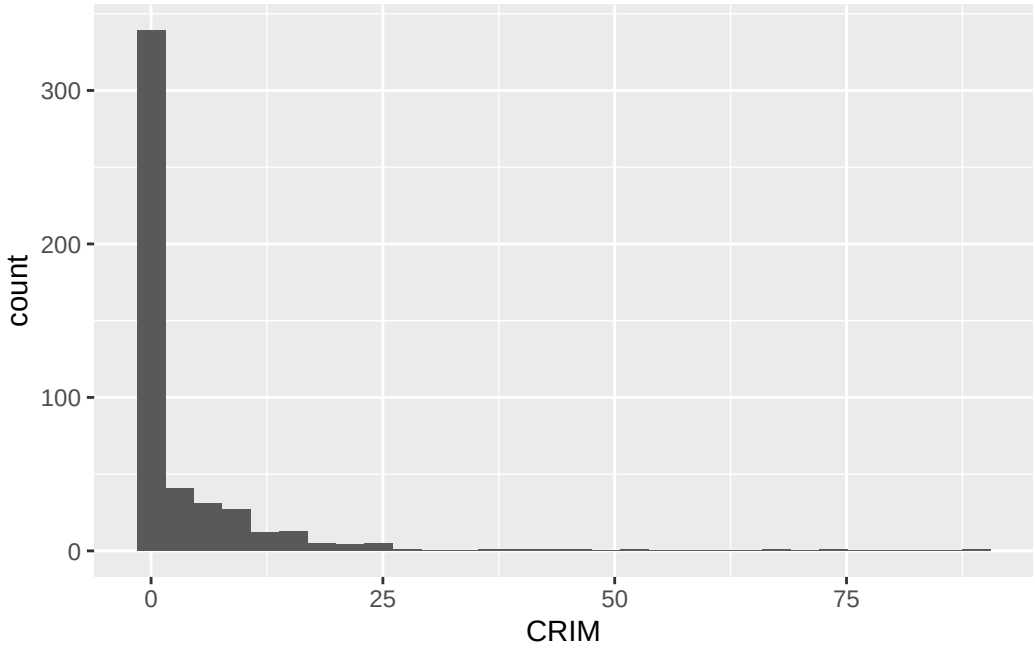
Here's a basic guide and examples of using `ggplot2`:

1.6.4.1 Histogram

```
library(ggplot2)
ggplot(mydata) +
  geom_histogram(aes(x = CRIM))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 20 rows containing non-finite outside the scale range
(`stat_bin()`).
```

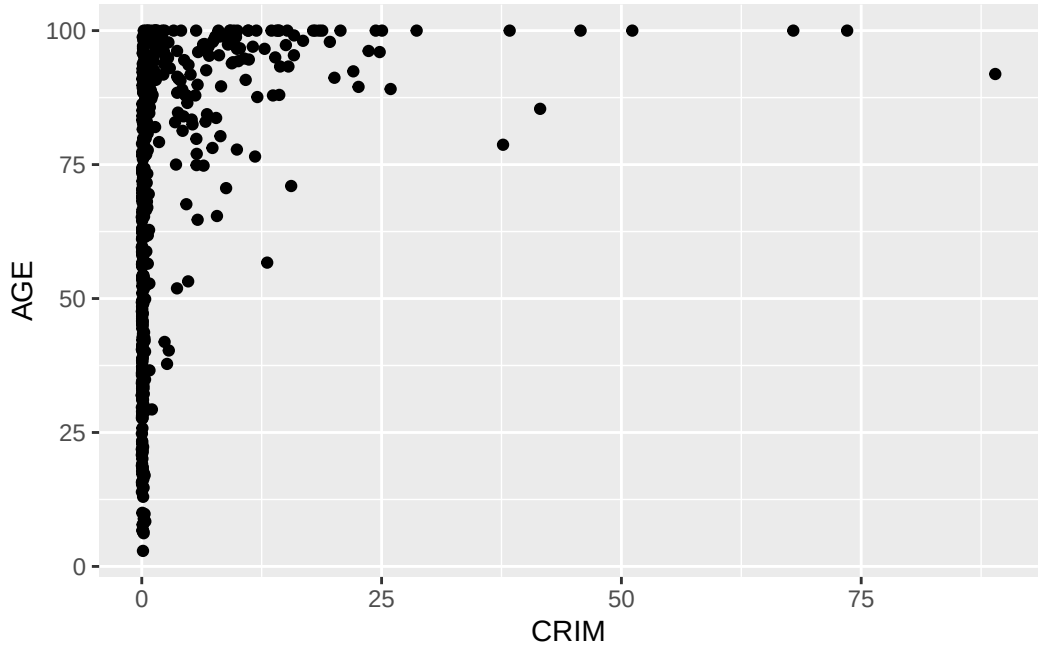



```
# x = variable_name
```

1.6.4.2 Scatter Plot

```
ggplot(mydata) +
  geom_point(aes(x = CRIM, y = AGE))
```

```
Warning: Removed 40 rows containing missing values or values outside the scale range
(`geom_point()`).
```

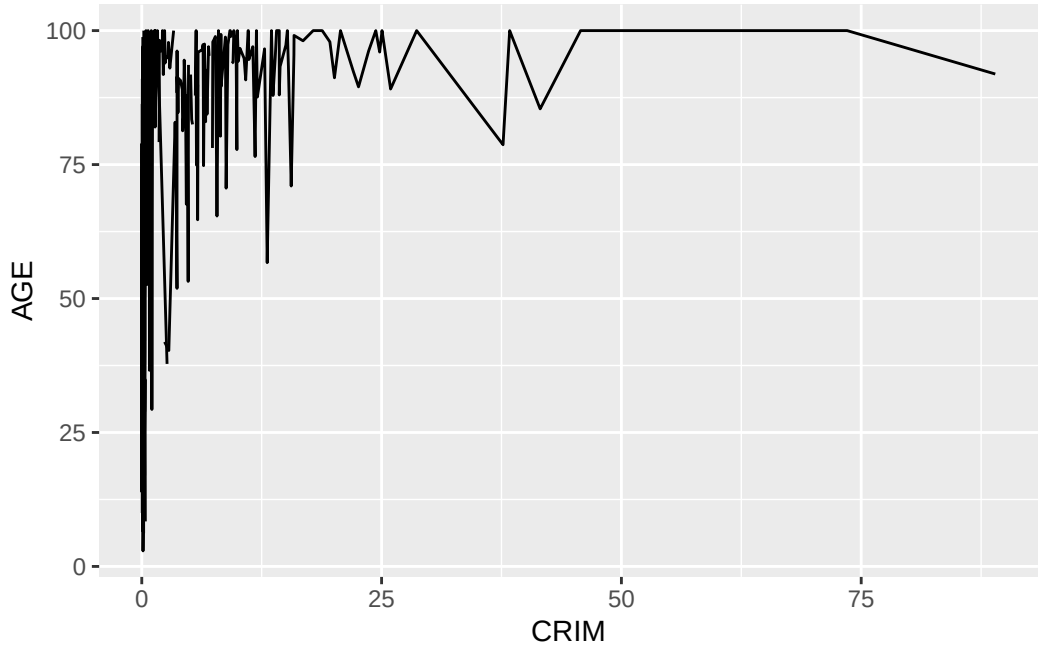



```
# x = variable1, y = variable2
```

1.6.4.3 Line Graph

```
ggplot(mydata) +
  geom_line(aes(x = CRIM, y = AGE, group = RAD))
```

```
Warning: Removed 21 rows containing missing values or values outside the scale range
(`geom_line()`).
```



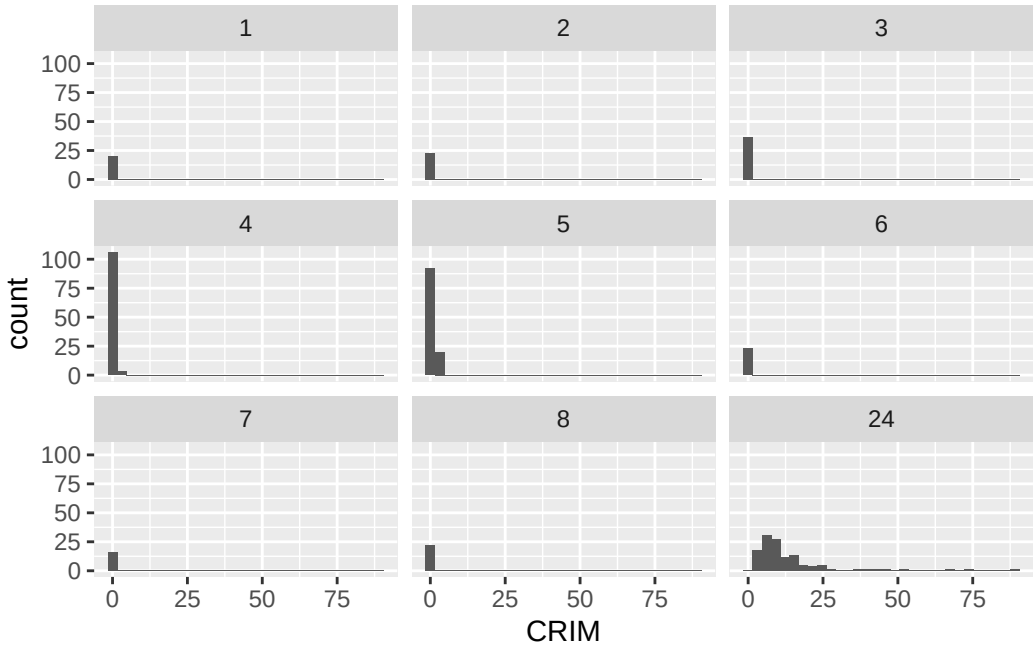

```
# x = time_variable, y = measurement_variable, group = category_variable
```

1.6.4.4 Faceting

```
ggplot(mydata) +
  geom_histogram(aes(x = CRIM)) +
  facet_wrap(~ RAD)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 20 rows containing non-finite outside the scale range
(`stat_bin()`).
```




```
# x = variable_name, ~ = category_variable
```

2 Hands-on 1: Packages

2.1 Load the Packages

To create plots using ggplot2, we need to load the `tidyverse` package, which includes ggplot2 and other useful packages. Loading `tidyverse` provides access to various help pages, functions, and datasets included in the package suite.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v lubridate 1.9.3     v tibble    3.2.1
v purrr     1.0.2     v tidyr     1.3.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

2.2 Open the Dataset

Use the `gapminder` dataset from the `gapminder` package. The dataset offers historical data on global life expectancy, GDP per capita, and population.

```
# Load the gapminder package
library(gapminder)

# View the first few rows of the dataset
head(gapminder)
```

```
# A tibble: 6 x 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.
```

The `gapminder` dataset includes the following columns:

- `country`: Country name (factor)
- `continent`: Continent name (factor)
- `year`: Year of observation (integer)
- `lifeExp`: Life expectancy (numeric)
- `pop`: Population (integer)
- `gdpPercap`: GDP per capita (numeric)

```
# Get a glimpse of the dataset structure
glimpse(gapminder)
```

```
Rows: 1,704
Columns: 6
$ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ~
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ~
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12~
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
```

The output provides a summary of the number of observations, types of variables, and a preview of the data:

- 1704 observations
- 6 variables (2 factor, 2 integer, and 2 numeric variables)

```
# Summarize the dataset
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
```

The summary function provides statistical insights into each variable:

- Frequencies for factor variables (`country`, `continent`)
- Minimum, 1st quartile, median, mean, 3rd quartile, and maximum values for numerical variables (`year`, `lifeExp`, `pop`, `gdpPercap`)

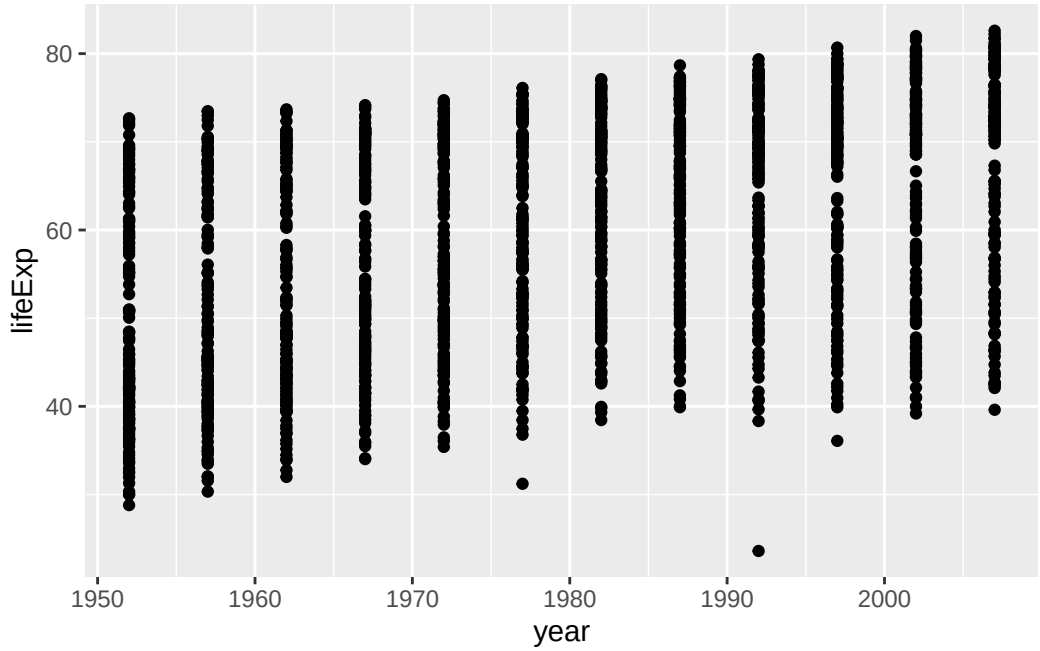
3 Hands-on 2: Scatterplots

3.1 Basic Scatterplot

Start by creating a basic scatter plot to visualize the relationship between the year and life expectancy in the gapminder dataset.

```r
# Load necessary libraries
library(tidyverse)
library(gapminder)

# Create a basic scatterplot
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = year, y = lifeExp))
```



Explanation:

- The `ggplot()` function initializes the plot and specifies the dataset (`gapminder`).
- The `geom_point()` function creates a scatter plot with points representing the data.

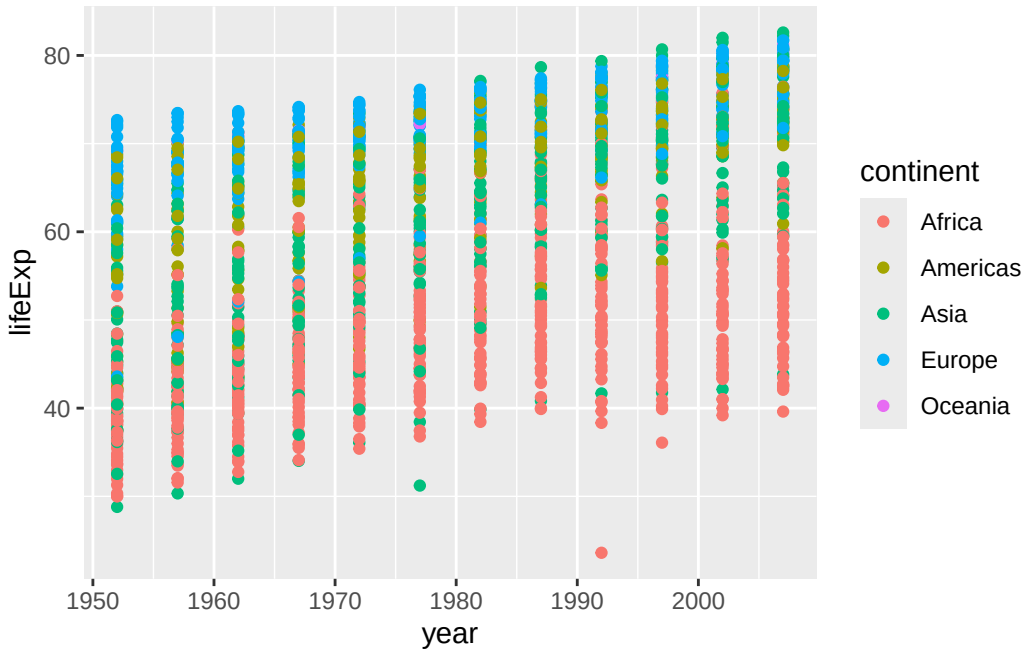
Observation:

- The plot shows an increase in life expectancy over the years, indicating a positive trend.

3.2 Adding Another Variable

Enhance the scatter plot by adding a third variable to differentiate data points by continent.

```r
# Create a scatterplot with continent colors
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = year, y = lifeExp, colour = continent))
```



Explanation:

- The `aes()` function inside `geom_point()` now includes `colour = continent`, which adds color differentiation for each continent.

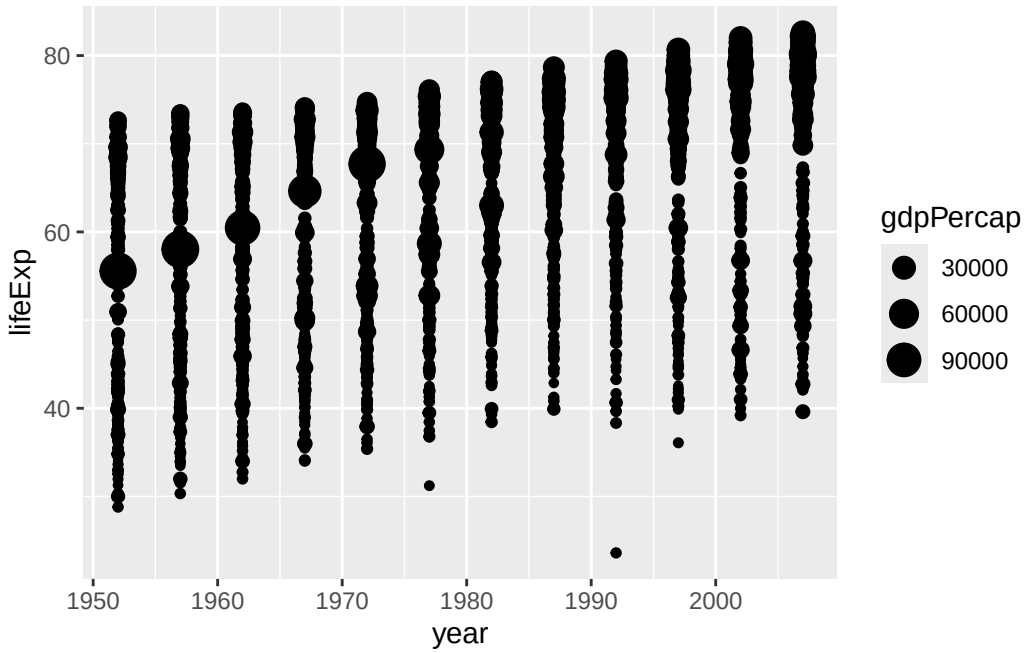
Observation:

- European countries generally have higher life expectancy.
- African countries tend to have lower life expectancy.
- Outliers with very low life expectancy are noticeable in Asia and Africa.

3.2.0.1 Adding GDP as a Size Variable

Modify the scatter plot to reflect GDP per capita (`gdpPercap`) as the size of the points.

```r
# Create a scatterplot with GDP per capita as the size of points
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = year, y = lifeExp, size = gdpPercap))
```



Explanation:

- The `size = gdpPercap` aesthetic scales the size of the points according to GDP per capita.

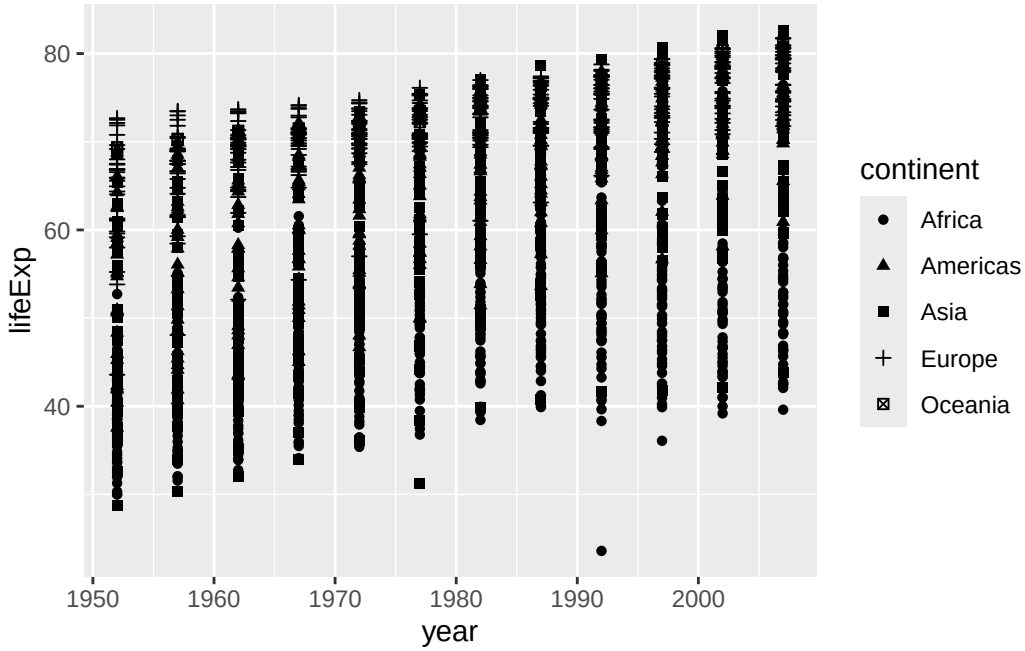
Observation:

- Countries with higher GDP generally have longer life expectancy.

3.2.0.2 Using Shape Instead of Color

You can use different shapes to represent continents when color is not an option, such as in black-and-white printouts.

```
# Create a scatterplot with different shapes for continents
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = year, y = lifeExp, shape = continent))
```



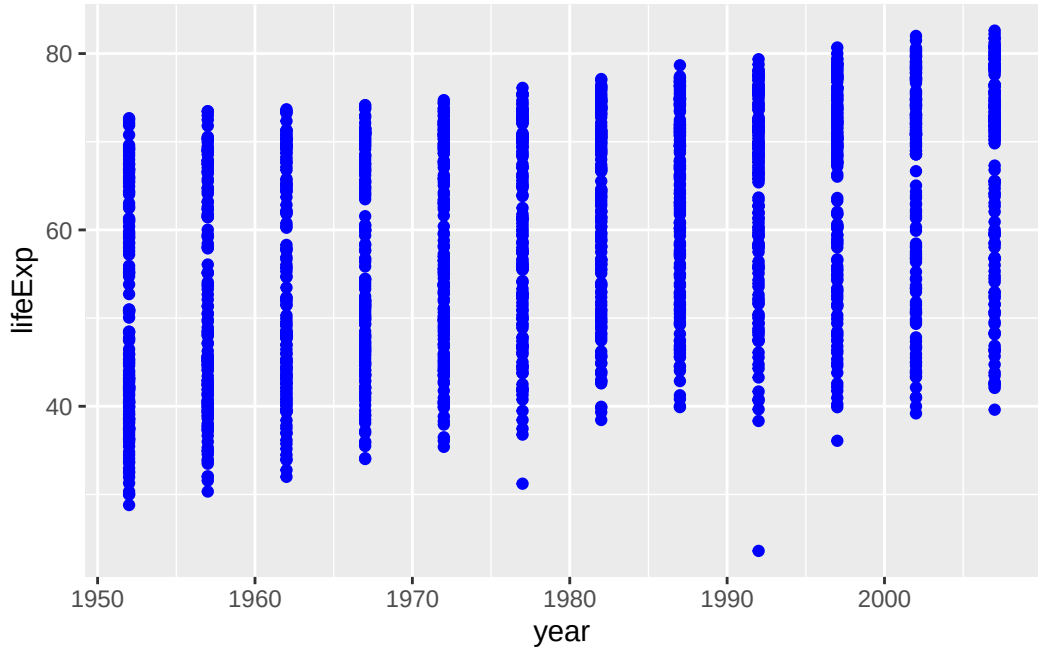
Explanation:

- The `shape = continent` aesthetic assigns different shapes to data points based on continent.

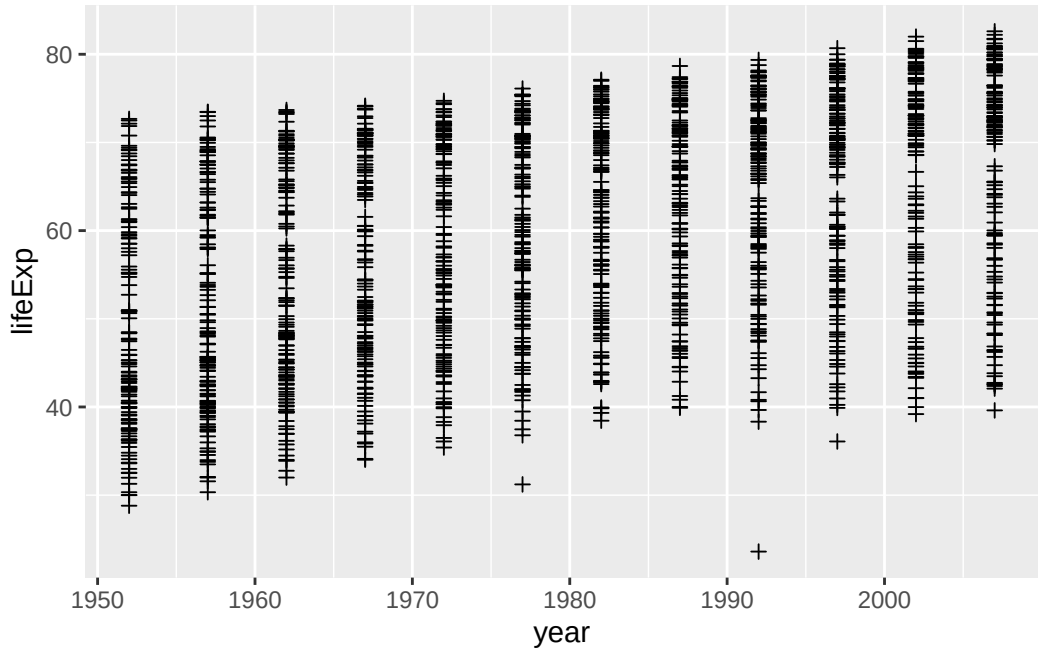
3.2.0.3 Setting Color and Shape Outside `aes()`

You can also set a uniform color or shape for all points by placing the arguments outside the `aes()` function.

```
# Create a scatterplot with all points colored blue
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = year, y = lifeExp), colour = 'blue')
```




```
# Create a scatterplot with all points shaped as plus signs
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = year, y = lifeExp), shape = 3)
```



Explanation:

- Setting `colour = 'blue'` or `shape = 3` outside `aes()` applies these attributes uniformly to all points.
- The shape argument uses numbers to represent different symbols (`?pch` provides a list of shape codes).
- Typing `?pch` in the R console will show you a reference for all the available point shapes and their corresponding codes.

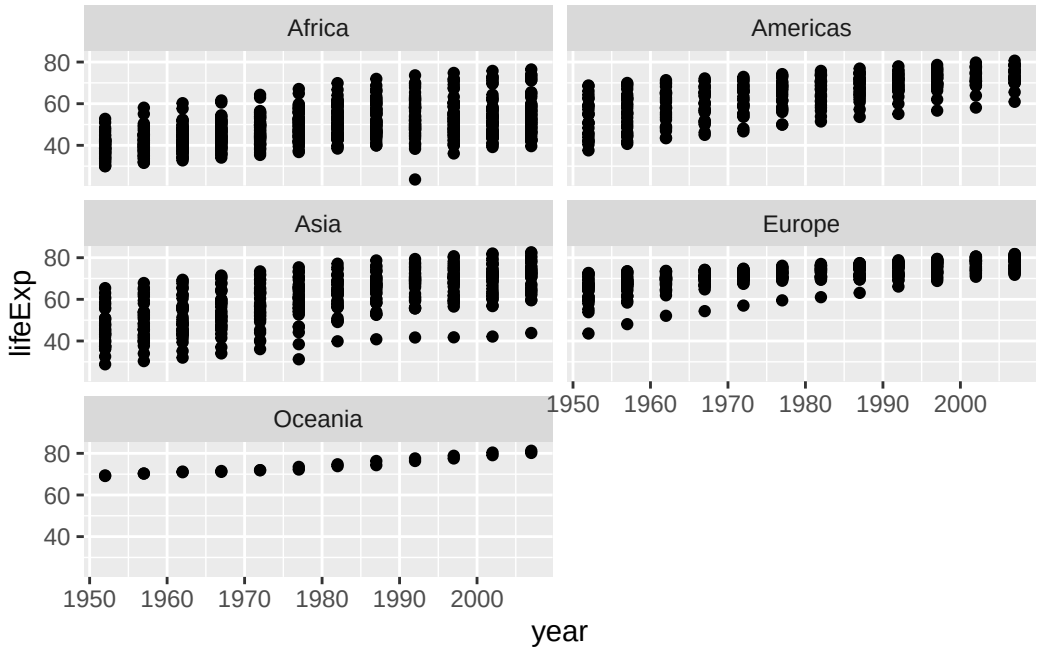
4 Hands-on 3: Subplots

To create subplots, we can use the `facet_wrap()` function to split our plots based on a factor variable. This allows for easy comparison of data subsets across different categories.

4.1 Creating Subplots Based on Continents

4.1.1 Create Subplots with 3 Rows

```
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = year, y = lifeExp)) +
  facet_wrap(~ continent, nrow = 3)
```

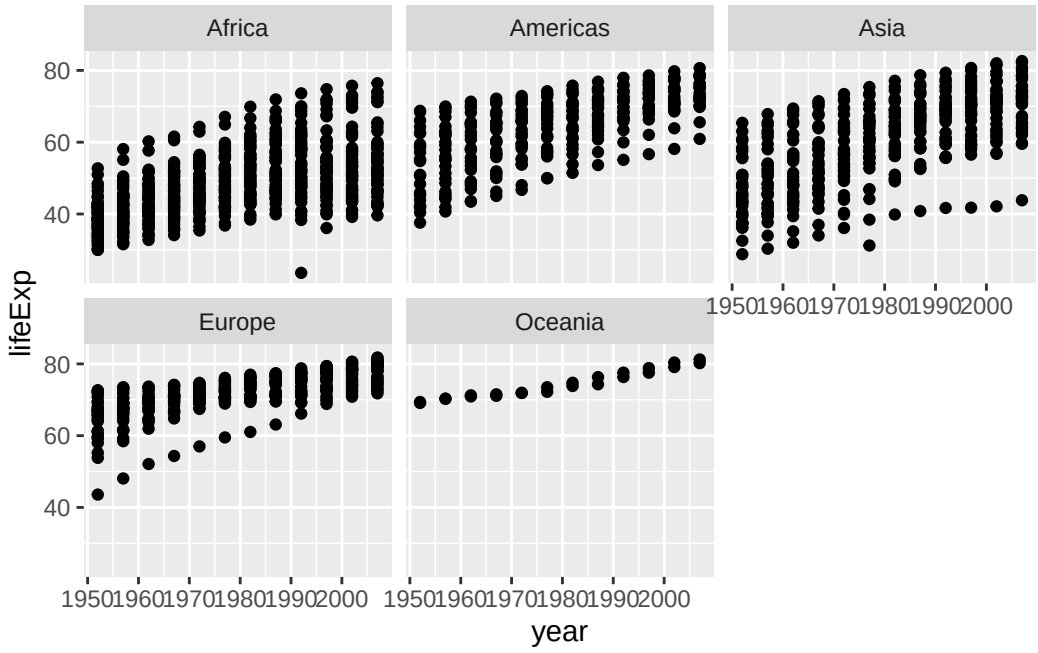


Explanation:

- `facet_wrap(~ continent)` splits the plot into subplots for each continent.
- `nrow = 3` arranges the subplots into 3 rows.

4.1.2 Create Subplots with 2 Rows

```
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = year, y = lifeExp)) +
  facet_wrap(~ continent, nrow = 2)
```



Explanation:

- Changing `nrow` to 2 arranges the subplots into 2 rows instead of 3, adjusting the layout for better readability based on your preferences or the amount of data.

4.2 Customizing Subplots

- You can further customize subplots by adjusting the number of columns (`ncol`), adding titles, modifying axis labels, and more to suit your analysis needs.
- **Subplots facilitate comparison:** By breaking down the data into subplots based on the `continent` variable, it's easier to compare trends and patterns across different continents.
- **Life expectancy trends:** Each subplot shows how life expectancy changes over time within each continent, making it clear how trends differ geographically.
- By utilizing `facet_wrap()`, you can create organized and insightful visualizations that highlight differences and similarities across subsets of your data.

5 Hands-on 4: Scatterplot, Smooth Plot, and Combining Plots

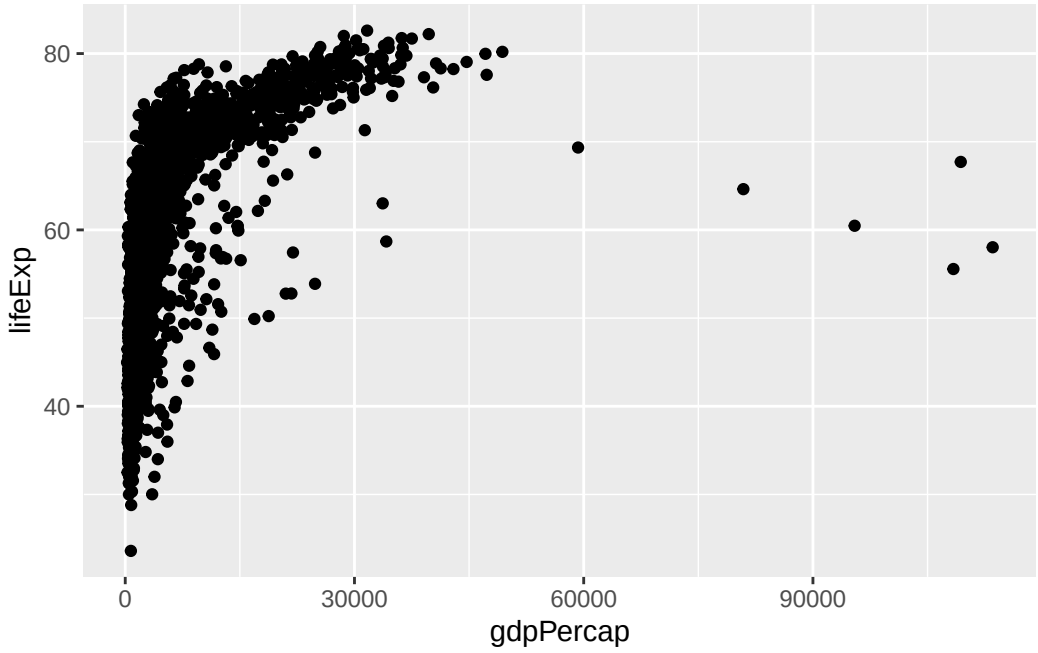
5.1 Scatterplot and Smooth Plot

In `ggplot2`, different geometric objects (`geom_X()`) represent different visualizations.

5.1.1 Scatterplot

```
# Load necessary libraries
library(tidyverse)
library(gapminder)

# Create a scatterplot
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = gdpPercap, y = lifeExp))
```



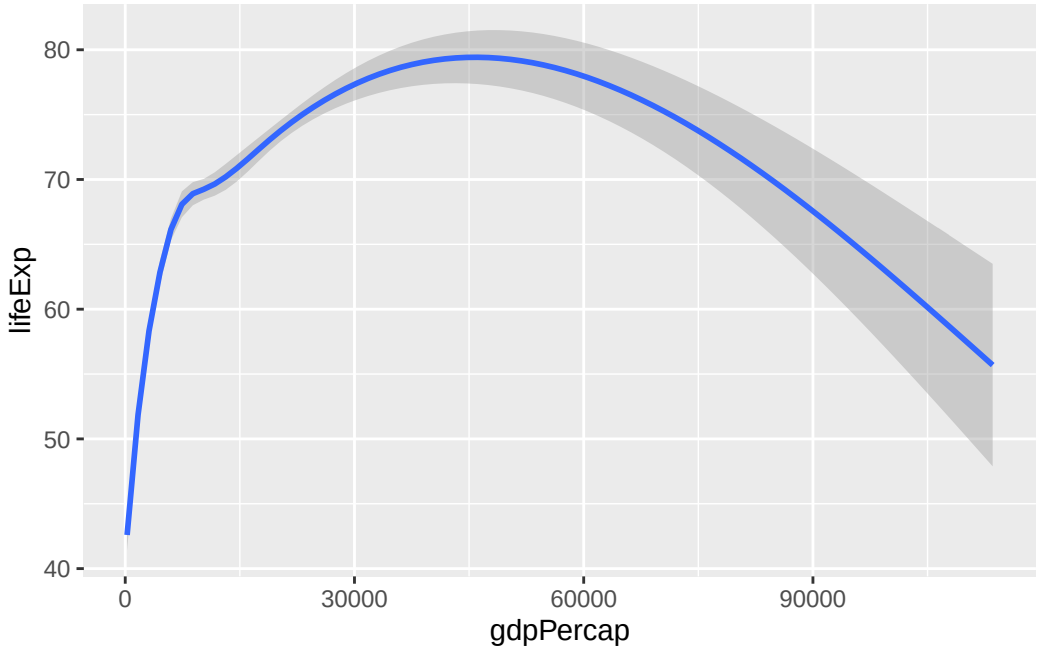
Explanation:

- `geom_point()` creates a scatter plot with GDP per capita on the x-axis and life expectancy on the y-axis.

5.1.2 Smooth Plot

```
ggplot(data = gapminder) +
  geom_smooth(mapping = aes(x = gdpPercap, y = lifeExp))
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



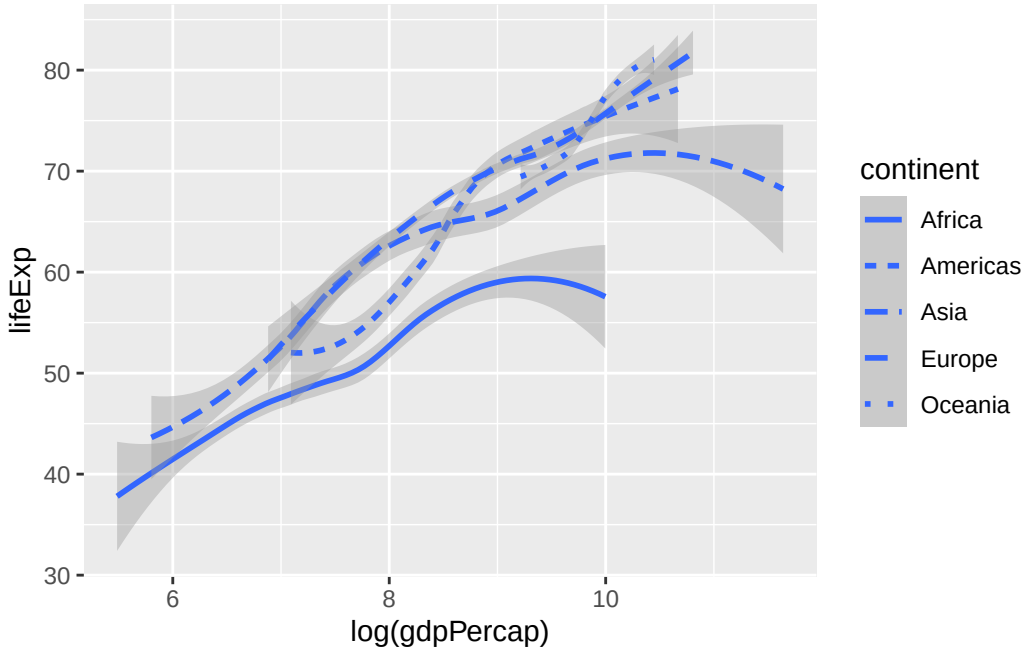
Explanation:

- `geom_smooth()` adds a smooth line to the plot, indicating trends in the data.

We can generate the smooth plot based on continent using the `linetype()`. We use `log(gdpPercap)` to reduce the skewness of the data.

```
ggplot(data = gapminder) +
  geom_smooth(mapping = aes(x = log(gdpPercap), y = lifeExp, linetype = continent))
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



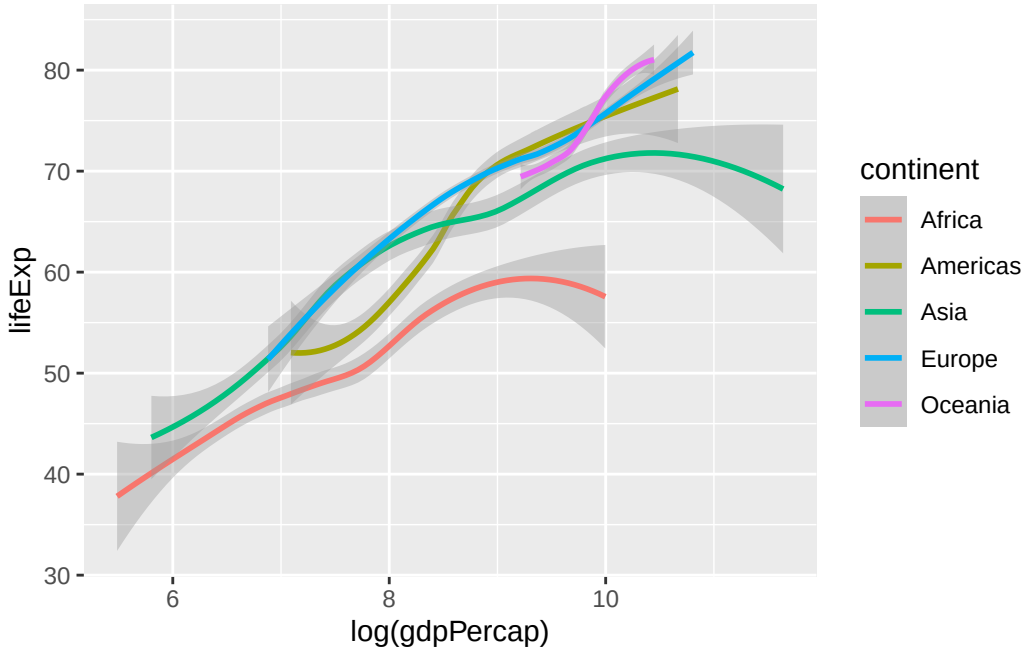
Explanation:

- `aes(x = log(gdpPercap), y = lifeExp, linetype = continent)` uses the logarithm of GDP per capita to reduce skewness and differentiates lines by continent.

We can also generate the smooth plot using colour.

```
ggplot(data = gapminder) +
  geom_smooth(mapping = aes(x = log(gdpPercap), y = lifeExp, colour = continent))
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Explanation:

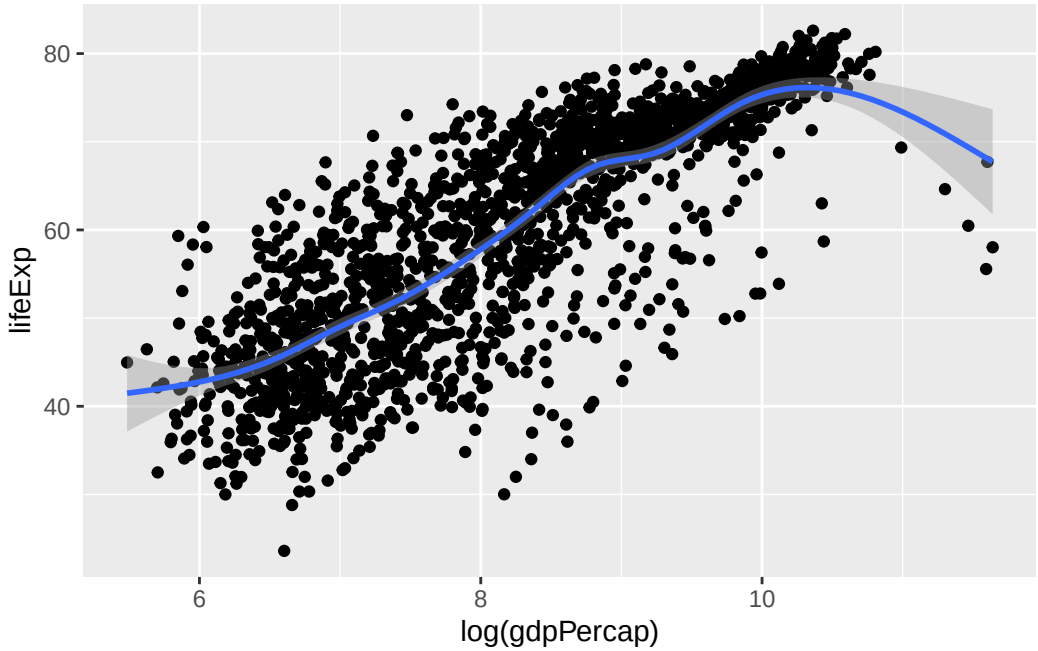
- `colour = continent` differentiates lines by continent using color.

5.2 Combining Geoms

We can overlay multiple geoms (geometric objects) in a single plot to visualize multiple aspects of the data simultaneously.

```
ggplot(data = gapminder) +
  geom_point(mapping = aes(x = log(gdpPercap), y = lifeExp)) +
  geom_smooth(mapping = aes(x = log(gdpPercap), y = lifeExp))
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



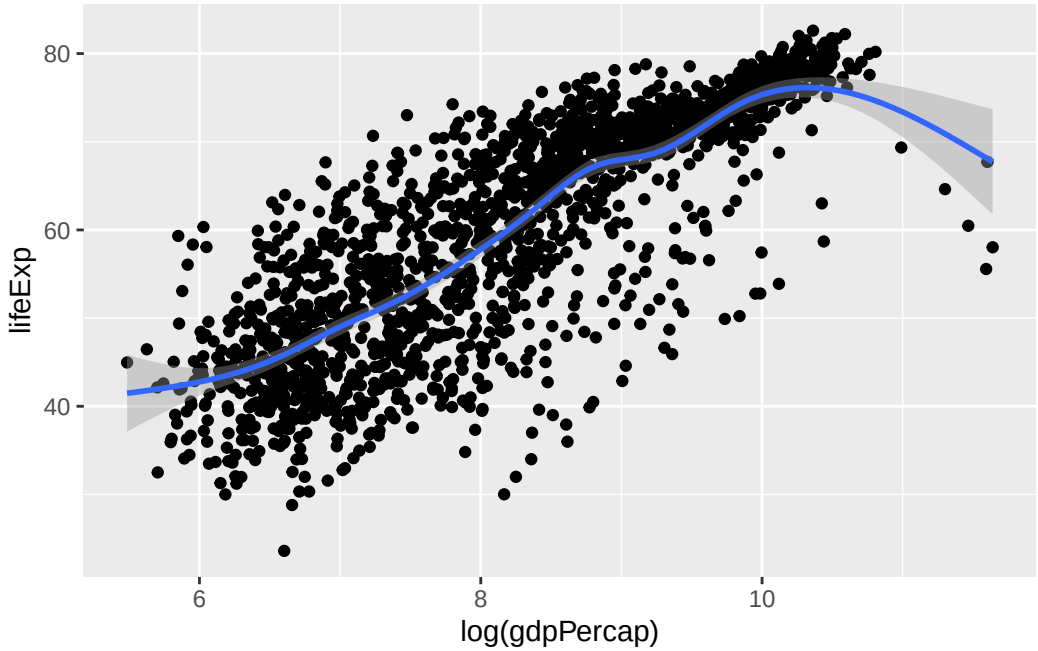
Explanation:

- This combines a scatterplot and a smooth plot, but it repeats the `mapping` for each geom.

The codes above show duplication or repetition. To avoid this, we can pass the mapping to `ggplot()`.

```
ggplot(data = gapminder, mapping = aes(x = log(gdpPercap), y = lifeExp)) +
  geom_point() +
  geom_smooth()
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



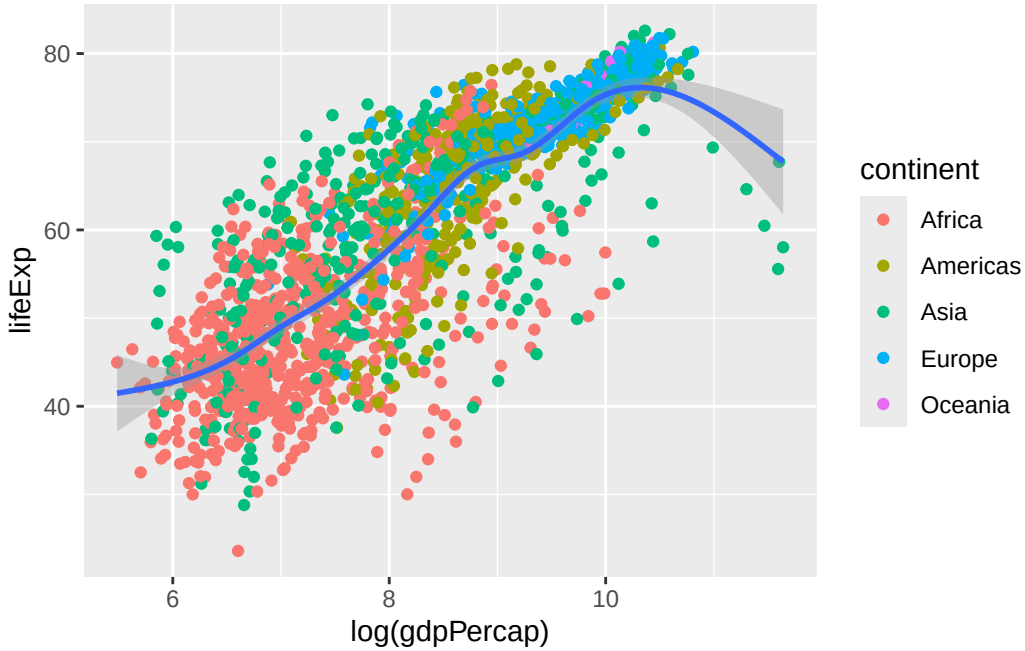
Explanation:

- Passing the `mapping` to `ggplot()` applies it to all geoms, reducing code duplication.

And we can expand this to make scatterplot with different colors for continents.

```
ggplot(data = gapminder, mapping = aes(x = log(gdpPercap), y = lifeExp)) +
  geom_point(mapping = aes(colour = continent)) +
  geom_smooth()
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



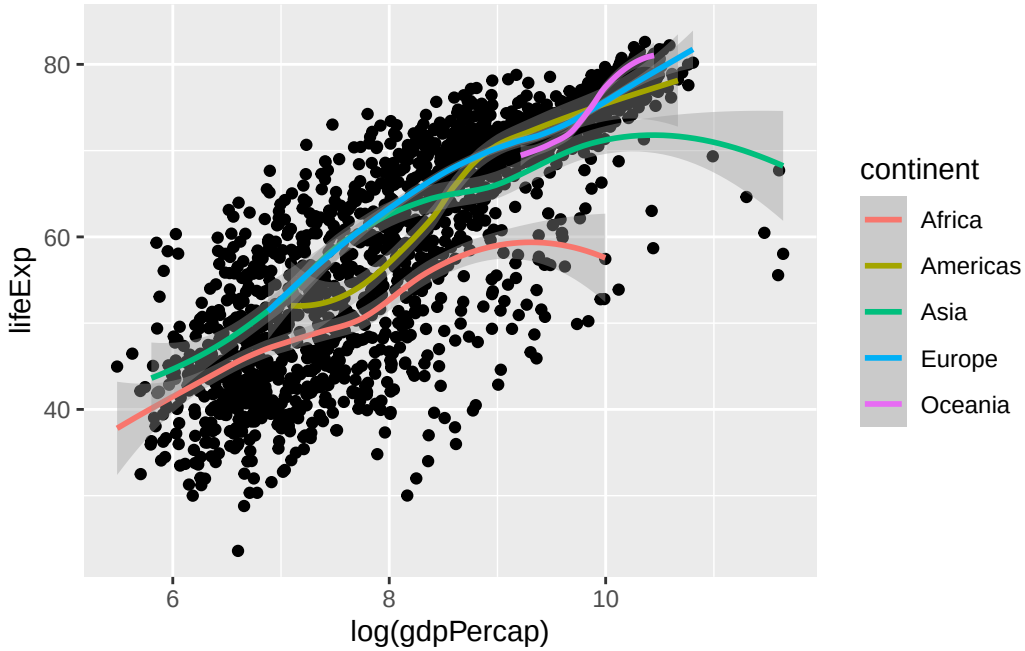
Explanation:

- This plot combines a scatterplot with points colored by continent and a smooth line.

Or expand this to make the smooth plot shows different colour for continent.

```
ggplot(data = gapminder, mapping = aes(x = log(gdpPercap), y = lifeExp)) +
  geom_point() +
  geom_smooth(mapping = aes(colour = continent))
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



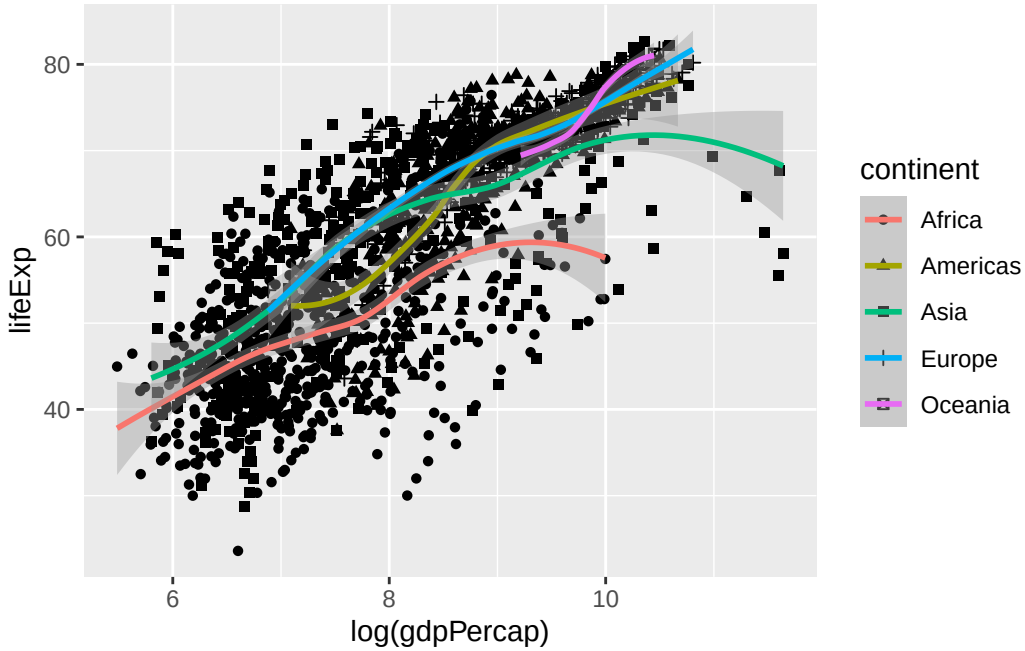
Explanation:

- This plot combines a scatterplot and a smooth plot, with the smooth lines colored by continent.

Or combining both scatterplot and smooth plot.

```
ggplot(data = gapminder, mapping = aes(x = log(gdpPercap), y = lifeExp)) +
  geom_point(mapping = aes(shape = continent)) +
  geom_smooth(mapping = aes(colour = continent))
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Explanation:

- This plot uses different shapes for continents in the scatterplot and different colors for continents in the smooth plot.

5.3 Summary

- **Scatterplot (`geom_point()`):** Visualizes individual data points.
- **Smooth Plot (`geom_smooth()`):** Adds trend lines to visualize the overall trend.
- **Combining Geoms:** Overlay different types of plots to provide more comprehensive visual analysis.
- **Aesthetics Mapping:** Use `aes()` to map data variables to visual properties like color, shape, and size.
- **Avoiding Duplication:** Pass common aesthetics mapping to `ggplot()` to avoid repetition and simplify code.

6 Hands-on 5: Bar Plot and Histogram

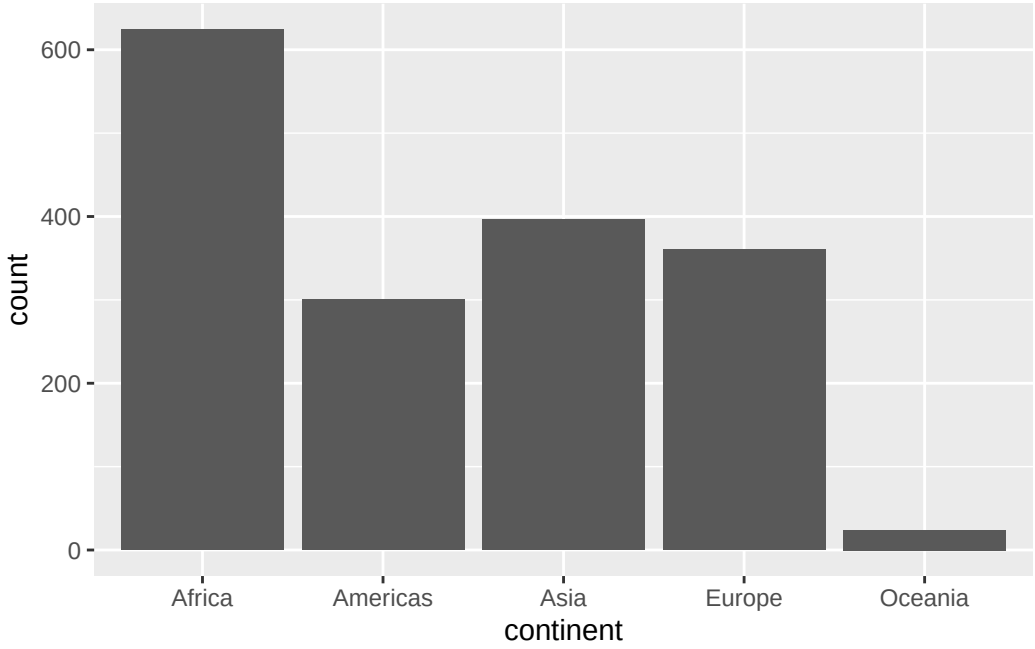
6.1 Bar Plot

A bar plot is useful for displaying the distribution of categorical data. Here, bar plots are created to show the frequency and proportion of continents in the `gapminder` dataset.

Bar Plot Showing Frequency

```
# Load necessary libraries
library(tidyverse)
library(gapminder)

# Create a bar plot showing frequency of each continent
ggplot(data = gapminder) +
  geom_bar(mapping = aes(x = continent))
```



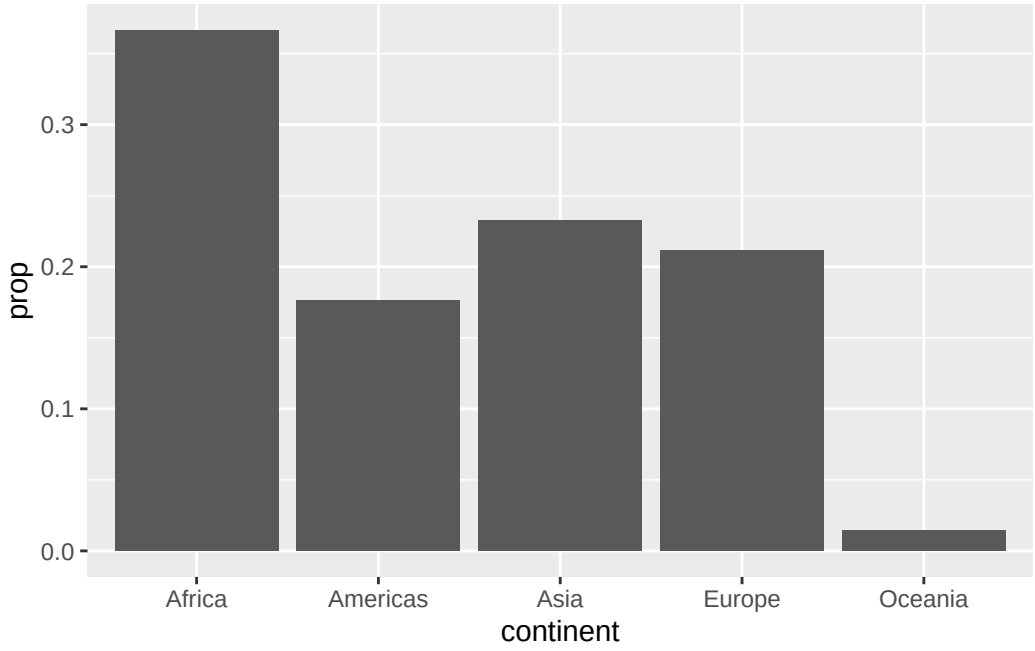
Explanation:

- `geom_bar()` creates a bar plot with continents on the x-axis and the count (frequency) of records on the y-axis.

Bar Plot Showing Proportion

```
# Create a bar plot showing proportion of each continent
ggplot(data = gapminder) +
  geom_bar(mapping = aes(x = continent, y = ..prop.., group = 1))
```

```
Warning: The dot-dot notation (`..prop..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(prop)` instead.
```



Explanation:

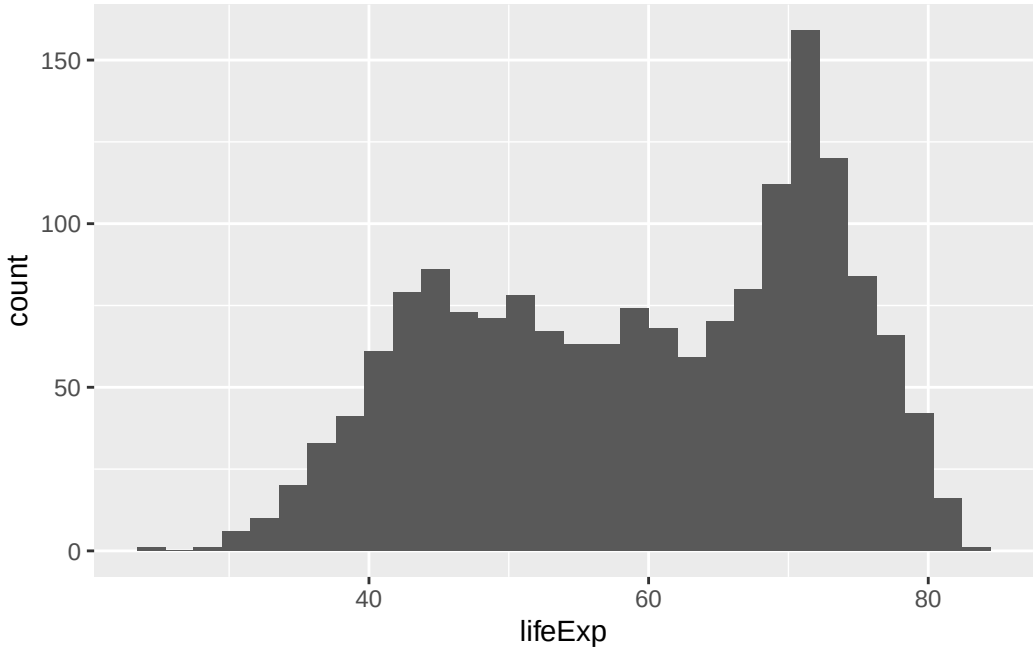
- `y = ..prop..` scales the y-axis to show the proportion of records for each continent instead of the count.
- `group = 1` ensures that proportions are calculated for the entire dataset.

6.2 Histogram

Histograms are used for visualizing the distribution of numerical data. Here, we create a histogram for life expectancy in the `gapminder` dataset.

```
# Create a histogram for life expectancy
ggplot(data = gapminder, aes(x = lifeExp)) +
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Explanation:

- `geom_histogram()` creates a histogram with life expectancy on the x-axis.
- By default, `geom_histogram()` uses 30 bins. You can adjust the `binwidth` parameter to get a more detailed or summarized view of the data distribution.

6.3 Summary

- **Bar Plot (`geom_bar()`):** Visualizes the distribution of categorical data. It can show frequencies or proportions.
- **Histogram (`geom_histogram()`):** Visualizes the distribution of numerical data by dividing the data into bins and counting the number of observations in each bin.
- **Proportional Bar Plot:** Use `..prop..` in the y aesthetic to display proportions instead of counts.

7 Hands-on 6: Meaningful and Beautiful Plots

7.1 Customizing Title

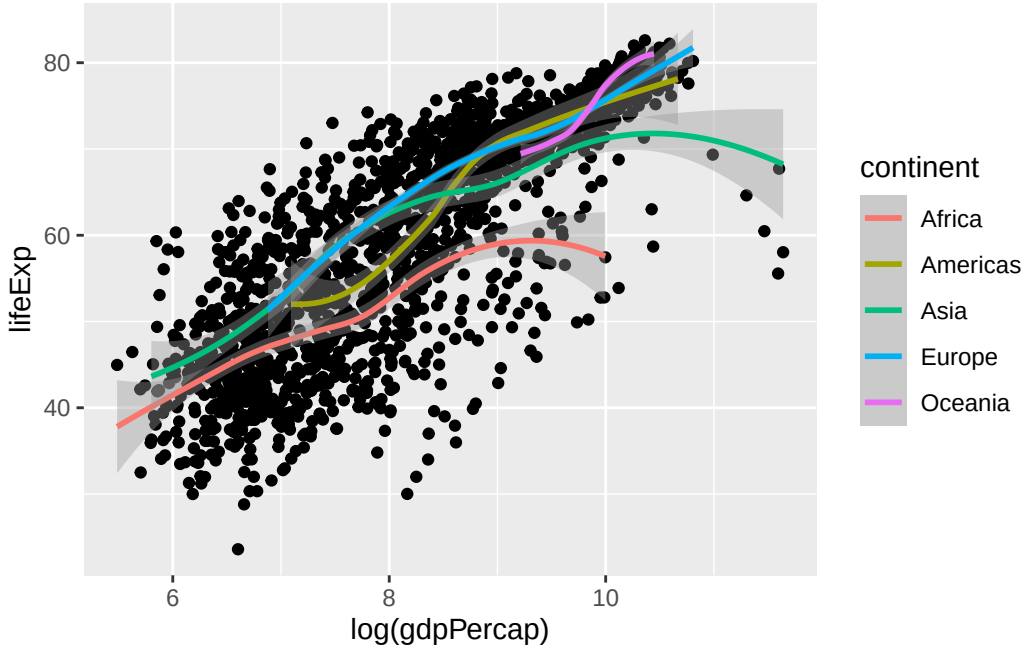
Customizing plot titles can make your visualizations more informative.

Creating the Initial Plot

```
# Load necessary libraries
library(tidyverse)
library(gapminder)

# Create a scatter plot with a smooth line
mypop <- ggplot(data = gapminder, mapping = aes(x = log(gdpPercap), y = lifeExp)) +
  geom_point() +
  geom_smooth(mapping = aes(colour = continent))
mypop
```

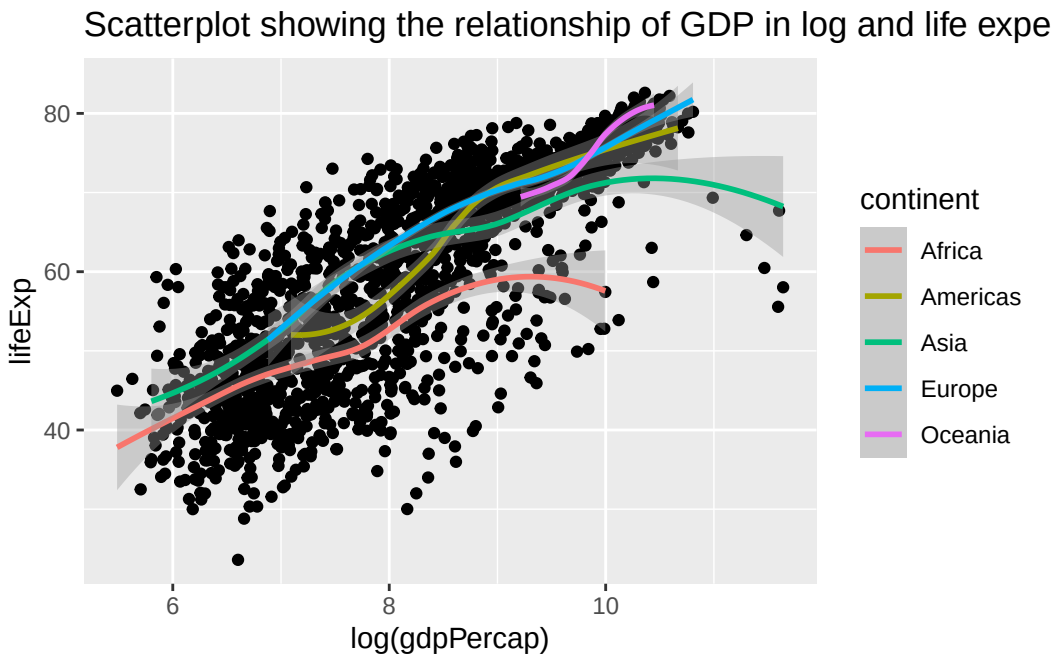
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Adding a Title

```
mypop +
  ggtitle("Scatterplot showing the relationship of GDP in log and life expectancy")
```

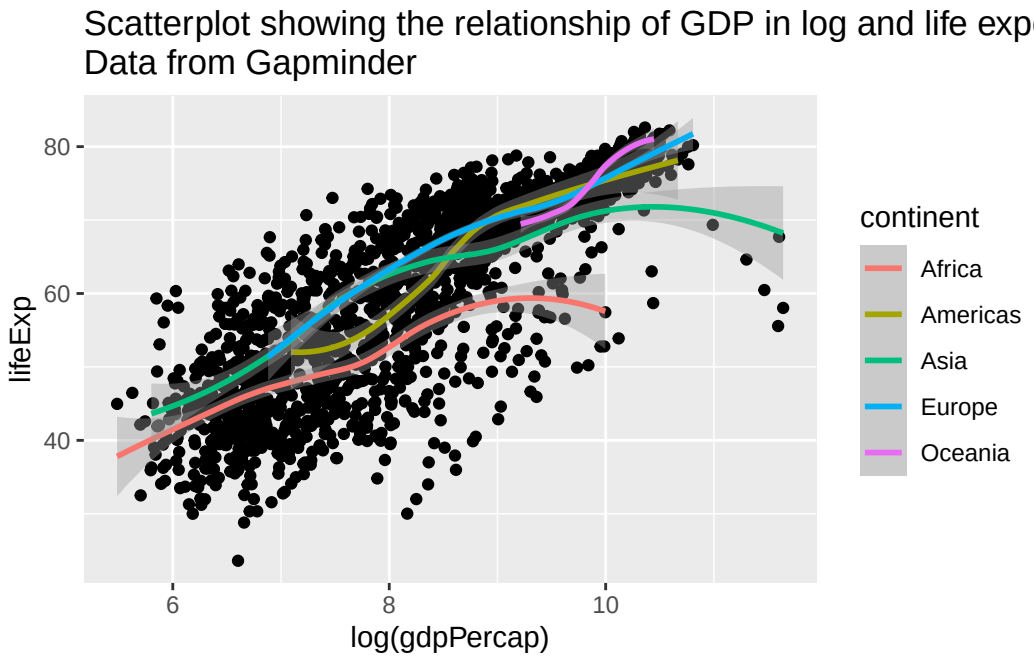
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Adding a Title with Multiple Lines

```
mypop +
  ggtitle("Scatterplot showing the relationship of GDP in log and life expectancy:\nData fro
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



7.2 Adjusting Axes

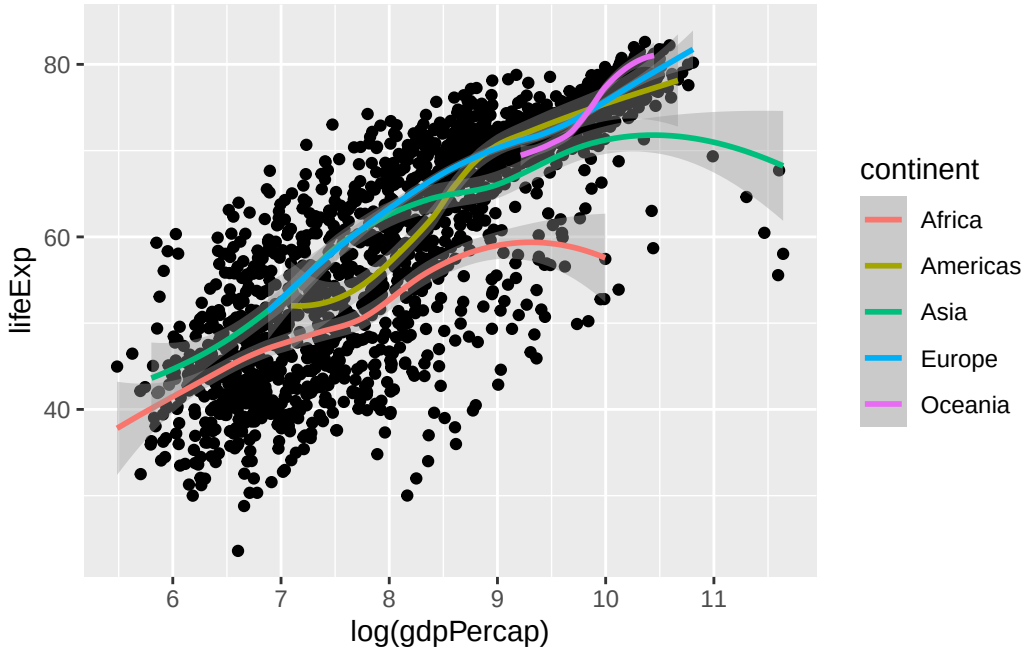
Adjusting axis tick marks and labels can enhance readability.

Specifying Tick Marks

- min = 0
- max = 12
- interval = 1

```
mypop +
  scale_x_continuous(breaks = seq(0, 12, 1))
```

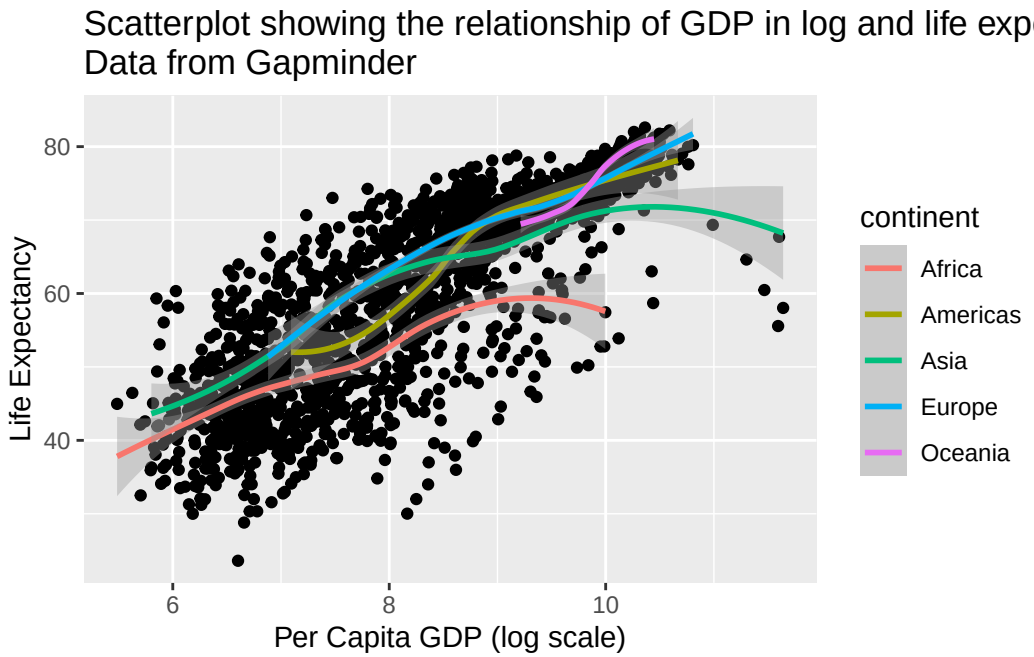
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Labeling Axes Add axis labels and a title with multiple lines.

```
mypop +
  ggtitle("Scatterplot showing the relationship of GDP in log and life expectancy:\nData fro
  ylab("Life Expectancy") +
  xlab("Per Capita GDP (log scale)")
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



7.3 Choosing Themes

`ggplot2` provides several themes to change the overall appearance of plots:

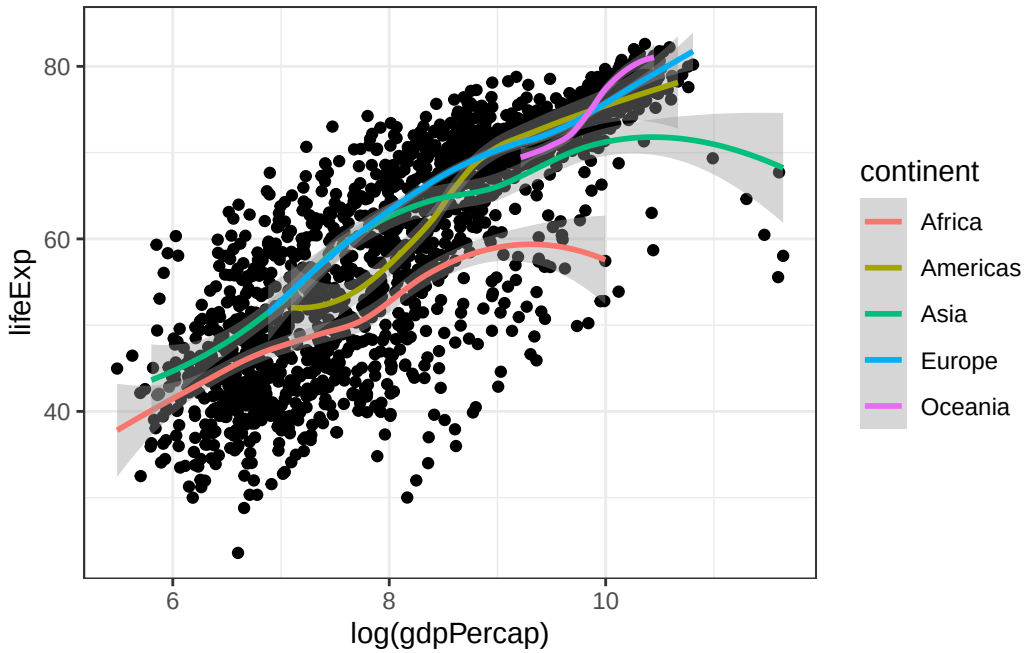
Default Theme (Gray)

- The default theme in ggplot2 is `theme_gray()`.

Black and White Theme

```
mypop + theme_bw()
```

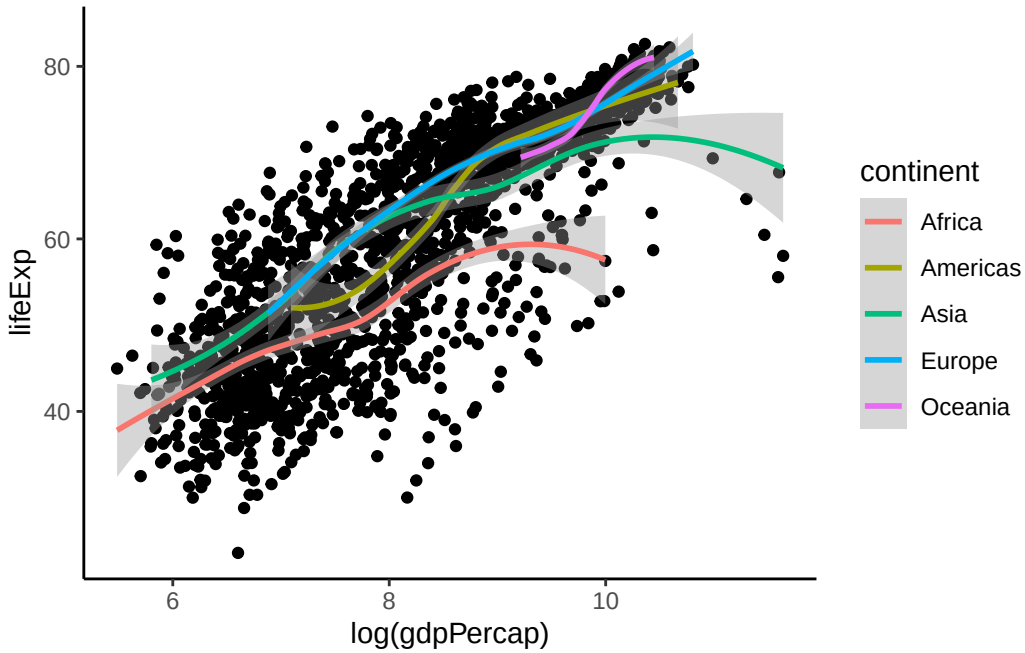
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Classic Theme

```
mypop + theme_classic()
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



7.4 Summary

- **Customizing Titles:** Adding informative titles and subtitles helps in understanding the context of the plot.
- **Adjusting Axes:** Customizing tick marks and axis labels makes the plot more readable.
- **Choosing Themes:** Different themes can enhance the aesthetic appeal of the plot and highlight different aspects of the data.

8 Hands-on 7: Saving Plots

8.1 Preferred Format for Saving

The preferred format for saving plots in R is often PDF because it preserves the quality and scalability of the graphics. However, other formats like PNG and JPG are also commonly used.

8.2 Saving Plots Using ggplot2

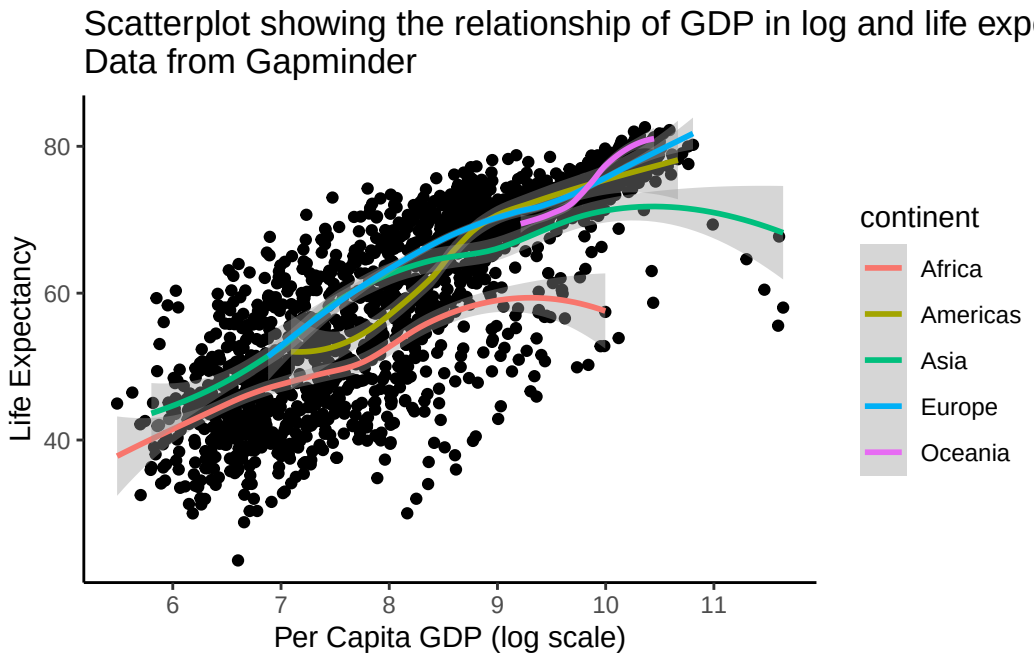
Creating a Complete Plot

```r
# Load necessary libraries
library(tidyverse)
library(gapminder)

# Create the plot
mypop <- ggplot(data = gapminder, mapping = aes(x = log(gdpPercap), y = lifeExp)) +
  geom_point() +
  geom_smooth(mapping = aes(colour = continent)) +
  ggtitle("Scatterplot showing the relationship of GDP in log and life expectancy:\nData fro
  ylab("Life Expectancy") +
  xlab("Per Capita GDP (log scale)") +
  scale_x_continuous(breaks = seq(0, 12, 1)) +
  theme_classic()

# Display the plot
mypop
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Saving the Plot

- You can save the current plot displayed on the screen to various file formats using `ggsave()`.

```r
# Save the plot as a PDF
ggsave("my_pdf_plot.pdf")
```

```
Saving 5.5 x 3.5 in image
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```r
# Save the plot as a PNG
ggsave("my_png_plot.png")
```

```
Saving 5.5 x 3.5 in image
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```r
# Save the plot as a JPG
ggsave("my_jpg_plot.jpg")
```

```
Saving 5.5 x 3.5 in image
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

Customizing the Saved Plot

- The dimensions and resolution of the saved plot can also be customized.

```r
# Save the plot as a customized PDF
ggsave("my_pdf_plot2.pdf", width = 10, height = 6, units = "cm")
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```r
# Save the plot as a customized PNG
ggsave("my_png_plot2.png", width = 10, height = 6, units = "cm")
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```r
# Save the plot as a customized JPG
ggsave("my_jpg_plot2.jpg", width = 10, height = 6, units = "cm")
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

8.3 Summary

- **Preferred Format:** PDF is often preferred for its quality and scalability, but PNG and JPG are also useful.
- **Saving Plots:** Use `ggsave()` to save plots in different formats.
- **Customization:** Customize the size and resolution of the saved plot by specifying the `width`, `height`, and `units` in the `ggsave()` function.

9 References

- Cookbook for R: A collection of R recipes for visualization and analysis. http://www.cookbook-r.com/Graphs/
- R Graph Gallery: A collection of charts made with the R programming language. https://www.r-graph-gallery.com/