Day 3: Data visualization with ggplot2 package

Qingyin Cai

Department of Applied Economics
University of Minnesota

WebR Status

🟢 Ready!

🎯 Learning Objectives

- Learn the basic operations of `ggplot2` package to create figures.
- You will be able to create:
  - scatter plot
  - line plot
  - bar plot
  - histogram
  - box plot
  - density plot
  - facet plot

✳ Reference

- ggplot2: Elegant Graphics for Data Analysis (3e)
- R for Data Science (2e), Ch1: Data Visualization

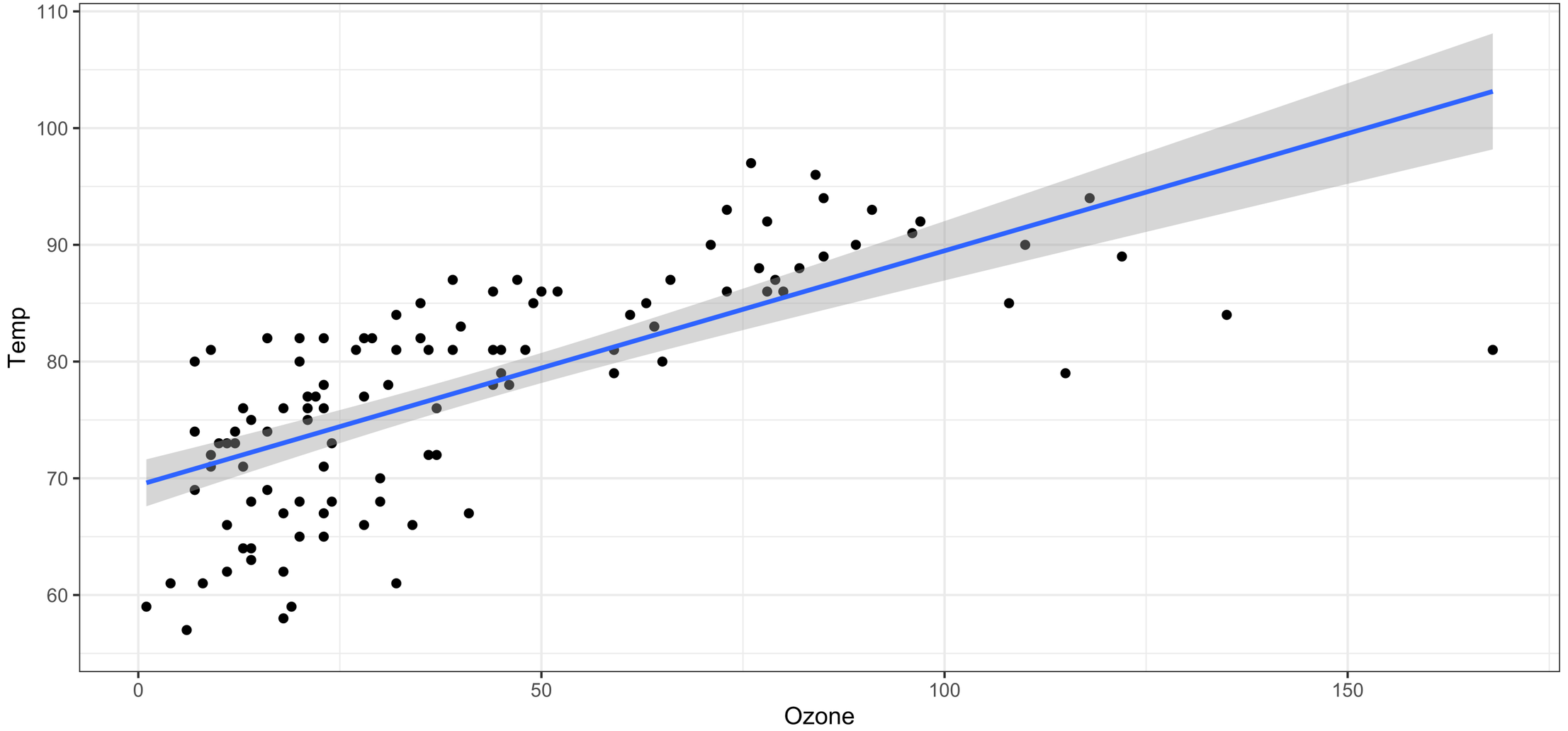
Today's Outline:

1. Taste of ggplot2 package
2. Introduction to ggplot 2
    - Anatomy of ggplot2
    - Scatter Plot
    - Different Types of Plot
    - Modify Aesthetic Attributes
    - Group Aesthetic
    - Collective geoms
    - Modify Axis, Labels, and Titles
3. Advanced Topics
    - Facet Plot
    - Multiple Datasets in One Figure
    - ggplot2 Themes
    - theme() Function
    - Save the Plot

Taste of ggplot2 package

By the end of the lecture, you will be able to create the figures like the following examples using `ggplot2` package.

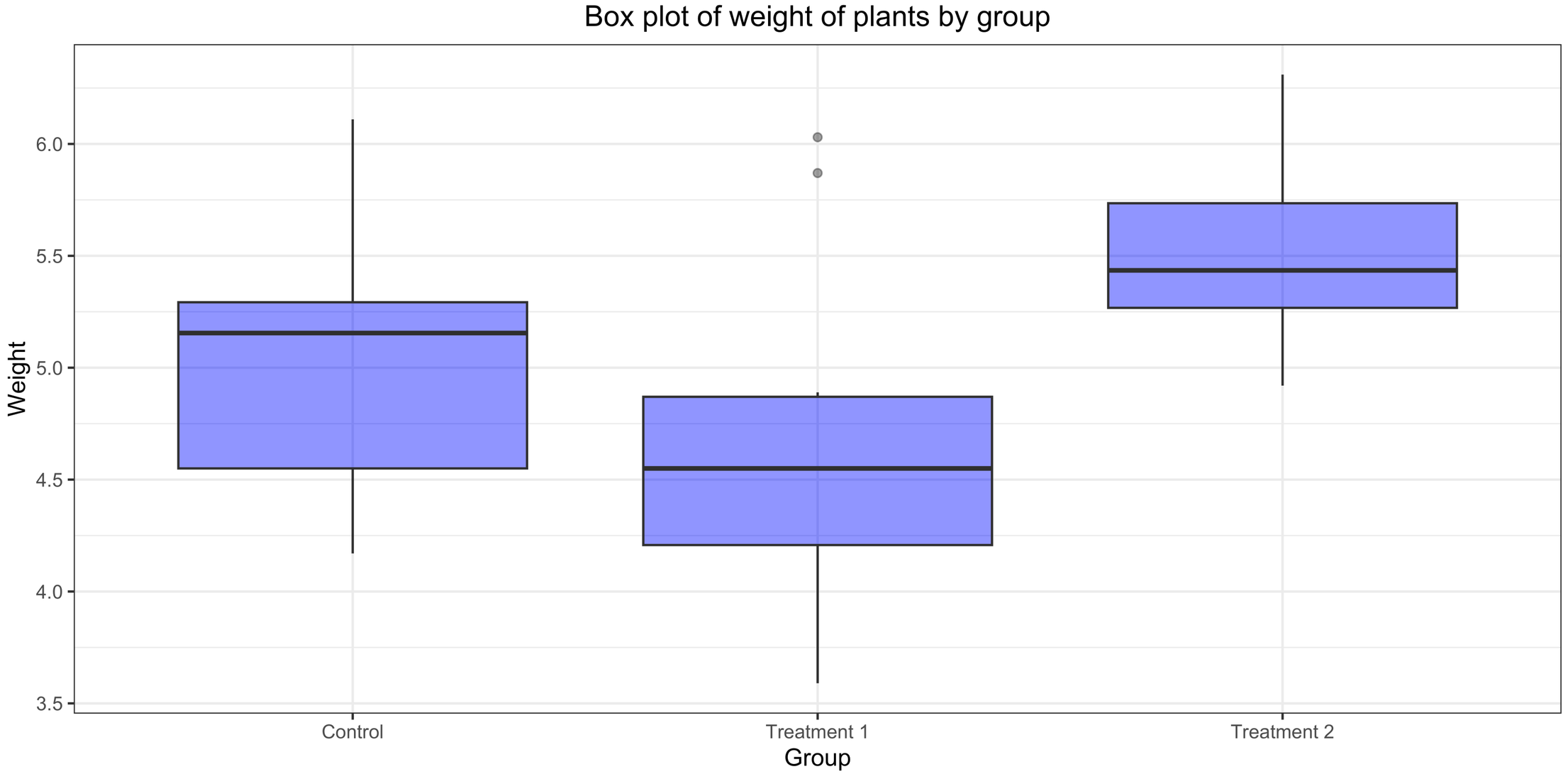
Example 1


Ozone (ppb) and temperature (degrees F) in New York, May to September 1973.
Temp
110
100
90
80
70
60
0
50
100
150
Ozone

Taste of ggplot2 package

By the end of the lecture, you will be able to create the figures like the following examples using `ggplot2` package.

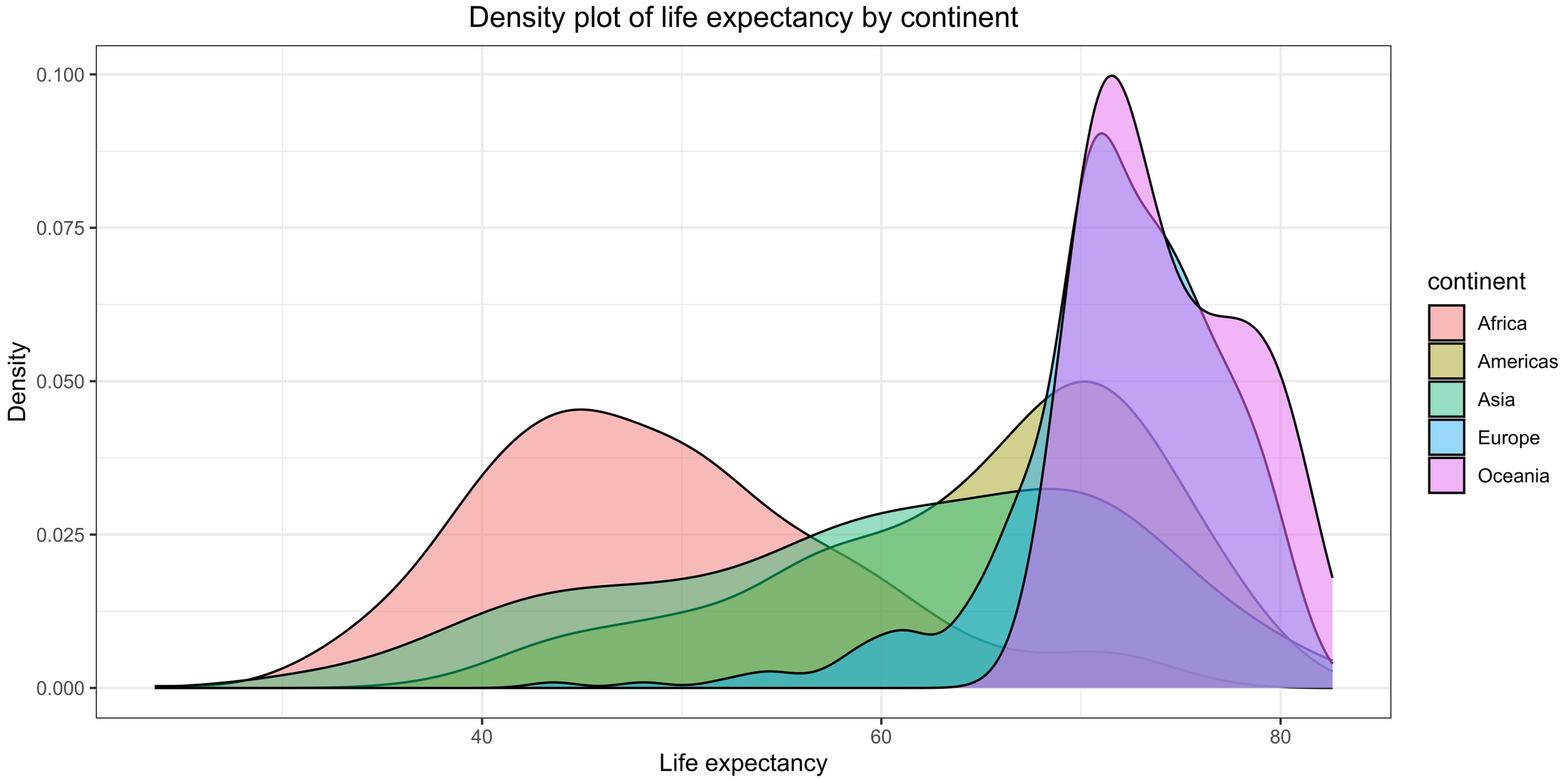
Example 2


Box plot of weight of plants by group
Weight
6.0
5.5
5.0
4.5
4.0
3.5
Control
Treatment 1
Treatment 2
Group

Taste of ggplot2 package

By the end of the lecture, you will be able to create the figures like the following examples using `ggplot2` package.

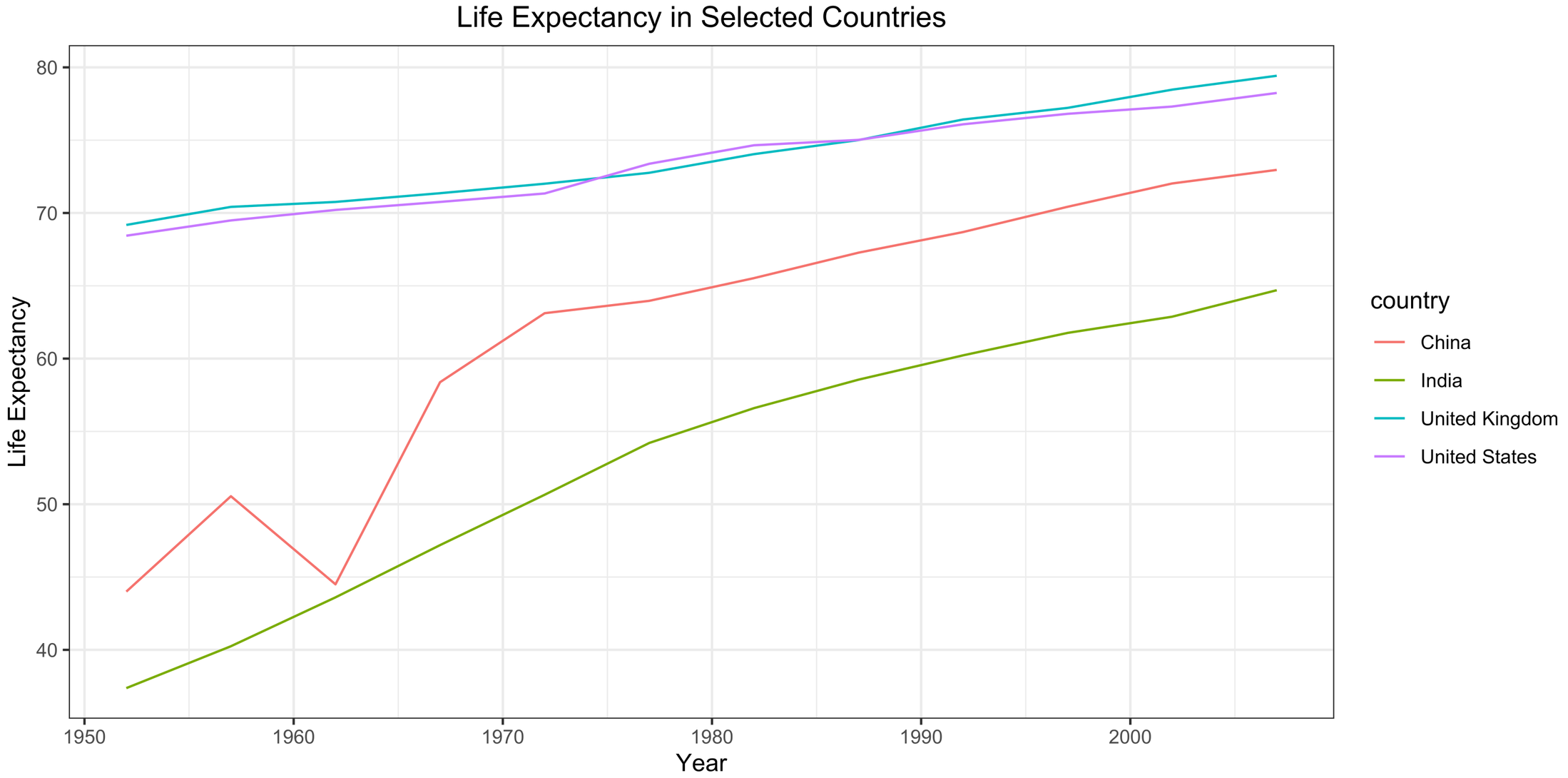
Example 3


Density plot of life expectancy by continent
0.100
0.075
0.050
0.025
0.000
Density
40
60
80
Life expectancy
continent
Africa
Americas
Asia
Europe
Oceania

Taste of ggplot2 package

By the end of the lecture, you will be able to create the figures like the following examples using `ggplot2` package.

Example 4

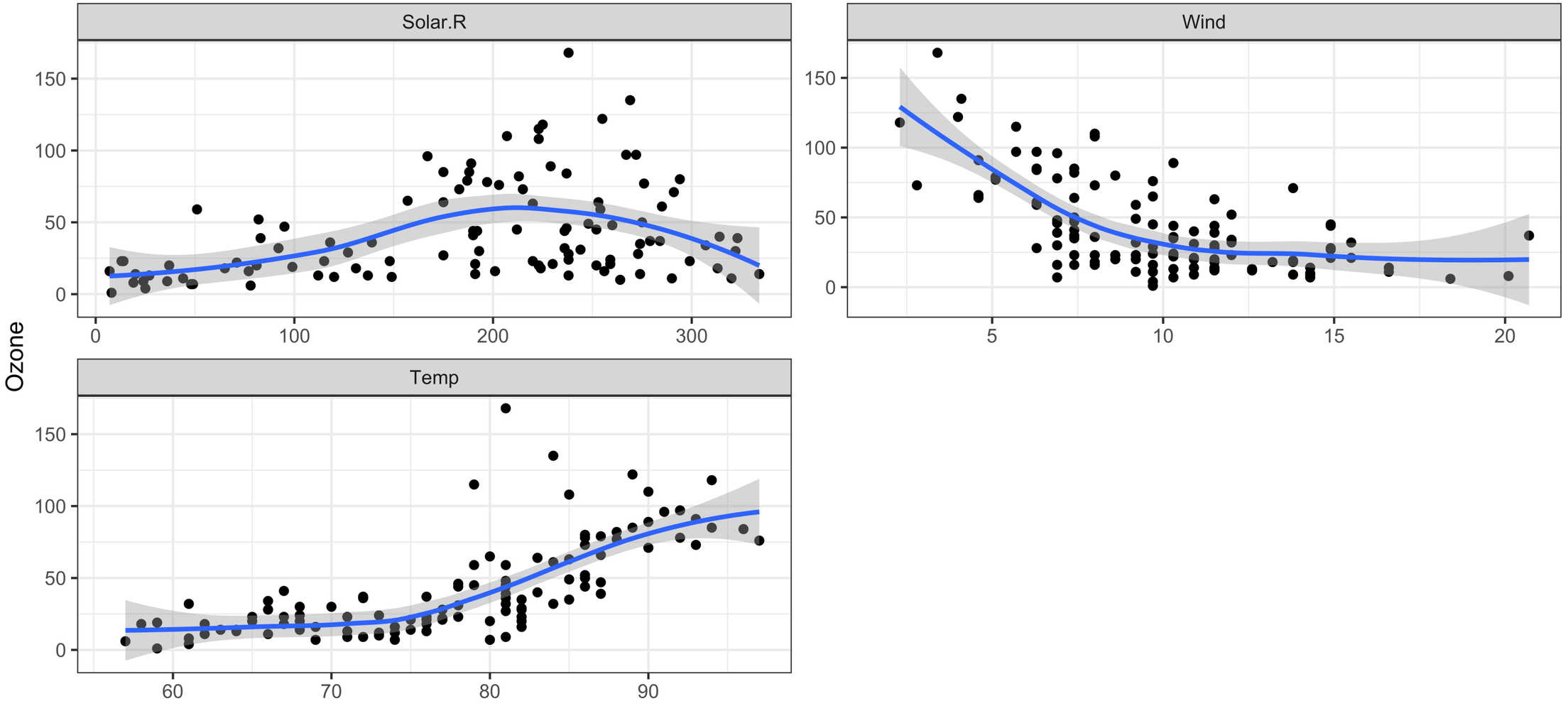

Life Expectancy in Selected Countries
Life Expectancy
80
70
60
50
40
1950
1960
1970
1980
1990
2000
Year
country
China
India
United Kingdom
United States

Taste of ggplot2 package

By the end of the lecture, you will be able to create the figures like the following examples using `ggplot2` package.

Example 5

Relationship between ozone and weather conditions in New York, May to September 1973



- There are many functions in the `ggplot2` package to create figures, and today's lecture is not a comprehensive guide to all of them.
- We will focus on the basic functions to create the most common types of figures.

Before Starting

Install the package `ggplot2` and `gapminder` locally if you haven't already done so.

```
install.packages('ggplot2')
install.packages('gapminder')
```

Once you have the package in R, let's load it.

Run Code

```
library(ggplot2)
library(gapminder)
```

Note

- There is a package called `tidyverse`, which is a collection of R packages designed for data science.
- When you load the `tidyverse` package, the `ggplot2` package is automatically loaded.

Introduction to `ggplot2`

What is it?

- As you know, there are already base (built-in) R functions to create figures (e.g., `plot()` and `hist()`)
  - pros: they are fast (especially for plotting a large dataset).
  - cons: The plots are difficult to customize.

- The `ggplot2` package provides more flexibility and customization options for creating figures with consistent syntax.
  - Check this out to see what kind of figures `ggplot2` can make.
- Variety of extensional packages built on top of `ggplot2` (e.g., `ggthemes`, `ggpubr`, `ggrepel`, `gganimate`, etc.) allows you to create more complex figures.
  - See this for examples.

Introduction to `ggplot2`

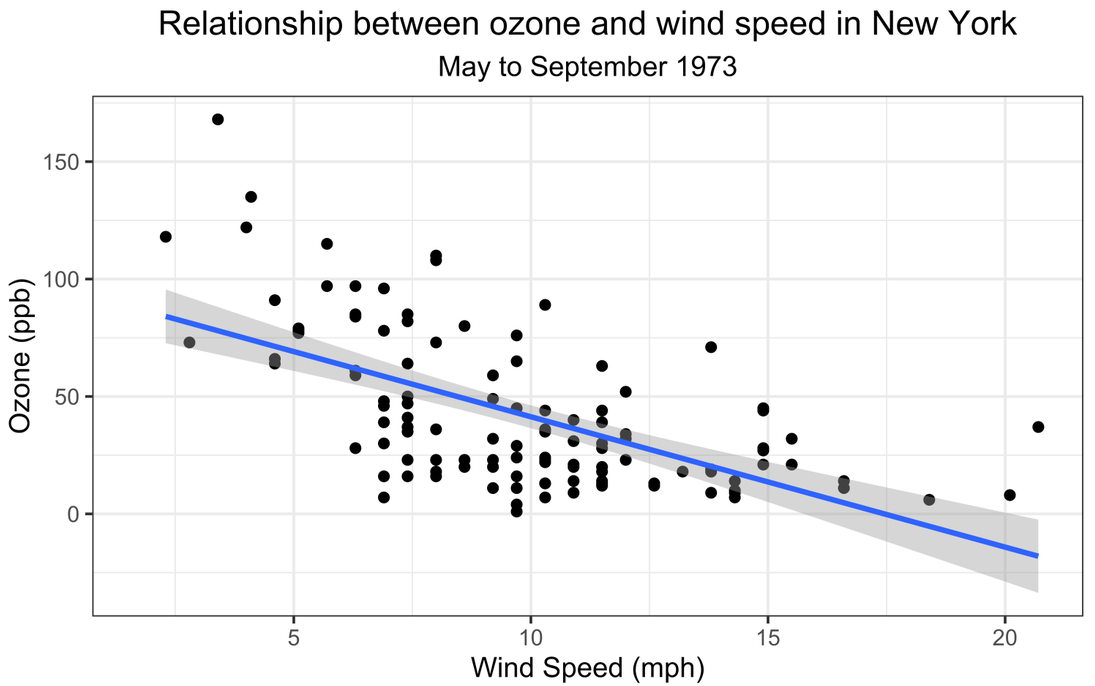
How does it work?

- `ggplot2` views a figure as the collection of multiple independent layers.
  - layers for geometric objects (e.g., points, lines, bars), layers for aesthetic attributes of the geometric objects (color, shape, size), layers of annotations and statistical summaries, … etc.
- Then, it combines these layers to create a single figure as a final output.

Anatomy of ggplot2

- Use the right-arrow (or down-arrow) key to move through the steps. The left column shows the code. The right column shows the plot it produces. Watch how the plot changes each time a new line of code is added.

```r
# Create a canvas for the plot
ggplot(data = airquality) +
  # Add x-axis
  aes(x = Wind) +
  # Add y-axis
  aes(y = Ozone) +
  # Add a scatter plot
  geom_point() +
  # Add a regression line
  geom_smooth(method = "lm") +
  # Change x-axis label
  labs(x = "Wind Speed (mph)") +
  # Change y-axis label
  labs(y = "Ozone (ppb)") +
  # Add title and subtitle
  labs(
    title = "Relationship between ozone and wind speed in New York",
    subtitle = "May to September 1973"
  ) +
  # Add caption
  labs(caption = "Data source:") +
  # Set the theme
  theme_bw() +
  # Center the title and subtitle position
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element text(hjust = 0.5)
```



- Note: This code is for demonstration purposes. Don’t imitate this code!

Anatomy of ggplot2 (continued)

- Every ggplot2 plot has three key components:
  - Data
  - A set of aesthetic mappings between variables in the data and visual properties.
  - At least one layer which describes how to render each observation. Layers are usually created with a `geom` function.

The very general syntax for creating a plot with `ggplot2` is as follows:

```
ggplot(data = ...) +
  geom_*(aes( ... ))
```

- `aes` stands for aesthetic mappings. It tells `ggplot2` how to map variables in the data to visual properties of the plot (e.g., x-axis, y-axis, color, shape, size, etc.)
- `+` operator tells R that you're adding another layer (e.g., line plot) to the current "canvas".
- Depending on the type of the figure you want to plot, use different `geom_*()` functions.
  - Eg. `geom_point()` for scatter plot, `geom_line()` for line plot, etc.

Example

Data

Let's use the `airquality` data for this example.

- `airquality` data is a built-in dataset in R. So, you don't need to load it.
- Type `airquality` in the console to see the data. (Type `?airquality` in the console for more information.)

Run Code

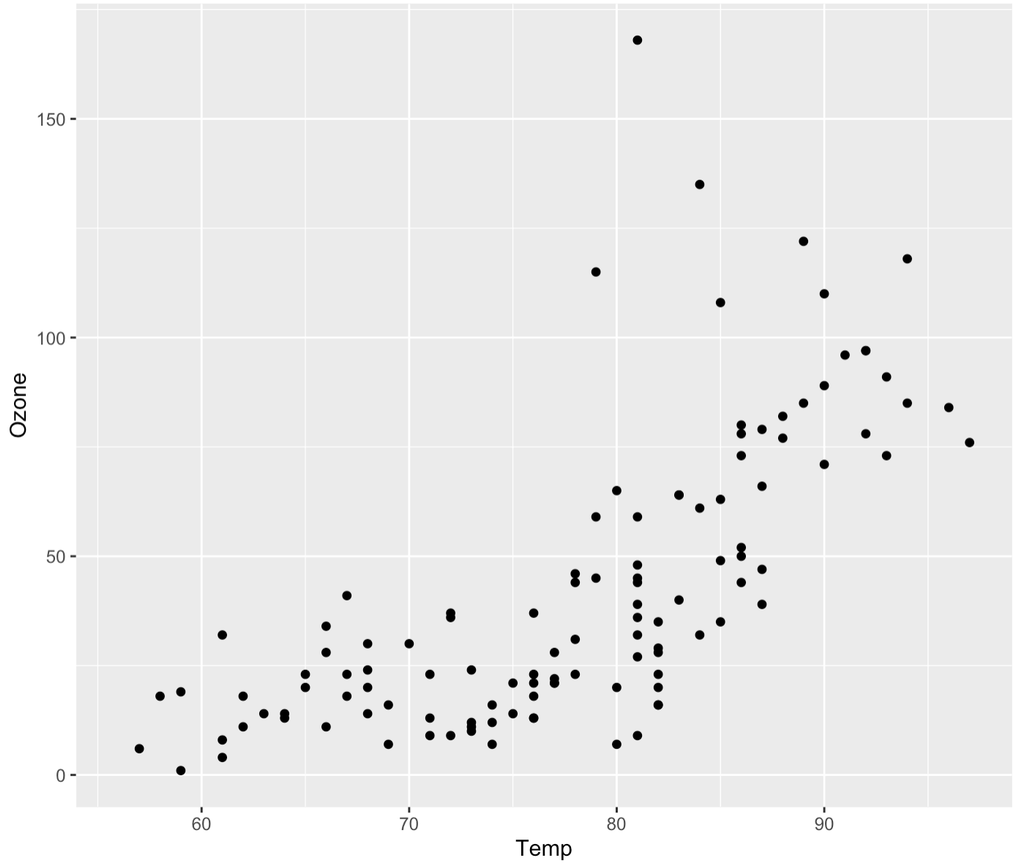
```
?airquality

# Take a look at the first few rows of the data
head(airquality)
```

Example

Let’s Create a Scatter Plot

We will create a scatter plot of `Ozone` (ozone level in the air) and `Temp` (Maximum daily temperature in degrees F) from the `airquality` data.

The final plot should look like the following:



Example

Step 1

Step 1: Start with `ggplot()`

- `ggplot(data = dataset)` initializes a ggplot object. In other words, it prepares a “canvas” for the plot.
- Here, let R know the dataset you are trying to visualize.

Try it! Why?

Run the following code. Can you see any output?

Run Code

```
1  ggplot(data = airquality)
```

Example

Step 2

Step 2: Draw figures with `geom_*()` functions, and add to the current canvas use + operator

- For example, we use `geom_point()` to create a scatter plot.
  - use `aes()` to specify which variable you want to use for x and y axis.
- `aes()` is used to tell R to look for the variables inside the dataset you specified in `ggplot()`, and use the information as specified.
- e.g., `aes(x = Temp, y = Ozone)` tells R to look for `Temp` and `Ozone` in the data, and to map the data to x-axis and y-axis, respectively.

Run Code

```
# You might see a warning message. Don't worry about it!
ggplot(data = airquality) +
  geom_point(aes(x = Temp, y = Ozone))
```

Summary

These are basic steps to create a figure with `ggplot2` package.

- Step1: Start with `ggplot()`
  - This function prepares a “canvas” for the figure.
- Step2: Draw a figure with `geom_*()` function, and add to the current canvas with `+` operator.
- Step3: Repeat Step2 and Step3 to add whatever layers you want to add.
- Step4 (optional): Add labels, titles, and other annotations to the plot with `labs()`, `theme()`, etc.

- Don’t forget to specify x and y variables in the `aes()` function.
  - Also, some `geom_*()` functions only require `x` variable (e.g., `geom_histogram()`).
- In step 3, layers can be added in any order, but the order of the layers affects the final appearance of the plot.
- When you want to make a simple x and y plot, the base R functions are sufficient (e.g., `with(data, plot(column_x, column_y))`)

In-class exercise

Questions

1. Create a scatter plot of `Temp` and `Wind` from the `airquality` data.
2. In the plot you just created, let's change the x-axis label to "Maximum temperature (degrees F)" and the y-axis label to "Wind Speed (mph)". For this, use `labs()` function.

Hint:

- `labs(x = new_x_label, y = new_y_label)`
- use `+` to add this layer to the plot.

Run Code

```
1  # Your code here
```

In-class exercise

Answers

1. Create a scatter plot of `Temp` and `Wind` from the `airquality` data.
2. In the plot you just created, let's change the x-axis label to "Maximum temperature (degrees F)" and the y-axis label to "Wind Speed (mph)". For this, use `labs()` function.

Hint:

- `labs(x = new_x_label, y = new_y_label)`
- use `+` to add this layer to the plot.

Run Code

```
ggplot(data = airquality) +
  geom_point(aes(x = Temp, y = Wind)) +
  labs(x = "Maximum temperature (degrees F)", y = "Wind Speed (mph)")
```

Different Types of Plot

- You can create various plots with the ggplot2 package by choosing the appropriate `geom_*()` function for the desired plot type.
- Here are some of the most commonly used `geom_*()` functions.
  - `geom_point()`: scatter plot
  - `geom_line()`: line plot
  - `geom_bar()`: bar plot
  - `geom_boxplot()`: box plot
  - `geom_histogram()`: histogram
  - `geom_density()`: density plot
    - This computes and draws kernel density estimates, and is a smoothed version of the histogram.
  - `geom_smooth()`: draws an OLS-estimated regression line (other regression methods available)
- see this for full list of `geom_*()`

Modify Aesthetic Attributes

Basics

We can modify how plots look by specifying color, shape, and size.

Here are list of options to control the aesthetics of figures. You use these options inside the `geom_*()`.

- `size`: control the size of points and text
  - e.g., `geom_point(size = 3)`
- `color`: control color of the points and lines
  - e.g.,`geom_point(color = "blue")`
- `fill`: control the color of the inside areas of figures like bars and boxes
  - e.g., `geom_density(fill = "blue")` fills the area under the density curve with blue color
- `alpha` controls the transparency of the fill color
  - e.g., `alpha=1` is opaque, `alpha=0` is completely transparent, usually between 0 and 1
- `shape`: controls the symbols of point, it takes integer values between 0 and 25
  - e.g., `geom_point(shape = 1)` for circle, `geom_point(shape = 2)` for triangle

- For point shapes available in R, see this.
- For further information about the options for aesthetics, see this.

Modify Aesthetic Attributes

Examples

Scatter Plot | Histogram | Line Plot

- `size = 3`: makes the points larger.
- `color = "red"`: changes the color of the points to red.
- `shape = 1`: changes the shape of the points to circle.

Run Code

```
1  ggplot(data = airquality) +
2    geom_point(aes(x = Temp, y = Ozone), size = 3, color = "red ", shape = 1)
```

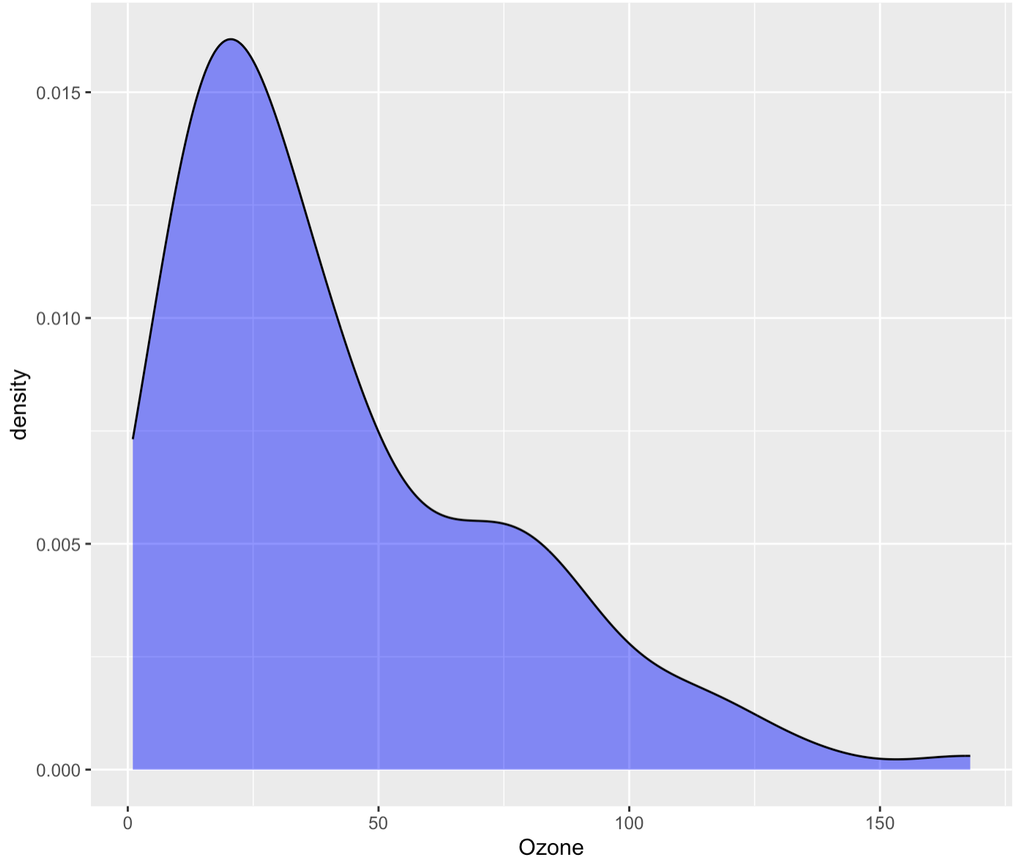
In-class Exercise

Exercise 1

Questions | Answers

Create a density plot of `Ozone` from the `airquality` data. Fill the area under the density curve with blue and make it semi-transparent (use `alpha = 0.5`).

The figure should look like the following:



Run Code

```
1  # Your code here
```

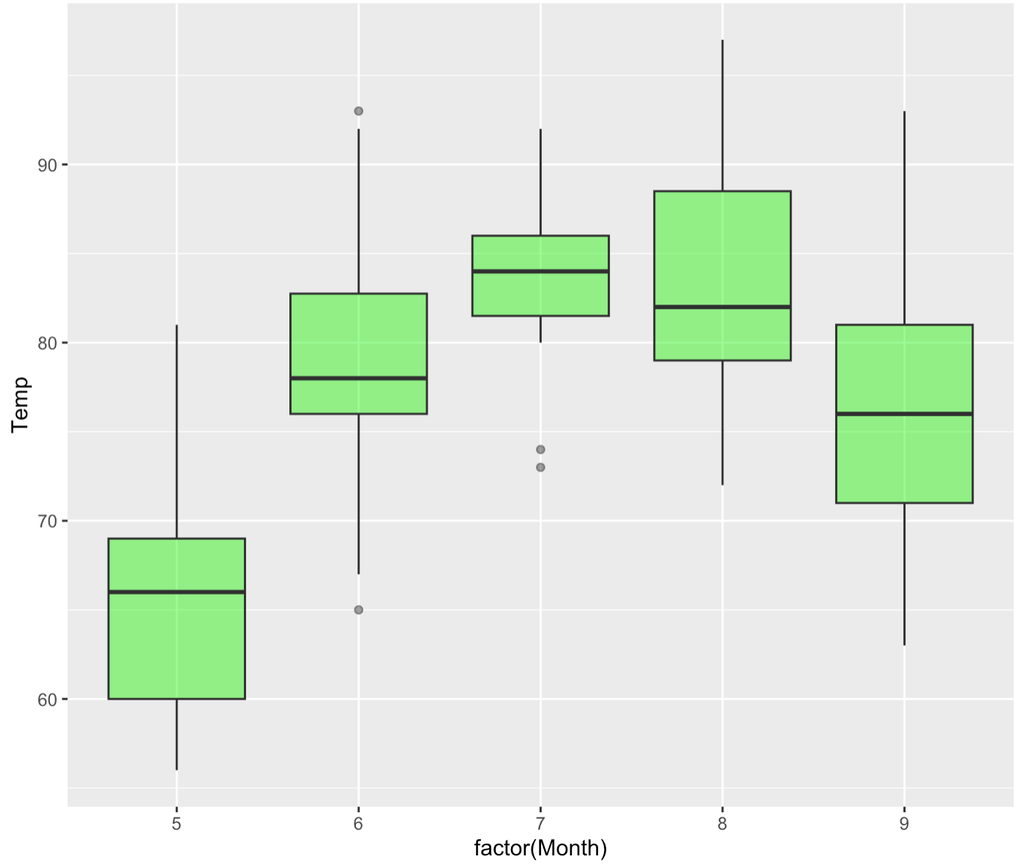
In-class Exercise

Exercise 2

Questions | Answers

Create box plots of monthly `Temp` from the `airquality` data. Fill the boxes with green and make it semi-transparent (use `alpha = 0.5`).

The figure should look like the following:



Hint

- This is a bit of a tricky problem, but very useful!
- We want to use `Month` as a categorical variable for the x-axis, but `Month` is a numeric variable in the data. How can we tell R to use it as a categorical (factor) variable?
  - Apply `factor()` function to a `Month` to convert it to a factor variable in `aes()` in the `geom_*()` function.

Run Code

```
1  # Your code here
```

Group Aesthetic

Basics

So far, we specified aesthetic attributes outside of the `aes()` function. Consequently, all the geometric objects in the plot have the same color, shape, and size, etc.

- e.g., `geom_point(aes(x = var_x, y = var_y), color = "red")`.

If you use those options inside the `aes()` function like `aes(color = var_z)`, R will display different colors by group based on the value of `var_z`. Usually `var_z` is a categorical variable.

- e.g., `geom_point(aes(x = var_x, y = var_y, color = var_z))` displays a scatter plot where the points are colored differently based on the value of `var_z`.

Example

Let's create density plots of `Temp` for each month, and use different colors (`fill` in this case) for different `Month`.

Run Code

```
# Now, fill color depends on factor(Month)
ggplot(data = airquality) +
  geom_density(aes(x = Temp, fill = factor(Month)), alpha = 0.5)
```

Group Aesthetic

In-class Exercise | In-class Exercise

Questions

1. Create a scatter plot of `Ozone` and `Temp` in the `airquality` data. Let's use different colors for different `Month`.
2. In addition to the previous plot, let's use different shapes for different `Month`.

NOTE: Remember that we need to tell R to use `Month` as a categorical variable.

Run Code

```
1  # Your code here
```

Collective geoms

Basics

So far, we used only one `geom_*()` function in a plot. But you can use multiple `geom_*()` functions in a single plot.

- This just overlays multiple layers of different geometric objects on the same “canvas”.
- Use `+` operator to add multiple `geom_*()` functions to the plot.

Example Syntax

```
ggplot(data = dataset) +
  geom_*(aes(x = column_x, y = column_y, fill = column_z)) +
  geom_*(aes(x = column_x, y = column_y)) +
  geom_*(aes(x = column_x, y = column_y)) +
  ...
```

If an additional layer has the same `aes()` mapping, you can specify it only once in the `ggplot()`.

```
# The above code is equivalent to the following code
ggplot(data = dataset, aes(x = column_x, y = column_y) +
  geom_*(fill = column_z)) +
  geom_*() +
  geom_*() +
  ...
```

Note

- Recall that `ggplot()` prepares a plot object.
- If you tell `ggplot()` to use `aes()` mapping from the beginning, you don’t need to specify it again in the `geom_*()` functions.

Collective geoms

Example

- Let's create a scatter plot of `Ozone` and `Temp` from the `airquality` data.
- In addition to the scatter plot, let's add a simple regression line to the plot using `geom_smooth()` function.

Run Code

```
ggplot(data = airquality, aes(x = Temp, y = Ozone)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x)

# The above code is equivalent to the following code
# ggplot(data = airquality) +
#   geom_point(aes(x = Temp, y = Ozone)) +
#   geom_smooth(aes(x = Temp, y = Ozone), method = "lm")
```

Modify Axis, Legend, and Plot Labels

Basics

- By default, x-axis, y-axis, and legend labels are the column names of the data, which are not always informative. Also, you might want to add a title and subtitle to the plot.
- You can modify the labels, titles, and other annotations of the plot using `labs()` function.

Example Syntax

```
ggplot(data = dataset) +
  geom_*(aes(x = column_x, y = column_y)) +
  labs(
    x = "X-axis label",
    y = "Y-axis label",
    title = "Title of the plot",
    subtitle = "Subtitle of the plot",
    caption = "Data source"
  )
```

Modify Axis, Legend, and Plot Labels

Examples | Examples

Example 1

Run Code

```
ggplot(data = airquality) +
  geom_point(aes(x = Temp, y = Ozone, color = factor(Month))) +
  labs(
    x = "Maximum temperature (degrees F)",
    y = "Ozone level (ppb)",
    title = "Relationship between ozone and temperature",
    color = "Month",
    subtitle = "May to September 1973, New York"
  )
```

Summary

Let's summarize what we have learned so far.

- the basic syntax of the `ggplot2` package.
- how to create a popular types of plots (scatter plot, line plot, bar plot, histogram, box plot, density plot).
- how to modify aesthetic attributes of the plot (color, shape, size, etc.)
- how to use group aesthetic to group the data by a variable
- how to use multiple `geom_*()` functions in a single plot.
- how to modify axis, legend, and plot labels with `labs()` function.

Exercise Problems

Exercise Problems 1

Data | Instructions | Solutions

Let's use the `economics` data, which is a dataset built into the `ggplot2` package. It was produced from US economic time series data available from Federal Reserve Economic Data. This contains the following variables:

- `date`: date in year-month format
- `pce`: personal consumption expenditures, in billions of dollars
- `pop`: total population in thousands
- `psavert`: personal savings rate
- `uempmed`: median duration of unemployment in weeks
- `unemploy`: number of unemployed in thousands

Exercise Problems

Exercise Problems 2

Data | Instructions | Solutions

For this exercise problem, we will use medical cost personal datasets descried in the book “Machine Learning with R” by Brett Lantz. The dataset provides $1,338$ records of medical information and costs billed by health insurance companies in 2013, compiled by the United States Census Bureau.

The dataset contains the following variables:

- `age`: age of primary beneficiary
- `sex`: insurance contractor gender, female, male
- `bmi`: body mass index, providing an understanding of body, weights that are relatively high or low relative to height
- `children`: number of children covered by health insurance
- `smoker`: smoking
- `region`: the beneficiary’s residential area in the US; northeast, southeast, southwest, northwest.
- `charges`: individual medical costs billed by health insurance

Download the data

Run Code

```
# === Download Data (Don't worry about this part.) === #
insurance_url <- "https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master/insurance.csv"
insurance <- fread(insurance_url) #You can use import() function of the rio package!
```

```
4
5  # fread() is a function from the data.table package used for fast and efficient reading of data from text
   files, particularly large CSV files.
6
7  # Of course, you can also use the import function from the rio package (this only works in RStudio, not
   here). For example, run
8  # insurance <- rio::import(insurance_url)
9
10 # === Take a look at the data === #
11 head(insurance)
```

```
     age    sex    bmi children smoker    region   charges
   <int> <char>  <num>    <int> <char>    <char>     <num>
1:    19 female 27.900        0    yes southwest 16884.924
2:    18   male 33.770        1     no southeast  1725.552
3:    28   male 33.000        3     no southeast  4449.462
4:    33   male 22.705        0     no northwest 21984.471
5:    32   male 28.880        0     no northwest  3866.855
6:    31 female 25.740        0     no southeast  3756.622
```

Section 2: Advanced Topics

Before We Start

For this section, we will continue to use the `economics` and `insurance` data we used in the previous exercise problems.

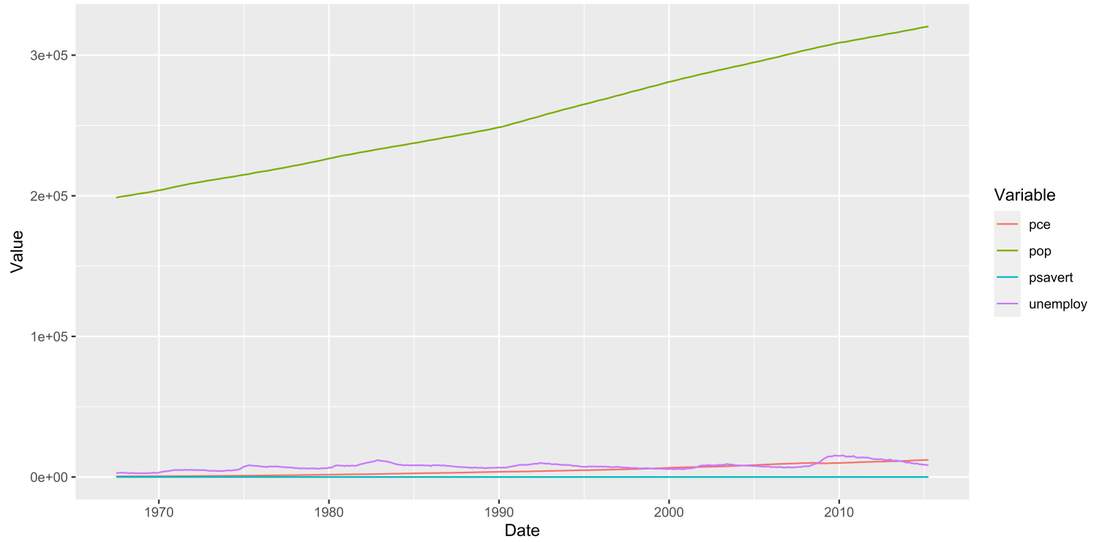
Facet Plot

Intro

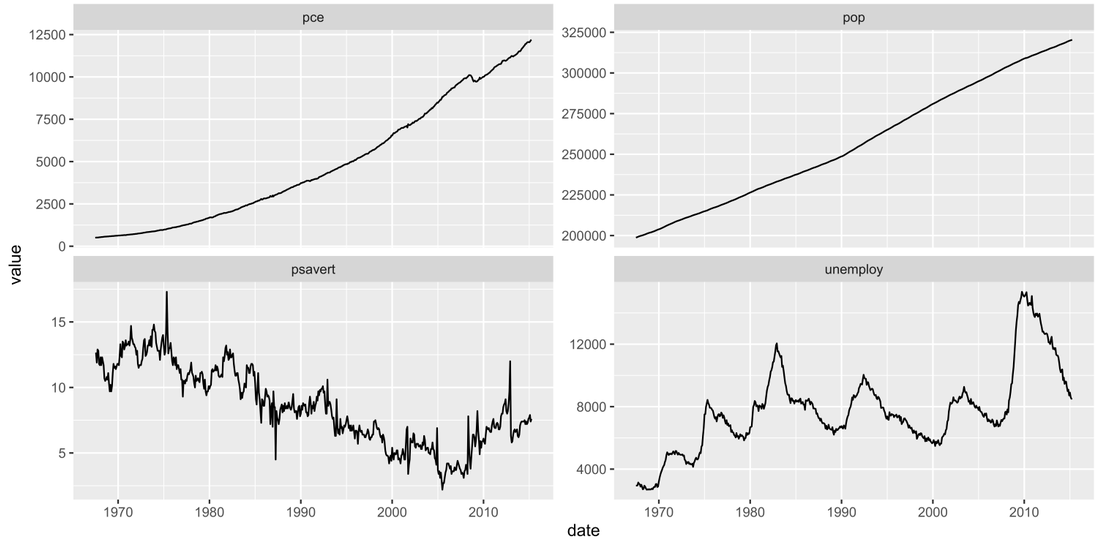
You can partition a plot into a matrix of panels and display a different subset of the data in each panel. This is useful when you want to compare patterns in the data by group.

Example 1 | Example 2

Without faceting



With faceting



Because the scales of the y-axis are different by variable, it is hard to compare the trends across variables in the same plot.

Facet Plot

facet_wrap()

`facet_wrap()` makes a long ribbon of panels (generated by any number of variables). You can also wrap it into 2 rows.

Syntax:

```
facet_wrap(vars(var_x, var_y), scales = "fixed", nrow = 2, ncol = 2)
```

- Inside `vars()`, specify variables used for faceting groups.
- `ncol` and `nrow` control the number of columns and rows (you only need to set one).
- `scales` controls the scales of the axes in the panel (either `"fixed"` (the default), `"free_x"`, or `"free_y"`, `"free"`).

Try it!

Play around with the `facet_wrap()` function in the code below. See how the choice of faceting groups, number of rows and columns and the scales of the axes affect the appearance of the plot.

Run Code

```
base_plot <- ggplot(insurance)+
  geom_histogram(aes(x = charges), fill = "blue", alpha = 0.5) +
  facet_wrap(vars(region), nrow = 1, scales = "fixed")

# try vars(region, sex)
```

Facet Plot

facet_grid()

`facet_grid()` produces a 2 row grid of panels defined by variables which form the rows and columns.

Syntax:

```
facet_grid(rows = vars(var_x), cols = var(var_y)), scales = "fixed")
```

- The graph is partitioned by the levels of the groups `var_x` and `var_y` in the rows and columns, respectively.
- `ncol` and `nrow` control the number of columns and rows (you only need to set one).
- `scales` controls the scales of the axes in the panel (either `fixed` (the default), `free_x`, or `free_y`, `free`).

Try it!

Run Code

```
ggplot(insurance)+
  geom_histogram(aes(x = charges), fill = "blue", alpha = 0.5) +
  facet_grid(rows = vars(sex), cols = vars(region), scales = "fixed")
```

facet_wrap() vs facet_grid()

Basics

So, when should you use `facet_wrap()` and `facet_grid()`?

1. In my opinion, if you have a single variable to make a facet, you should use `facet_wrap()`. Unlike `facet_grid()`, `facet_wrap()` can control the number of rows and columns in the panel.
2. If you have two variables to make a facet, you should use `facet_grid()`.
3. In `facet_grid()`, you don't always need to provide both `rows` and `columns` variables. If only one is specified, the produced plot will look like the one from `facet_wrap()` but you cannot wrap the panels into 2 rows.

facet_wrap() vs facet_grid()

Example

One Faceting Variables

Run Code

```
# Prepare the base histogram plot
base <- ggplot(insurance)+
  geom_histogram(aes(x = charges), fill = "blue", alpha = 0.5)

# === fact_wrap === #
base +
  facet_wrap(vars(region), scales = "fixed", nrow = 2)

# === fact_grid === #
base +
  facet_grid(cols = vars(region), scales = "fixed") # use vars(region) for the rows
# Note that you cannot wrap the panels into 2 rows.


base +
  facet_grid(rows = vars(region), scales = "fixed") # use vars(region) for the rows
```

facet_wrap() vs facet_grid()

Example

Two Faceting Variables

Run Code

```
# Prepare the base histogram plot
base <- ggplot(insurance)+
  geom_histogram(aes(x = charges), fill = "blue", alpha = 0.5)

# === fact_wrap === #
base +
  facet_wrap(vars(region, sex), scales = "fixed", nrow = 2)

# === fact_grid === #
base +
  facet_grid(rows = vars(sex), cols = vars(region), scales = "fixed")
```

How can we modify the facet labels?

See the following document: How can I set different axis labels for facets?. You can use the `labeller` argument in the `facet_wrap()` and `facet_grid()` function to modify the facet labels.

Here, I will show another way to modify the facet labels. You can

Preparation

First, I re-define region and sex as factor variables. **In doing so, I will add labels for each level of the variables.** If labels are attached to the variables, ggplot use those names in the facet labels.

Run Code

```
# To use the following code, make sure that your insurance data is data.table class.
insurance[,`:=`(
  # --- re-define region as factor variable --- #
  region =
    factor(
      region,
      levels = c("northeast", "northwest", "southeast", "southwest"),
      labels = c("North East", "North West", "South East", "South West")
    ),
  # --- re-define sex as factor variable --- #
  sex =
    factor(
      sex,
      levels = c("female", "male"),
      labels = c("Female", "Male")
    )
```

```
17      )]
```

How can we modify the facet labels?

See the following document: How can I set different axis labels for facets?. You can use the `labeller` argument in the `facet_wrap()` and `facet_grid()` function to modify the facet labels.

Here, I will show another way to modify the facet labels. You can

facet_wrap

Now, the facet labels are changed to “North East”, “North West”, “South East”, and “South West” for the `region` variable.

Run Code

```
ggplot(insurance)+
  geom_histogram(aes(x = charges), fill = "blue", alpha = 0.5)+
  facet_wrap(vars(region), nrow = 2)
```

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

How can we modify the facet labels?

See the following document: How can I set different axis labels for facets?. You can use the `labeller` argument in the `facet_wrap()` and `facet_grid()` function to modify the facet labels.

Here, I will show another way to modify the facet labels. You can

facet_grid

Run Code

```
ggplot(insurance)+
  geom_histogram(aes(x = charges), fill = "blue", alpha = 0.5)+
  facet_grid(rows = vars(region), cols = vars(sex))
```

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

Multiple Datasets in One Figure

Basics

So far, we have been using the same dataset for each layer of the plot. But you can use multiple datasets in a single plot.

Note

- If you specify data in `ggplot()` at the beginning (e.g., `ggplot(data = dataset)`), the data applies to ALL the subsequent `geom_*()`s unless overwritten locally inside individual `geom_*()`s.
- To use multiple datasets in a single plot, you just need to specify what dataset to use locally inside individual `geom_*()`s.

Multiple Datasets in One Figure

Example

- `insurance_southwest` is a subset of the `insurance` data where `region` is `southwest`.
- `insurance_northeast` is a subset of the `insurance` data where `region` is `northeast`.

Run Code

```
insurance_southwest <- insurance[region == "South West",]
insurance_northeast <- insurance[region == "North East",]

# === Create economics_month dataset === #
ggplot() +
  geom_point(
    data = insurance_southwest,
    aes(x = bmi, y = charges),
    color = "blue"
  ) +
  geom_point(
    data = insurance_northeast,
    aes(x = bmi, y = charges),
    color = "red"
  )
```

Multiple Datasets in One Figure

How can we make the legend?

You can do something like this:

Run Code

```
insurance_southwest <- insurance[region == "South West",]
insurance_northeast <- insurance[region == "North East",]

# === Create economics_month dataset === #
ggplot() +
  geom_point(
    data = insurance_southwest,
    aes(x = bmi, y = charges, color = "South West"),
  ) +
  geom_point(
    data = insurance_northeast,
    aes(x = bmi, y = charges, color = "North East")
  ) +
  labs(color = "Region")
```

ggplot2 Themes (Optional)

You can change the theme of the plot.

- `ggplot2` ships several pre-made themes that you can apply to your plots. (e.g, `theme_minimal()`, `theme_bw()` (I use this often), `theme_classic()`). See this.
- `ggthemes` package provides additional ggplot themes. See this for full list of available themes.

Try it!

Run Code

```
library(ggthemes)

ggplot(data = insurance) +
  geom_point(aes(x = bmi, y = charges)) +
  theme_stata()
```

theme() Function (Optional)

`theme()` function let you tweak the details of all non-data related components of a plot (e.g., font type in the plot, position of the legend and title, etc.). There are so many components you can modify with the `theme()` function. See this for full list of options.

For more information, see:

- Chapter 17 Themes, ggplot2: Elegant Graphics for Data Analysis (3e)

Try it!

For example, you can change the position of the title and legend with the following `theme()` options.

Run Code

```
ggplot(data = insurance) +
  geom_point(aes(x = bmi, y = charges, color = region)) +
  labs(titile = "Scatter plot of BMI and smoke status") +
  theme(
    plot.title = element_text(hjust = 0.5), #center the title
    legend.position = "bottom" # move the legend to the bottom
  )
```

Save the Plot

Basics

Two options + Use the `ggsave()` function from the `ggplot2` package. + Use the “Export” button in the RStudio plot viewer.

Syntax:

```
ggsave(filename, plot = plot_object)
```

- `filename`: the name of the file (including path) to save the plot to. (e.g., filename = “Data/plot.png”)
- `plot`: the plot object to save.

Example

Run the following code on your RStudio. Make sure you are opening the RProject.

```
library(ggplot2)
library(rio)

insurance_url <- "https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master/insurance.csv"
insurance <- import(insurance_url)

ggplot(data = insurance) +
  geom_boxplot(aes(x = sex, y = charges, fill = region)) +
  labs(
    x = "Sex",
    y = "Medical costs",
    title = "Distribution of individual medical mosts by sex and region"
  )

# --- Sve plot --- #
ggsave(filename = "Data/insurance.png")
ggsave(filename = "Data/insurance.pdf")
ggsave(filename = "Data/insurance2.png", plot = plot_insurance)
```

:::

Summary 2

For this second section, you learned a few advanced topics in ggplot2.

Now, you know;

- how to create facet plots with `facet_wrap()` and `facet_grid()`.
- when to use `facet_wrap()` and `facet_grid()`.
- how to visualize multiple datasets in a single plot.
- how to save the plot.

That's it!

Exercise Problems

Exercise Problem 1 (basic) | Exercise Problem 2 (basic) | Exercise Problem 3 (challenging)

Data | Instructions | Solutions

For this exercise problem, you will use the `gapminder` data from the `gapminder` package.

Run Code

```
# install the gapminder package if you haven't done so.
# install.packages("gapminder")

# Load the data from the gapminder package
data(gapminder, package="gapminder")

gapminder <- as.data.table(gapminder) # Convert the data to a data.table object
```