Introduction to the Grammar of Graphics

Prof. Wells

STA 209, 1/30/23

Outline

In this lecture, we will. . .

Outline

In this lecture, we will...

- Discuss the Grammar of Graphics
- Decompose particular graphics using the GG paradigm
- Introduce the `ggplot2` package for R graphics

Section 1

The Grammar of Graphics

The Guiding Principle

A statistical graphic is a mapping of `data` variables to `aesthetic` attributes of `geometric` objects.

The Guiding Principle

A statistical graphic is a mapping of `data` variables to `aesthetic` attributes of `geometric` objects.

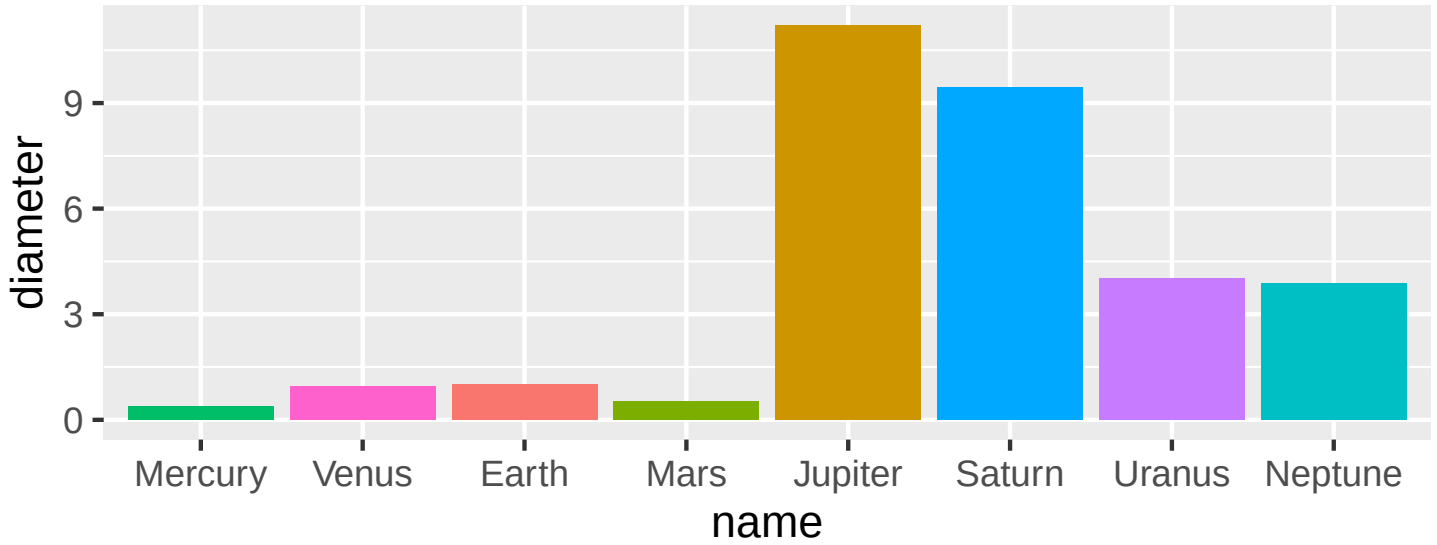
data	aesthetics	geometric object
Planet Name	x position	bar
Planet Diameter	y height	bar
Planet Name	color	bar

The Guiding Principle

A statistical graphic is a mapping of data variables to aesthetic attributes of geometric objects.

data	aesthetics	geometric object
Planet Name	x position	bar
Planet Diameter	y height	bar
Planet Name	color	bar

```
ggplot(data = planets_df) +
  geom_bar(stat = "identity",  mapping = aes(x = name, y = diameter, fill  = name)
```



GG Definitions

- **data**: data frame that contains the raw data and variables of interest

GG Definitions

- **data**: data frame that contains the raw data and variables of interest
- **geom**: geometric shape that the data are mapped to. Can be points, lines, bars, etc.

GG Definitions

- **data**: data frame that contains the raw data and variables of interest
- **geom**: geometric shape that the data are mapped to. Can be points, lines, bars, etc.
- **aesthetic**: visual properties of the **geom** object, like x position, y position, color, fill, shape

GG Definitions

- **data**: data frame that contains the raw data and variables of interest
- **geom**: geometric shape that the data are mapped to. Can be points, lines, bars, etc.
- **aesthetic**: visual properties of the **geom** object, like x position, y position, color, fill, shape
- **scale**: controls how data are mapped to the visual values of the aesthetic.
  i.e. specifying particular colors or shapes

GG Definitions

- **data**: data frame that contains the raw data and variables of interest
- **geom**: geometric shape that the data are mapped to. Can be points, lines, bars, etc.
- **aesthetic**: visual properties of the **geom** object, like x position, y position, color, fill, shape
- **scale**: controls how data are mapped to the visual values of the aesthetic.
  i.e. specifying particular colors or shapes
- **guide**: a legend to help user convert visual display back to the data

Plotting the Planets

Consider the planets data frame, `planets_df`:

name	type	diameter	rotation	rings	distance
Mercury	Terrestrial planet	0.382	58.64	FALSE	0.4
Venus	Terrestrial planet	0.949	-243.02	FALSE	0.7
Earth	Terrestrial planet	1.000	1.00	FALSE	1.0
Mars	Terrestrial planet	0.532	1.03	FALSE	1.5
Jupiter	Gas giant	11.209	0.41	TRUE	5.2
Saturn	Gas giant	9.449	0.43	TRUE	9.5
Uranus	Gas giant	4.007	-0.72	TRUE	19.2
Neptune	Gas giant	3.883	0.67	TRUE	30.1

Plotting the Planets

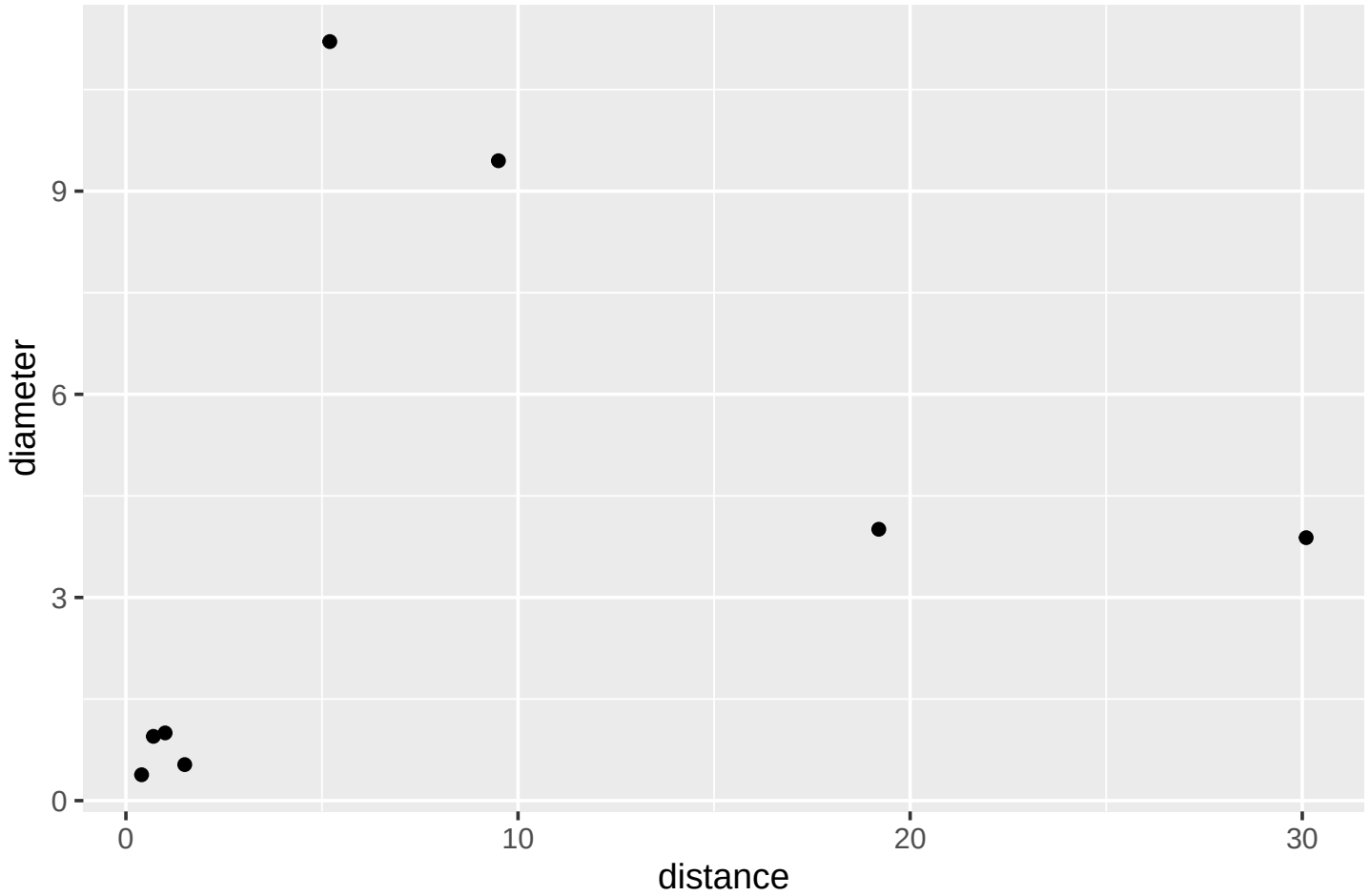
Consider the planets data frame, `planets_df`:

name	type	diameter	rotation	rings	distance
Mercury	Terrestrial planet	0.382	58.64	FALSE	0.4
Venus	Terrestrial planet	0.949	-243.02	FALSE	0.7
Earth	Terrestrial planet	1.000	1.00	FALSE	1.0
Mars	Terrestrial planet	0.532	1.03	FALSE	1.5
Jupiter	Gas giant	11.209	0.41	TRUE	5.2
Saturn	Gas giant	9.449	0.43	TRUE	9.5
Uranus	Gas giant	4.007	-0.72	TRUE	19.2
Neptune	Gas giant	3.883	0.67	TRUE	30.1

Describe how to create a plot of distance vs. diameter.

Plotting the Planets

```
ggplot(data = planets_df, mapping = aes(x = distance, y = diameter)) +
  geom_point()
```



Decomposing Graphics

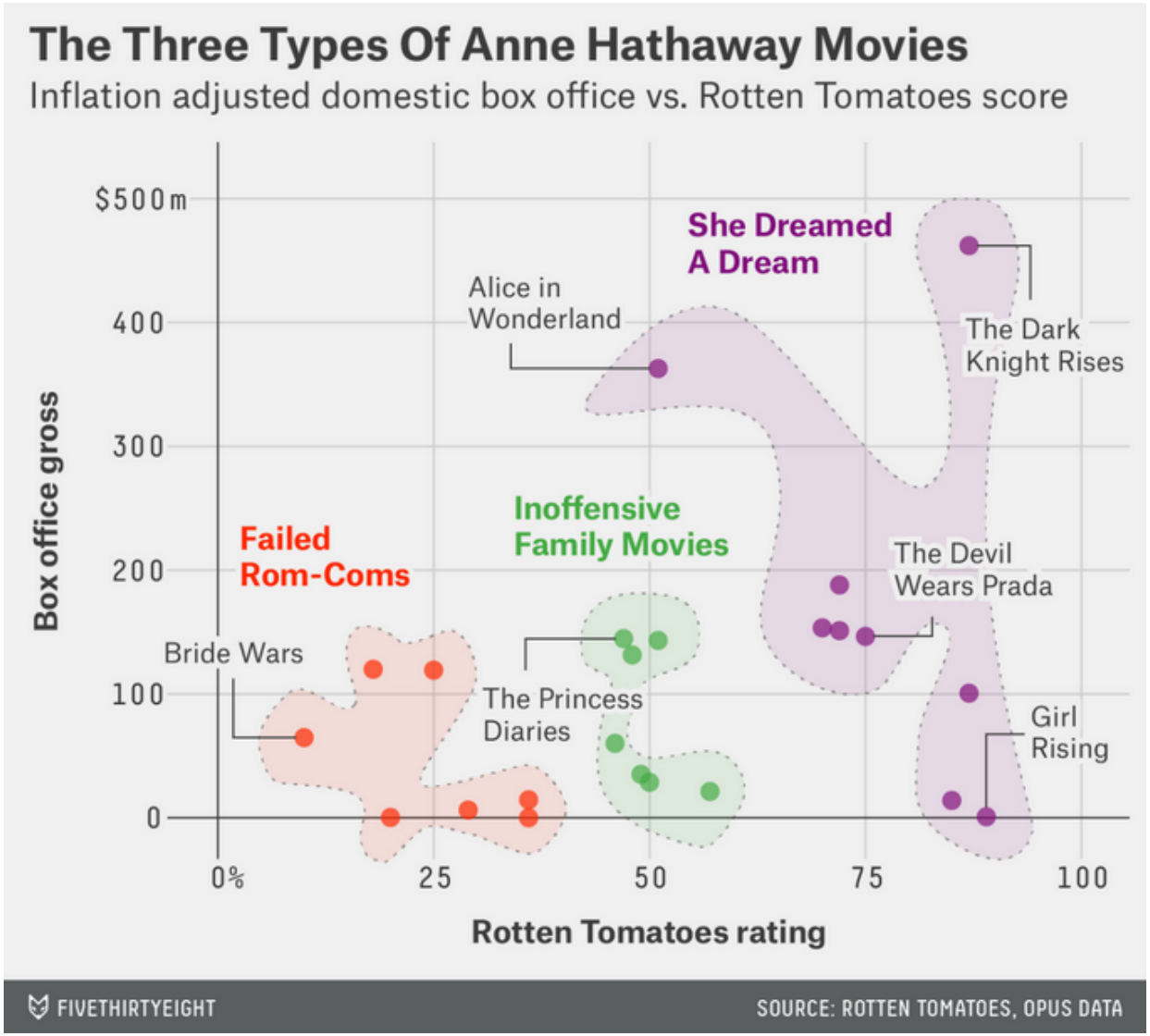
Let's get some practice decomposing visualizations using the grammar of graphics. For each of images on the next slides, we'll answer the following:

Decomposing Graphics

Let's get some practice decomposing visualizations using the grammar of graphics. For each of images on the next slides, we'll answer the following:

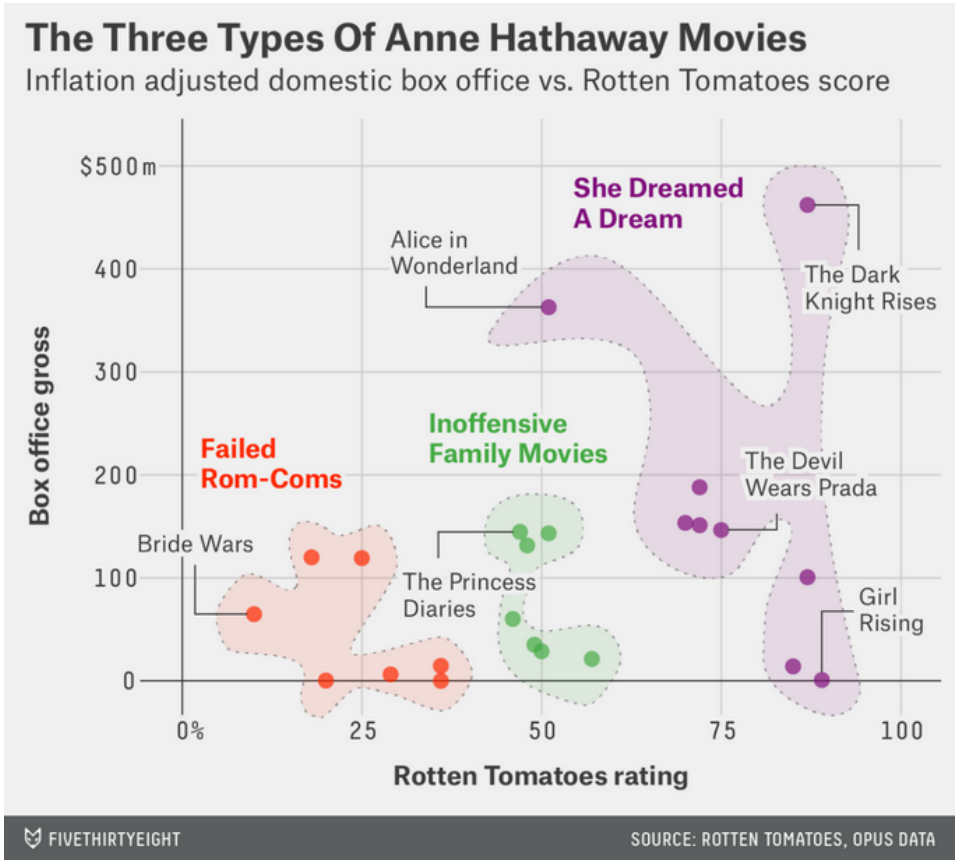
1. What is the story this graphic is telling?
2. What are the variables here?
3. What **geom** are the variables mapped to?
4. What are the **aesthetics** of the **geom**? Which variable sets the value of that **aesthetic**?
5. What additional context does this graphic provide?

Example 1 Graphic

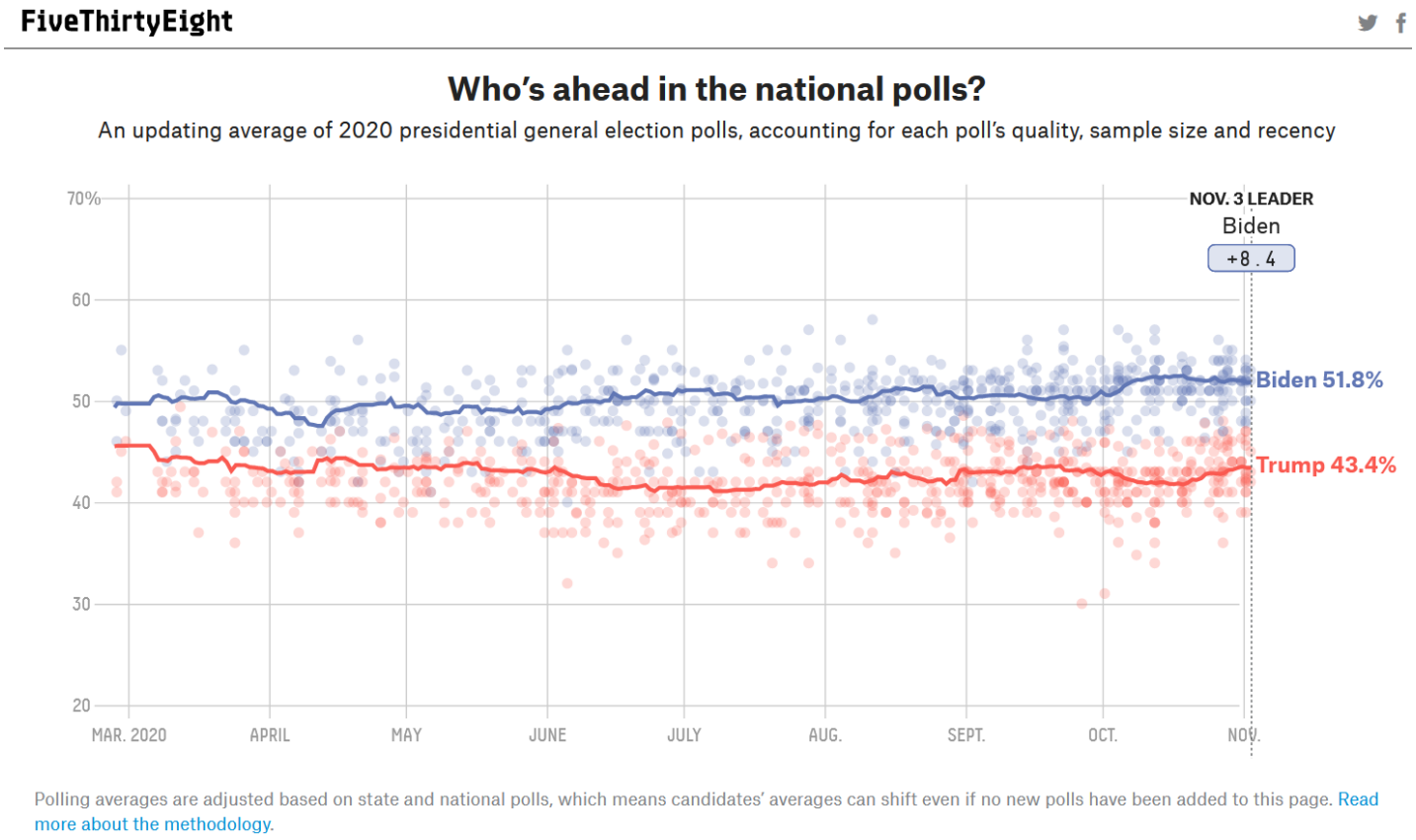


Example 1

1. What is the story this graphic is telling?
2. What are the variables here?
3. What **geom** are the variables mapped to?
4. What are the **aesthetics** of the **geom**? Which variable sets the value of that **aesthetic**?
5. What additional context does this graphic provide?

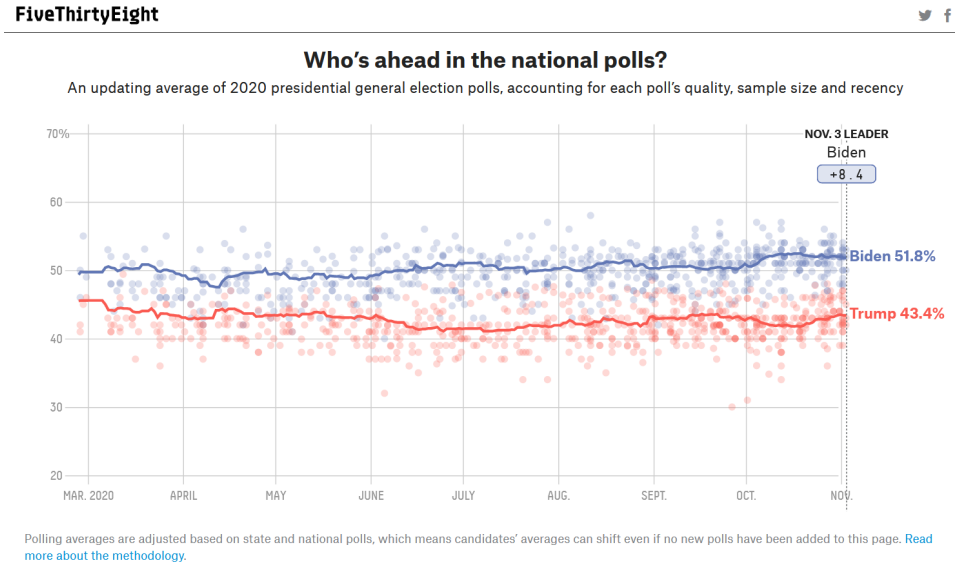


Example 2 Graphic



Example 2

1. What is the story this graphic is telling?
2. What are the variables here?
3. What **geom** are the variables mapped to?
4. What are the **aesthetics** of the **geom**? Which variable sets the value of that **aesthetic**?
5. What additional context does this graphic provide?



Best Practices for Graphics

- Determine what story your graphic should tell. And ensure that all aspects of your graphic serve this purpose.

Best Practices for Graphics

- Determine what story your graphic should tell. And ensure that all aspects of your graphic serve this purpose.
- Label axes and units for improved clarity.

Best Practices for Graphics

- Determine what story your graphic should tell. And ensure that all aspects of your graphic serve this purpose.
- Label axes and units for improved clarity.
- Include a legend to translate from aesthetics to variables.

Best Practices for Graphics

- Determine what story your graphic should tell. And ensure that all aspects of your graphic serve this purpose.
- Label axes and units for improved clarity.
- Include a legend to translate from aesthetics to variables.
- Specify your data source for reproducibility/verification.

Best Practices for Graphics

- Determine what story your graphic should tell. And ensure that all aspects of your graphic serve this purpose.
- Label axes and units for improved clarity.
- Include a legend to translate from aesthetics to variables.
- Specify your data source for reproducibility/verification.
- Minimize/eliminate extraneous elements that do not serve main purpose.

Section 2

The ggplot2 Package

The ggplot2 syntax

- We will use the `ggplot` function in the `ggplot2` package for data visualization in accordance with the grammar of graphics.

The ggplot2 syntax

- We will use the `ggplot` function in the `ggplot2` package for data visualization in accordance with the grammar of graphics.
- Recall the guiding principle:
  A statistical graphic is a mapping of data variables to aesthetic attributes of geometric objects.

The ggplot2 syntax

- We will use the `ggplot` function in the `ggplot2` package for data visualization in accordance with the grammar of graphics.
- Recall the guiding principle:
  A statistical graphic is a mapping of data variables to aesthetic attributes of geometric objects.
- The code for graphics will (almost) always take the following general form:

```
ggplot(data = ---, mapping = aes(---)) +
  geom_---(---)
```

The ggplot2 syntax

- We will use the `ggplot` function in the `ggplot2` package for data visualization in accordance with the grammar of graphics.
- Recall the guiding principle:
  A statistical graphic is a mapping of data variables to aesthetic attributes of geometric objects.
- The code for graphics will (almost) always take the following general form:

```
ggplot(data = ---, mapping = aes(---)) +
  geom_---(---)
```

- For brevity, the above code can also be written as:

```
ggplot(---, aes(---)) +
  geom_---(---)
```

- R will assume that the first argument is the data argument and the second argument is the mapping argument.

Why ggplot?

- Several other applications have capability of plotting graphics.

Why ggplot?

- Several other applications have capability of plotting graphics.
  - Excel and Google Spreadsheets each have separate buttons to produced bar plots, scatter plots, line plots, etc. from data sets.

Why ggplot?

- Several other applications have capability of plotting graphics.
  - Excel and Google Spreadsheets each have separate buttons to produced bar plots, scatter plots, line plots, etc. from data sets.
- What advantages does ggplot2 (and the Grammar of Graphics) have over these other tools?

Why ggplot?

- Several other applications have capability of plotting graphics.
  - Excel and Google Spreadsheets each have separate buttons to produced bar plots, scatter plots, line plots, etc. from data sets.
- What advantages does ggplot2 (and the Grammar of Graphics) have over these other tools?
  - Control
  - Intentionality
  - Consistency
  - Ability to create publication quality graphs with minimal tuning

The Five Named Graphs

- We focus on just 5 graphs fundamental to statistics:

The Five Named Graphs

- We focus on just 5 graphs fundamental to statistics:
  1. Scatterplots (`geom_point`)
  2. Linegraphs (`geom_line`)
  3. Histograms (`geom_histogram`)
  4. Boxplots (`geom_boxplot`)
  5. Barplots (`geom_bar`)

The Five Named Graphs

- We focus on just 5 graphs fundamental to statistics:
  1. Scatterplots (`geom_point`)
  2. Linegraphs (`geom_line`)
  3. Histograms (`geom_histogram`)
  4. Boxplots (`geom_boxplot`)
  5. Barplots (`geom_bar`)
- Other common graph types you may encounter:
  - Violin plots (`geom_violin`)
  - Interpolation (`geom_smooth`)
  - Geographic maps (`geom_map`)
  - Polygon areas (`geom_poly`)
  - Density plots (`geom_density`)

Portland Biketown

- We'll use a common data set to investigate each graph: the Portland Biketown data:

```
biketown <- read_csv("biketown.csv")
```

Portland Biketown

- We'll use a common data set to investigate each graph: the Portland Biketown data:

```
biketown <- read_csv("biketown.csv")
```

- Biketown PDX is a bike-sharing system in Portland, OR, owned by PBOT, operated by Lyft, and sponsored by Nike

Portland Biketown

- We'll use a common data set to investigate each graph: the Portland Biketown data:

```
biketown <- read_csv("biketown.csv")
```

- Biketown PDX is a bike-sharing system in Portland, OR, owned by PBOT, operated by Lyft, and sponsored by Nike
- Users can purchase a single-ride fare, a day pass, or an annual membership. They can borrow a bike from any bike station, and return the bike to any station.

Portland Biketown

- We'll use a common data set to investigate each graph: the Portland Biketown data:

```
biketown <- read_csv("biketown.csv")
```

- Biketown PDX is a bike-sharing system in Portland, OR, owned by PBOT, operated by Lyft, and sponsored by Nike
- Users can purchase a single-ride fare, a day pass, or an annual membership. They can borrow a bike from any bike station, and return the bike to any station.
- Bike stations automatically log data on each trip.

Portland Biketown

- We'll use a common data set to investigate each graph: the Portland Biketown data:

```
biketown <- read_csv("biketown.csv")
```

- Biketown PDX is a bike-sharing system in Portland, OR, owned by PBOT, operated by Lyft, and sponsored by Nike
- Users can purchase a single-ride fare, a day pass, or an annual membership. They can borrow a bike from any bike station, and return the bike to any station.
- Bike stations automatically log data on each trip.
- The `biketown` data was obtained from the `BiketownPDX` website and contains a random sample of all bike share rides between July and September, 2017.

Biketown Preview

- First, let's preview the data frame:

```
glimpse(biketown)
```

```
## Rows: 9,999
## Columns: 19
## $ RouteID          <dbl> 4074085, 3719219, 3789757, 3576798, 3459987, 3947695,~
## $ PaymentPlan      <chr> "Subscriber", "Casual", "Casual", "Subscriber", "Casu~
## $ StartHub         <chr> "SE Elliott at Division", "SW Yamhill at Director Par~
## $ StartLatitude    <dbl> 45.50513, 45.51898, 45.52990, 45.52389, 45.53028, 45.~
## $ StartLongitude   <dbl> -122.6534, -122.6813, -122.6628, -122.6722, -122.6547~
## $ StartDate        <chr> "8/17/2017", "7/22/2017", "7/27/2017", "7/12/2017", "~
## $ StartTime        <time> 10:44:00, 14:49:00, 14:13:00, 13:23:00, 19:30:00, 10~
## $ EndHub           <chr> "Blues Fest - SW Waterfront at Clay - Disabled", "SW ~
## $ EndLatitude      <dbl> 45.51287, 45.52142, 45.55902, 45.53409, 45.52990, 45.~
## $ EndLongitude     <dbl> -122.6749, -122.6726, -122.6355, -122.6949, -122.6628~
## $ EndDate          <chr> "8/17/2017", "7/22/2017", "7/27/2017", "7/12/2017", "~
## $ EndTime          <time> 10:56:00, 15:00:00, 14:42:00, 13:38:00, 20:30:00, 10~
## $ TripType         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ BikeID           <dbl> 6163, 6843, 6409, 7375, 6354, 6088, 6089, 5988, 6857,~
## $ BikeName         <chr> "0488 BIKETOWN", "0759 BIKETOWN", "0614 BIKETOWN", "0~
## $ Distance_Miles   <dbl> 1.91, 0.72, 3.42, 1.81, 4.51, 5.54, 1.59, 1.03, 0.70,~
## $ Duration         <dbl> 11.500, 11.383, 28.317, 14.917, 60.517, 53.783, 23.86~
## $ RentalAccessPath <chr> "keypad", "keypad", "keypad", "keypad", "keypad", "ke~
## $ MultipleRental   <lgl> FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE~
```

A Deeper Dive I

What do the first few entries look like?

A Deeper Dive I

What do the first few entries look like?

```
head(biketown)
```

```
## # A tibble: 6 x 19
##   RouteID Payme~1 Start~2 Start~3 Start~4 Start~5 Start~6 EndHub EndLa~7 EndLo~8
##     <dbl> <chr>   <chr>     <dbl>   <dbl> <chr>   <time>  <chr>    <dbl>   <dbl>
## 1 4074085 Subscr~ SE Ell~    45.5   -123. 8/17/2~ 10:44   Blues~    45.5   -123.
## 2 3719219 Casual  SW Yam~    45.5   -123. 7/22/2~ 14:49   SW 2n~    45.5   -123.
## 3 3789757 Casual  NE Hol~    45.5   -123. 7/27/2~ 14:13   NE Al~    45.6   -123.
## 4 3576798 Subscr~ NW Cou~    45.5   -123. 7/12/2~ 13:23   NW Ra~    45.5   -123.
## 5 3459987 Casual  NE 11t~    45.5   -123. 7/3/20~ 19:30   NE Ho~    45.5   -123.
## 6 3947695 Casual  SW Moo~    45.5   -123. 8/8/20~ 10:01   SW 3r~    45.5   -123.
## # ... with 9 more variables: EndDate <chr>, EndTime <time>, TripType <lgl>,
## #   BikeID <dbl>, BikeName <chr>, Distance_Miles <dbl>, Duration <dbl>,
## #   RentalAccessPath <chr>, MultipleRental <lgl>, and abbreviated variable
## #   names 1: PaymentPlan, 2: StartHub, 3: StartLatitude, 4: StartLongitude,
## #   5: StartDate, 6: StartTime, 7: EndLatitude, 8: EndLongitude
## # i Use `colnames()` to see all variable names
```

A Deeper Dive II

- To access 1 variable of a data set, separate the dataframe and variable name with $

A Deeper Dive II

- To access 1 variable of a data set, separate the dataframe and variable name with $

```
biketown$Distance_Miles
```

```
##    [1]  1.91  0.72  3.42  1.81  4.51  5.54  1.59  1.03  0.70  1.72  1.79  3.15
##   [13]  0.81  0.55  5.78  0.41  3.76  2.22  1.77  4.96  3.19  2.56  2.55  0.68
##   [25]  0.59  0.71  3.15  5.89  1.34  1.56  3.50  1.81  2.74  4.56  3.99  1.03
##   [37]  1.51  2.87  2.60  1.48  0.96  2.82  0.66  0.37  2.38  5.92  1.27  0.78
##   [49]  0.79  3.38  1.73  3.64  1.40  2.61  1.85  1.04  1.55  0.63  3.41  4.94
##   [61]  3.93  0.40  1.00  7.19  7.15  0.96  0.33  0.79  2.80  1.08  2.27  0.62
##   [73]  0.50  2.15  0.23  3.06  1.85  5.00  0.42  3.05  0.42  1.00  4.09  0.45
##   [85]  2.53  0.66  0.26  1.89  1.63  0.99  1.62  1.87  6.73 12.95  3.44  0.43
##   [97]  0.82  0.72  1.51  1.70  0.34  0.55  2.84  1.31  2.78  1.09  1.25  5.04
##  [109]  1.18  1.15  1.62  0.63  3.88  4.67  1.25  0.34  3.11  5.29  1.00  1.67
##  [121]  0.61  0.47  0.68  0.66  0.71  0.02  0.87  1.61  4.50  1.47  4.53  0.10
##  [133]  0.25  5.50  2.05  4.98  0.66  0.12  4.79  0.47  4.19  0.43  1.57  0.27
##  [145]  0.17  1.08  0.36  5.16  6.74  2.54  0.48  0.91  1.80  0.19  2.71  1.32
##  [157]  2.75  1.14  0.65  2.58  3.77  0.66  3.55  1.37  0.98  1.41  1.01  1.87
##  [169]  0.51  0.37  1.12  0.84  0.55  0.12  3.64  4.69  0.15  2.94  5.06  1.24
##  [181]  0.83  2.32  1.25  2.82  0.61  1.80  1.41  1.16  1.09  2.03  1.34  0.55
##  [193]  0.45  4.79  4.30  0.45  2.05  0.71  0.16  0.31  0.01  1.49  3.27  3.11
##  [205]  0.78  2.62  0.63  2.09  1.83  0.35  0.82  1.39  2.39  0.58  0.36  0.28
##  [217]  1.65  0.79  1.90  1.27  3.71  2.96  7.12  3.20  0.40  1.50  0.93  1.97
##  [229]  0.73  0.68  0.91  3.20  2.27  2.67  2.37  0.05  0.82  2.50  2.17  0.44
##  [241]  2.79  0.07  0.66  1.93  1.78  0.59  1.54  1.59  1.47  0.62  1.63  2.68
##  [253]  0.27  1.46  0.68  0.16  0.42  0.55  1.47  0.27  2.02  0.85  0.63  0.58
##  [265]  0.60  0.56  2.12  1.48  0.60  0.88  0.62  0.48  2.55  0.82  0.97  3.45
```

A Deeper Dive II

- To determine the variable type in R, use `class`

```
class(biketown$Distance_Miles)
```

```
## [1] "numeric"
```

```
class(biketown$PaymentPlan)
```

```
## [1] "character"
```

A Deeper Dive II

- To determine the variable type in R, use `class`

```
class(biketown$Distance_Miles)
```

```
## [1] "numeric"
```

```
class(biketown$PaymentPlan)
```

```
## [1] "character"
```

- To get variable names for a dataframe, use `names`

A Deeper Dive II

- To determine the variable type in R, use `class`

```
class(biketown$Distance_Miles)
```

```
## [1] "numeric"
```

```
class(biketown$PaymentPlan)
```

```
## [1] "character"
```

- To get variable names for a dataframe, use `names`

```
names(biketown)
```

```
##  [1] "RouteID"          "PaymentPlan"      "StartHub"         "StartLatitude"
##  [5] "StartLongitude"   "StartDate"        "StartTime"        "EndHub"
##  [9] "EndLatitude"      "EndLongitude"     "EndDate"          "EndTime"
## [13] "TripType"         "BikeID"           "BikeName"         "Distance_Miles"
## [17] "Duration"         "RentalAccessPath" "MultipleRental"
```

Section 3

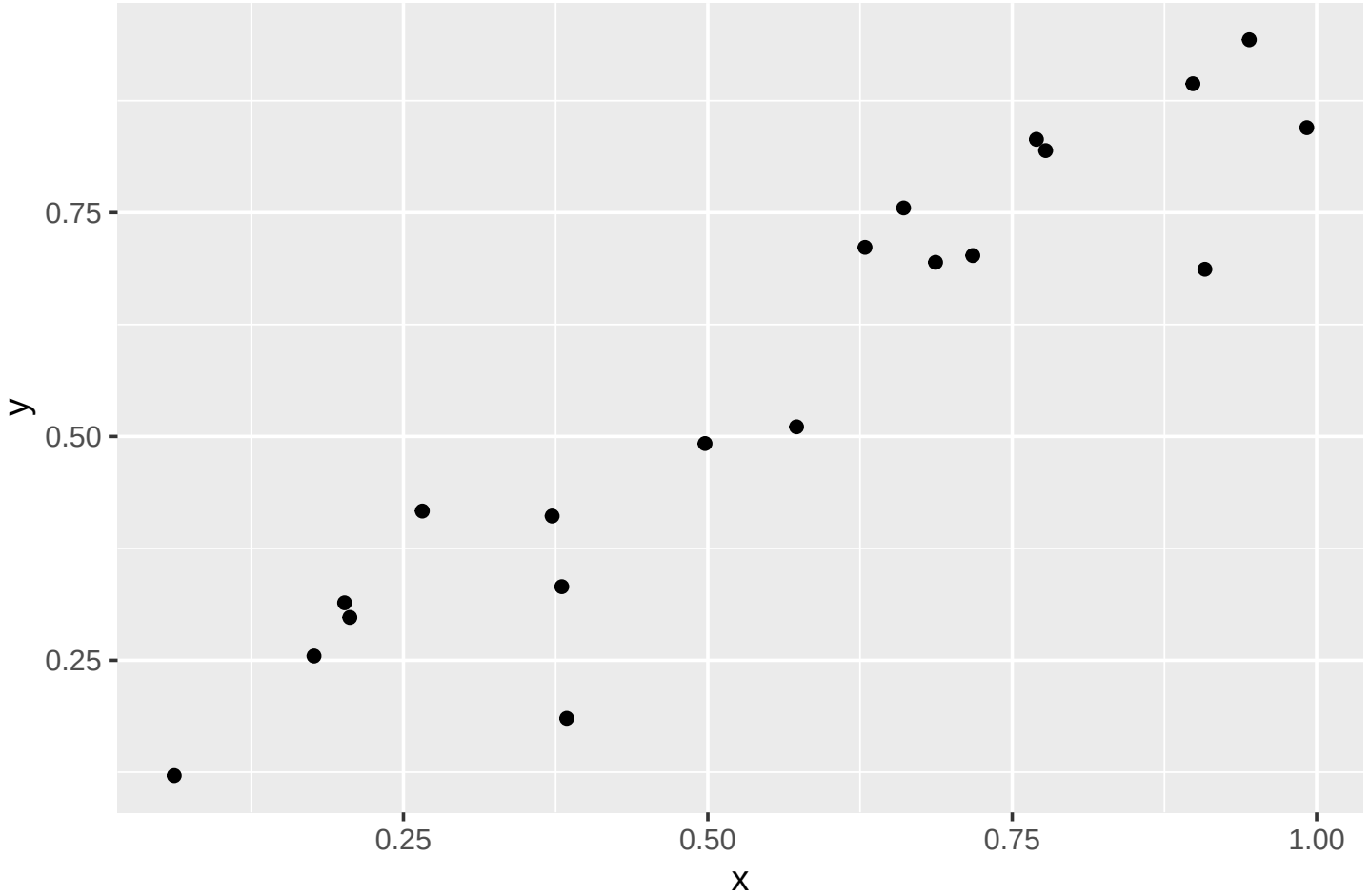
Types of Graphics

Scatterplots

- Scatterplots show relationships between a pair of **quantitative** variables.

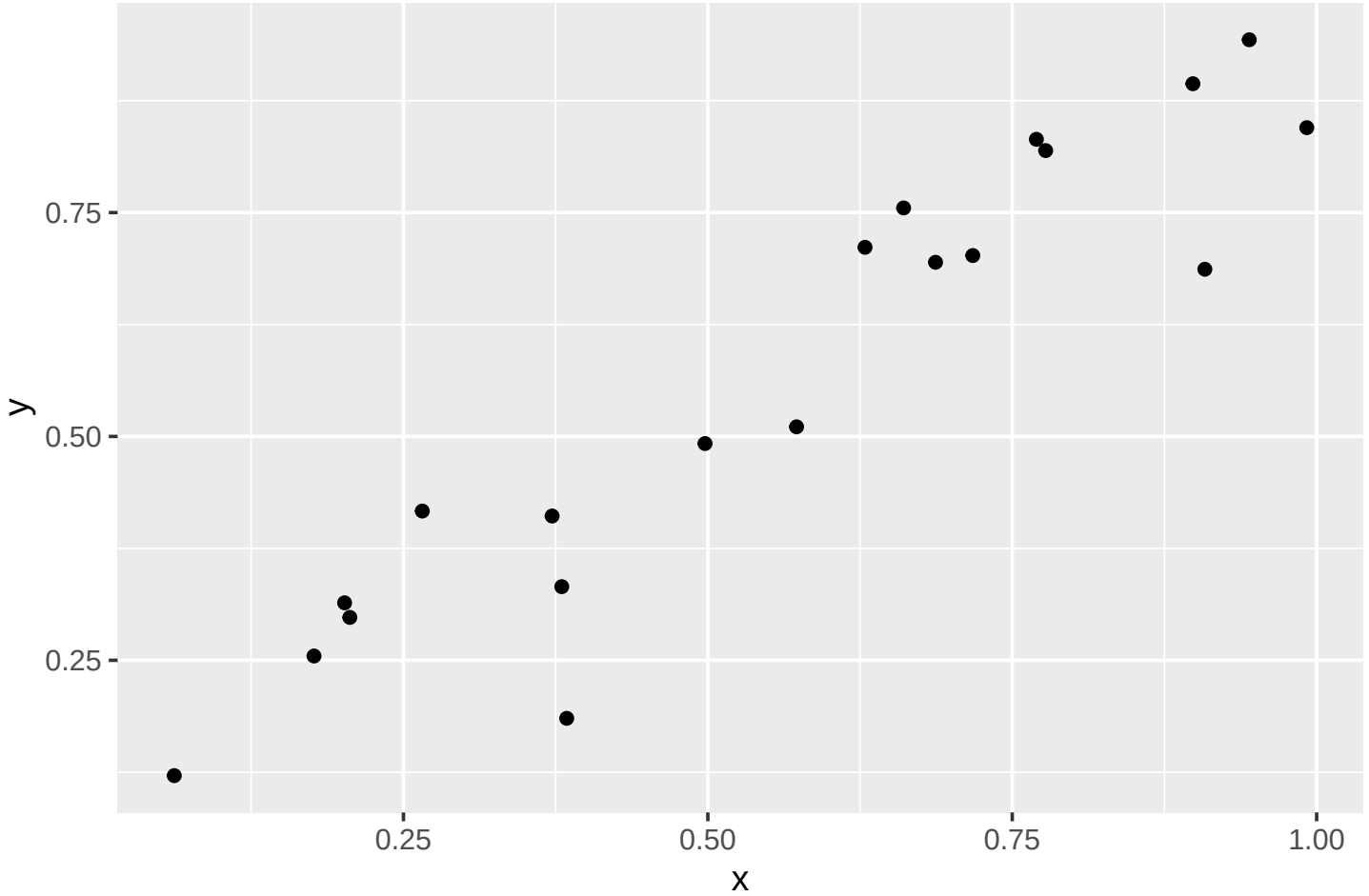
Scatterplots

- Scatterplots show relationships between a pair of **quantitative** variables.



Scatterplots

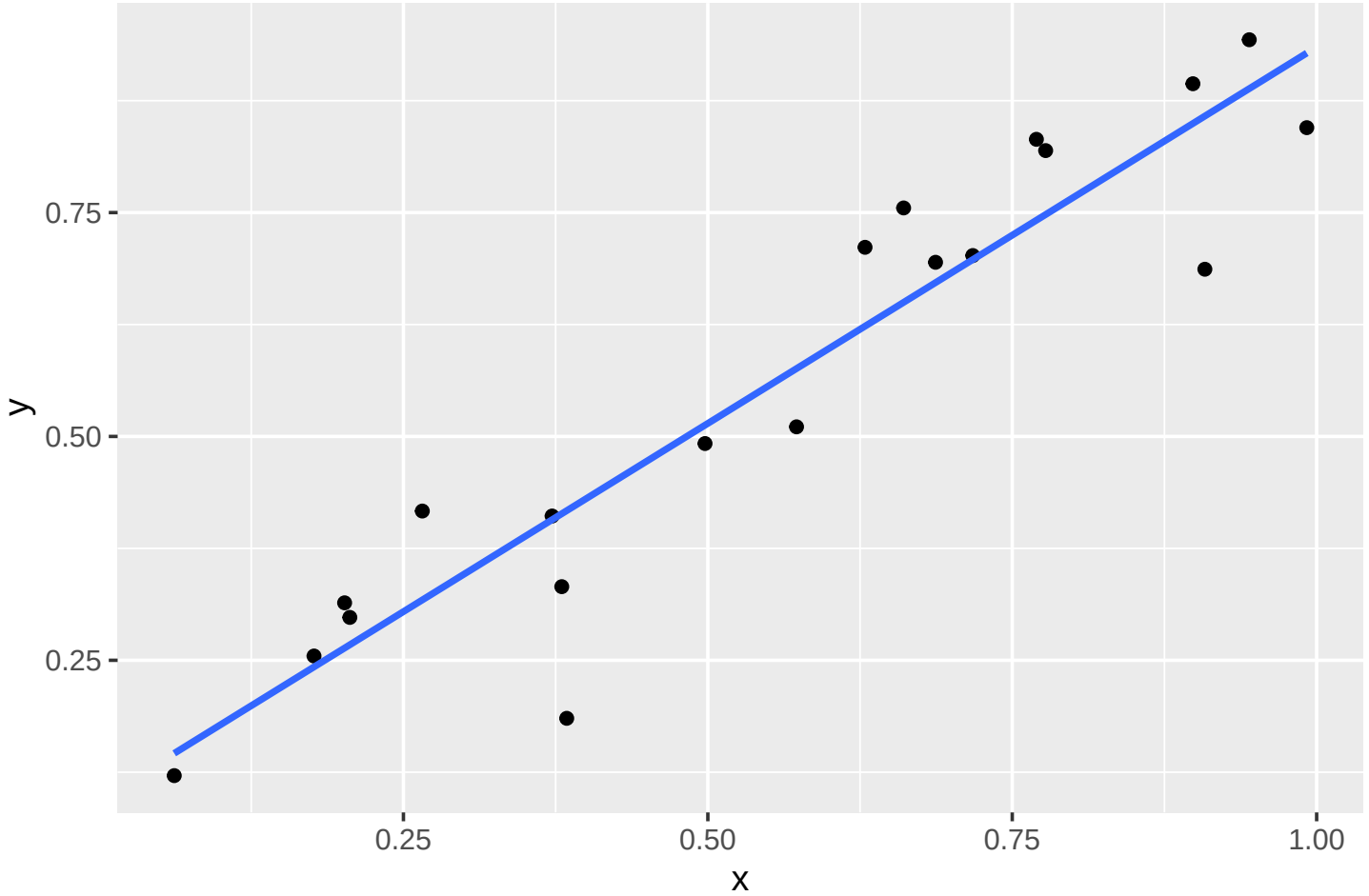
- Scatterplots show relationships between a pair of **quantitative** variables.



- In particular, we are often interested in **linear** relationships.

Scatterplots

- Scatterplots show relationships between a pair of **quantitative** variables.



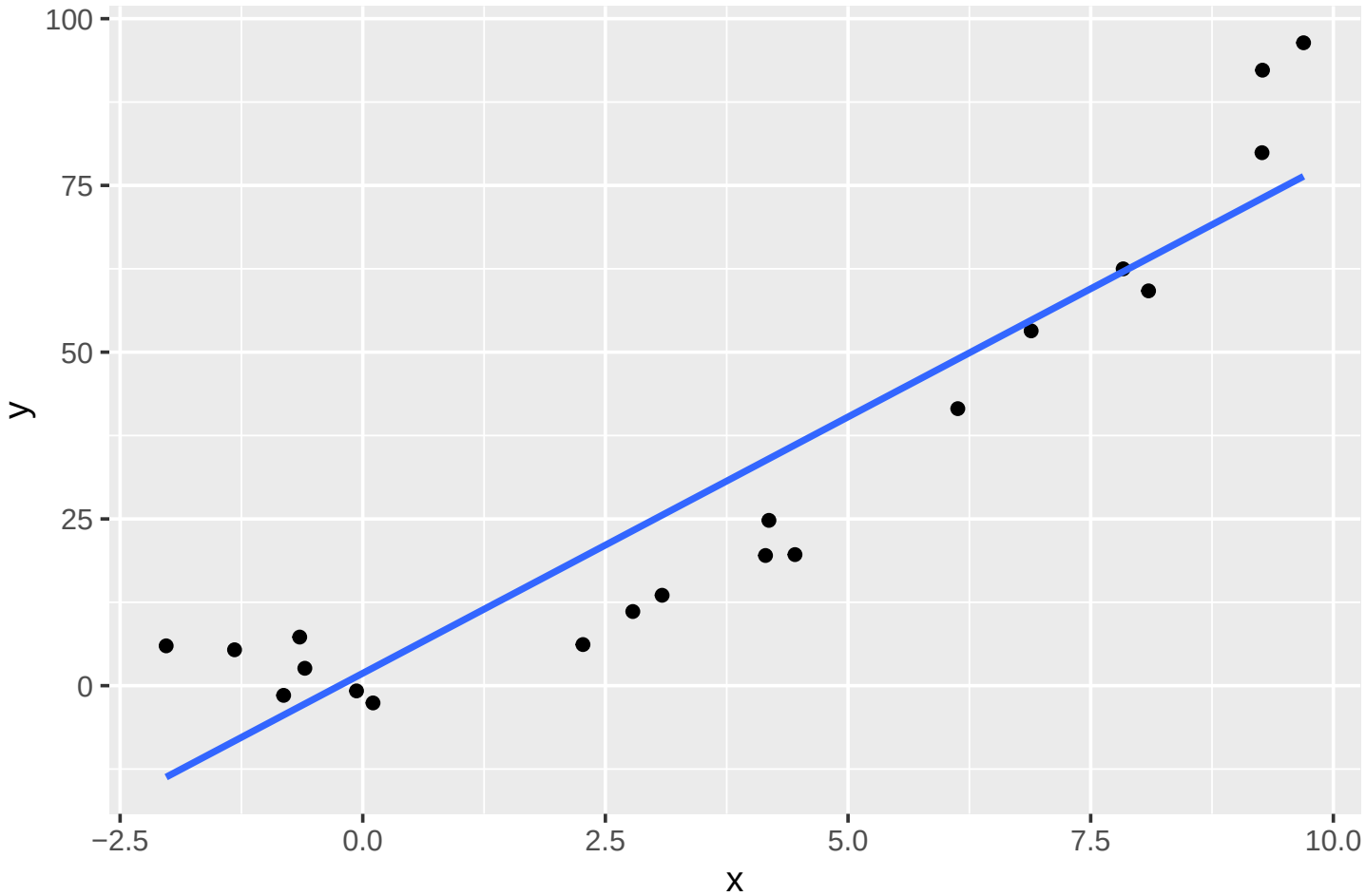
- In particular, we are often interested in **linear** relationships.

Linear Relationships

- Two variables have a **positive** relationship provided the values of one increase as the values of the other also increase.

Linear Relationships

- Two variables have a **positive** relationship provided the values of one increase as the values of the other also increase.

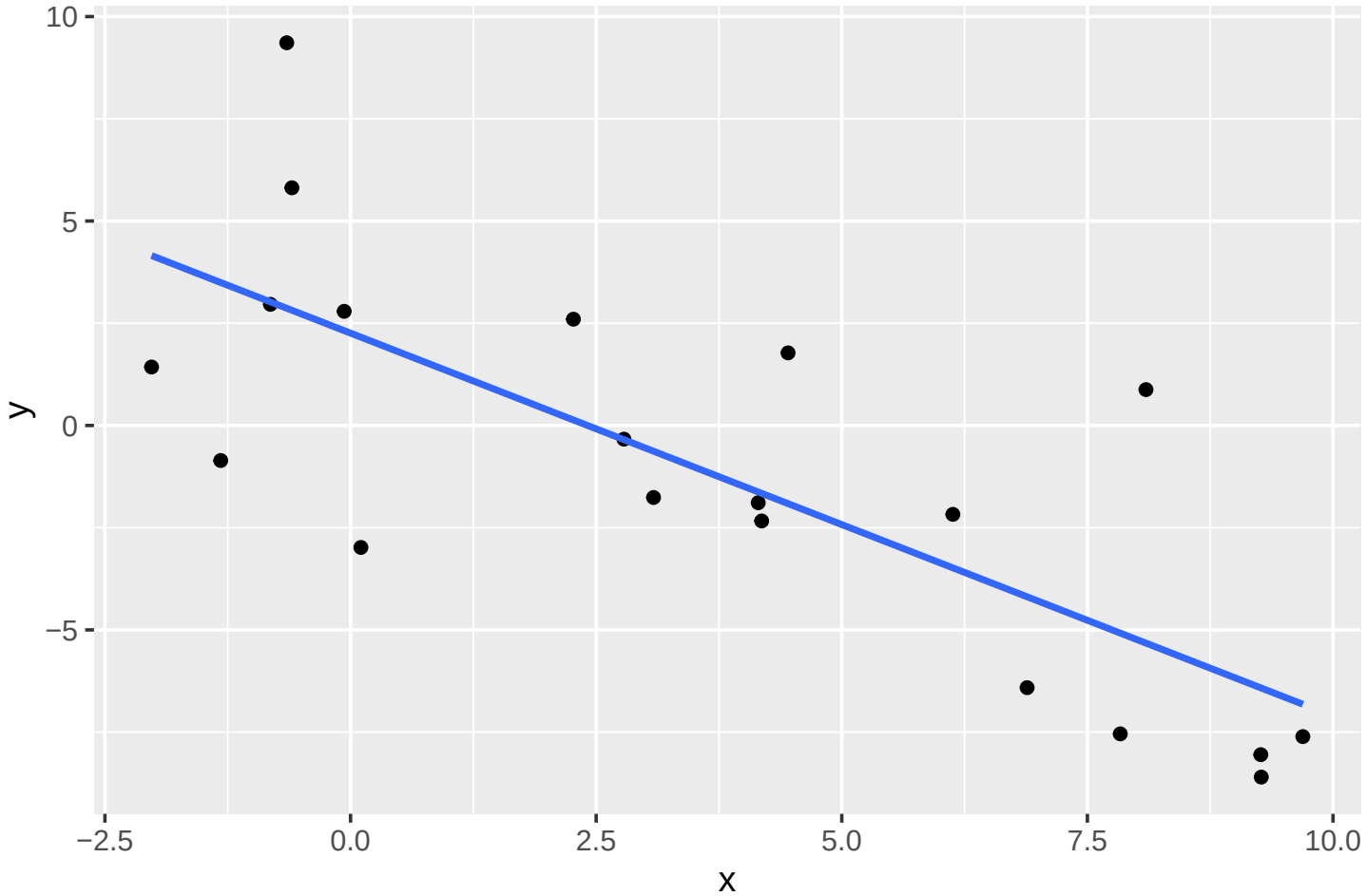


Linear Relationships

- Two variables have a **negative** relationship provided the values of one decrease as the values of the other also increase.

Linear Relationships

- Two variables have a **negative** relationship provided the values of one decrease as the values of the other also increase.

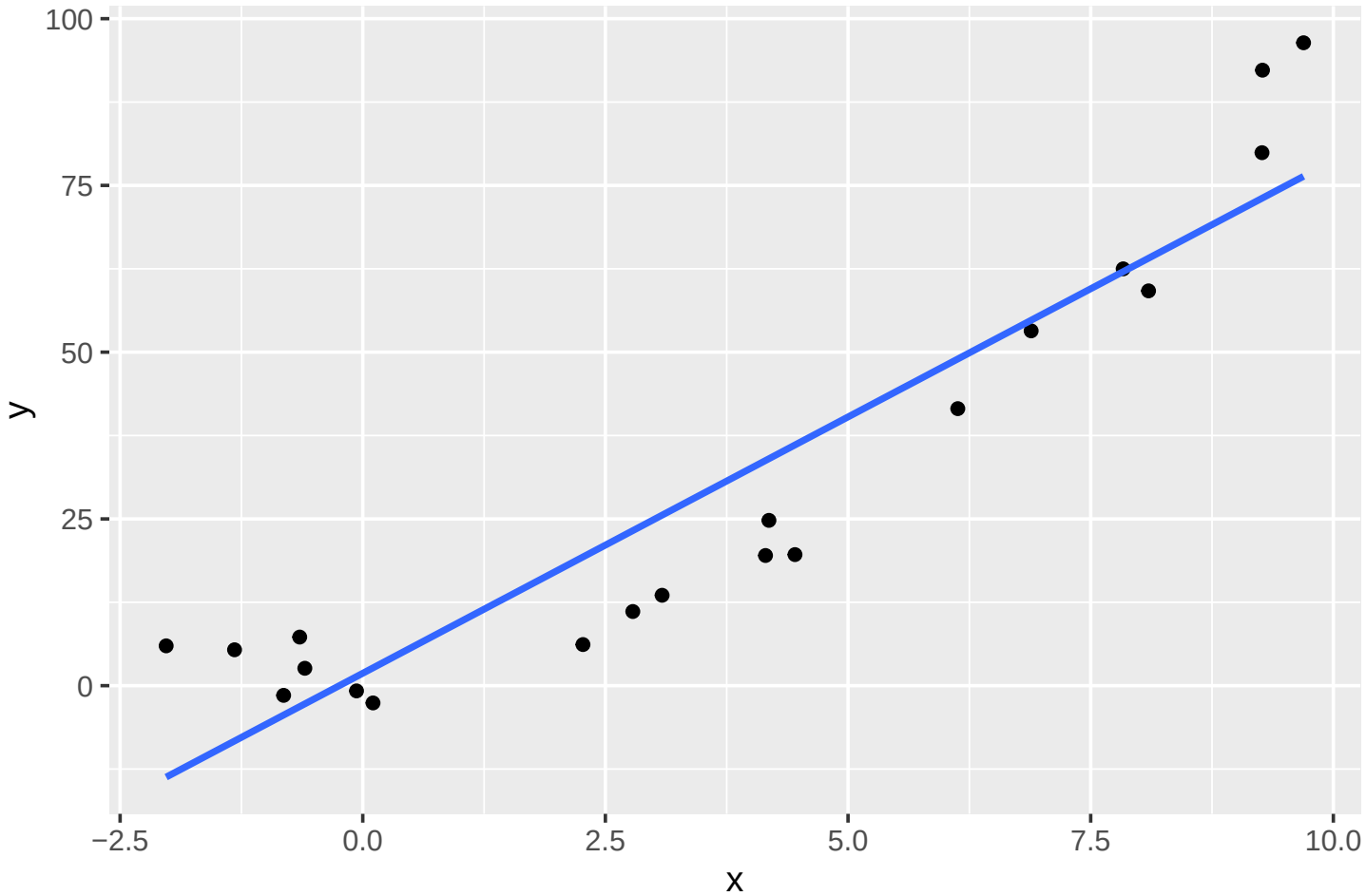


Linear Relationships

- What type of relationshop do we expect if the values of one variable **decrease** as the values of the other also **decrease**?

Linear Relationships

- What type of relationshop do we expect if the values of one variable **decrease** as the values of the other also **decrease**?

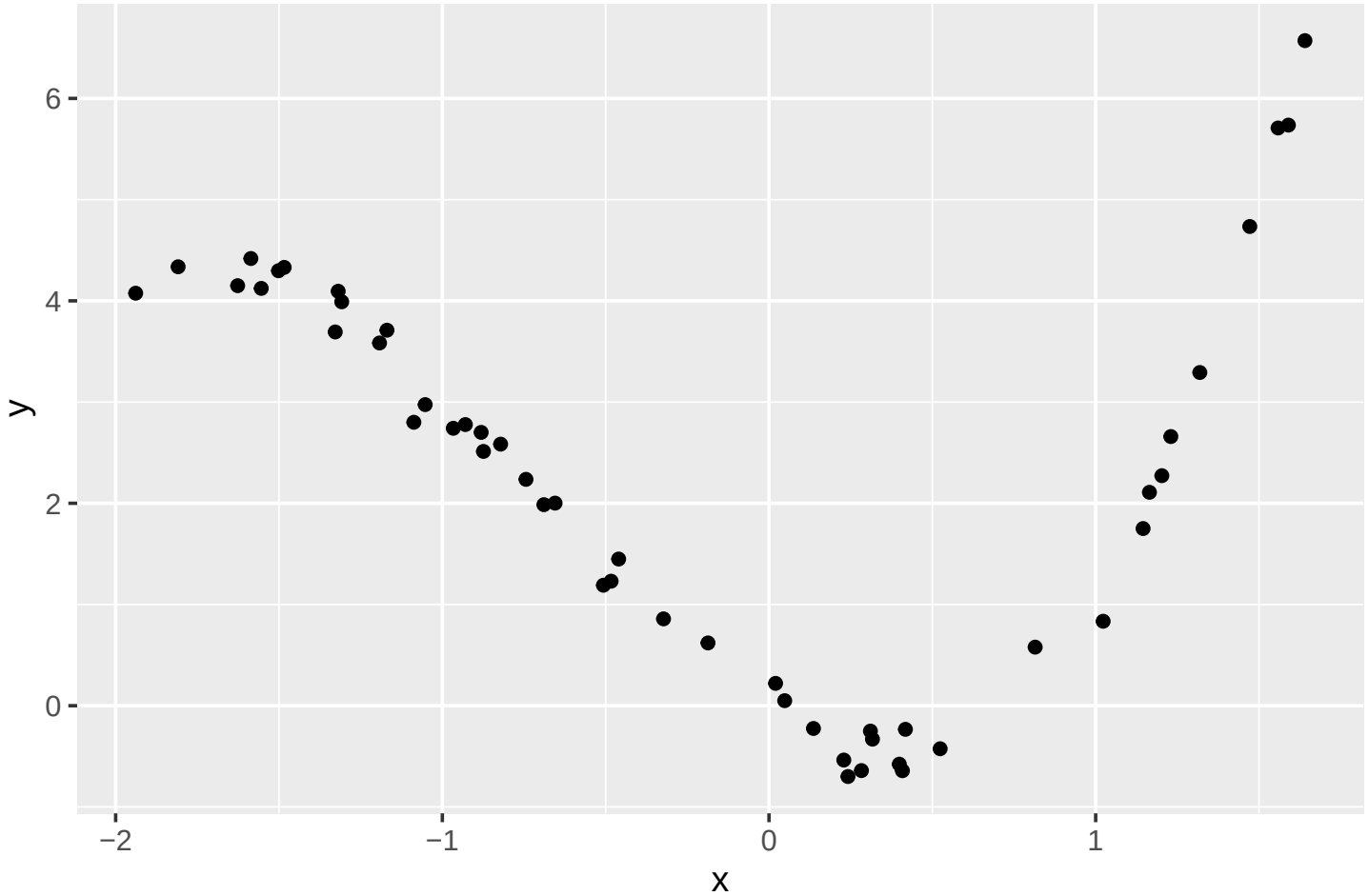


Non-Linear Relationships

- Of course, sometimes variables have strong association, but no linear relationship:

Non-Linear Relationships

- Of course, sometimes variables have strong association, but no linear relationship:



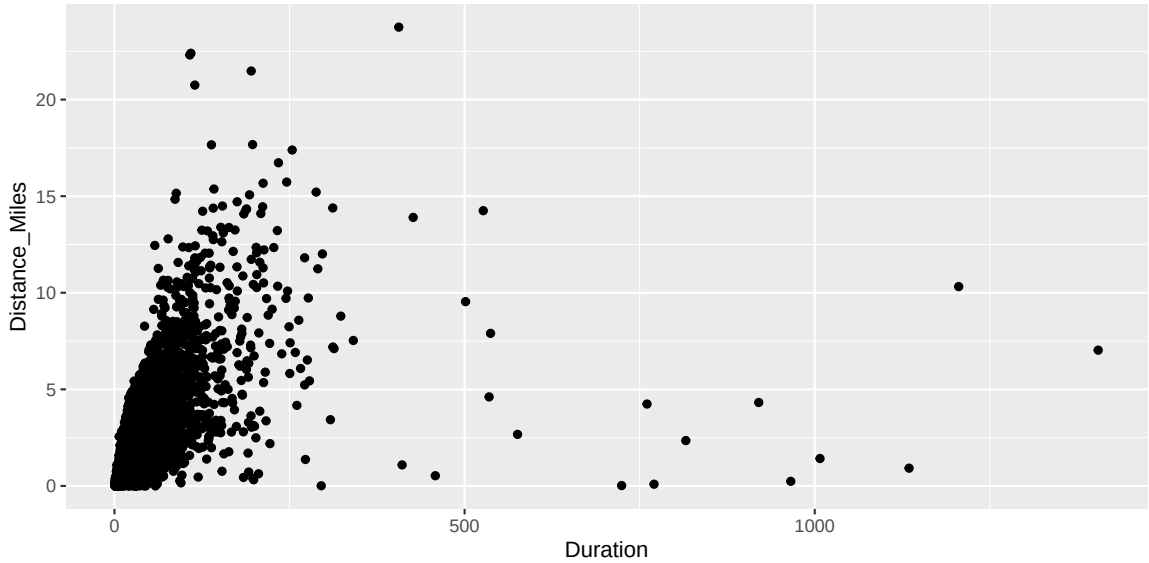
Creating Scatterplots

- In biketown data, what do you expect to be the relationship between `Duration` and `Distance_Miles`?

Creating Scatterplots

- In biketown data, what do you expect to be the relationship between `Duration` and `Distance_Miles`?

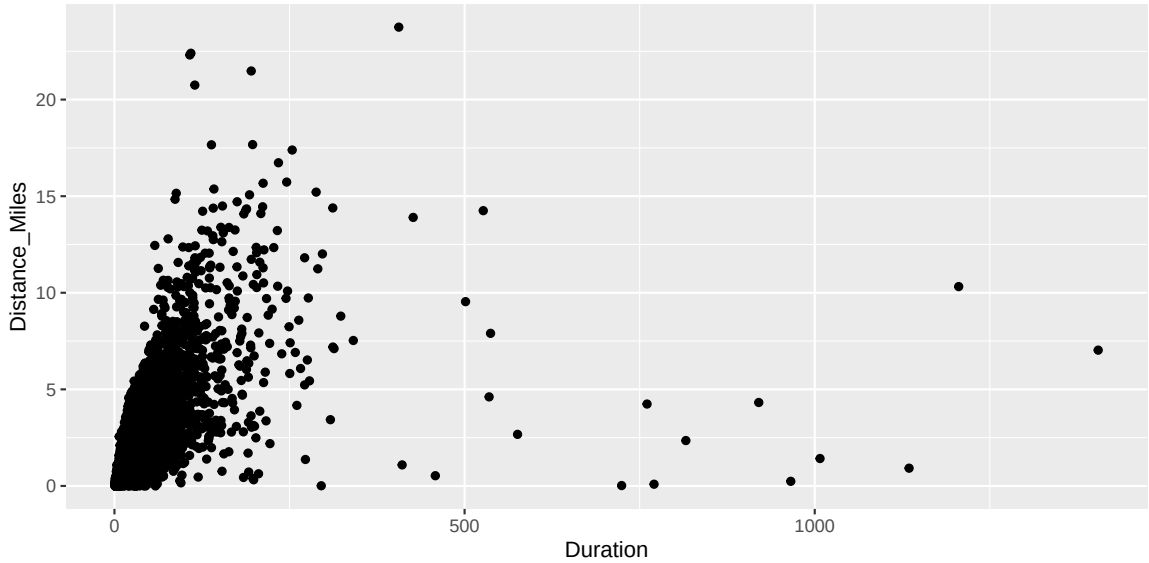
```
ggplot(data = biketown, mapping = aes(x = Duration, y = Distance_Miles)) +
  geom_point()
```



Creating Scatterplots

- In biketown data, what do you expect to be the relationship between `Duration` and `Distance_Miles`?

```
ggplot(data = biketown, mapping = aes(x = Duration, y = Distance_Miles)) +
  geom_point()
```



- Problems with the graphic?

Overplotting

- Overplotting occurs when a large number of points are plotted in close proximity, making it difficult to accurately distinguish true number of points in a region.

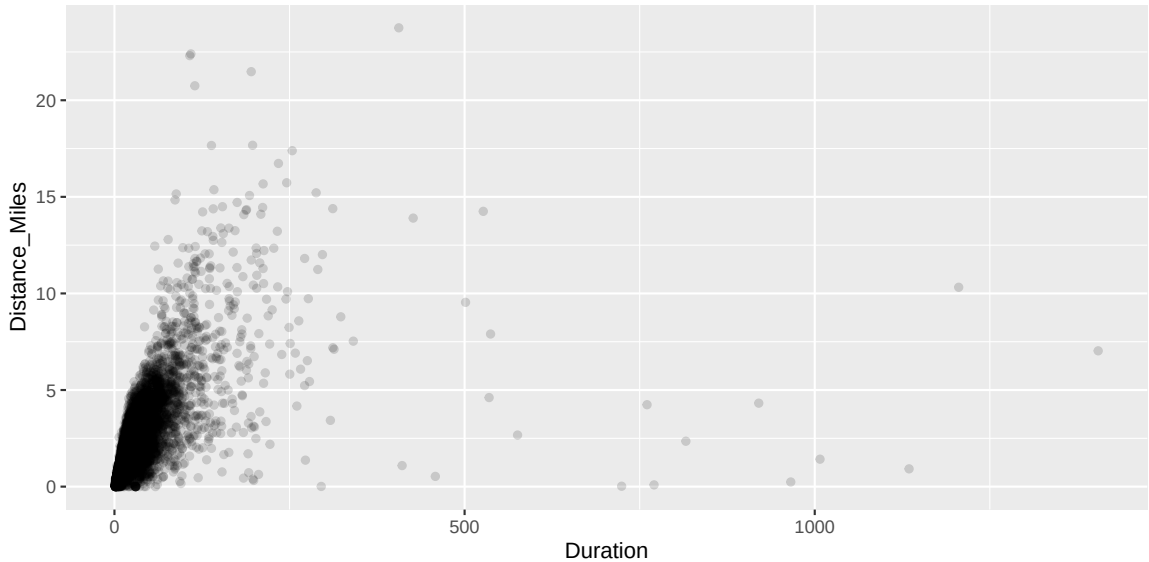
Overplotting

- Overplotting occurs when a large number of points are plotted in close proximity, making it difficult to accurately distinguish true number of points in a region.
  - Can be corrected by making points more transparent via the `alpha` aesthetic:

Overplotting

- Overplotting occurs when a large number of points are plotted in close proximity, making it difficult to accurately distinguish true number of points in a region.
  - Can be corrected by making points more transparent via the `alpha` aesthetic:

```
ggplot(data = biketown, mapping = aes(x = Duration, y = Distance_Miles)) +
  geom_point(alpha = 0.15)
```



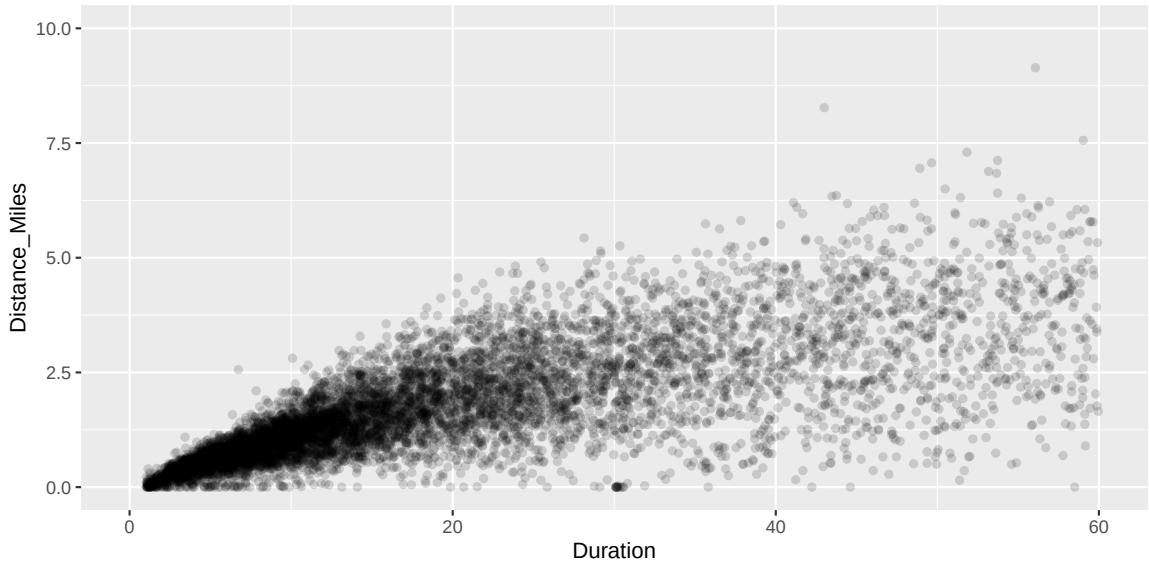
Overplotting II

- We can also focus on just part of the graph by controlling the limits of the axes:

Overplotting II

- We can also focus on just part of the graph by controlling the limits of the axes:

```
ggplot(data = biketown, mapping = aes(x = Duration, y = Distance_Miles)) +
  geom_point(alpha = .15)+
  scale_x_continuous(limits = c(0, 60))+
  scale_y_continuous(limits = c(0, 10))
```



Overplotting III

- Alternatively, can manipulate data set by jittering points a small random amount so that they no longer lie on top of each other.

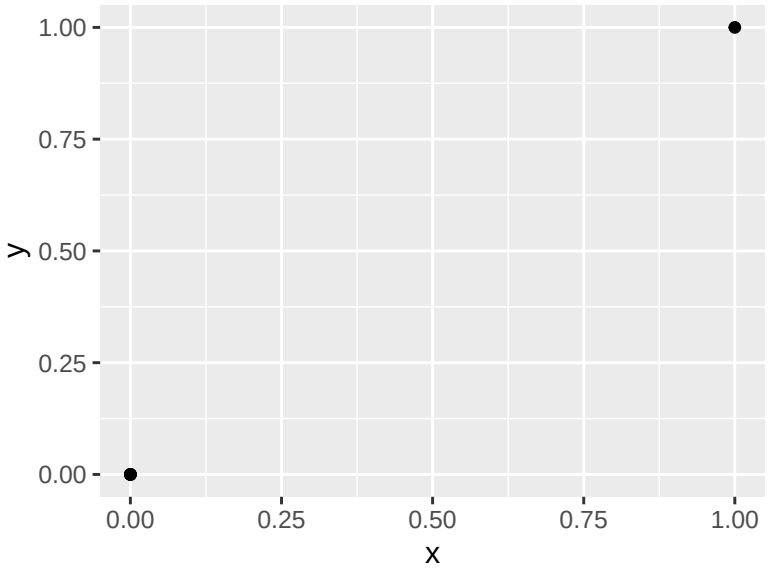
Overplotting III

- Alternatively, can manipulate data set by jittering points a small random amount so that they no longer lie on top of each other.
- Consider the data set consisting of $(0, 0), (0, 0), (0, 0), (0, 0)$ and $(1, 1)$:

Overplotting III

- Alternatively, can manipulate data set by jittering points a small random amount so that they no longer lie on top of each other.
- Consider the data set consisting of $(0, 0), (0, 0), (0, 0), (0, 0)$ and $(1, 1)$:

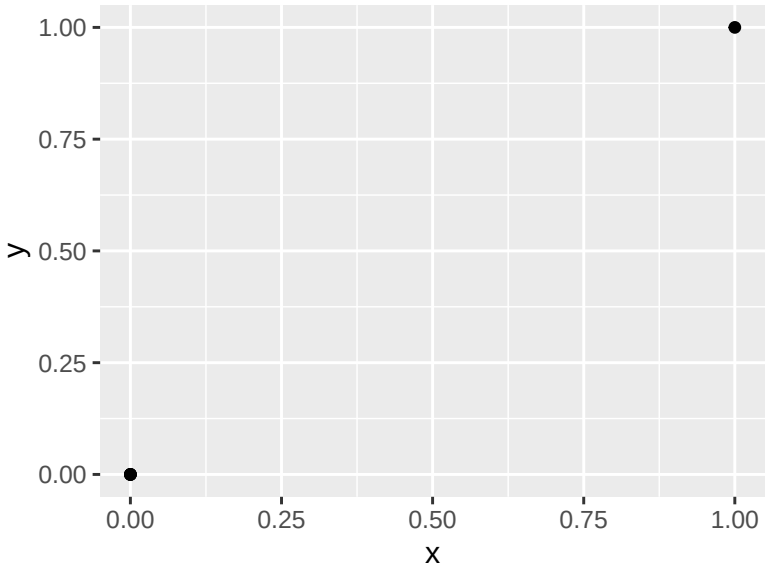
```
ggplot(data = jiggle_df, mapping = aes(x = x, y = y)) +
  geom_point()
```



Overplotting III

- Alternatively, can manipulate data set by jittering points a small random amount so that they no longer lie on top of each other.
- Consider the data set consisting of $(0, 0), (0, 0), (0, 0), (0, 0)$ and $(1, 1)$:

```
ggplot(data = jiggle_df, mapping = aes(x = x, y = y)) +
  geom_point()
```

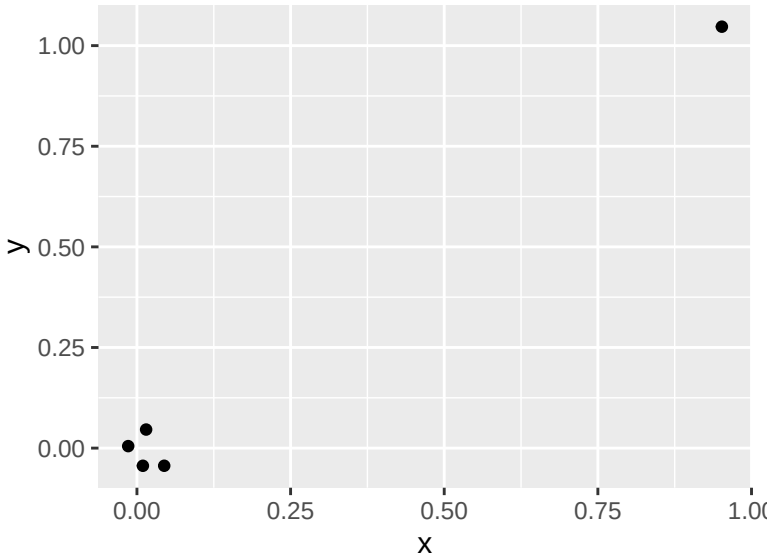


- It looks like there are just 2 observations!

Overplotting III

- Alternatively, can manipulate data set by jittering points a small random amount so that they no longer lie on top of each other.
- Consider the data set consisting of $(0, 0), (0, 0), (0, 0), (0, 0)$ and $(1, 1)$:

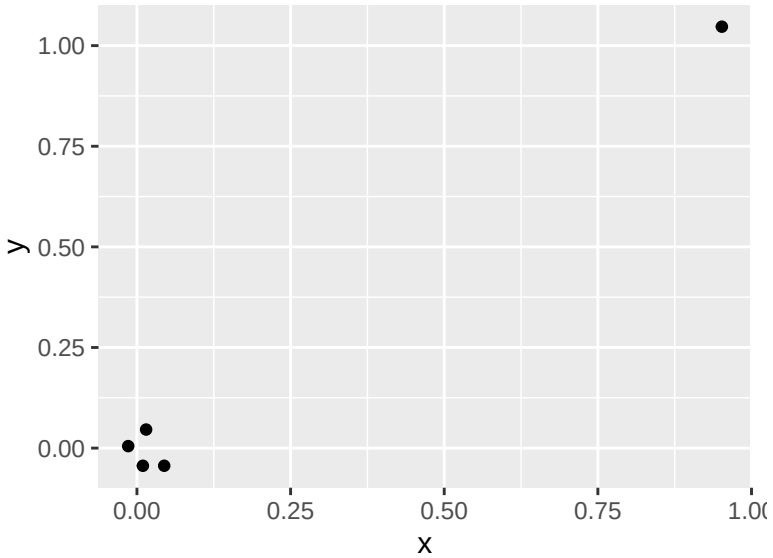
```
ggplot(data = jiggle_df, mapping = aes(x = x, y = y)) +
  geom_jitter(width = .05, height = .05)
```



Overplotting III

- Alternatively, can manipulate data set by jittering points a small random amount so that they no longer lie on top of each other.
- Consider the data set consisting of $(0, 0), (0, 0), (0, 0), (0, 0)$ and $(1, 1)$:

```
ggplot(data = jiggle_df, mapping = aes(x = x, y = y)) +
  geom_jitter(width = .05, height = .05)
```



- To jitter points, use the layer `geom_jitter(width = ..., height = ...)` instead of `geom_points()`