Introduction to the Grammar of Graphics II

Nate Wells

Math 141, 2/1/21

Outline

In this lecture, we will. . .

Outline

In this lecture, we will...

- Introduce the ggplot2 package for R graphics
- Create scatterplots and linegraphs

Section 1

The ggplot2 Package

The ggplot2 syntax

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

The ggplot2 syntax

- We will use the `ggplot` function in the `ggplot2` package for data vizualization 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 vizualization 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 Planets

Let's take a look at the planets data frame `planets_df` using the `glimpse` function:

```
glimpse(planets_df)
```

```
## Rows: 8
## Columns: 6
## $ name     <fct> Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune
## $ type     <fct> Terrestrial planet, Terrestrial planet, Terrestrial planet...
## $ diameter <dbl> 0.382, 0.949, 1.000, 0.532, 11.209, 9.449, 4.007, 3.883
## $ rotation <dbl> 58.64, -243.02, 1.00, 1.03, 0.41, 0.43, -0.72, 0.67
## $ rings    <lgl> FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE, TRUE
## $ distance <dbl> 0.4, 0.7, 1.0, 1.5, 5.2, 9.5, 19.2, 30.1
```

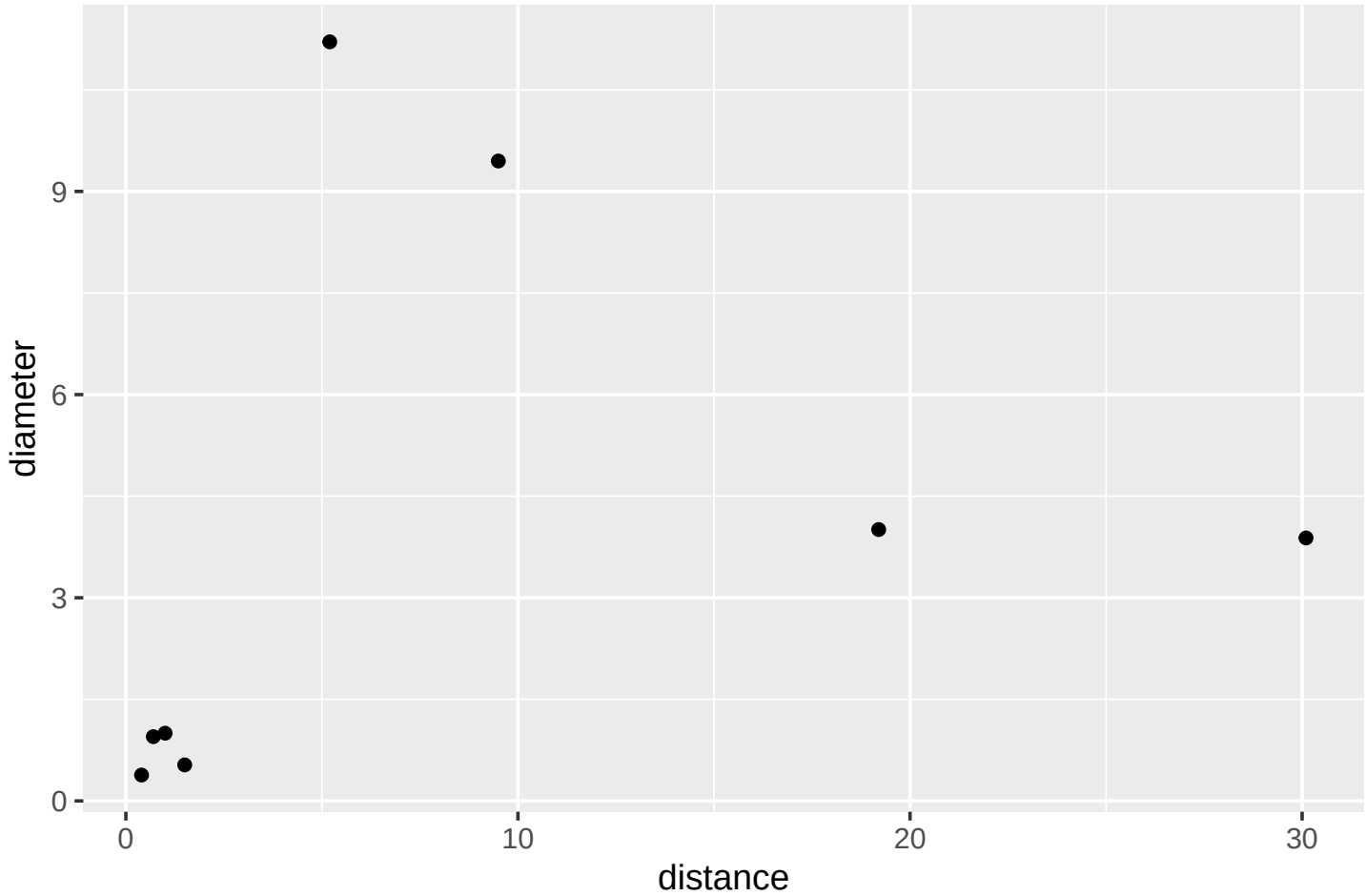
Plotting the Planets

- Create a plot of distance vs. diameter based on the `planets_df` data frame.

Plotting the Planets

- Create a plot of distance vs. diameter based on the `planets_df` data frame.

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



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
  - Intentionaility
  - Ability to create publication quality graphs with minimal tuning

The Five Named Graphs

- We focus on just 5 graphs fundamental to statistics (although other types exist)

The Five Named Graphs

- We focus on just 5 graphs fundamental to statistics (although other types exist)
  1. Scatterplots
  2. Linegraphs
  3. Histograms
  4. Boxplots
  5. Barplots

The Five Named Graphs

- We focus on just 5 graphs fundamental to statistics (although other types exist)
  1. Scatterplots
  2. Linegraphs
  3. Histograms
  4. Boxplots
  5. Barplots
- We'll use a common data set to investigate each graph: the Portland Biketown data

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

Biketown Preview

- First, let's preview the data frame:

```
glimpse(biketown)
```

```
## Rows: 9,999
## Columns: 19
## $ RouteID           <dbl> 4074085, 3719219, 3789757, 3576798, 3459987, 39476...
## $ PaymentPlan       <chr> "Subscriber", "Casual", "Casual", "Subscriber", "C...
## $ StartHub          <chr> "SE Elliott at Division", "SW Yamhill at Director ...
## $ StartLatitude     <dbl> 45.50513, 45.51898, 45.52990, 45.52389, 45.53028, ...
## $ StartLongitude    <dbl> -122.6534, -122.6813, -122.6628, -122.6722, -122.6...
## $ 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,...
## $ EndHub            <chr> "Blues Fest - SW Waterfront at Clay - Disabled", "...
## $ EndLatitude       <dbl> 45.51287, 45.52142, 45.55902, 45.53409, 45.52990, ...
## $ EndLongitude      <dbl> -122.6749, -122.6726, -122.6355, -122.6949, -122.6...
## $ 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,...
## $ TripType          <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ BikeID            <dbl> 6163, 6843, 6409, 7375, 6354, 6088, 6089, 5988, 68...
## $ BikeName          <chr> "0488 BIKETOWN", "0759 BIKETOWN", "0614 BIKETOWN",...
## $ Distance_Miles    <dbl> 1.91, 0.72, 3.42, 1.81, 4.51, 5.54, 1.59, 1.03, 0....
## $ Duration          <dbl> 11.500, 11.383, 28.317, 14.917, 60.517, 53.783, 23...
## $ RentalAccessPath <chr> "keypad", "keypad", "keypad", "keypad", "keypad", ...
```

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 PaymentPlan StartHub StartLatitude StartLongitude StartDate StartTime
##     <dbl> <chr>       <chr>            <dbl>          <dbl> <chr>     <time>
## 1 4074085 Subscriber  SE Elli~          45.5           -123. 8/17/2017 10:44
## 2 3719219 Casual      SW Yamh~          45.5           -123. 7/22/2017 14:49
## 3 3789757 Casual      NE Holl~          45.5           -123. 7/27/2017 14:13
## 4 3576798 Subscriber  NW Couc~          45.5           -123. 7/12/2017 13:23
## 5 3459987 Casual      NE 11th~          45.5           -123. 7/3/2017  19:30
## 6 3947695 Casual      SW Mood~          45.5           -123. 8/8/2017  10:01
## # ... with 12 more variables: EndHub <chr>, EndLatitude <dbl>,
## #   EndLongitude <dbl>, EndDate <chr>, EndTime <time>, TripType <lgl>,
## #   BikeID <dbl>, BikeName <chr>, Distance_Miles <dbl>, Duration <dbl>,
## #   RentalAccessPath <chr>, MultipleRental <lgl>
```

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 $

```
planets_df$diameter
```

```
## [1]  0.382  0.949  1.000  0.532 11.209  9.449  4.007  3.883
```

A Deeper Dive II

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

```
planets_df$diameter
```

```
## [1]  0.382  0.949  1.000  0.532 11.209  9.449  4.007  3.883
```

What do you think `head(biketown$Distance_Miles)` will do?

A Deeper Dive II

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

```
planets_df$diameter
```

```
## [1]  0.382  0.949  1.000  0.532 11.209  9.449  4.007  3.883
```

What do you think `head(biketown$Distance_Miles)` will do?

```
head(biketown$Distance_Miles)
```

```
## [1] 1.91 0.72 3.42 1.81 4.51 5.54
```

To determine the variable type, use `class`

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

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

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

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

A Deeper Dive II

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

```
planets_df$diameter
```

```
## [1]  0.382  0.949  1.000  0.532 11.209  9.449  4.007  3.883
```

What do you think `head(biketown$Distance_Miles)` will do?

```
head(biketown$Distance_Miles)
```

```
## [1] 1.91 0.72 3.42 1.81 4.51 5.54
```

To determine the variable type, use `class`

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

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

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

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

What happens if we apply `class` to `biketown`?

A Deeper Dive II

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

```
planets_df$diameter
```

```
## [1]  0.382  0.949  1.000  0.532 11.209  9.449  4.007  3.883
```

What do you think `head(biketown$Distance_Miles)` will do?

```
head(biketown$Distance_Miles)
```

```
## [1] 1.91 0.72 3.42 1.81 4.51 5.54
```

To determine the variable type, use `class`

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

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

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

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

What happens if we apply `class` to `biketown`?

```
class(biketown)
```

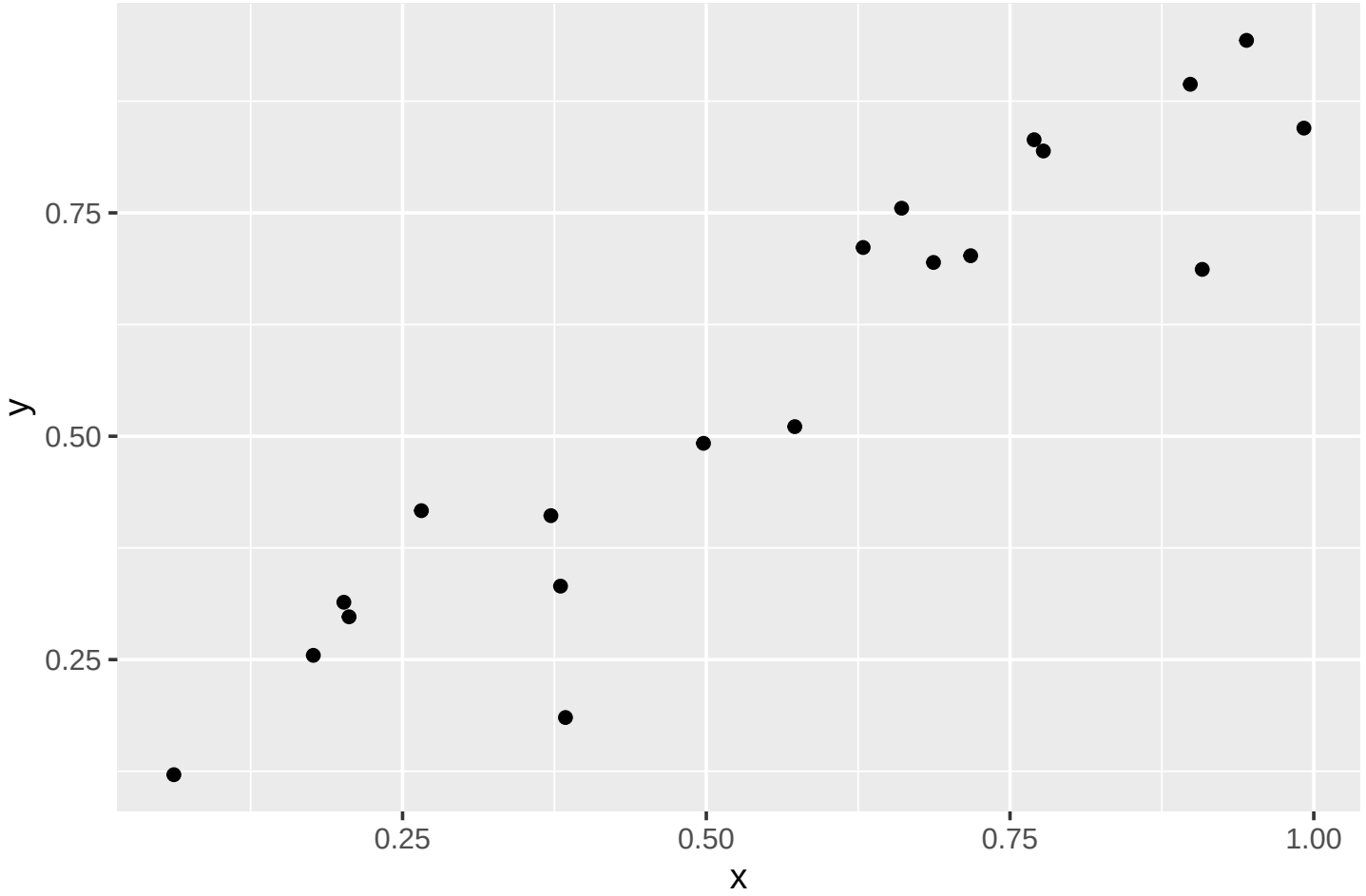
```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

Section 2

Types of Graphics

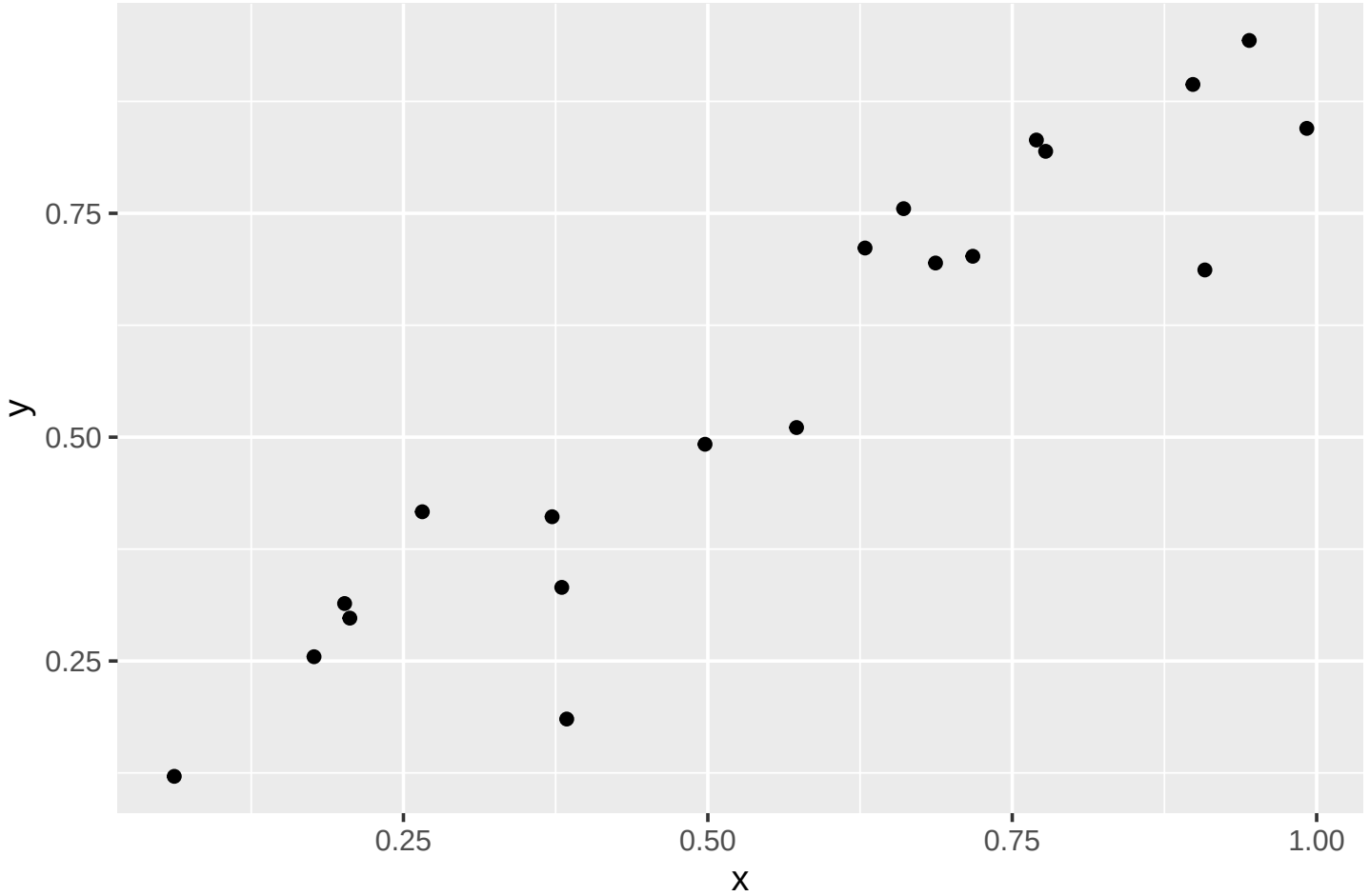
Scatterplots

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



Scatterplots

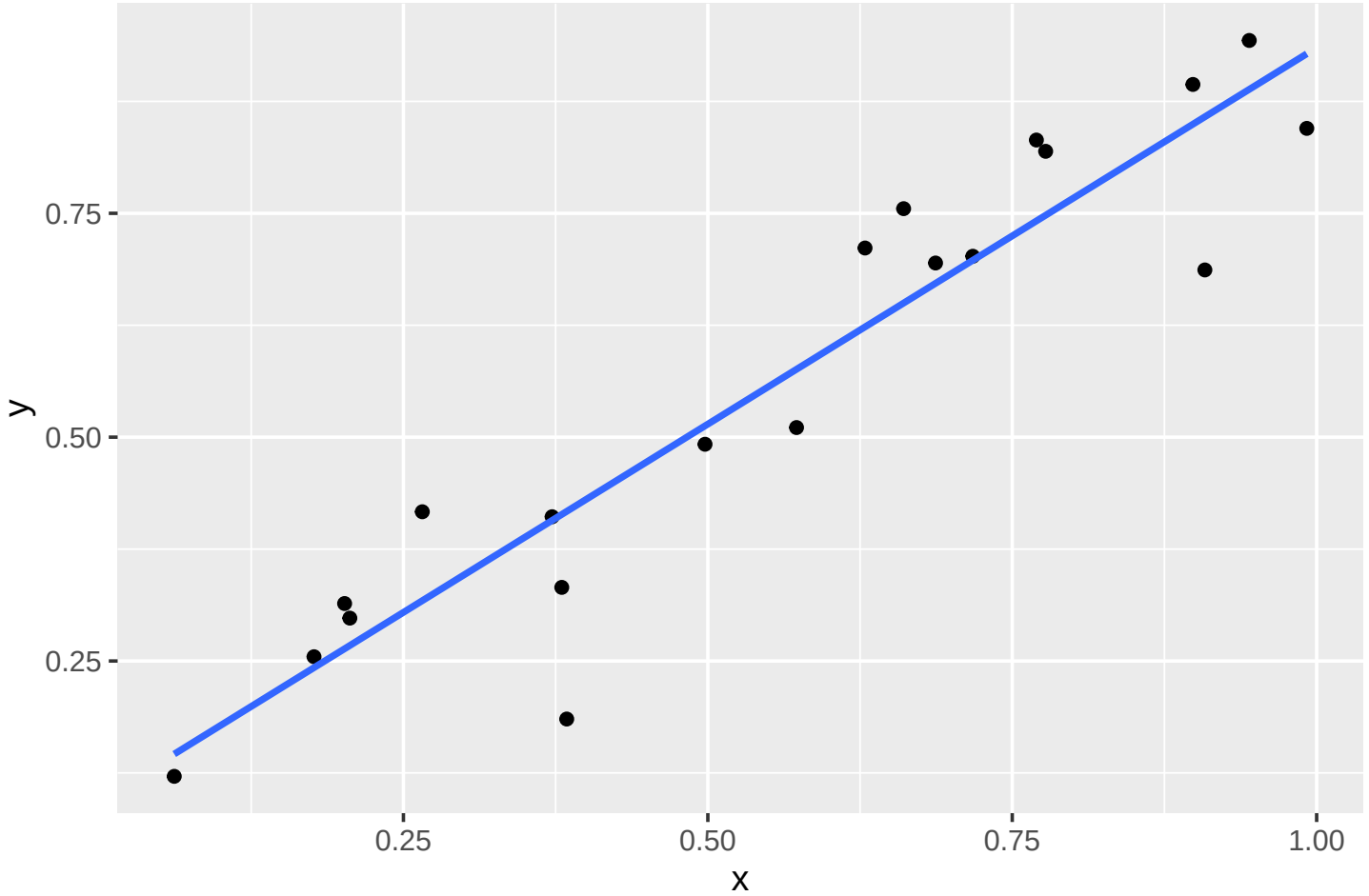
- Scatterplots show relationships between a pair of **quantitative** variables.



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

Scatterplots

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



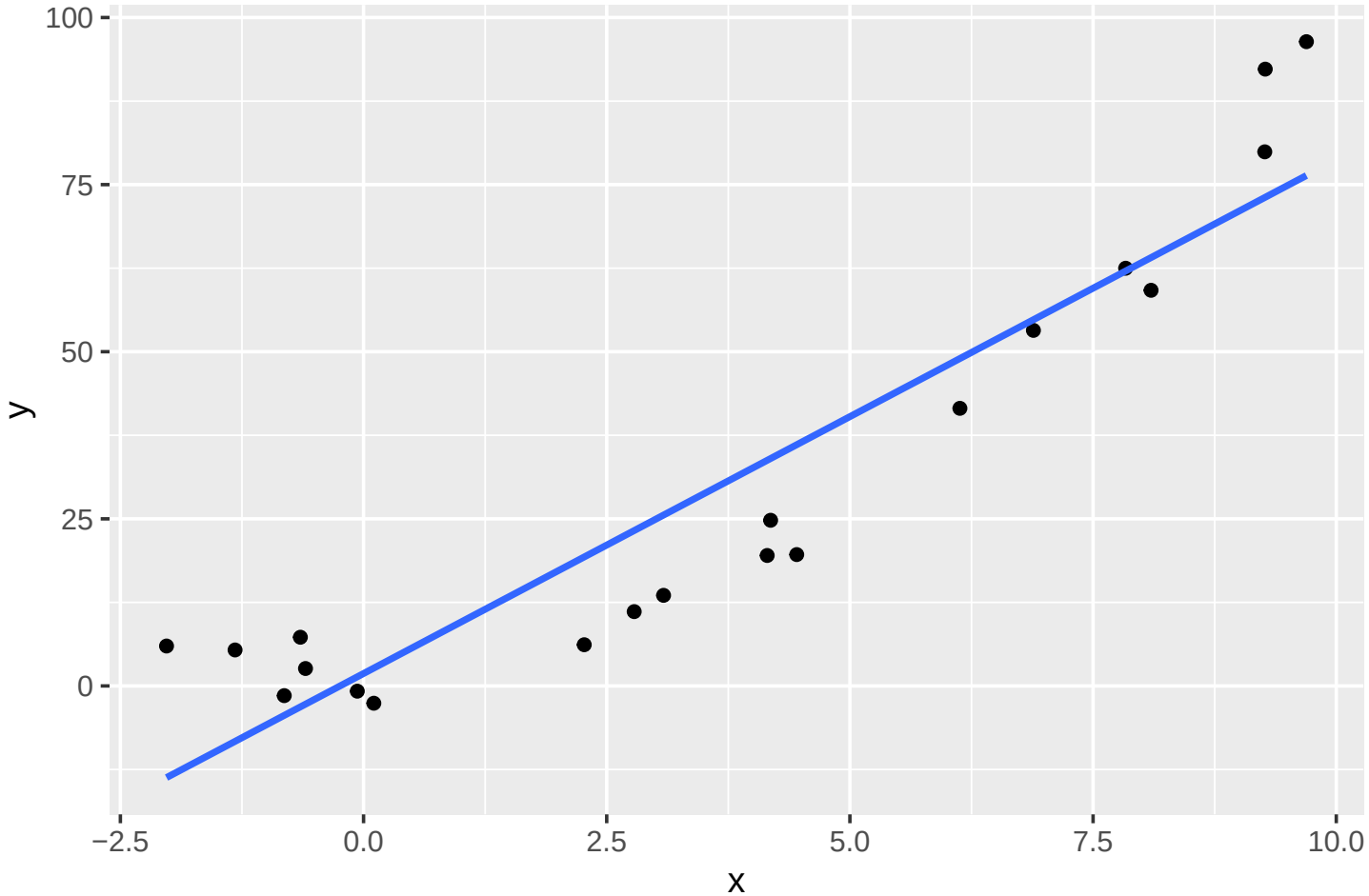
- In particular, we are often interested in **linear** relationships.

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.

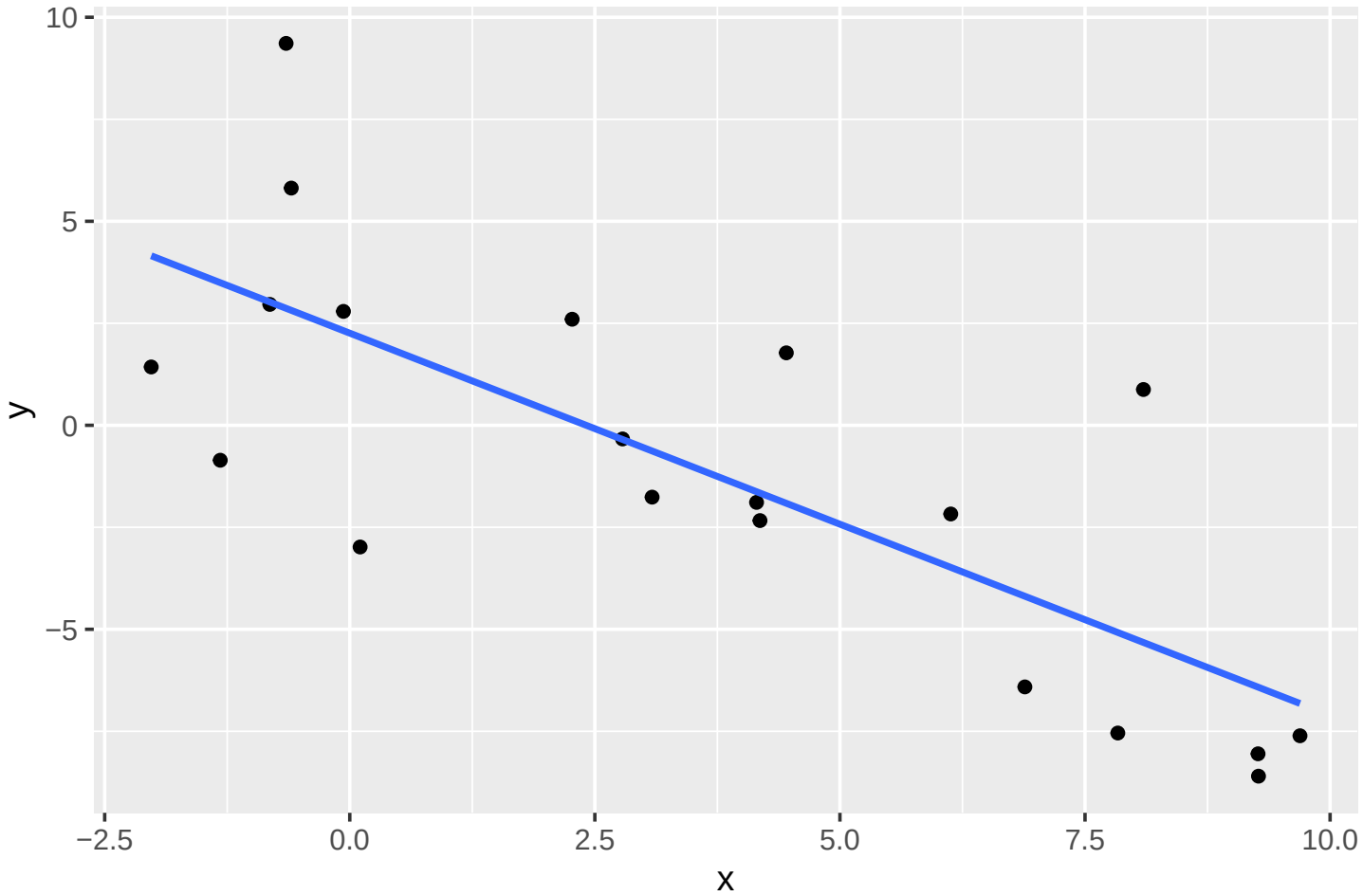


Linear Relationships

- 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.

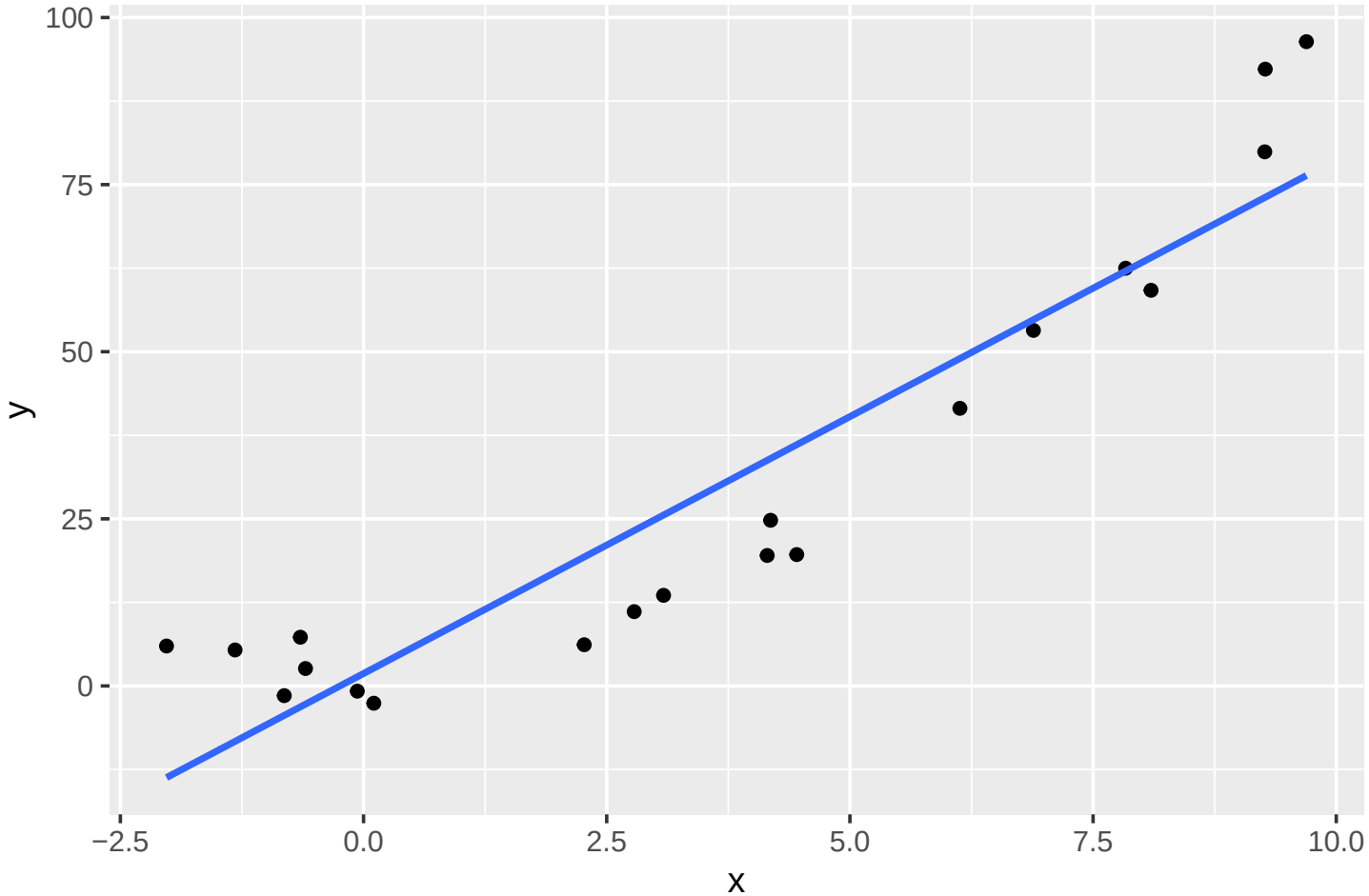


Linear Relationships

- 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**?

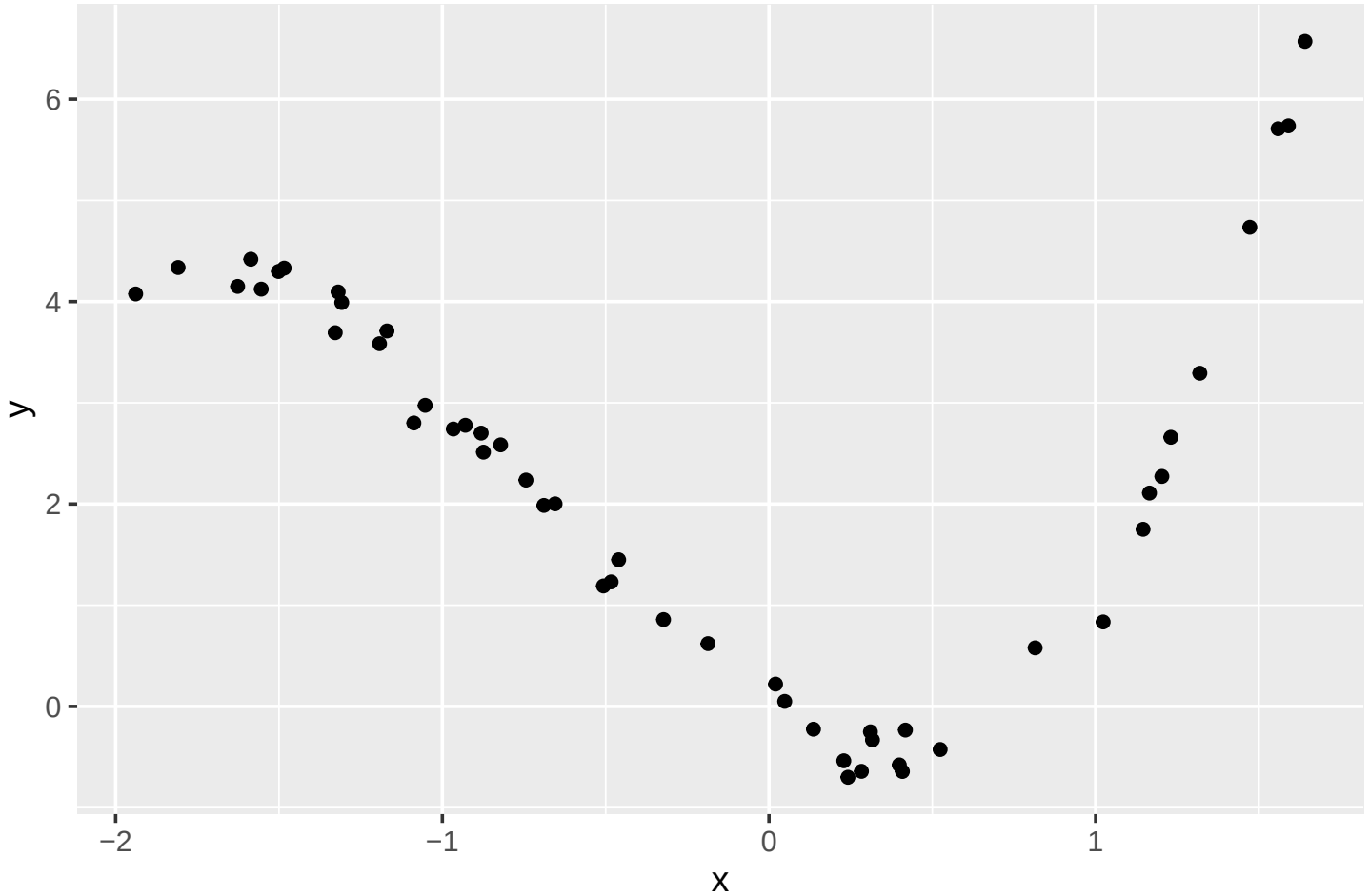


Non-Linear Relationships

- 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:



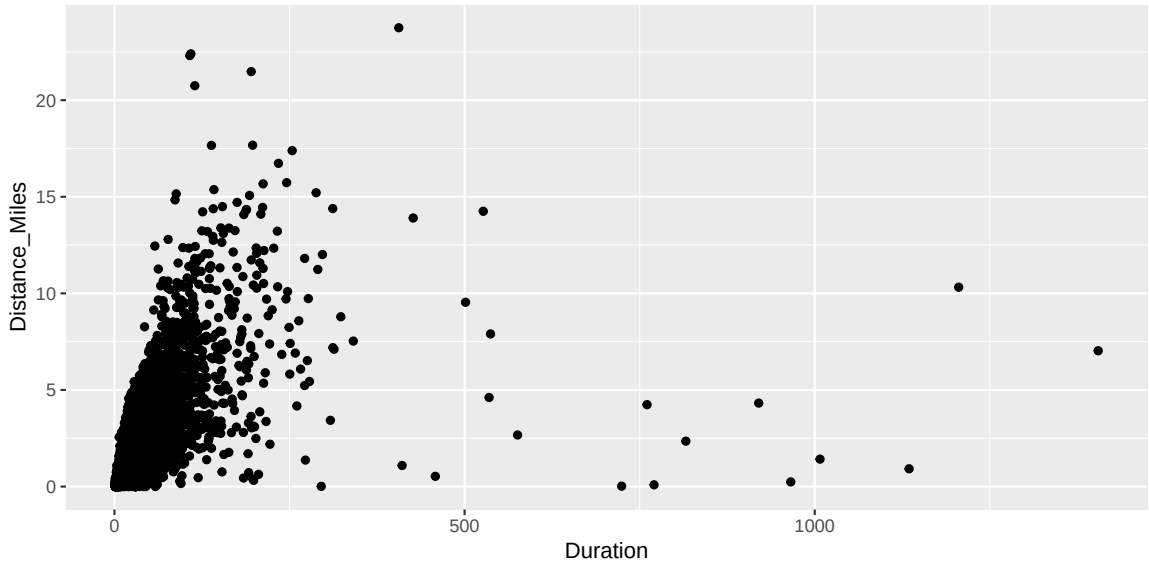
Creating Scatterplots

- 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`?

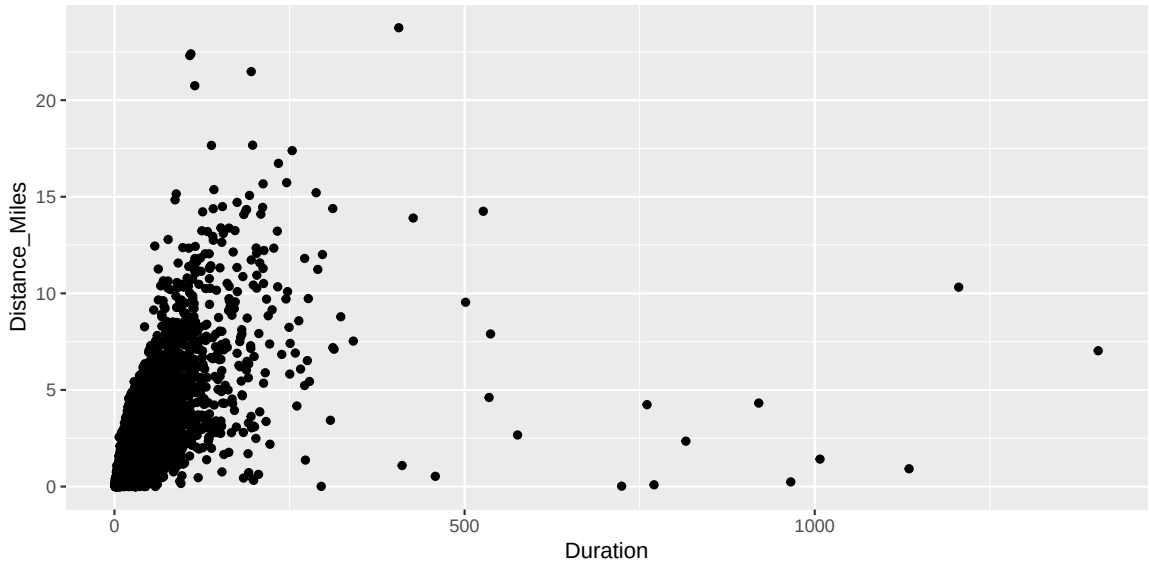
```
ggplot(data = biketown, mapping = aes(x = Duration, y = Distance_Miles)) +
  geom_point()
```



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.

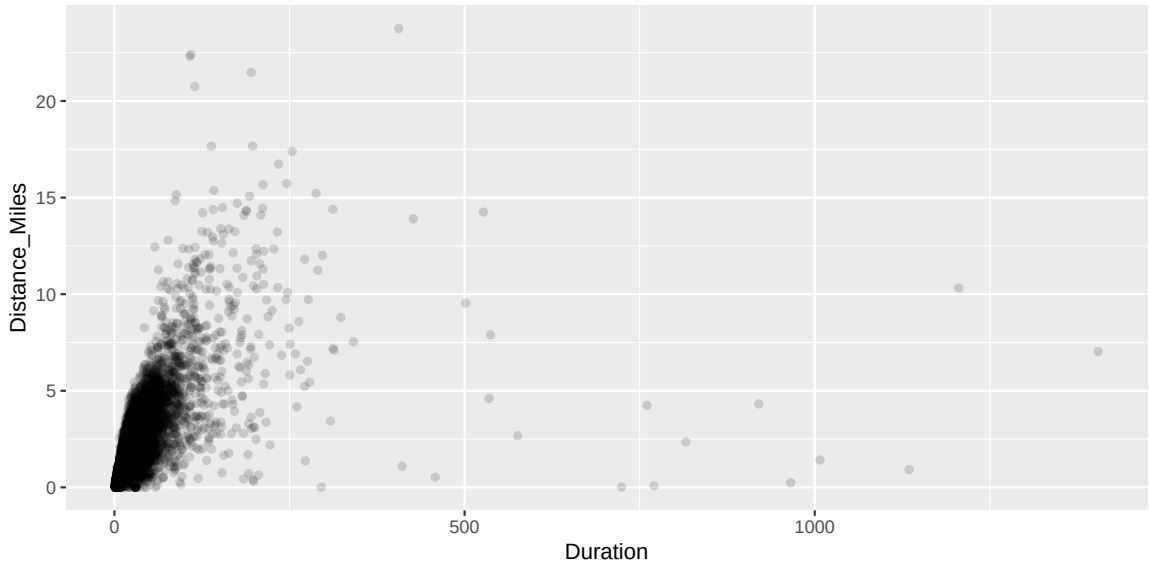
Overplotting

- 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)
```



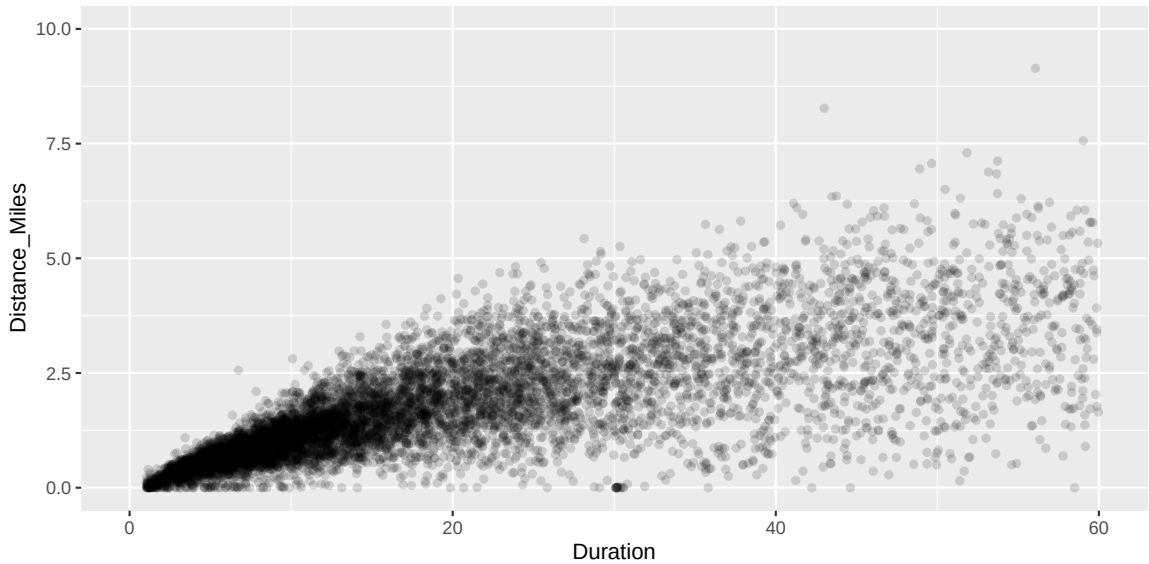
Overplotting II

- 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.

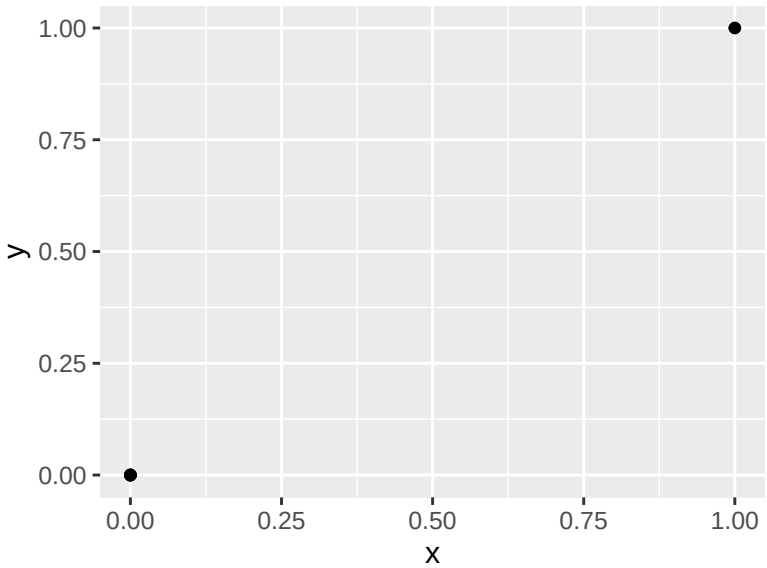
Overplotting III

- 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 $(1, 1)$:

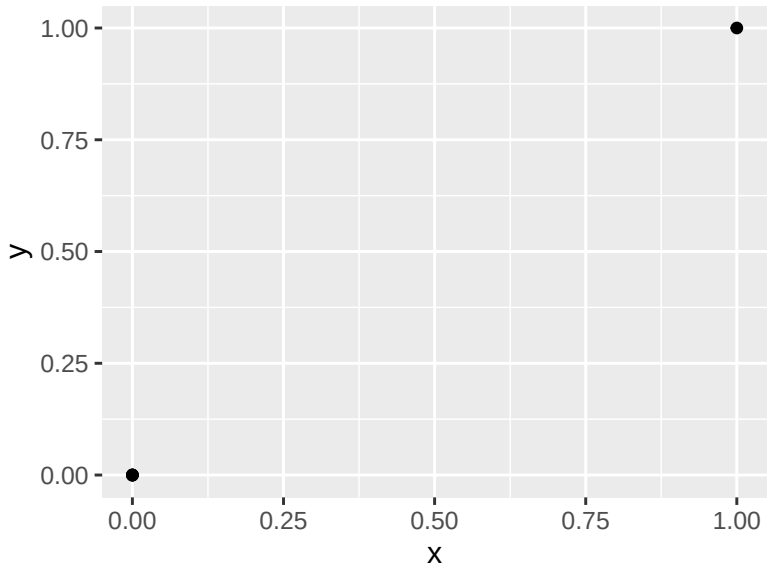
```
ggplot(data = jiggle_df, mapping = aes(x = x, y = y)) +
  geom_point()
```



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()
```

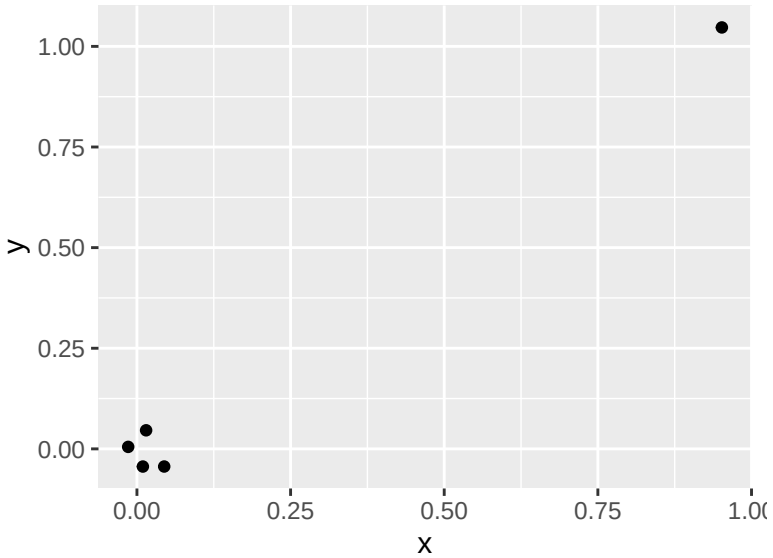


- It looks like there are just 2 observations!

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 $(1, 1)$:

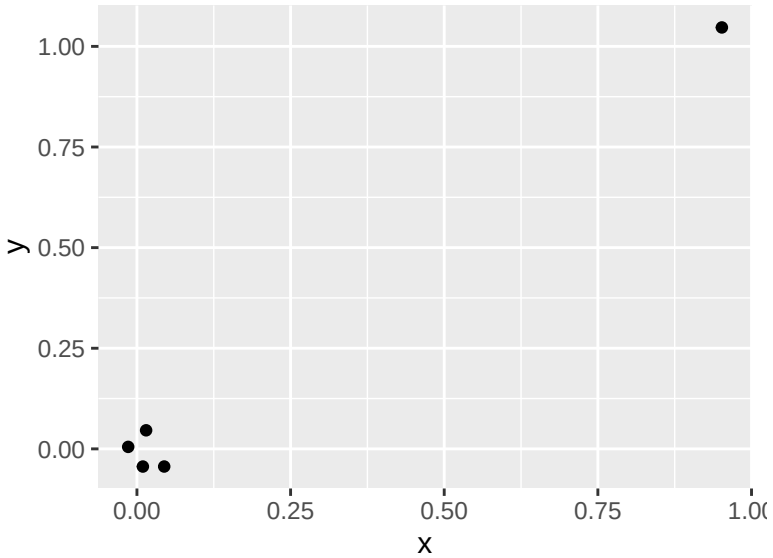
```
ggplot(data = jiggle_df, mapping = aes(x = x, y = y)) +
  geom_jitter(width = .05, height = .05)
```



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()`

Line Graphs

How do bike use patters change throughout the day?

Line Graphs

How do bike use patters change throughout the day?

```
biketown2 <- count(biketown, StartHour)
biketown2
```

```
## # A tibble: 24 x 2
##    StartHour     n
##        <int> <int>
##  1         0   118
##  2         1    69
##  3         2    50
##  4         3    20
##  5         4    35
##  6         5    71
##  7         6   104
##  8         7   270
##  9         8   492
## 10         9   392
## # ... with 14 more rows
```

Line Graphs

- Frequently, we compare two quantitative variables where one variable represents time. It is illustrative to connect neighboring points with a smooth curve.

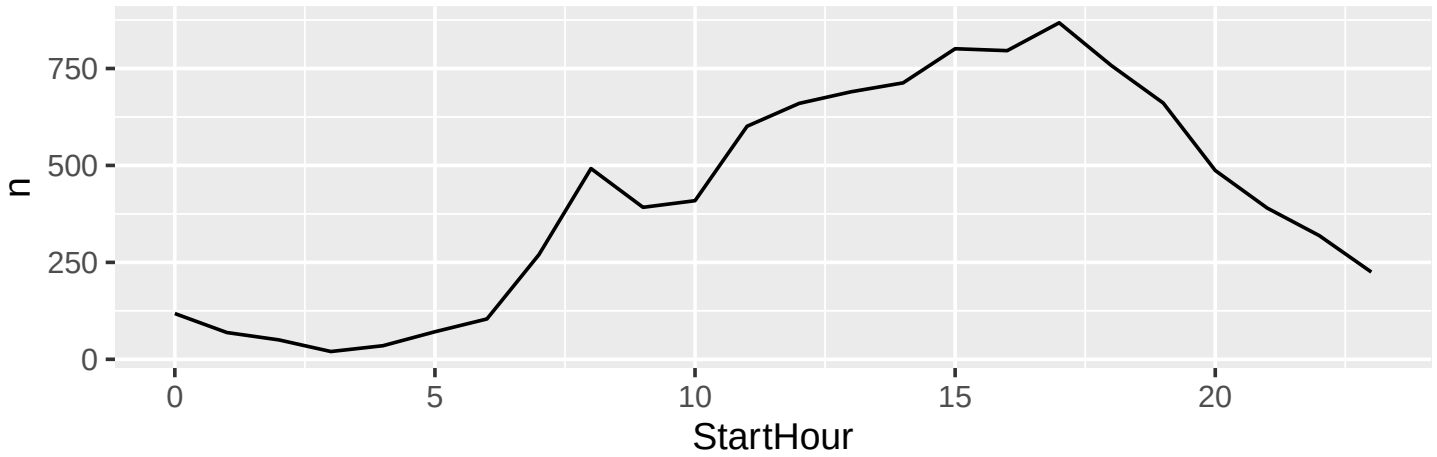
Line Graphs

- Frequently, we compare two quantitative variables where one variable represents time. It is illustrative to connect neighboring points with a smooth curve.
- These **line graphs** (or time series) provide stronger sequential and/or cyclic visual cues.

Line Graphs

- Frequently, we compare two quantitative variables where one variable represents time. It is illustrative to connect neighboring points with a smooth curve.
- These **line graphs** (or time series) provide stronger sequential and/or cyclic visual cues.

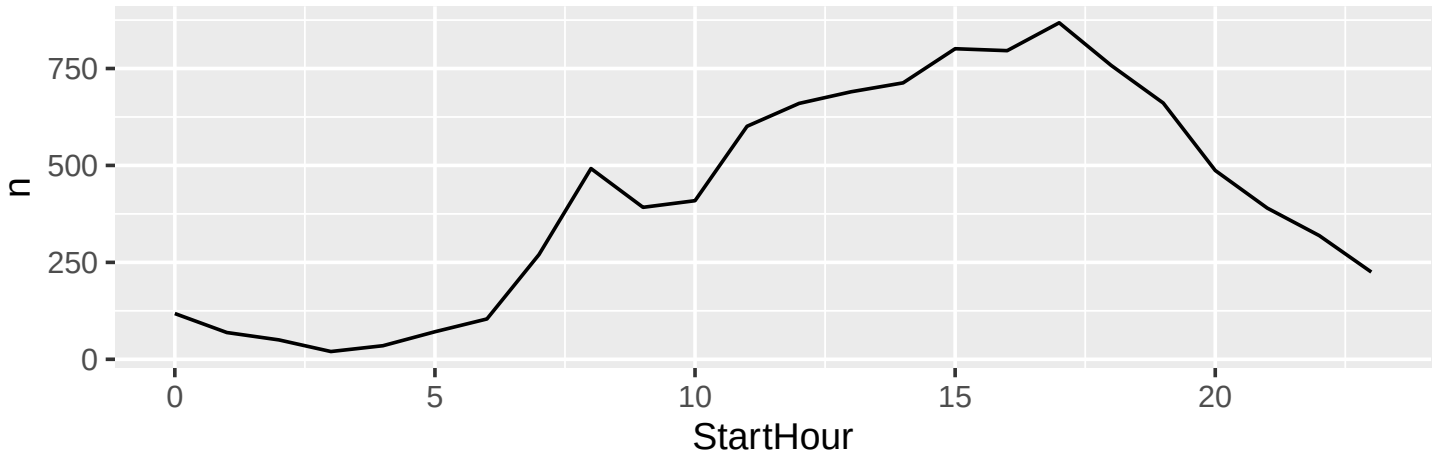
```
ggplot(data = biketown2, mapping = aes(x = StartHour, y = n)) +
  geom_line()
```



Line Graphs

- Frequently, we compare two quantitative variables where one variable represents time. It is illustrative to connect neighboring points with a smooth curve.
- These **line graphs** (or time series) provide stronger sequential and/or cyclic visual cues.

```
ggplot(data = biketown2, mapping = aes(x = StartHour, y = n)) +
  geom_line()
```



- To construct a line graph , use `geom_line()` with the aesthetic mapping `aes(x = ... , y = ...)`.