Visualizing Data in R

Pwani R Workshop

10th July 2024

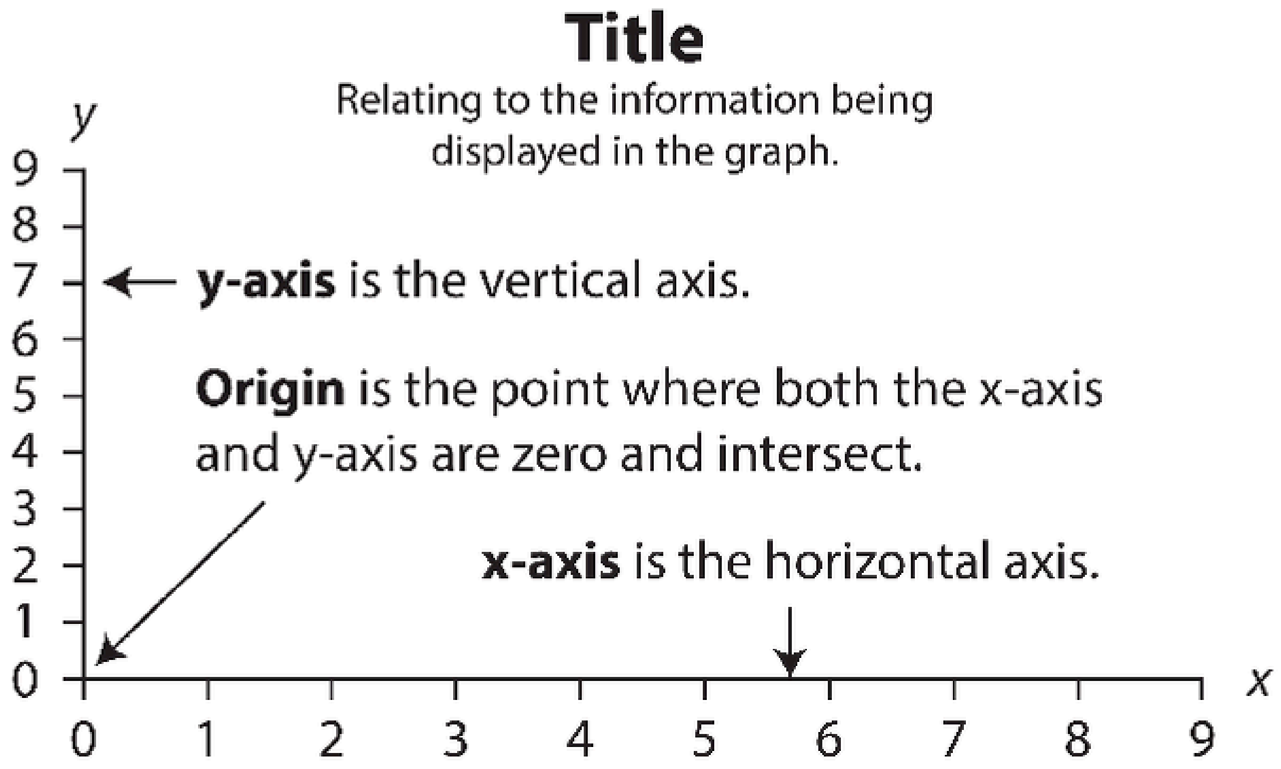
Session objective

- To create basic graphs using R.

Fundamentals data visualization

- Graphs must be labelled properly.
- Graphs should be as intuitive as possible and not misleading.

Anatomy of a graph


Title
Relating to the information being displayed in the graph.
y
9
8
7
6
5
4
3
2
1
0
y-axis is the vertical axis.
Origin is the point where both the x-axis and y-axis are zero and intersect.
x-axis is the horizontal axis.
x
0
1
2
3
4
5
6
7
8
9

Plots types

For one numerical variable:

- **Density plots**
- **Histogram**
- **Box plots**

For two numerical variables:

- **Scatter plots**
- **Line graph**

For categorical variables:

- **Bar plots**
- **Pie charts - NEVER USE THEM!!!**

Plots for numerical data

Density plot: shows the distribution of a numerical variable

density

Plots for numerical data

Histogram: shows the distribution of a numerical variable

histogram

Plots for numerical data

Box plot: shows the distribution of a numerical variable.

boxplot

Plots for numerical data

Box plot: mostly used to compare distribution between groups.

boxplot_compare

Plots for categorical data

Bar plot: useful for summarising frequencies of categories.

barplot

R graphical frameworks

1. base R
2. grid
3. lattice
4. ggplot2 - our focus this week!
5. plotly

ggplot framework

- A package for producing graphics - gg = *Grammar of Graphics*
- Created by Hadley Wickham in 2005
- Belongs to “Tidyverse” family of packages
- *“Make a ggplot”* = Make a plot with the use of ggplot2 package

Resources:

- https://ggplot2-book.org/
- https://www.opencasestudies.org/
- https://ggplot2.tidyverse.org/articles/ggplot2.html

Why learn ggplot2?

Extremely powerful/flexible

Very customizable:

- branding
- making plots interactive
- combining plots

Easier plot automation (creating plots in scripts)

Faster (eventually)

ggplot2

ggplot2 is designed to work iteratively:

- You start with a layer that draws the axes
- Add a layer that shows the raw data
- Add layers of annotations and statistical summaries

Layers are are placed on top of each other using **+**

Every ggplot2 plot has three key components

data, i.e., the data that should be visualized.

aesthetics, i.e., which variable should be mapped to which axis using `aes()`.

geometrics, i.e., the type of graph that should be created, e.g., scatter plot or boxplot.

However, you **can** specify for more details, for instance:

- **scale** used in the X and Y axes
- **themes** - e.g., change background color
- **facets** - i.e., specify subplots

First plot with `ggplot2` package

First order of business!

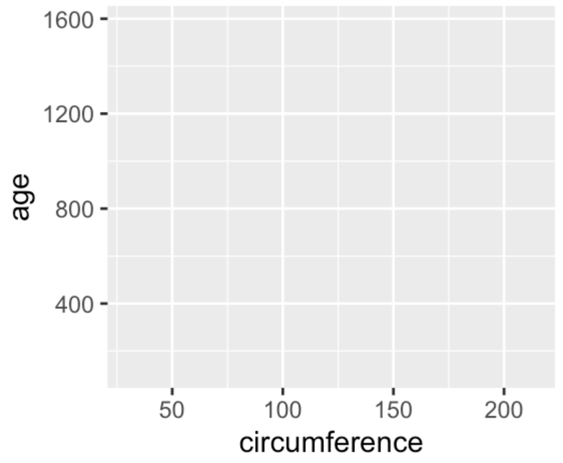
- Install the *ggplot2* package
    - install.packages(“ggplot2”)
- load all libraries that were used on Day 2 using `library()`
- load the *ggplot2* package

First layer of code with `ggplot2` package

Aesthetic mapping `aes(x= , y =)` describes how variables in our data are mapped to elements of the plot - Note you don't need to use `mapping` but it is helpful to know what we are doing.

```
library(ggplot2) # don't forget to load ggplot2
# This is not code but shows the general format
ggplot({data_to_plot}, aes(x = {var in data to plot},
                                   y = {var in data to plot}))
```

```
ggplot(Orange, aes(x = circumference, y = age))
```



Next layer code with **`ggplot2`** package

There are many to choose from, to list just a few:

- `geom_point()` – points
- `geom_line()` – lines to connect observations
- `geom_boxplot()` – boxplots
- `geom_histogram()` – histogram
- `geom_bar()` – bar plot
- `geom_tile()` – blocks filled with color

Next layer code with `ggplot2` package

When to use what plot? A few examples:

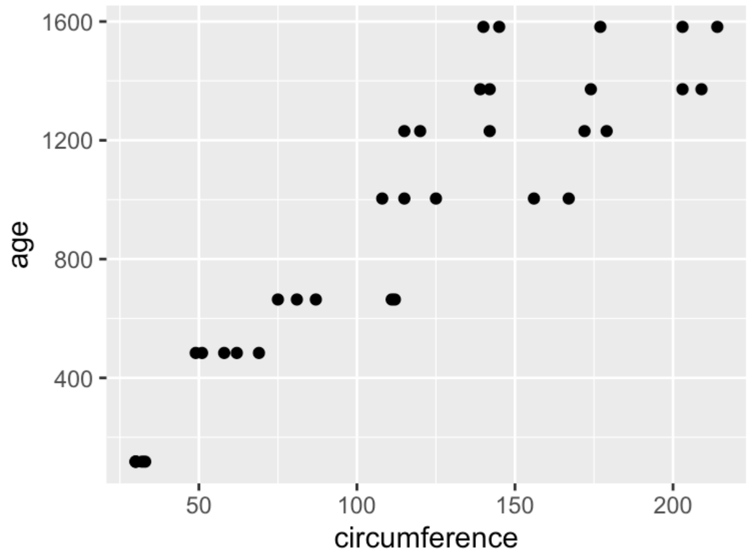
- **scatterplot (`geom_point()`)**: to examine the relationship between two sets of continuous numeric data
- **barplot (`geom_bar()`)**: to compare the distribution of a quantitative variable (numeric) between groups or categories
- **histogram (`geom_hist()`)**: to observe the overall distribution of numeric data
- **boxplot (`geom_boxplot()`)**: to compare values between different factor levels or categories

Next layer code with **`ggplot2`** package

Need the `+` sign to add the next layer to specify the type of plot

```
ggplot({data_to plot}, aes(x = {var in data to plot},
                                  y = {var in data to plot})) +

  geom_{type of plot}</div>
```

```
ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point()
```



Tip - plus sign **+** must come at end of line

Having the + sign at the beginning of a line will not work!

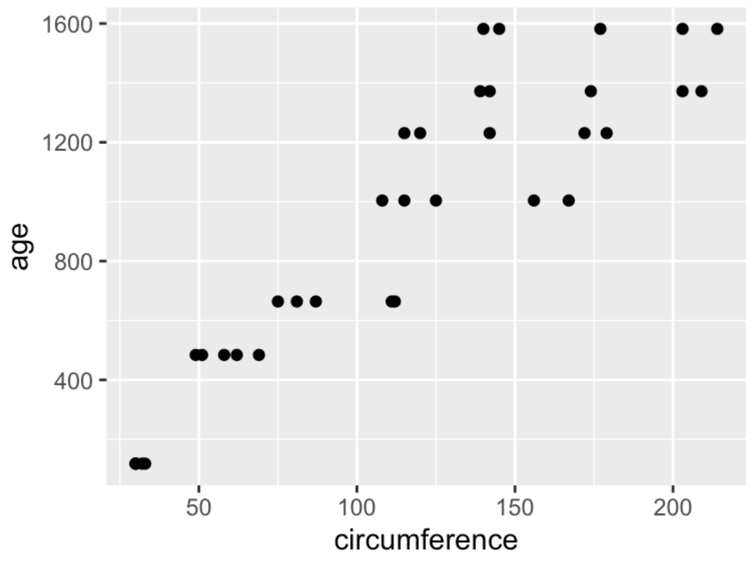
```
ggplot(Orange, aes(x = circumference, y = age))
 + geom_point()
```

Pipes will also not work in place of `+`!

```
ggplot(Orange, aes(x = circumference, y = age)) %>%
   geom_point()
```

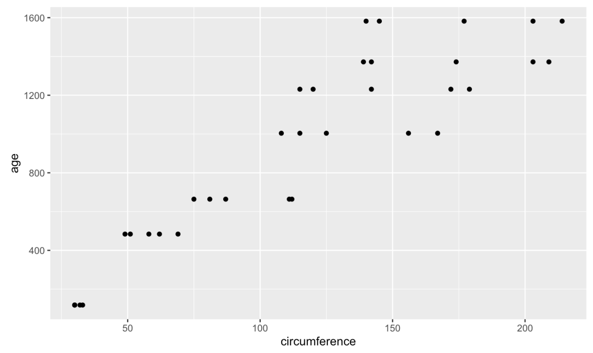
Plots can be assigned as an object

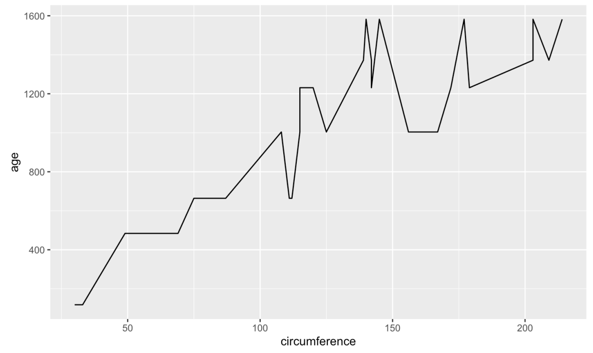
```
plt1 <- ggplot(Orange, aes(x = circumference, y = age)) +
          geom_point()

plt1
```



Examples of different geoms

```
plt1 <- ggplot(Orange, aes(x = circumference, y = age)) +
          geom_point()

plt2 <- ggplot(Orange, aes(x = circumference, y = age)) +
          geom_line()

plt1 # fig.show = "hold" makes plots appear
plt2 # next to one another in the chunk settings
```

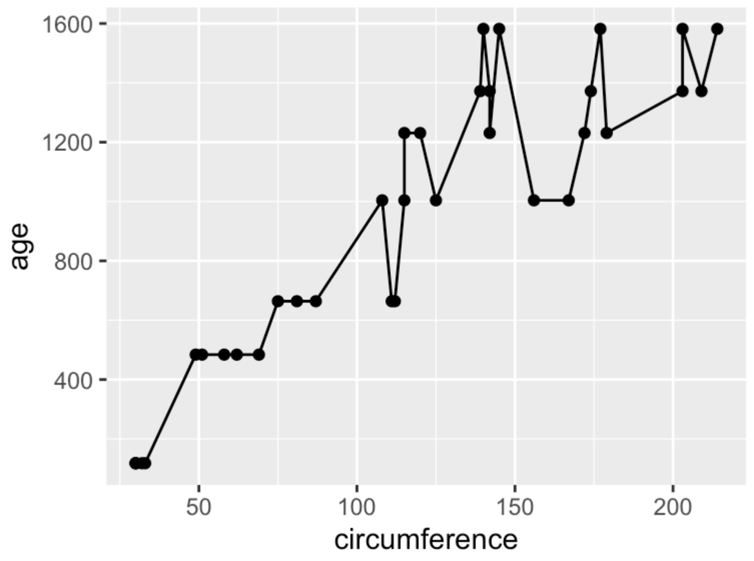




Specifying plot layers: combining multiple layers

Layer a plot on top of another plot with `+`

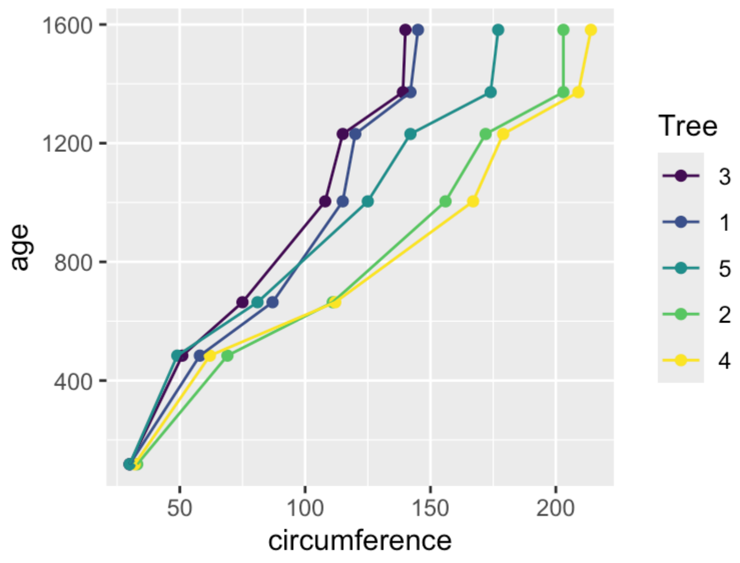
```
ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point() +
  geom_line()
```



Adding color

You can map color to a variable

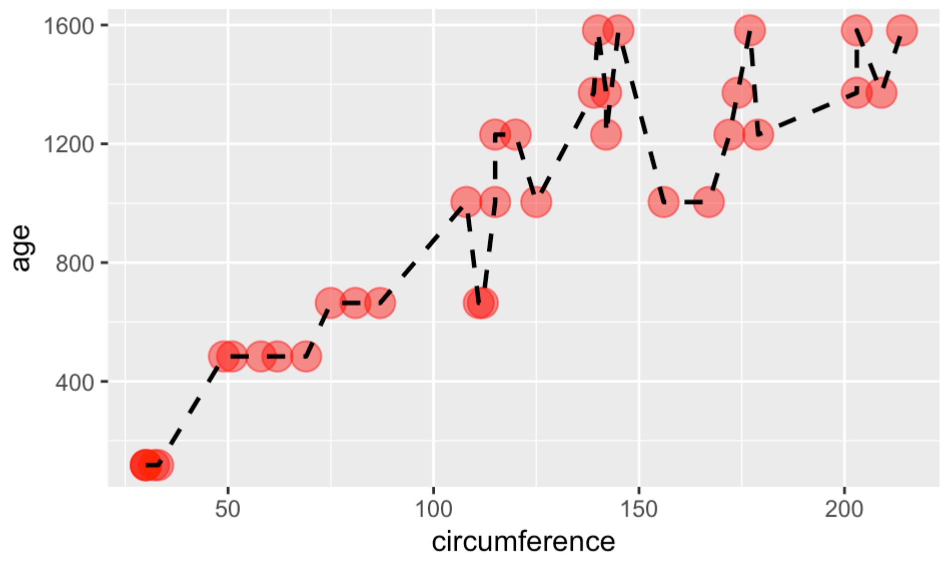
```
ggplot(Orange, aes(x = circumference, y = age, color = Tree)) +
  geom_point() +
  geom_line()
```



Adding color or change the color of each plot layer

You can change look of each layer separately. Note the arguments like `linetype` and `alpha` that allow us to change the opacity of the points and style of the line respectively.

```
ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "black", linetype = 2)
```



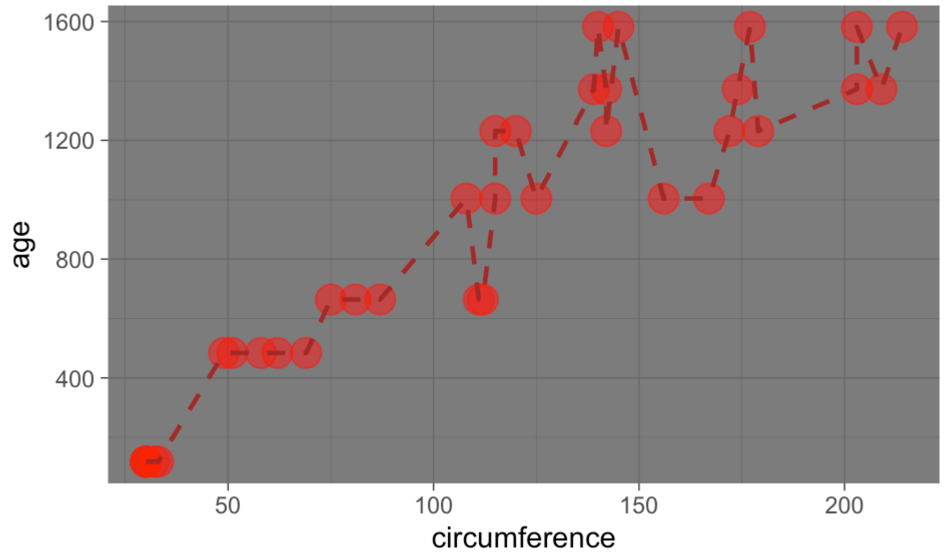
`linetype` can be given as a number. See the docs for what numbers correspond to what `linetype`!

Customize the look of the plot

Customize the look of the plot

You can change the look of whole plot using `theme_*()` functions.

```
ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  theme_dark()
```



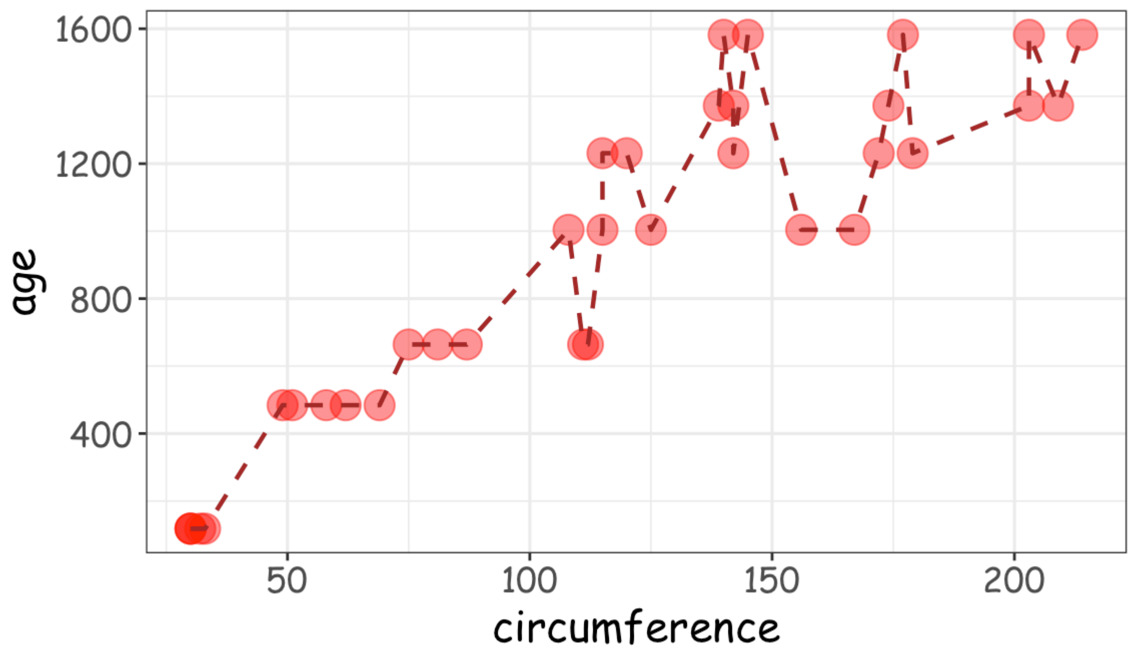
More themes!

There’s not only the built in ggplot2 themes but all kinds of themes from other packages! - ggthemes - ThemePark package - hrbr themes

Customize the look of the plot

You can change the look of whole plot - **specific elements, too** - like changing font and font size - or even more fonts

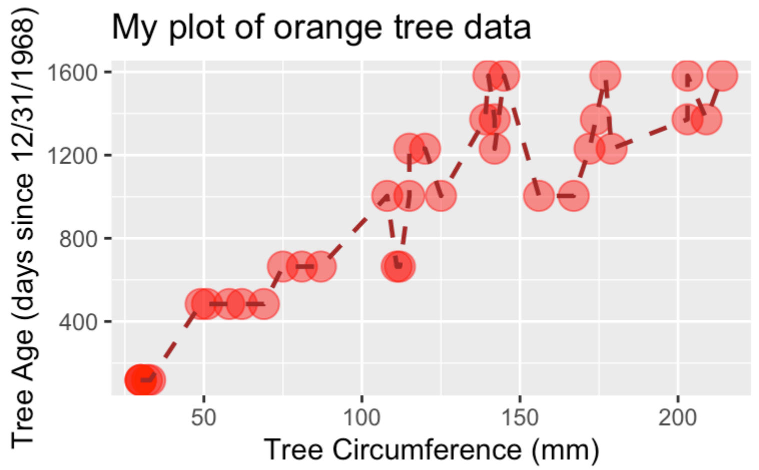
```
ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  theme_bw() +
  theme(text=element_text(size=16,  family="Comic Sans MS"))
```



Adding labels

The `labs()` function can help you add or modify titles on your plot. The `title` argument specifies the title. The x argument specifies the x axis label. The y argument specifies the y axis label.

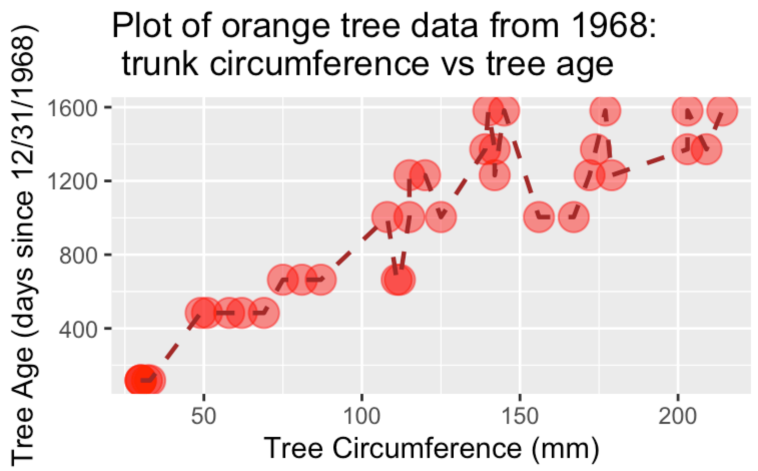
```
ggplot(Orange, aes(x = circumference, y = age)) +
            geom_point(size = 5, color = "red", alpha = 0.5) +
            geom_line(size = 0.8, color = "brown", linetype = 2) +
            labs(title = "My plot of orange tree data",
              x = "Tree Circumference (mm)",
              y = "Tree Age (days since 12/31/1968)")
```



Adding labels line break

Line breaks can be specified using `\n` within the `labs()` function to have a label with multiple lines.

```
ggplot(Orange, aes(x = circumference, y = age)) +
            geom_point(size = 5, color = "red", alpha = 0.5) +
            geom_line(size = 0.8, color = "brown", linetype = 2) +
            labs(title = "Plot of orange tree data from 1968: \n trunk circumference vs tree age",
              x = "Tree Circumference (mm)",
              y = "Tree Age (days since 12/31/1968)")
```



Changing axis: specifying axis scale

`scale_x_continuous()` and `scale_y_continuous()` can change how the axis is plotted. Can use the `breaks` argument to specify how you want the axis ticks to be.

```
range(pull(Orange, circumference))
```

```
## [1]  30 214
```

```
range(pull(Orange, age))
```

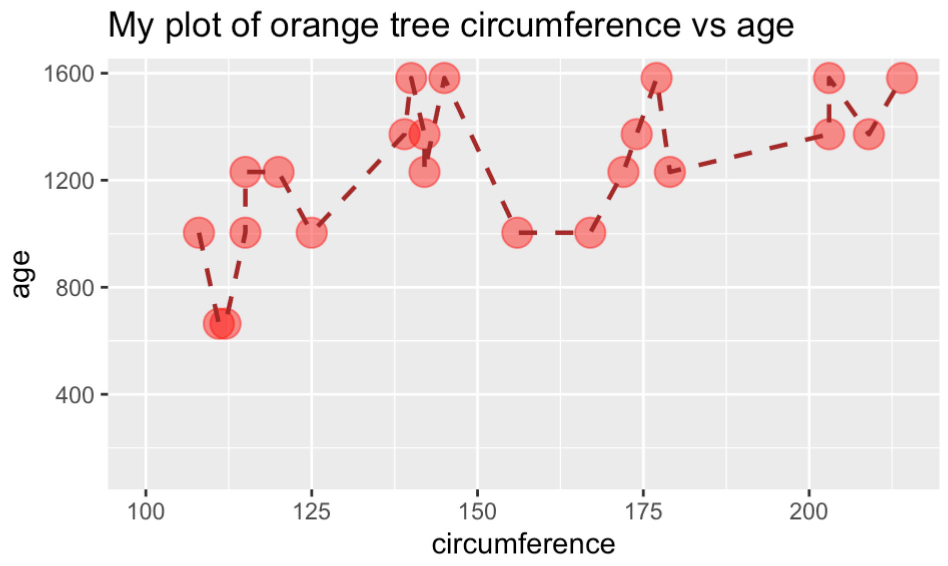
```
## [1]  118 1582
```

```
plot_scale <-ggplot(Orange, aes(x = circumference, y = age)) +
                geom_point(size = 5, color = "red", alpha = 0.5) +
                geom_line(size = 0.8, color = "brown", linetype = 2) +
                scale_x_continuous(breaks = seq(from = 20, to = 240, by = 20)) +
                scale_y_continuous(breaks = seq(from = 100, to = 1600, by = 200))
```

Changing axis: specifying axis limits

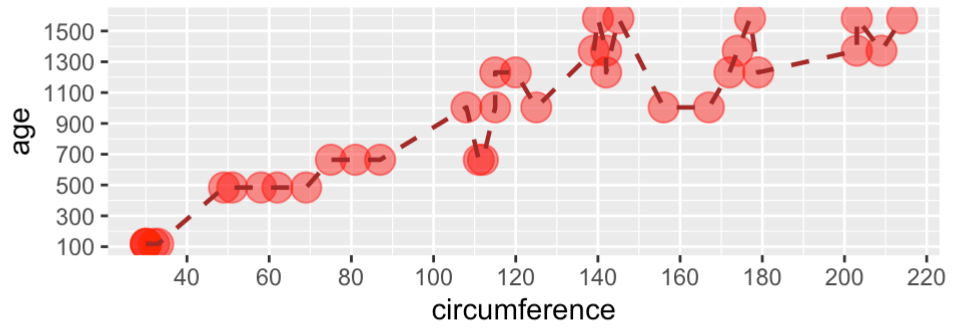
`xlim()` and `ylim()` can specify the limits for each axis

```
ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "My plot of orange tree circumference vs age") +
  xlim(100, max(pull(Orange, circumference)))
```

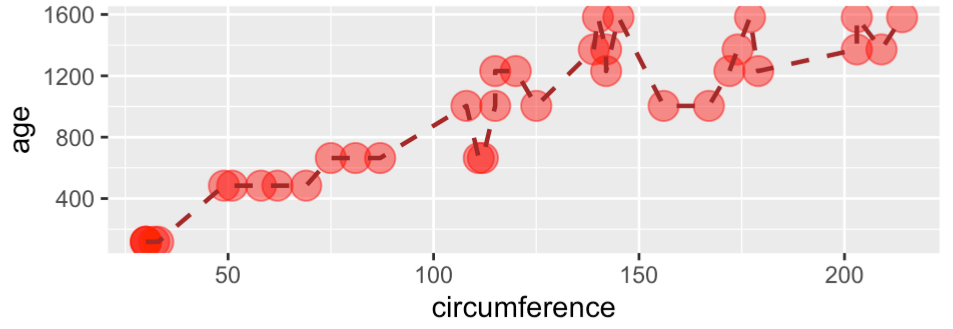


Changing axis: specifying axis scale

```
plot_scale
```



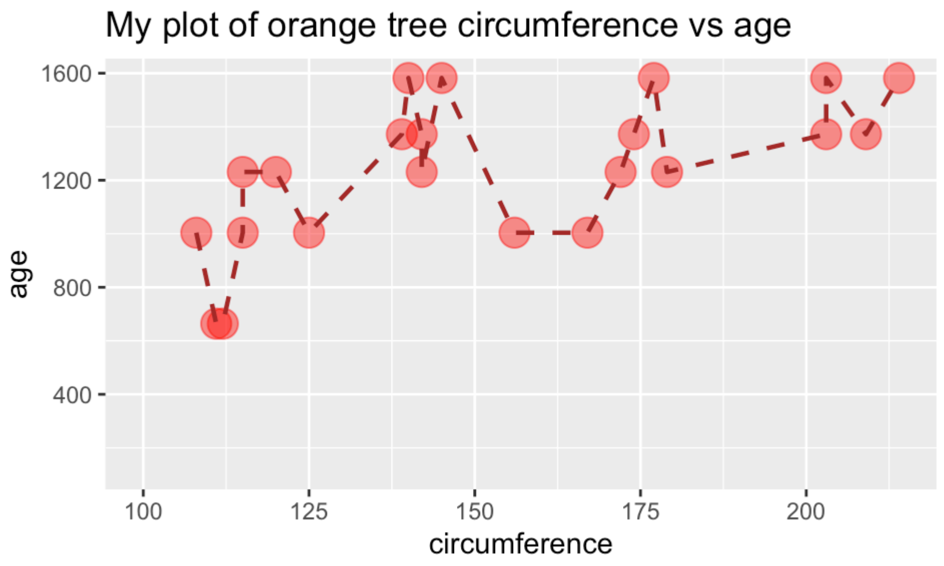
```
ggplot(Orange, aes(x = circumference, y = age)) +
                geom_point(size = 5, color = "red", alpha = 0.5) +
                geom_line(size = 0.8, color = "brown", linetype = 2)
```



Changing axis: specifying axis limits

`xlim()` and `ylim()` can specify the limits for each axis

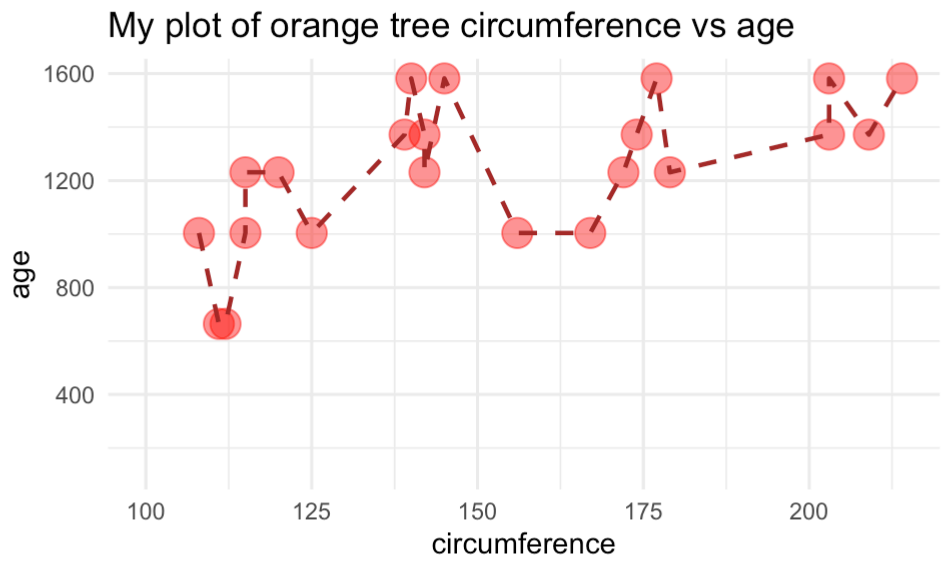
```
ggplot(Orange, mapping = aes(x = circumference, y = age)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "My plot of orange tree circumference vs age") +
  xlim(100, max(pull(Orange, circumference)))
```



Modifying plot objects

You can add to a plot object to make changes! Note that we can save our plots as an object like `plt1` below. And now if we reference `plt1` again our plot will print out!

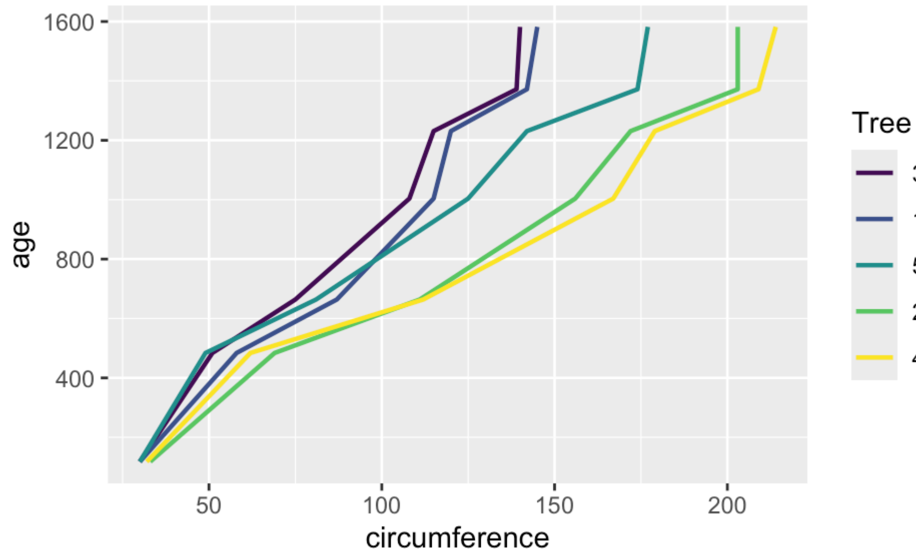
```r
plt1 <- ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "My plot of orange tree circumference vs age") +
  xlim(100, max(pull(Orange, circumference)))

plt1 + theme_minimal()
```



Overwriting specifications

It’s possible to go in and change specifications with newer layers

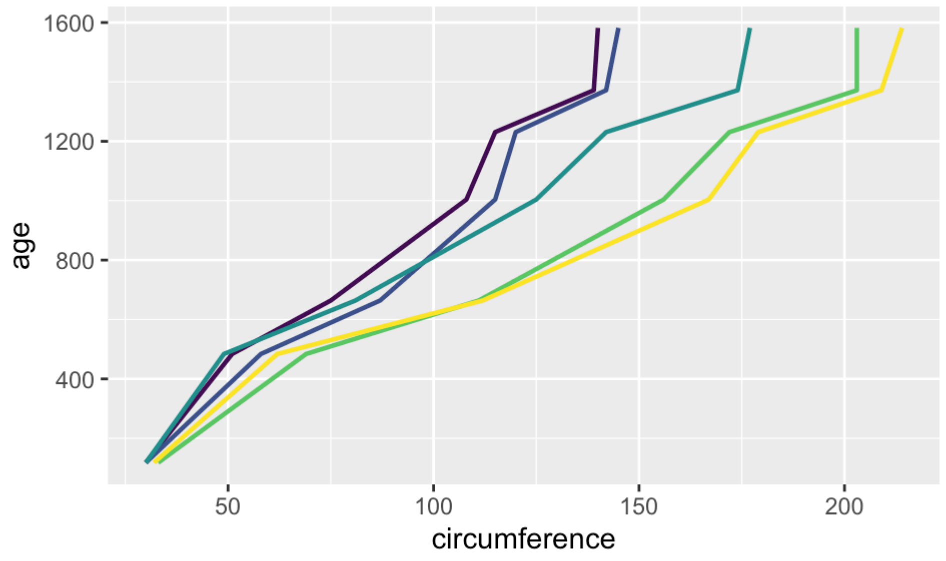
```
Orange %>% ggplot(aes(x = circumference,
                      y = age,
                      color = Tree)) +
  geom_line(size = 0.8)
```



Removing the legend label

You can use `theme(legend.position = "none")` to remove the legend.

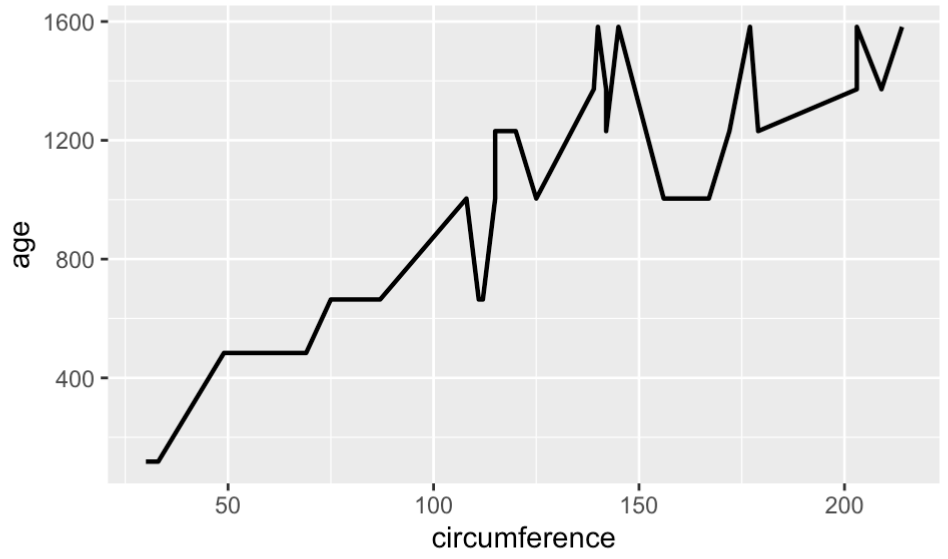
```
Orange %>% ggplot(aes(x = circumference,
                      y = age,
                      color = Tree)) +
  geom_line(size = 0.8) +
  theme(legend.position = "none")
```



Overwriting specifications

It’s possible to go in and change specifications with newer layers

```
Orange %>% ggplot(aes(x = circumference,
                      y = age,
                      color = Tree)) +
  geom_line(size = 0.8, color = "black")
```



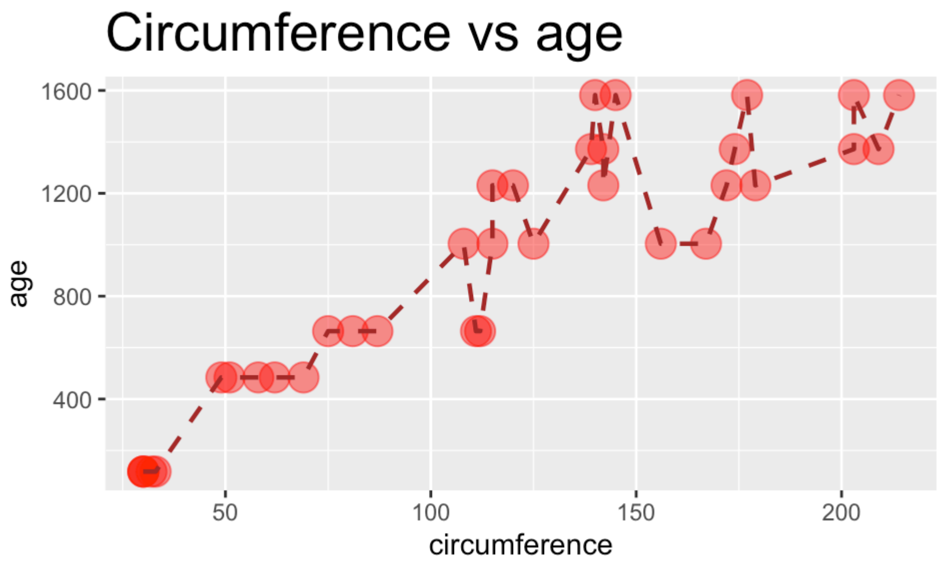
Summary

- `ggplot()` specifies what data use and what variables will be mapped to where
- inside `ggplot()`, `aes(x = , y = , color =)` specify what variables correspond to what aspects of the plot in general
- layers of plots can be combined using the `+` at the **end** of lines
- special `theme_*()` functions can change the overall look
- individual layers can be customized using arguments like: `size`, `color` `alpha` (more transparent is closer to 0), and `linetype`
- labels can be added with the `labs()` function and `x`, `y`, `title` arguments - the `\n` can be used for line breaks
- `xlim()` and `ylim()` can limit or expand the plot area
- `scale_x_continuous()` and `scale_y_continuous()` can modify the scale of the axes
- by default, `ggplot()` removes points with missing values from plots.

theme() function:

The `theme()` function can help you modify various elements of your plot. Here we will adjust the font size of the plot title.

```
ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "Circumference vs age") +
  theme(plot.title = element_text(size = 20))
```



theme() function

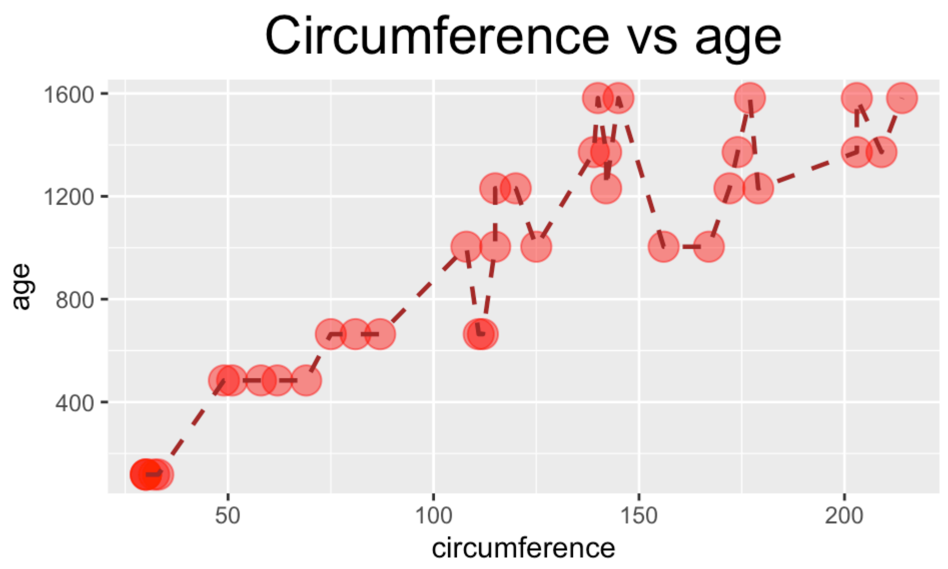
The `theme()` function always takes:

1. an object to change (use `?theme()` to see - `plot.title`, `axis.title`, `axis.ticks` etc.)
2. the aspect you are changing about this: `element_text()`, `element_line()`, `element_rect()`, `element_blank()`
3. what you are changing:
   - text: `size`, `color`, `fill`, `face`, `alpha`, `angle`
   - position: `"top"`, `"bottom"`, `"right"`, `"left"`, `"none"`
   - rectangle: `size`, `color`, `fill`, `linetype`
   - line: `size`, `color`, `linetype`

theme() function: center title and change size

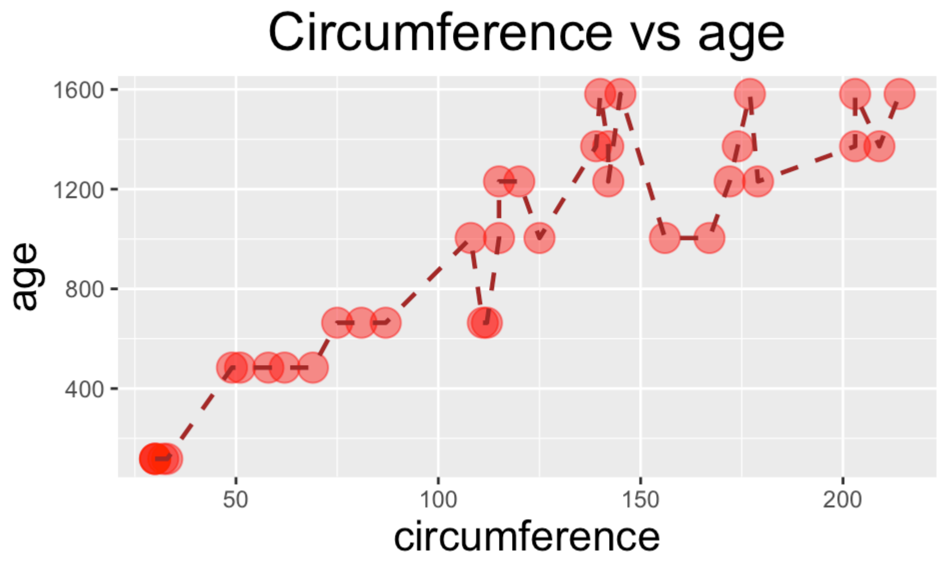
The `theme()` function can help you modify various elements of your plot. Here we will adjust the horizontal justification (`hjust`) of the plot title.

```
ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "Circumference vs age") +
  theme(plot.title = element_text(hjust = 0.5, size = 20))
```



theme() function: change title and axis format

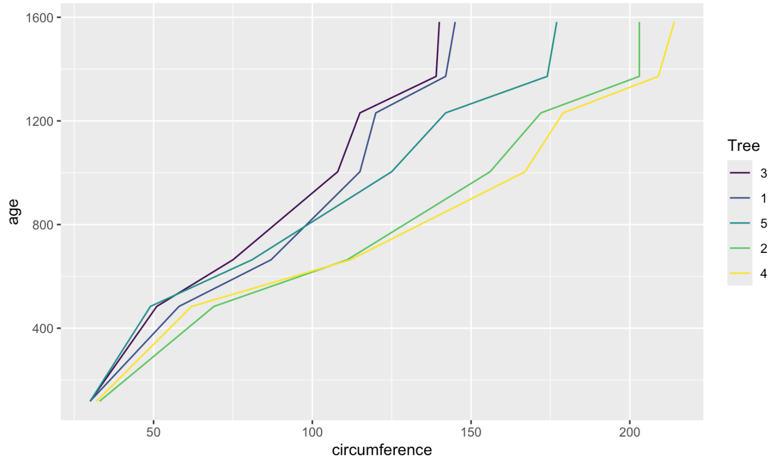
```
ggplot(Orange, aes(x = circumference, y = age)) +
  geom_point(size = 5, color = "red", alpha = 0.5) +
  geom_line(size = 0.8, color = "brown", linetype = 2) +
  labs(title = "Circumference vs age") +
  theme(plot.title = element_text(hjust = 0.5, size = 20),
        axis.title = element_text(size = 16))
```

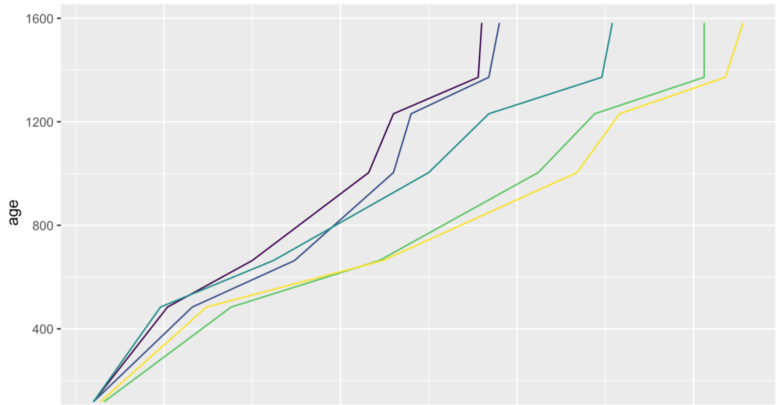


theme() function: moving (or removing) legend

If specifying position - use: “top”, “bottom”, “right”, “left”, “none”

```
ggplot(Orange, aes(x = circumference, y = age, color = Tree)) +
  geom_line()

ggplot(Orange, aes(x = circumference, y = age, color = Tree)) +
  geom_line() +
  theme(legend.position = "none")
```





Cheatsheet about theme

https://github.com/claragranell/ggplot2/blob/main/ggplot_theme_system_cheatsheet.pdf

Keys for specifications

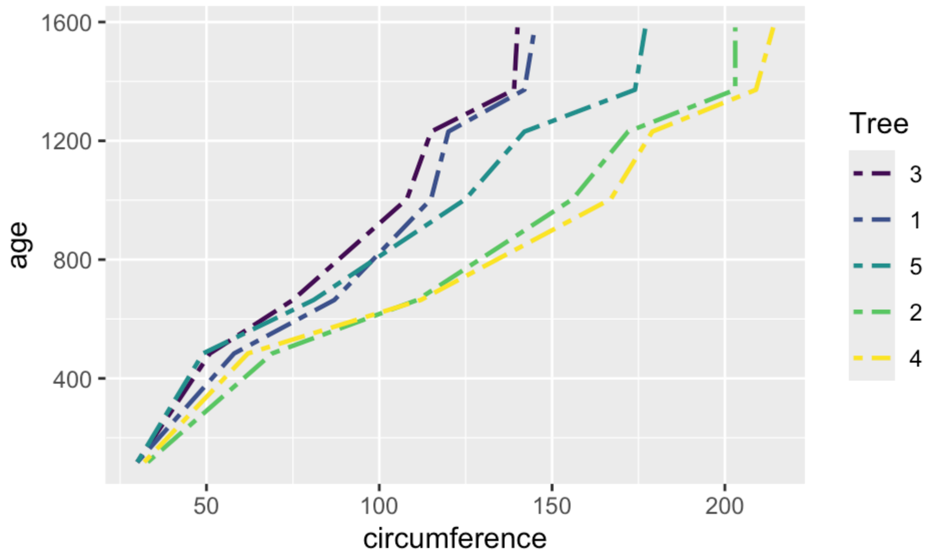
`linetype`

source

Linetype key

- *geoms* that draw lines have a `linetype` parameter
- these include values that are strings like “blank”, “solid”, “dashed”, “dotdash”, “longdash”, and “twodash”

```
Orange %>% ggplot(aes(x = circumference,
                      y = age,
                      color = Tree)) +
  geom_line(size = 0.8, linetype = "twodash")
```



Keys for specifications

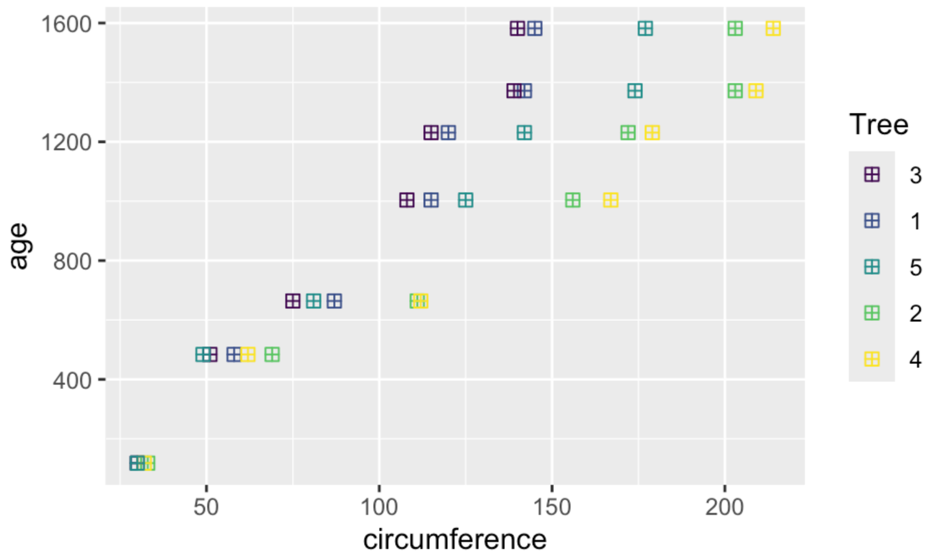
`shape`

source

shape key

- *geoms* that draw have points have a `shape` parameter
- these include numeric values (don't need quotes for these) and some characters values (need quotes for these)

```
Orange %>% ggplot(aes(x = circumference,
                      y = age,
                      color = Tree)) +
  geom_point(size = 2, shape = 12)
```



Can make your own theme to use on plots!

Guide on how to: https://rpubs.com/mclaire19/ggplot2-custom-themes

Group and/or color by variable's values

First, we will read in some data for the purpose of demonstration about food prices over time.