wiki.qcbs.ca
CENTRE DE LA SCIENCE DE LA BIODIVERSITÉ DU QUÉBEC
QUEBEC CENTRE FOR BIODIVERSITY SCIENCE

Basic steps for effective and reproducible data visualization

Tania L. Maxwell [123]

QCBS R Symposium October 10-12, 2019

 tania.maxwell@inra.fr

 tania_maxwell7

[1] INRA, Bordeaux Sciences Agro, UMR 1391 ISPA, 33140 Villenave d'Ornon, France

[2] Université de Bordeaux, 35 place Pey Berland, 33000 Bordeaux, France

[3] Université Laval, Département des sciences du bois et de la forêt, Québec, Canada

References used throughout the document

- Wilke, C. O. (2019). *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures*. O'Reilly Media.
- Rougier, N. P., Droettboom, M., & Bourne, P. E. (2014). Ten simple rules for better figures. PLoS Comput Biol 10(9): e1003833.
- Raab, G., Calitri, F., & Schiedung, M. (2019, April). *Visualizing Science*. Presentation at the European Geosciences Union General Assembly, Vienna, Austria.

Useful ressources

- Wilke, C. O. (2019). *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures*. O'Reilly Media.
  - Online version: https://serialmentor.com/dataviz/
- Link to useful R graphs:
  - https://www.r-graph-gallery.com/index.html
  - http://shinyapps.stat.ubc.ca/r-graph-catalog/#
- Rougier, N. P., Droettboom, M., & Bourne, P. E. (2014). Ten simple rules for better figures. PLoS Comput Biol 10(9): e1003833.
- https://wiki.qcbs.ca/r_workshop3
- https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

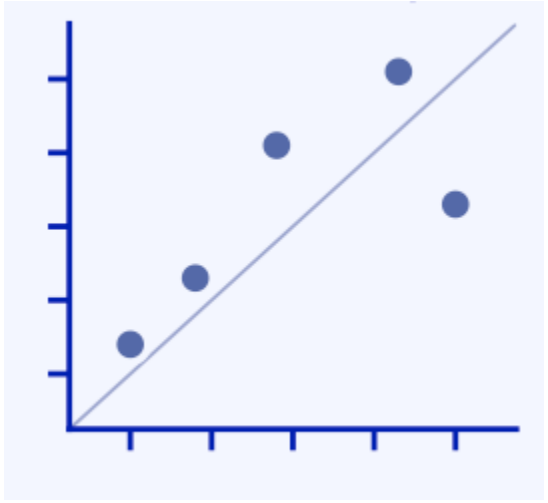
Workshop outline

1. Rules for better figures
2. Using R
   - 2.1: Building your plot (Orange tree data)
   - 2.2: Building a 2nd graph (Temperature data)
   - 2.3: Plotting averages using ddply
   - 2.4: Fitting a model (i.e. a logistical model)
   - 2.5: Saving your figure
3. Have any tips?

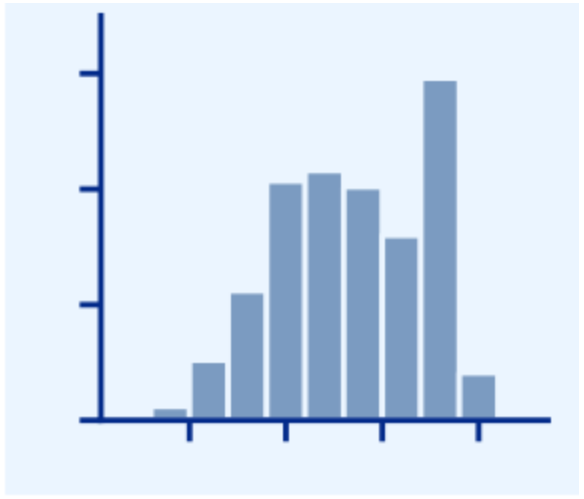
Part 1: Rules for better figures

1. What's your story?

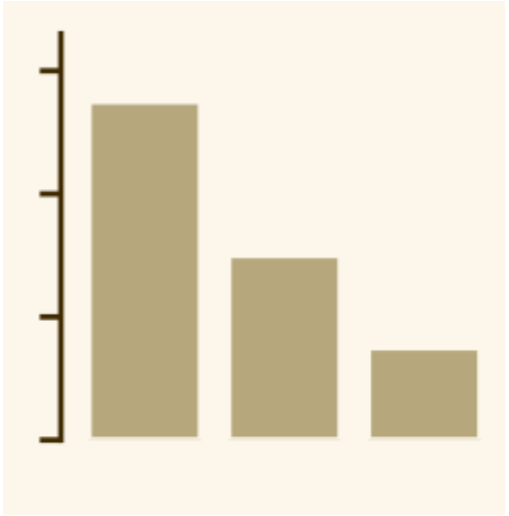
Relationship



Distribution



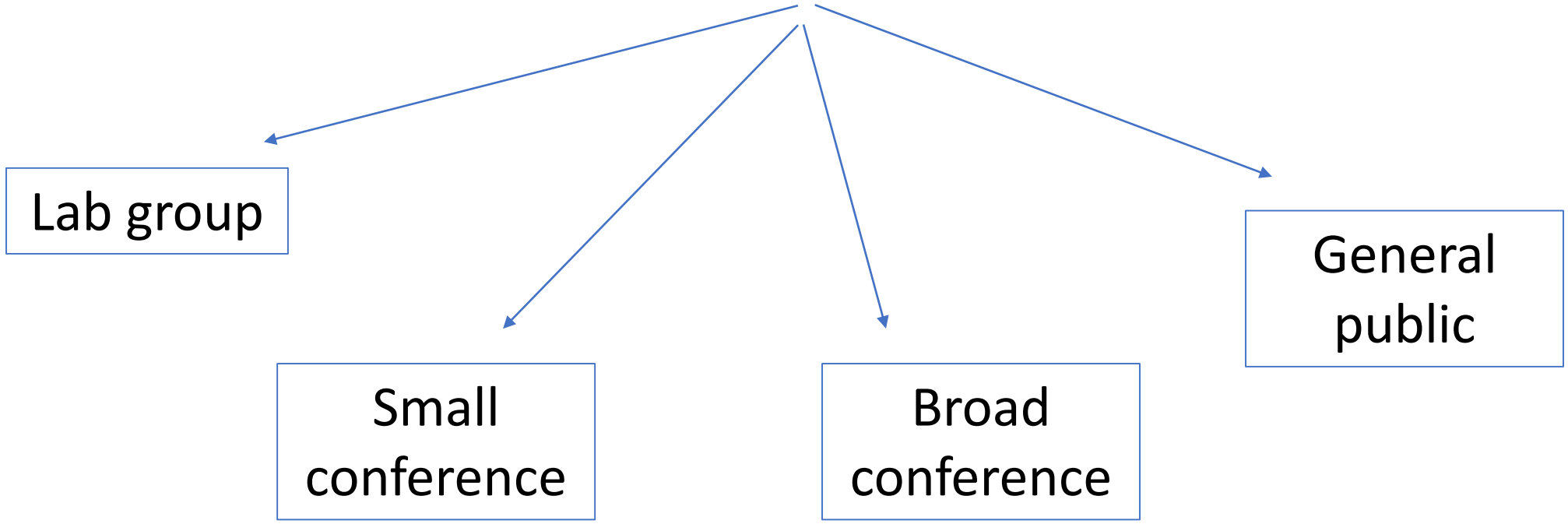
Comparison



Figures: Wilke 2019

1. What's your story?

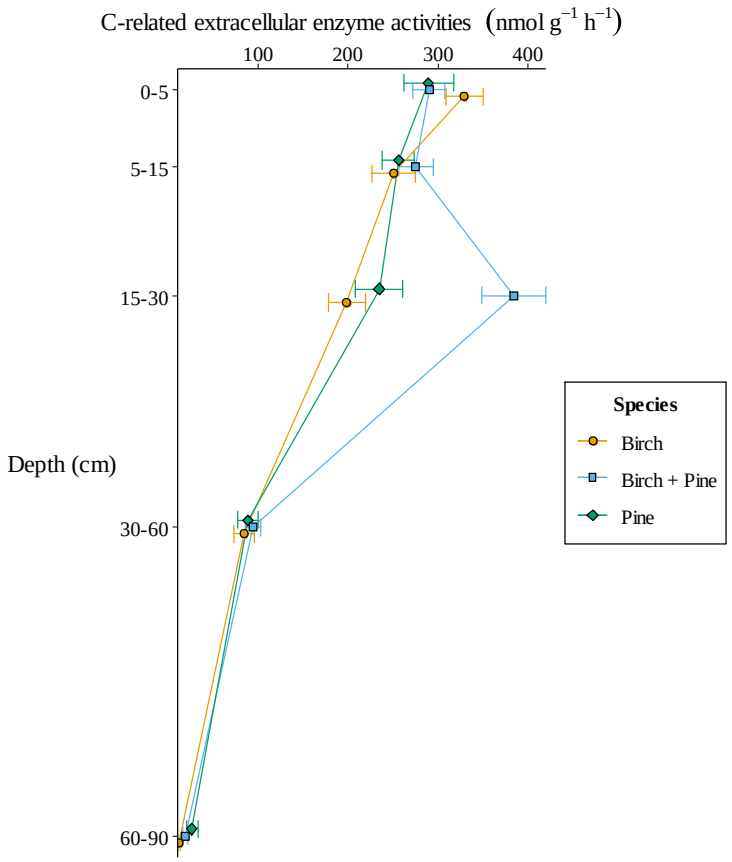
2. Know your audience



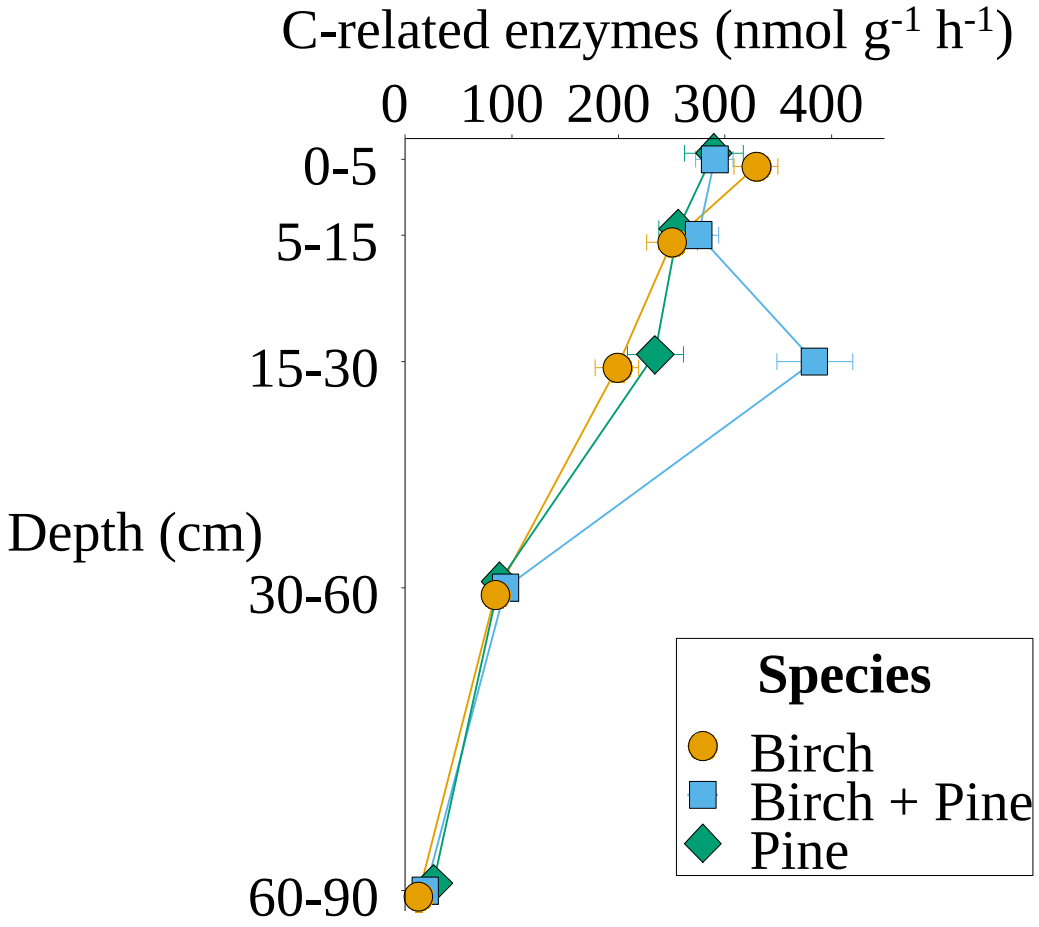
1. What's your story?

2. Know your audience

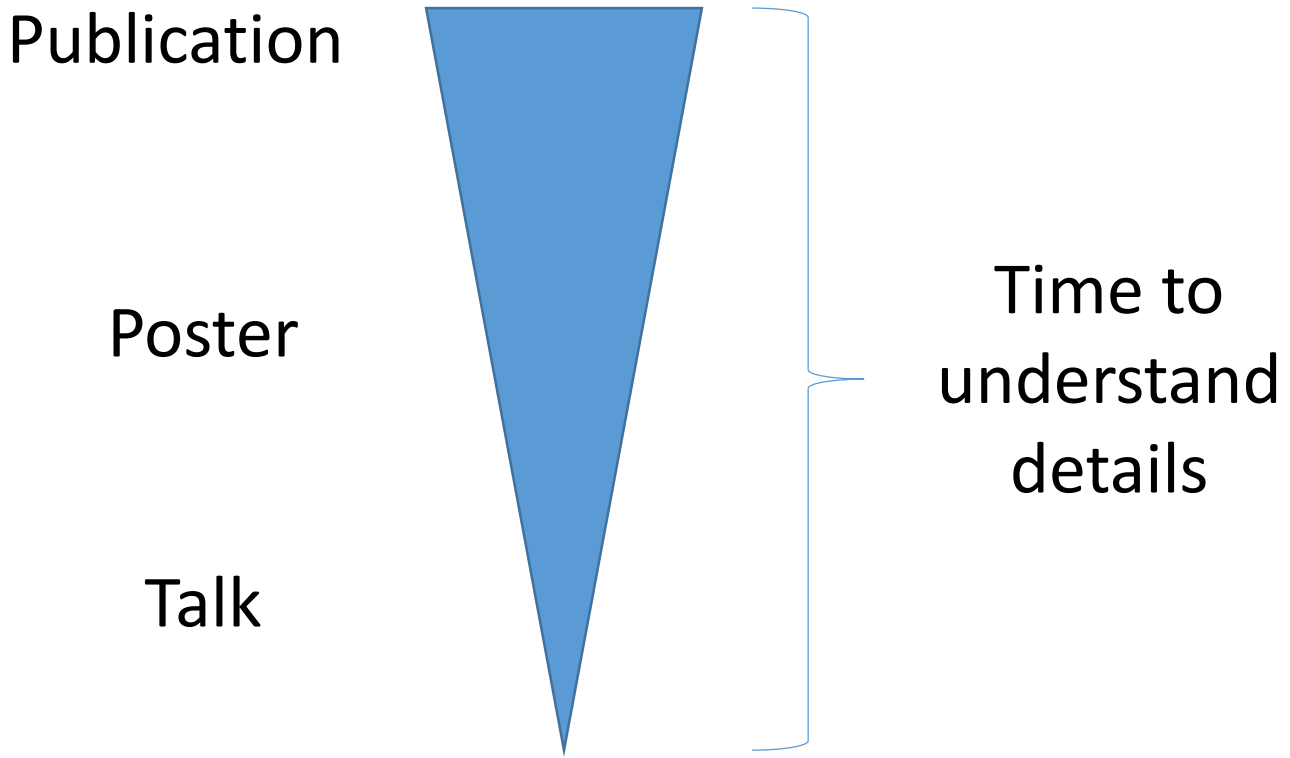
3. Adapt the figure



Maxwell et al. (In prep)



3. Adapt the figure (cont.)



Raab et al. 2019, EGU

3. Adapt the figure (cont.)

4. Include Captions

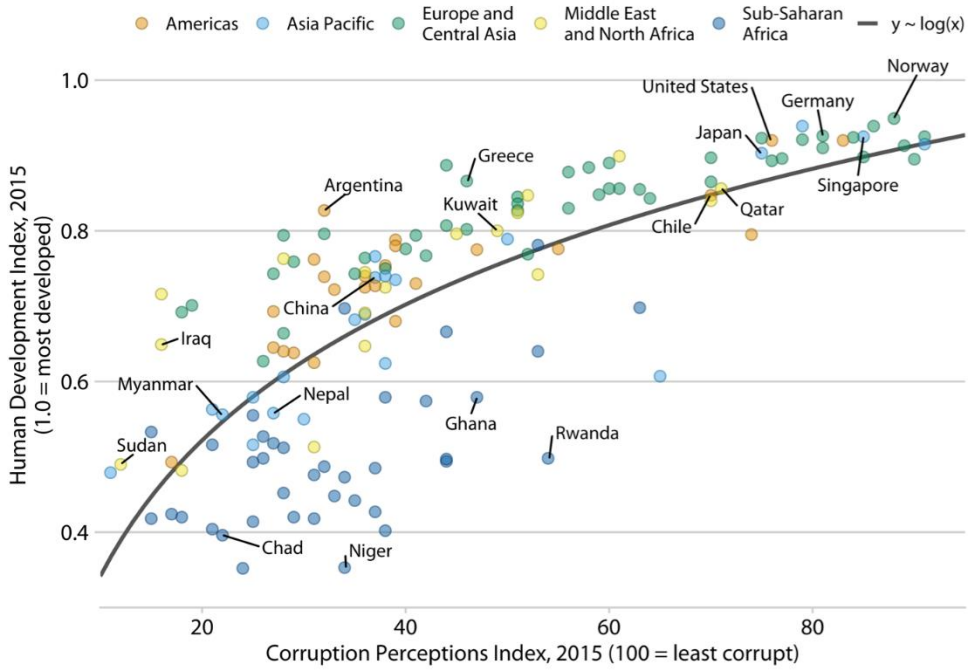
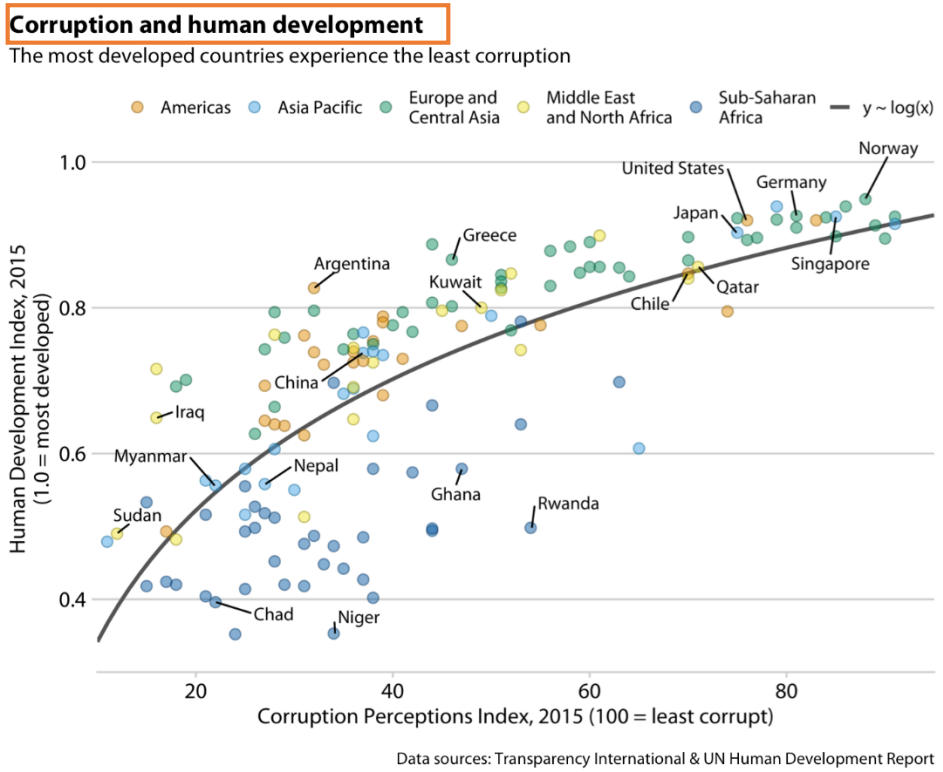


Figure 22.1: Corruption and human development: The most developed countries experience the least corruption. This figure was inspired by a posting in The Economist online (2011). Data sources: Transparency International & UN Human Development Report



Figures from Wilke (2019)

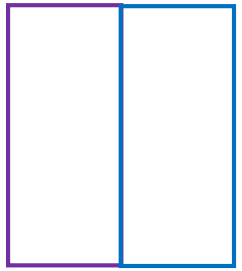
3. Adapt the figure (cont.)

4. Include Captions

5. Choose your Graph maker

Creating / editing figures manually

Excel



R

Image editor
(i.e. Inkscape)

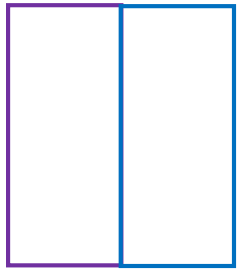
3. Adapt the figure (cont.)

4. Include Captions

5. Choose your Graph maker

Creating / editing figures manually

Excel



R

Image editor
(i.e. Inkscape)

1. Irreproducible
2. Unlikely to re-do figure if asked to be modified
3. You might forget what you did

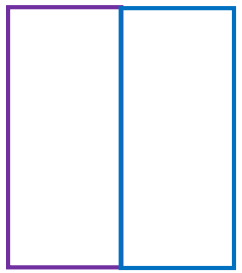
3. Adapt the figure (cont.)

4. Include Captions

5. Choose your Graph maker

Creating / editing figures manually

Excel



R

Image editor (i.e. Inkscape)

1. Irreproducible
2. Unlikely to re-do figure if asked to be modified
3. You might forget what you did

Figure preparation pipeline

R Ex: ggplot2

Reproducible

Part 2: Using R

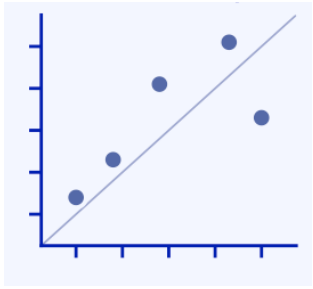
& some more rules

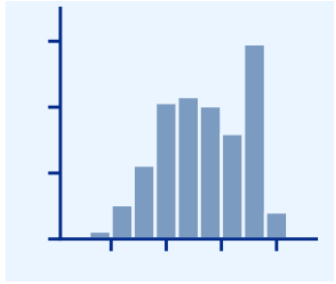
Part 2.1: Building your plot (Orange tree data)

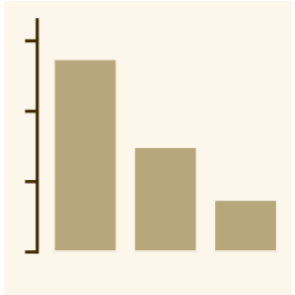
```
tab<- Orange #import data from 'datasets' package
```

```
Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':   35 obs.
of  3 variables:
 $ Tree         : Ord.factor w/ 5 levels "3"<"1"<"5"<"2"<..: 2 2 2 2 2 2 2 4 4 4 ...
 $ age          : num  118 484 664 1004 1231 ...
 $ circumference: num  30 58 87 115 120 142 145 33 69 111 ...
```

How do I want to present my data?







Figures: Wilke 2019

Basic ggplot structure

```
tab<- Orange #import data from 'datasets' package

p<- ggplot(tab, aes(x=age, y=circumference))
p
```

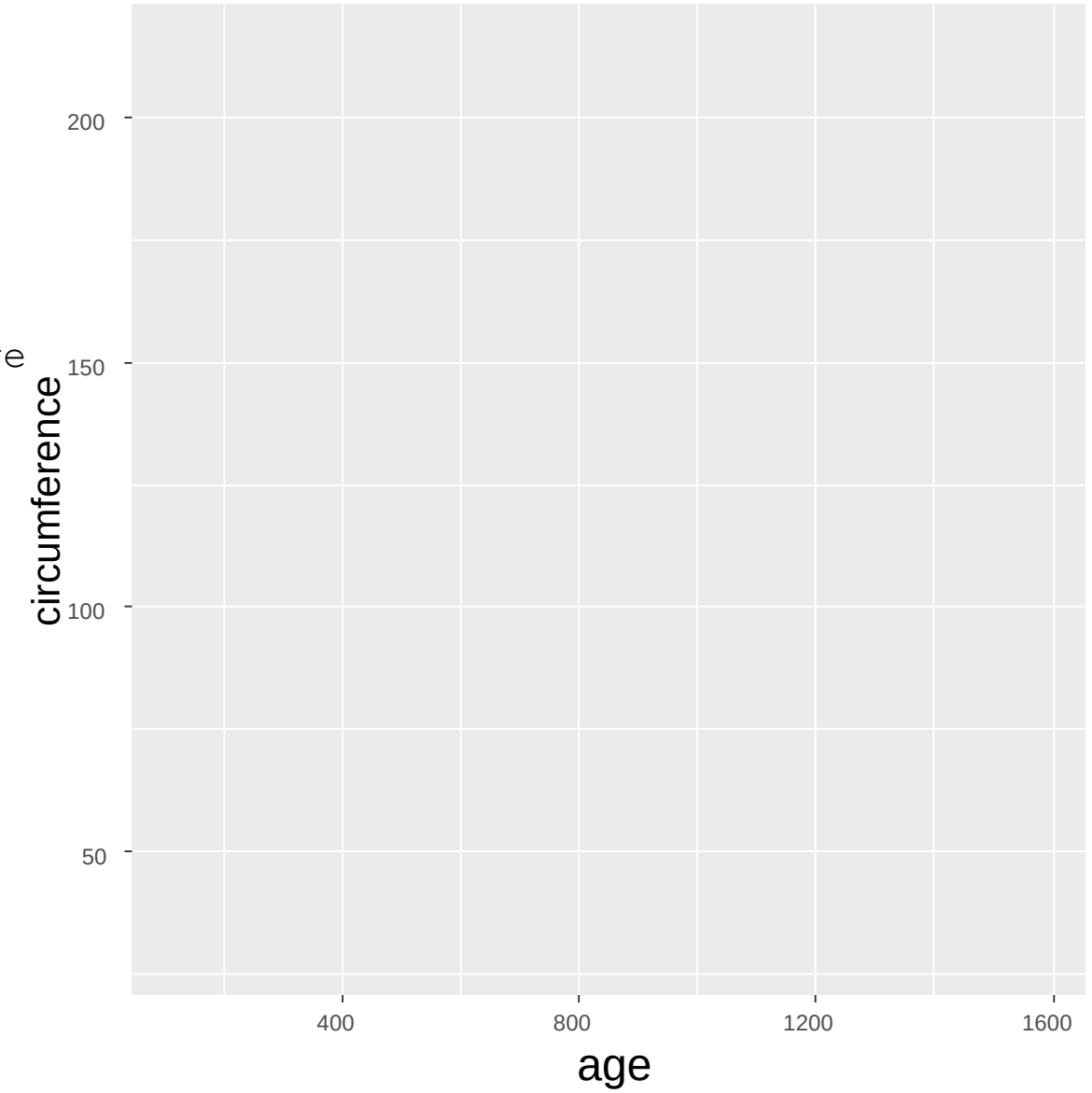
dataset

Coordinate system

Aesthetics = aes

- Quantifiable features

Place graph in an object named "p";
Display the object

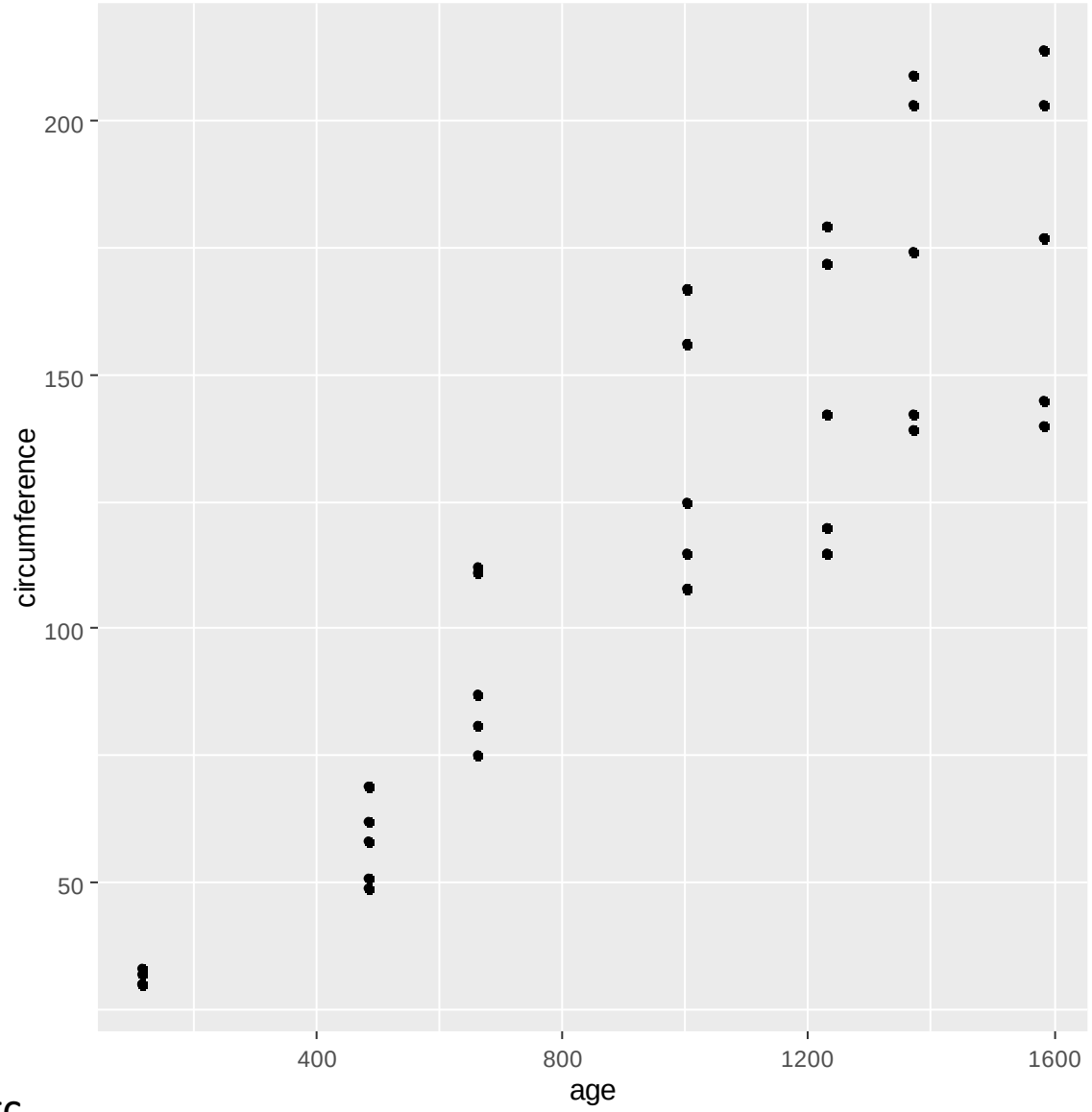


Adding geom_*

```
p<- ggplot(tab, aes(x=age, y=circumference))+
geom_point()
p
```

Geom_*
Visual mark that represents data

Adding layers



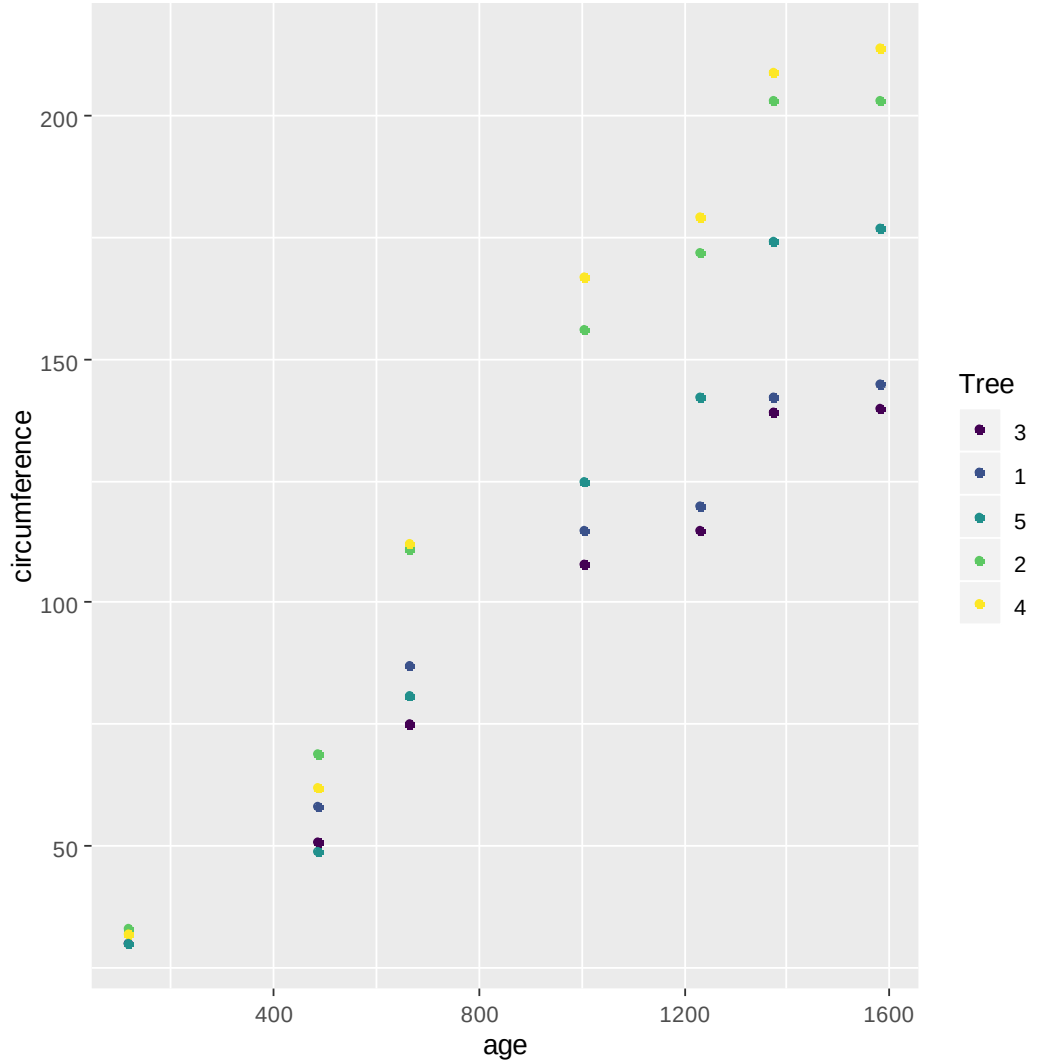
Geom_point, geom_line, geom_bar, geom_boxplot, geom_text, etc…

Grouping by a factor (i.e. Tree)

```
p<- ggplot(tab, aes(x=age, y=circumference))+
  geom_point(aes(group=Tree, color=Tree))
p
```

Geom_*
Visual mark that represents data

Adding layers



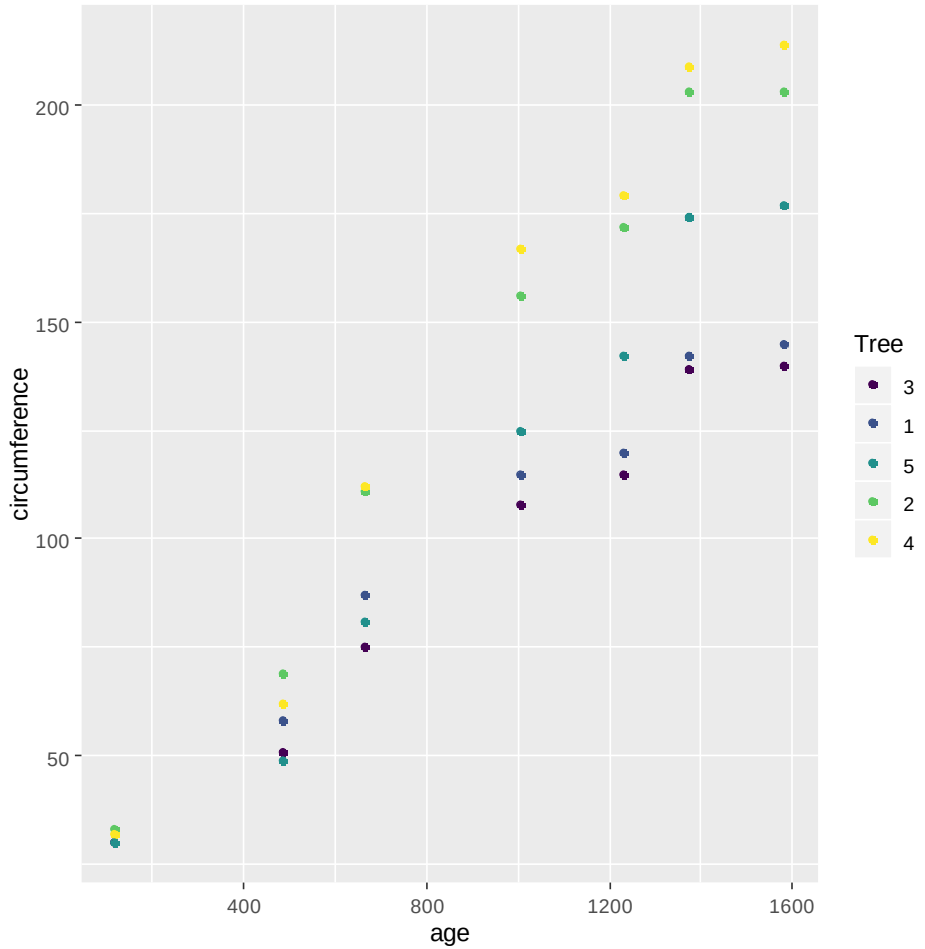
Why are the Trees in the order 3,1,5,2,4 ?

```
tab$Tree <-factor(tab$Tree, levels = c("1","2","3","4","5"))
```

R

```
tab$Tree <-factor(tab$Tree, levels = c("1","2","3","4",
```

```
p<- ggplot(tab, aes(x=age, y=circumference))+
  geom_point(aes(group=Tree, color=Tree))
p
```



Why are the Trees in the order 3,1,5,2,4 ?

R

```
tab$Tree <-factor(tab$Tree, levels = c("1","2","3","4","5"))
```

You can use specify the level order with the above function.

Tree graph reordered

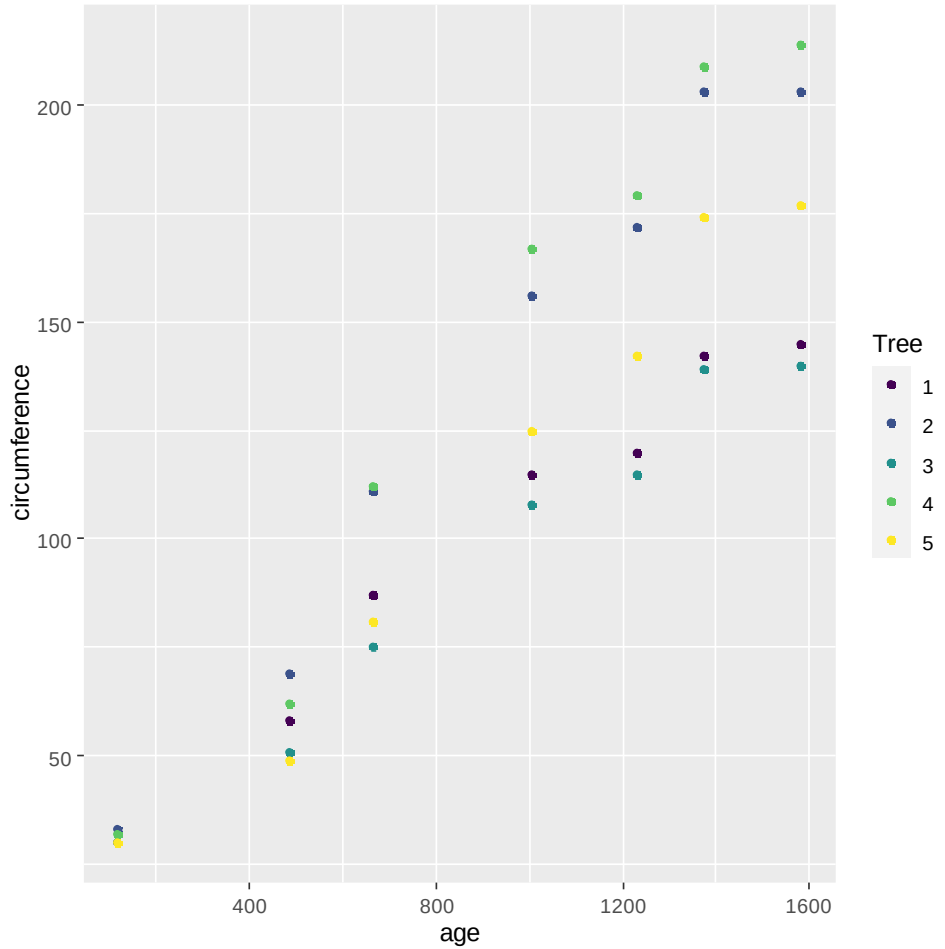
R

```
tab$Tree <-factor(tab$Tree, levels = c("1","2","3","4","5"))
```

You can use specify the level order with the above function.

Rerun your previous code

```
p<- ggplot(tab, aes(x=age, y=circumference))+
  geom_point(aes(group=Tree, color=Tree))
p
```



Tree graph reordered

R

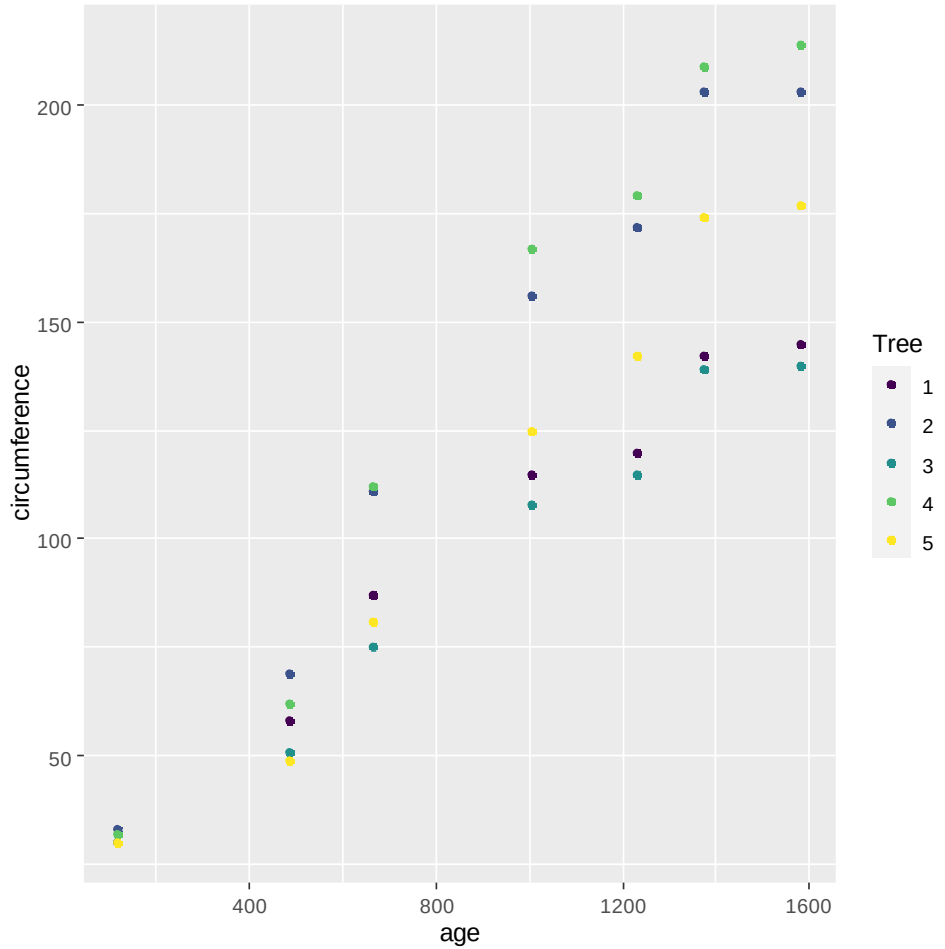
```
tab$Tree <-factor(tab$Tree, levels = c("1","2","3","4","5"))
```

You can use specify the level order with the above function.

Rerun your previous code

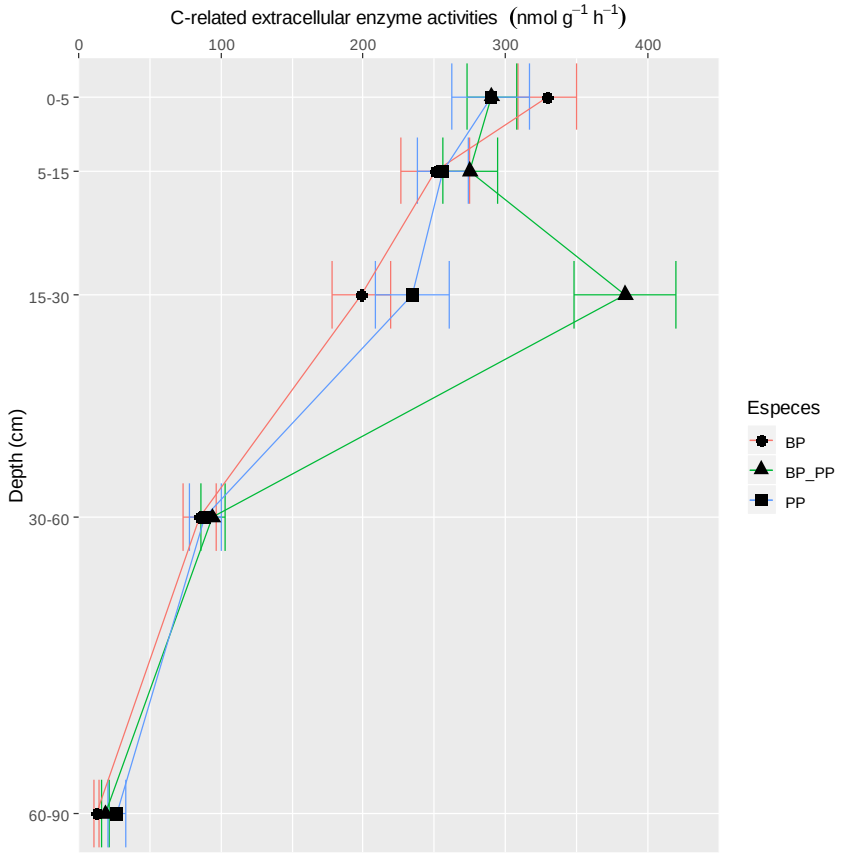
```
p<- ggplot(tab, aes(x=age, y=circumference))+
  geom_point(aes(group=Tree, color=Tree))
p
```

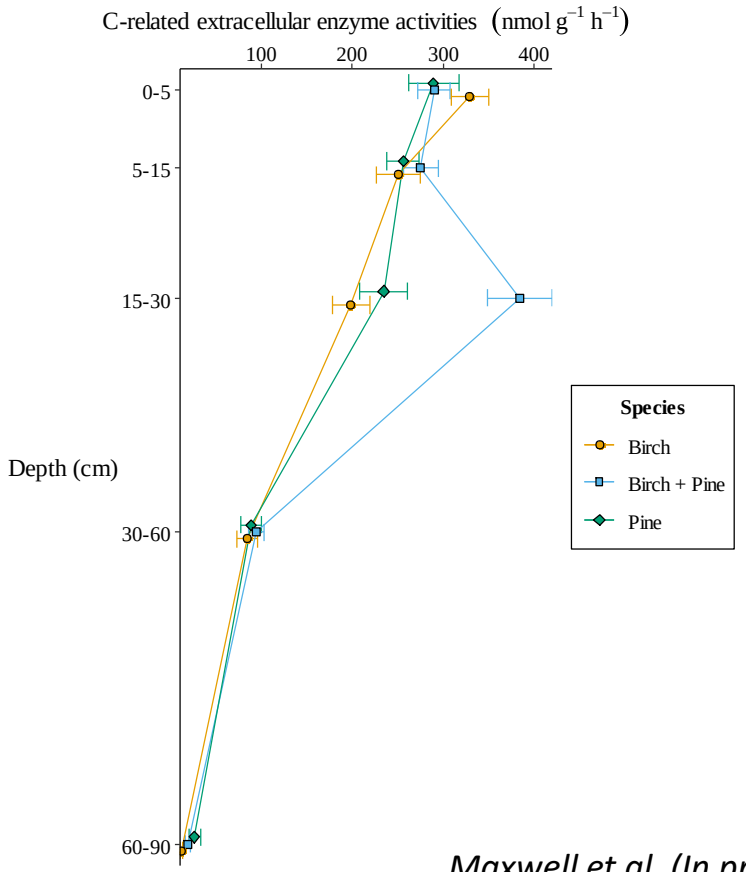
What can you notice about the order of the colors?



6. Don’t trust default settings

Color, shape, axes, legend, etc…





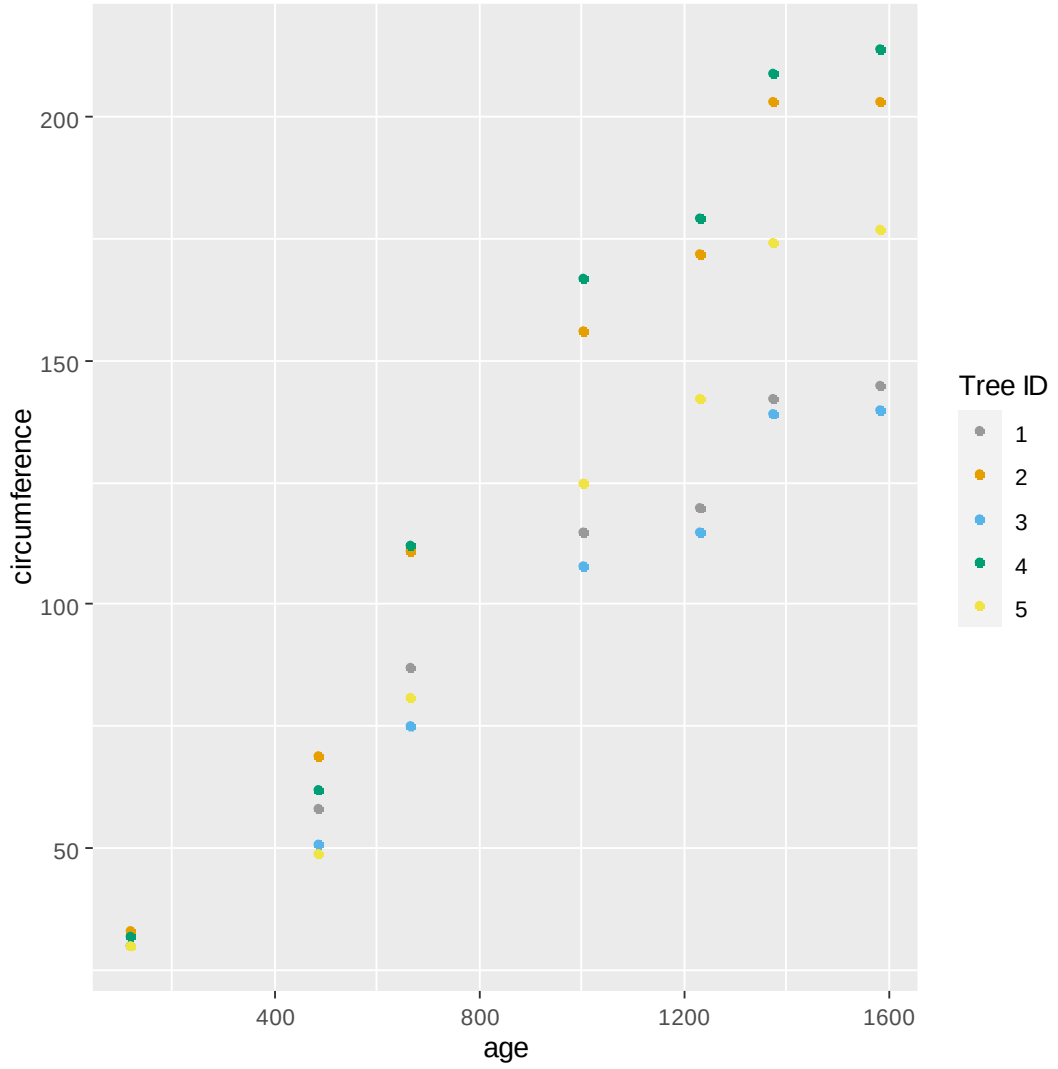
Maxwell et al. (In prep)

Changing your colors

R

```
creating your own palette
cbPalette <- c("#999999", "#E69F00", "#56B4E9",
"#009E73", "#F0E442", "#0072B2", "#D55E00",
"#CC79A7")
```

http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/#a-colorblind-friendly-palette



Changing your colors

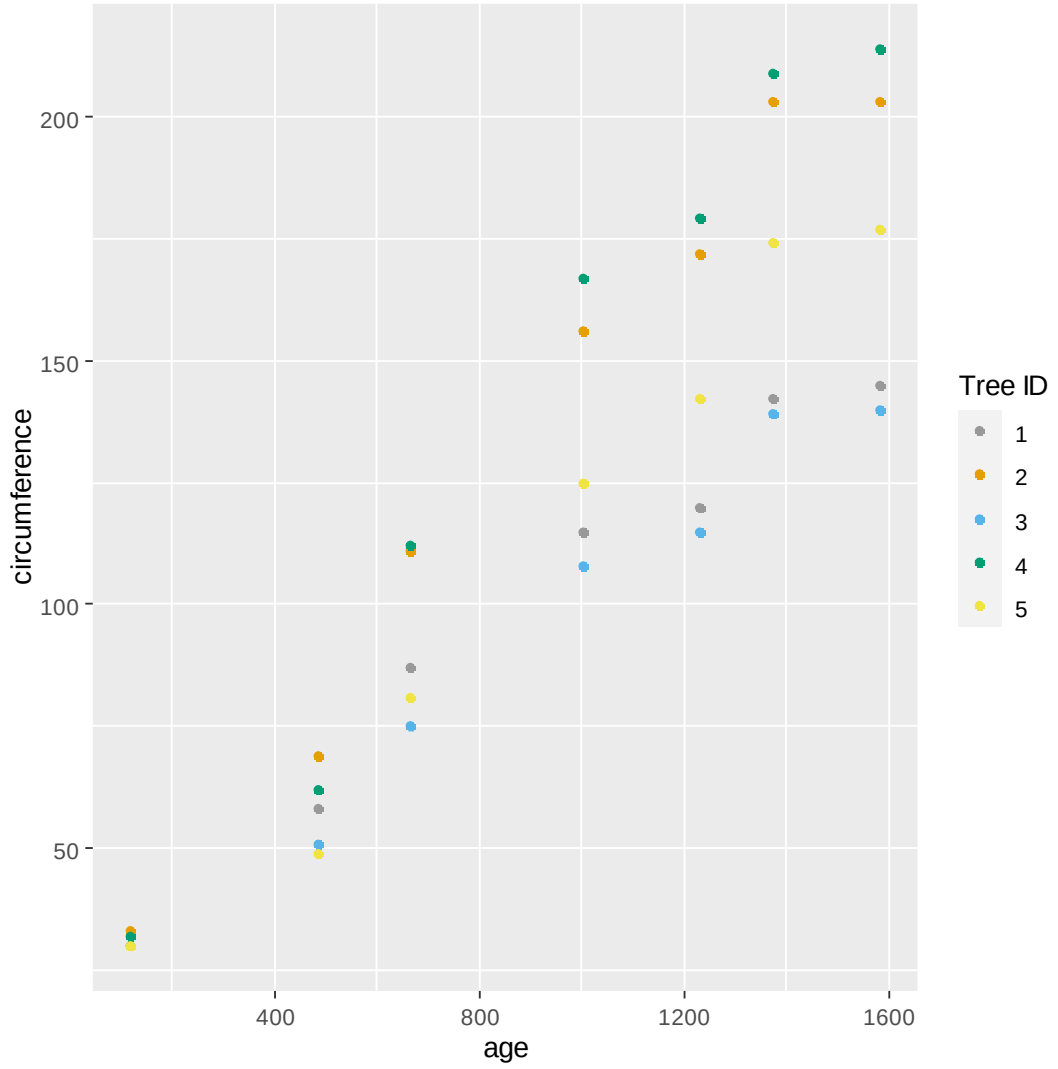
R

```
creating your own palette
cbPalette <- c("#999999", "#E69F00", "#56B4E9",
"#009E73", "#F0E442", "#0072B2", "#D55E00",
"#CC79A7")
```

http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/#a-colorblind-friendly-palette

```
Add to your previous graph
p<- ggplot(tab, aes(x=age, y=circumference))+
  geom_point(aes(group=Tree, color=Tree))+
  scale_color_manual(name = "Tree ID",
    values=cbPalette)
P
```

Add this to your code



Changing your colors

R

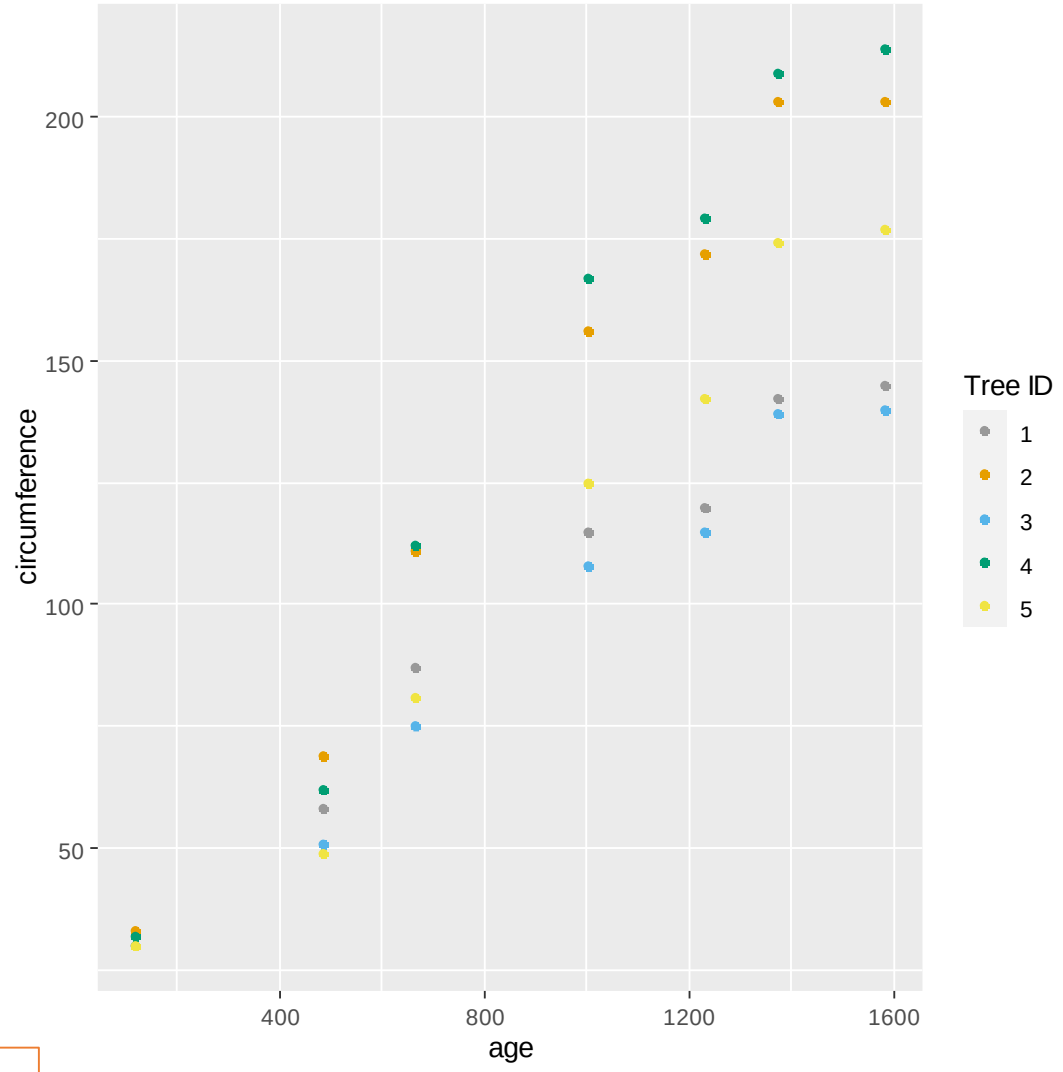
```
creating your own palette
cbPalette <- c("#999999", "#E69F00", "#56B4E9",
"#009E73", "#F0E442", "#0072B2", "#D55E00",
"#CC79A7")
```

http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/#a-colorblind-friendly-palette

```
Add to your previous graph
p<- ggplot(tab, aes(x=age, y=circumference))+
  geom_point(aes(group=Tree, color=Tree))+
  scale_color_manual(name = "Tree ID",
    values=cbPalette)
P
```

Note: since there are five levels of Trees, R will take the first five colors in your palette.

You can rename the legend title here

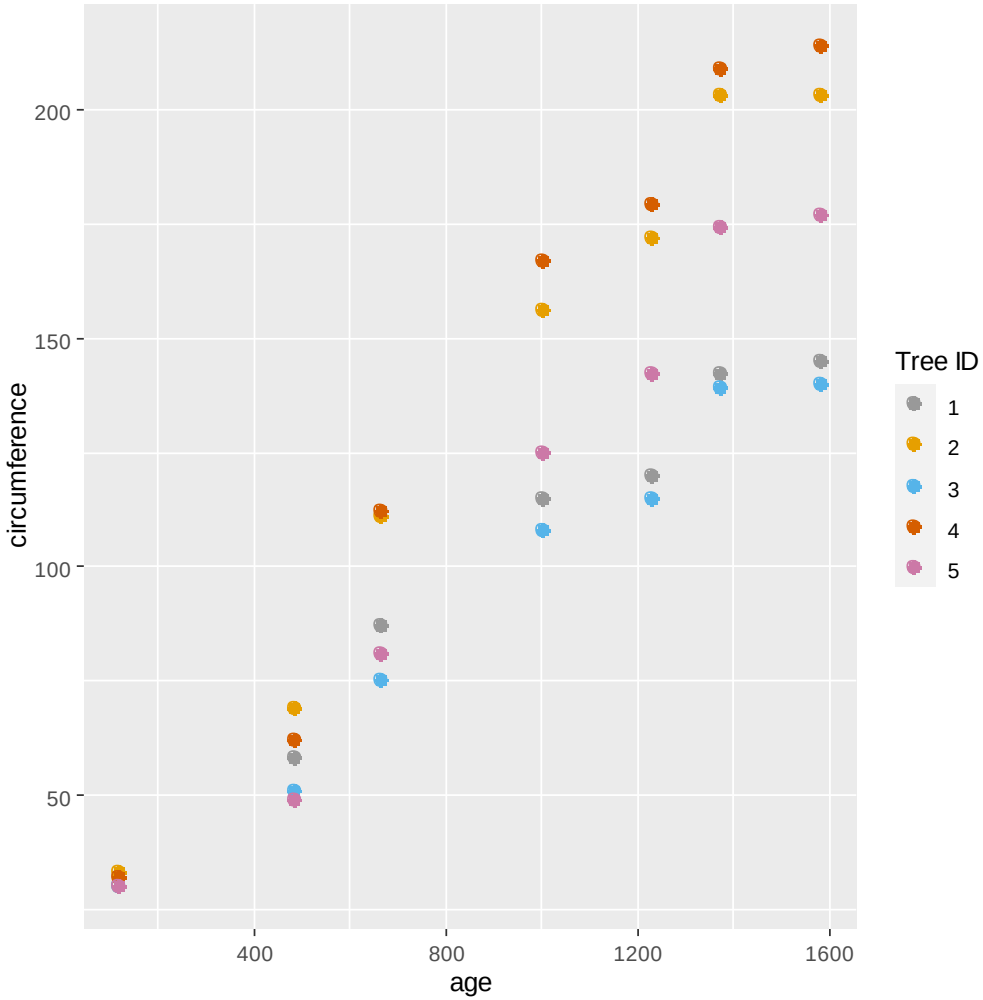


Changing your colors

```
p<- ggplot(tab, aes(x=age, y=circumference))+
  geom_point(aes(group=Tree, color=Tree), size =2.5)+
  scale_color_manual(name = "Tree",labels=
    c("1","2","3","4","5"), values=c("#999999",
    "#E69F00", "#56B4E9", "#D55E00", "#CC79A7"))
p
```

You can also add a list of colors with manual values

To increase your point size

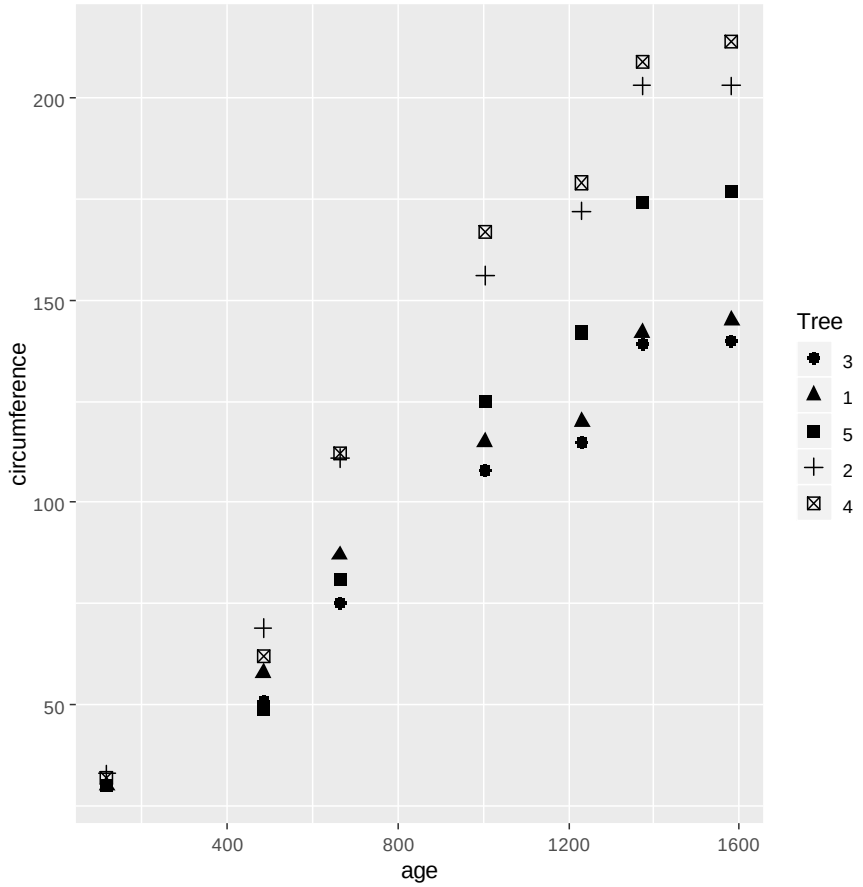


Resource to pick colors:

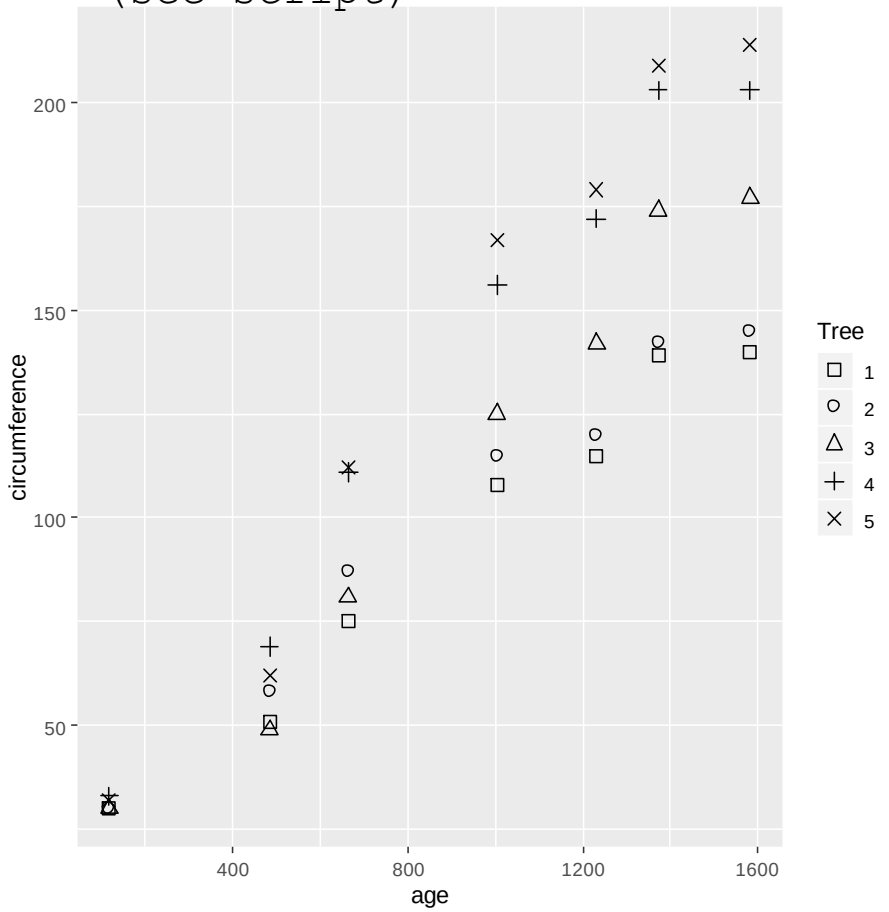
http://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3

Or, you can group by shape

```
p<- ggplot(tab, aes(x=age, y=circumference))+
  geom_point(aes(group=Tree, shape=Tree),size=2.5)
p
```



Manually choosing shapes (see script)



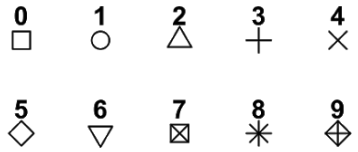
Grouping by color & shape - Caution

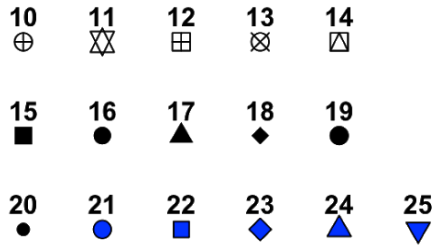
R

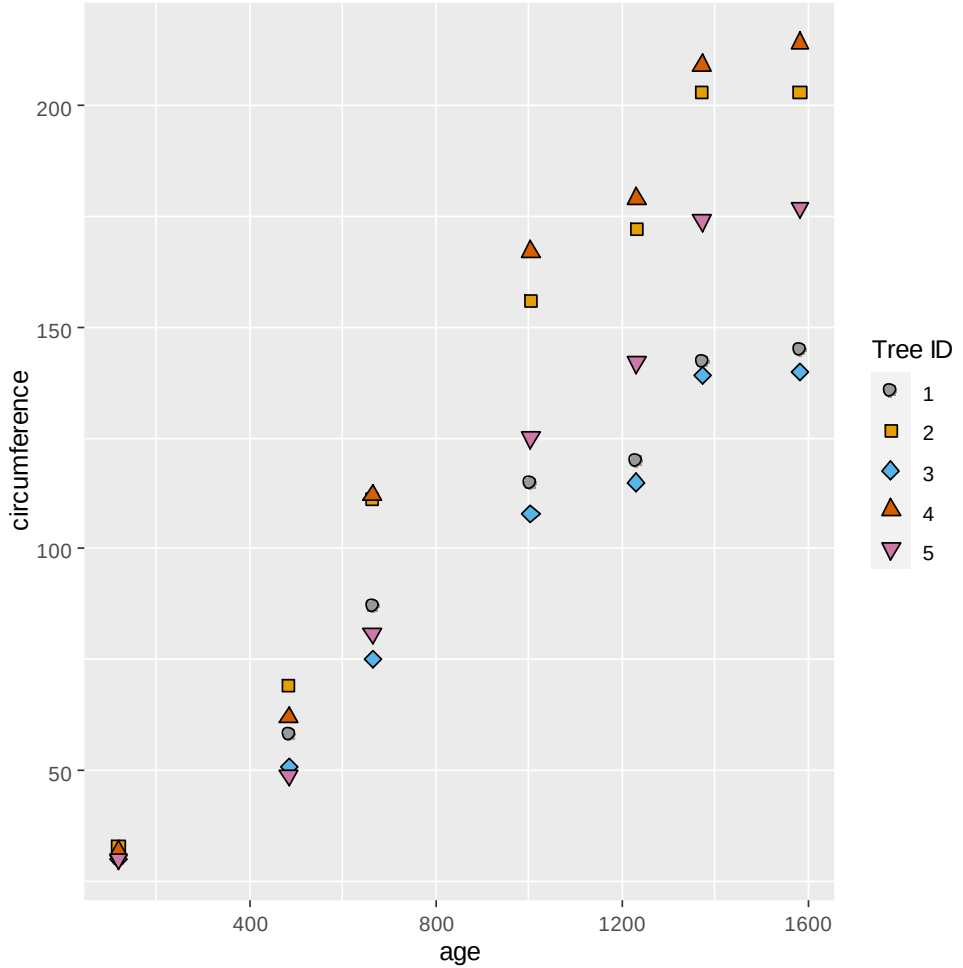
Our previous plot (shortened in presentation)

```
p<- p +
    scale_shape_manual(name = "Tree",labels=
    c("1","2","3","4","5"),values=c(21,22,23,24,25))+
    scale_fill_manual(name="Tree", labels=
    c("1","2","3","4","5"),
    values=c("#999999", "#E69F00", "#56B4E9", "#D55E00",
    "#CC79A7"))
p
```

These are different shape styles, which are identified by a number





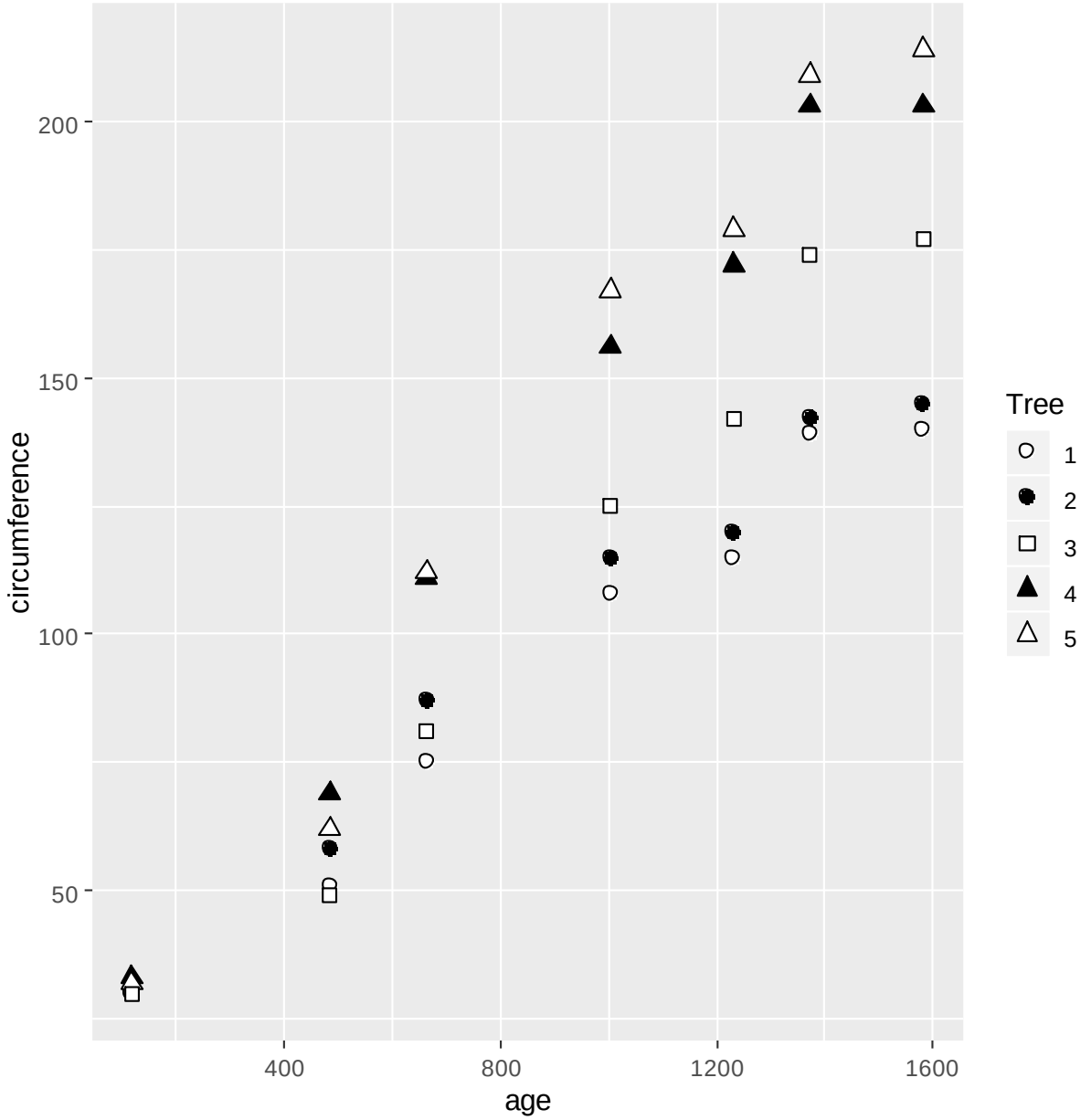


7. Use colors and symbols effectively

- Find the greyscale equivalent

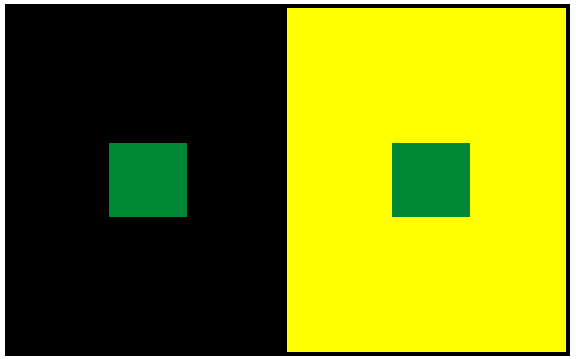
  https://toolstud.io/color/rgb.php

- Is black and white possible?

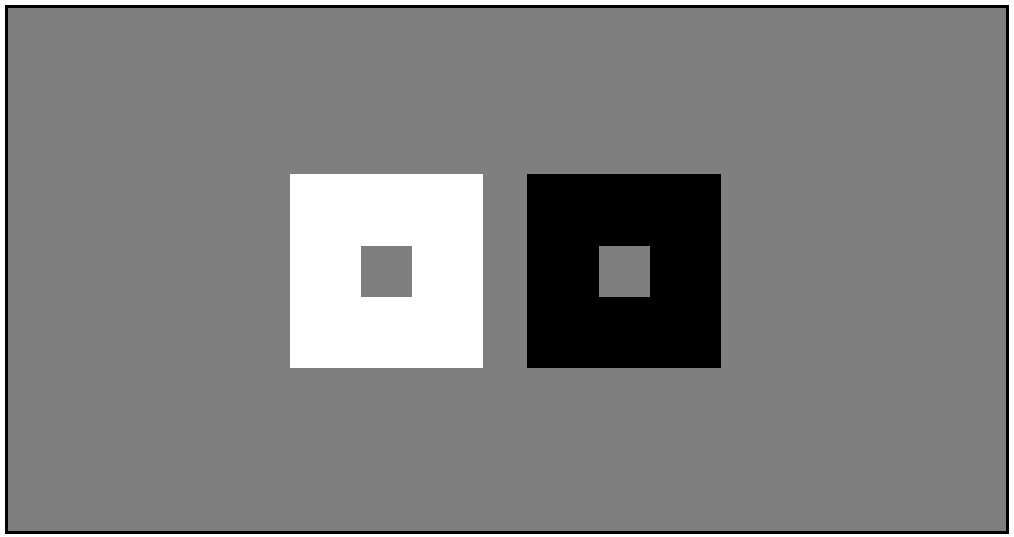


7. Use colors and symbols effectively

(Continued)



- Colors can change with background



- Think about overlap of points

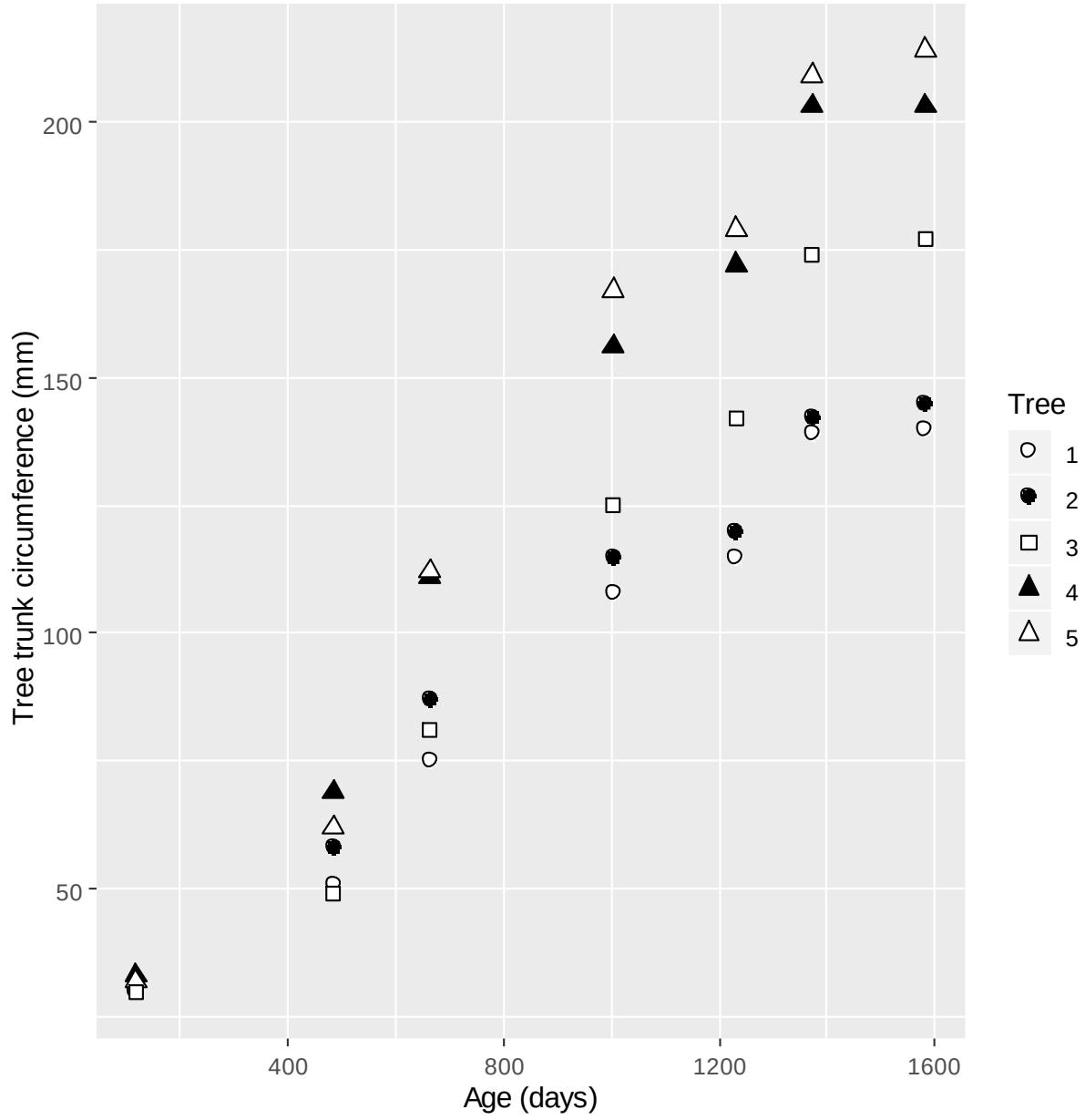
Images from: https://www.extremetech.com/extreme/49034-colors-affect-colors

Changing your axes

R

```
p<- p +
  labs(x="Age (days)", y="Tree trunk
    circumference (mm) ")
p
```

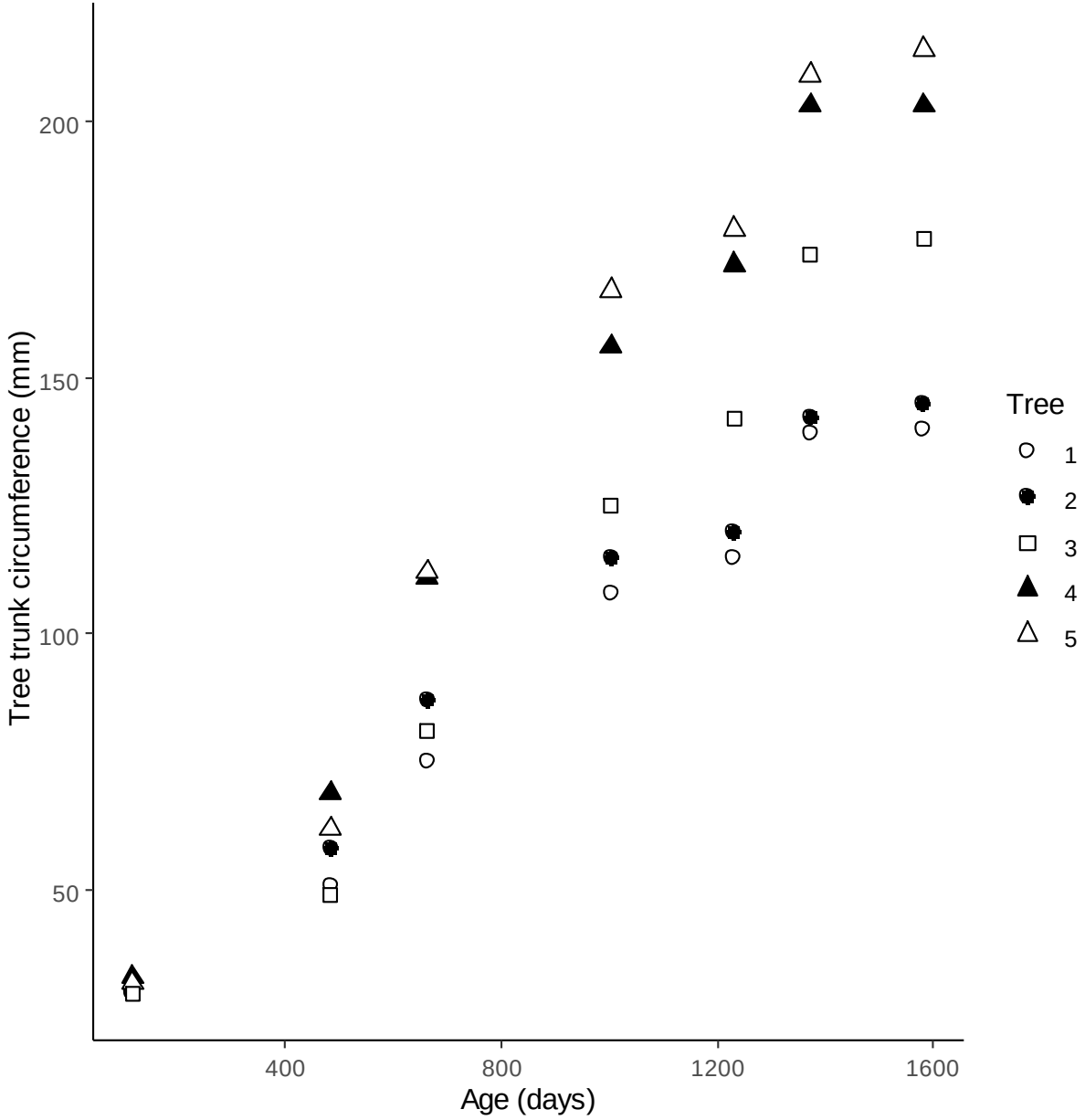
Give as much information to the reader as possible



Add a theme

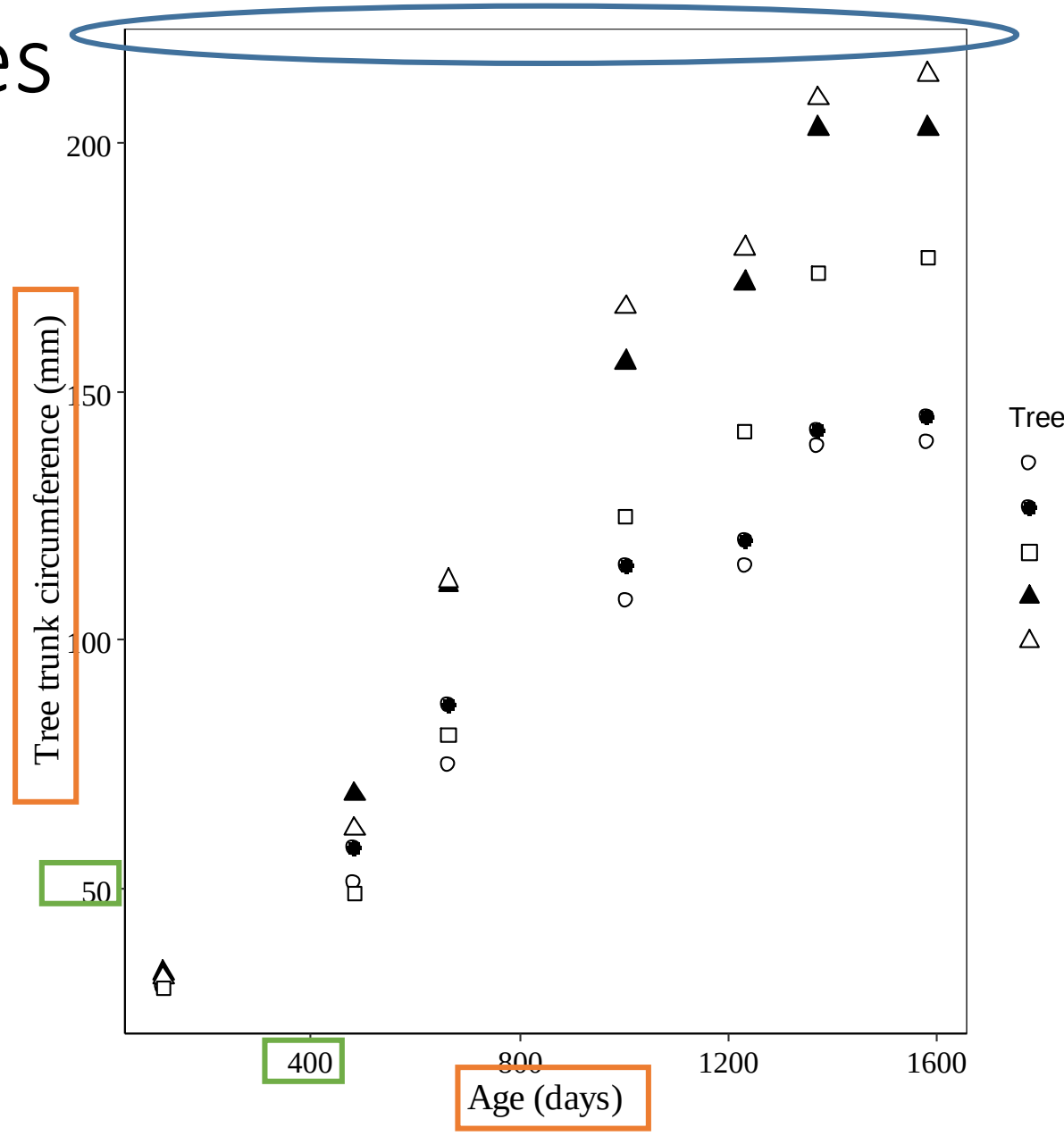
```
p <- p +
  theme_classic()
p
```

One of the built-in ggplot themes



Personalize your theme: axes

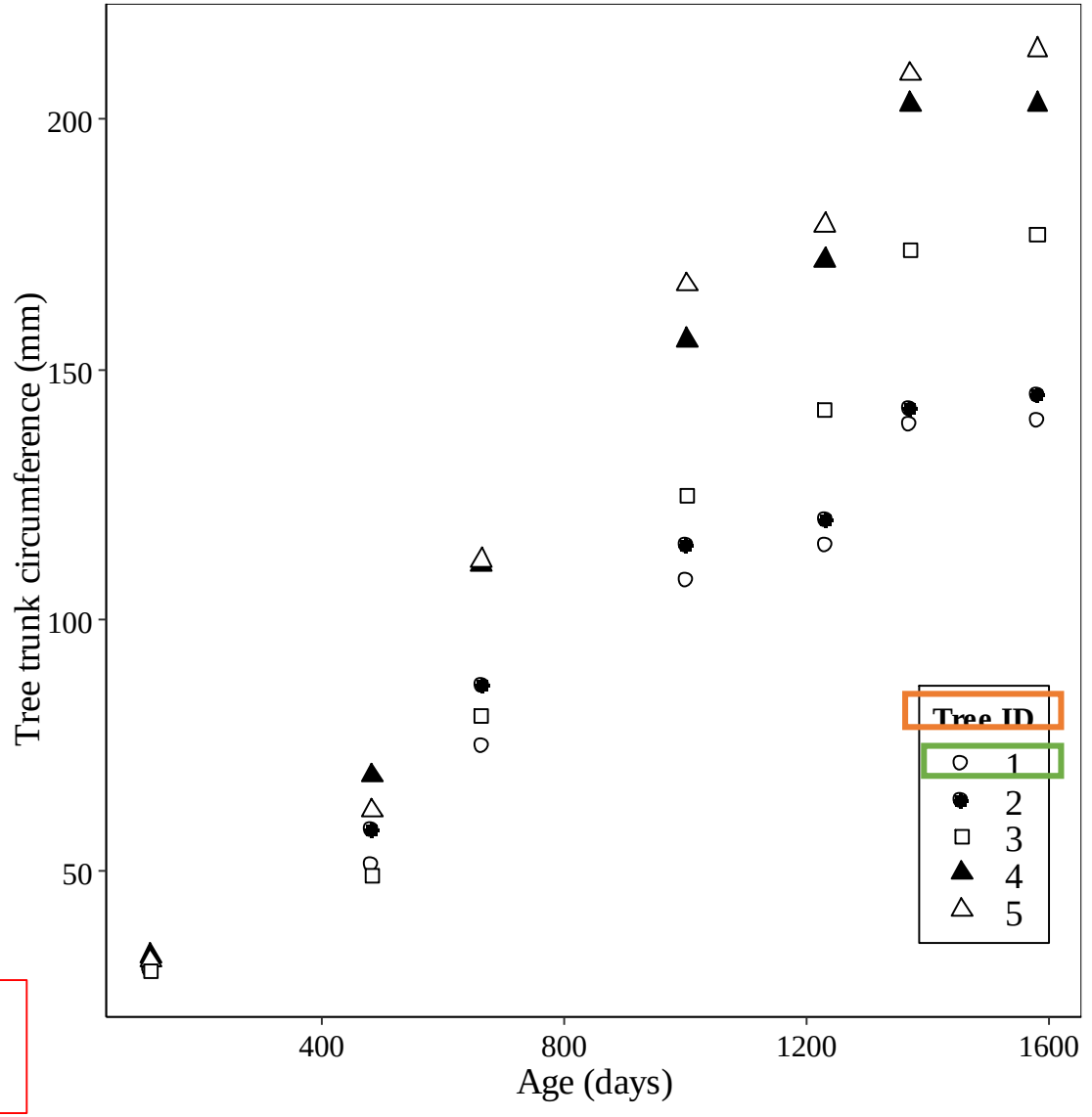
```
p <- p + theme_classic()+
    theme(panel.border = element_rect(linetype
    = "solid", fill = "NA"))+

    theme(
    axis.title.x=element_text(angle=0,
    size=14, family="serif",color = "black"),

    axis.text.x = element_text(angle =0,
    size=12,family="serif",color = "black"),

    axis.title.y=
    element_text(angle=90, size=14,
    hjust=.5,vjust=.5,family="serif",color
    =black"),

    axis.text.y =
    element_text(size=12,family="serif",color =
    "black"))
p
```



Personalize your theme: legend

```
p <- p+
    theme(
     legend.title=element_text(size=12,
face="bold", hjust=.5, family="serif"), #legend
title
        legend.text=element_text(size=14,
family="serif"), #legend text (the tree numbers)
        legend.background =
element_rect(linetype="solid", colour ="black"),
#box around the legend
        legend.key.size = unit(0.5,"cm"),
#legend height
        legend.key.width = unit(0.8, "cm"),
#legend width
        legend.position = c(0.9, 0.2))#position
of legend within the figure
pp
```

Caution: this may need to be changed when increasing font sizes

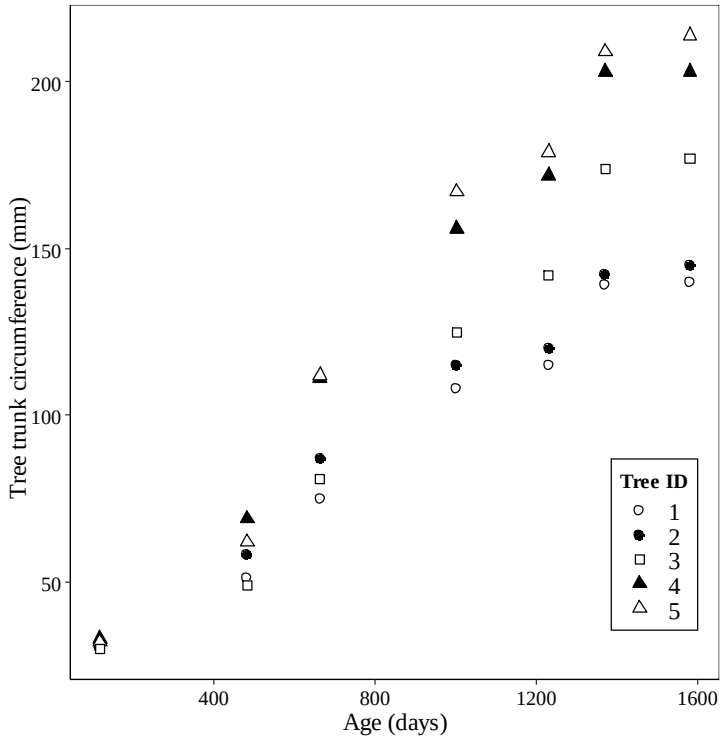


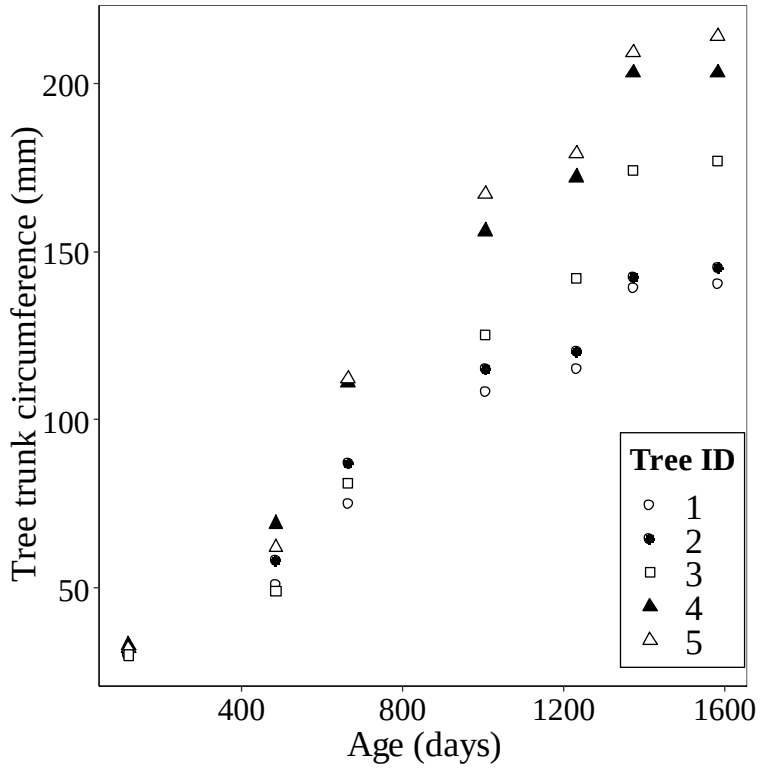
You can save your theme

Make two themes:

1. For figures to be printed
2. Figures for presentations


```
newtheme_print <- theme(...)

newtheme_pres <- theme(...)
```



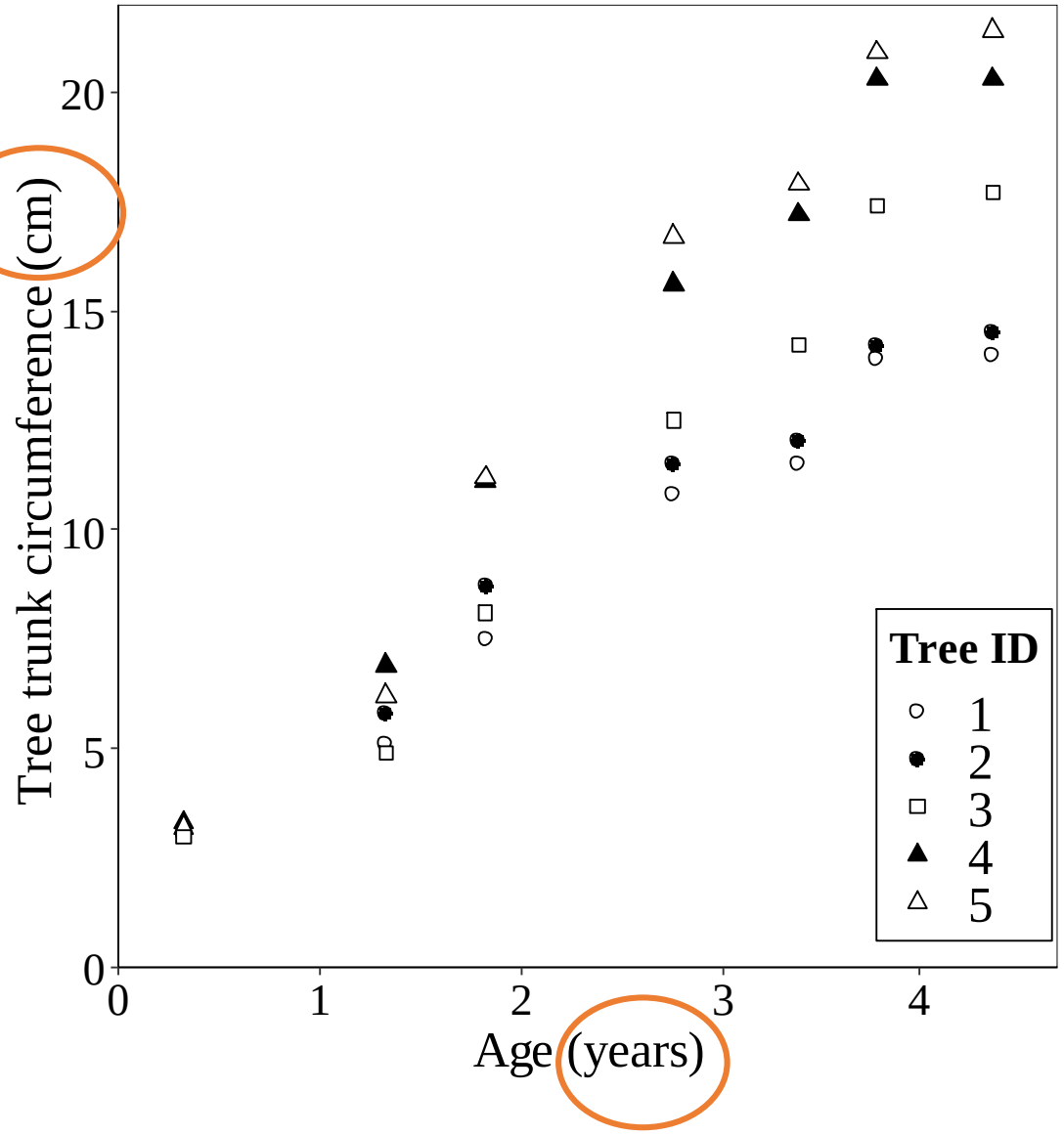


Extra: change your axis

R

```
p <- p  +
    scale_x_continuous(name="Age (years)",
    limits=c(0,1700), expand=c(0,0),
    breaks=c(0,365,730,1095,1450),
    labels=c("0","1", "2", "3","4"))+

    scale_y_continuous(name="Tree trunk
    circumference (cm)",limits=c(0,220),
    expand=c(0,0),breaks=c(0,50,100,150,200),
    labels=c("0","5", "10", "15","20"))
p
```

Manualy write where the axis ticks are located and the text for each of them



Part 2.2 Building a 2nd graph (Temperature data)

Linking growth to annual temperature could be interesting?

Import new data set to create new graph

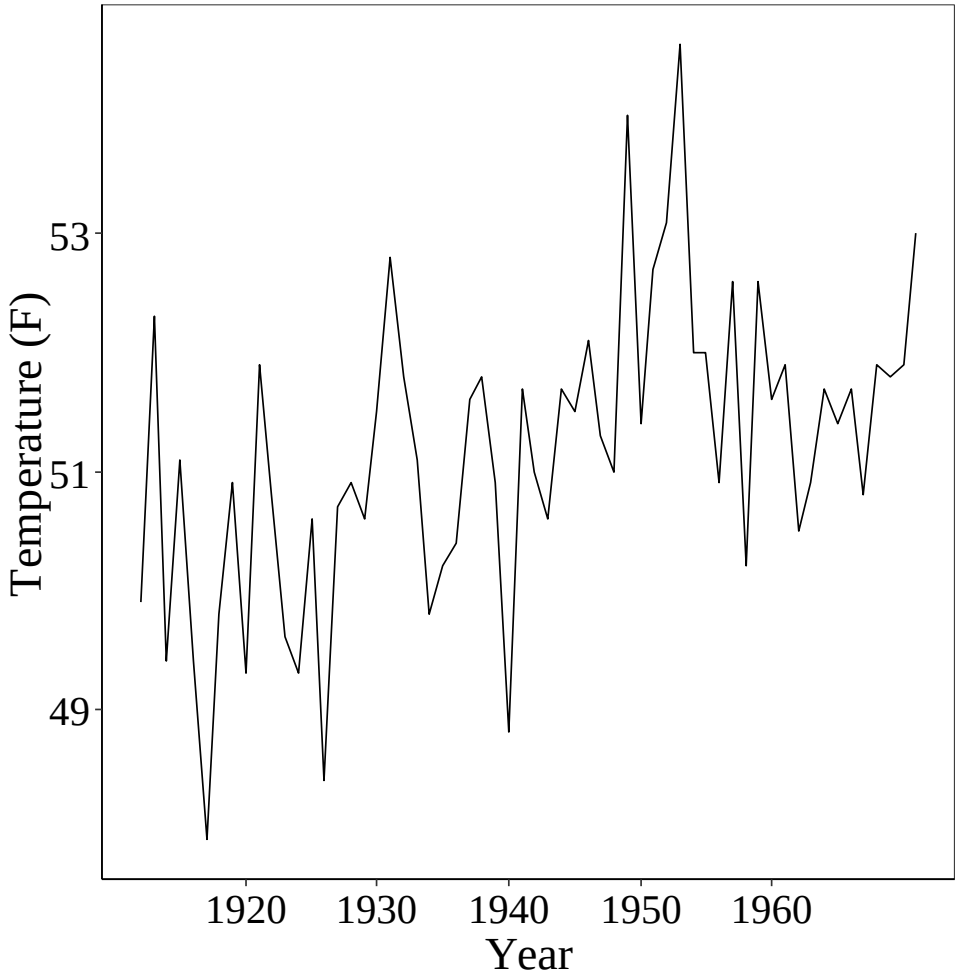
R

```
tab2<-as.data.frame(nhtemp)
str(tab2)
'data.frame': 60 obs. of  1 variable:
 $ x: Time-Series  from 1912 to 1971: 49.9 52.3 49.4 51.1 49.4...

date<-list(1912:1971)
tab_temp<-cbind(date,tab2)
```

Note: this data is unrelated

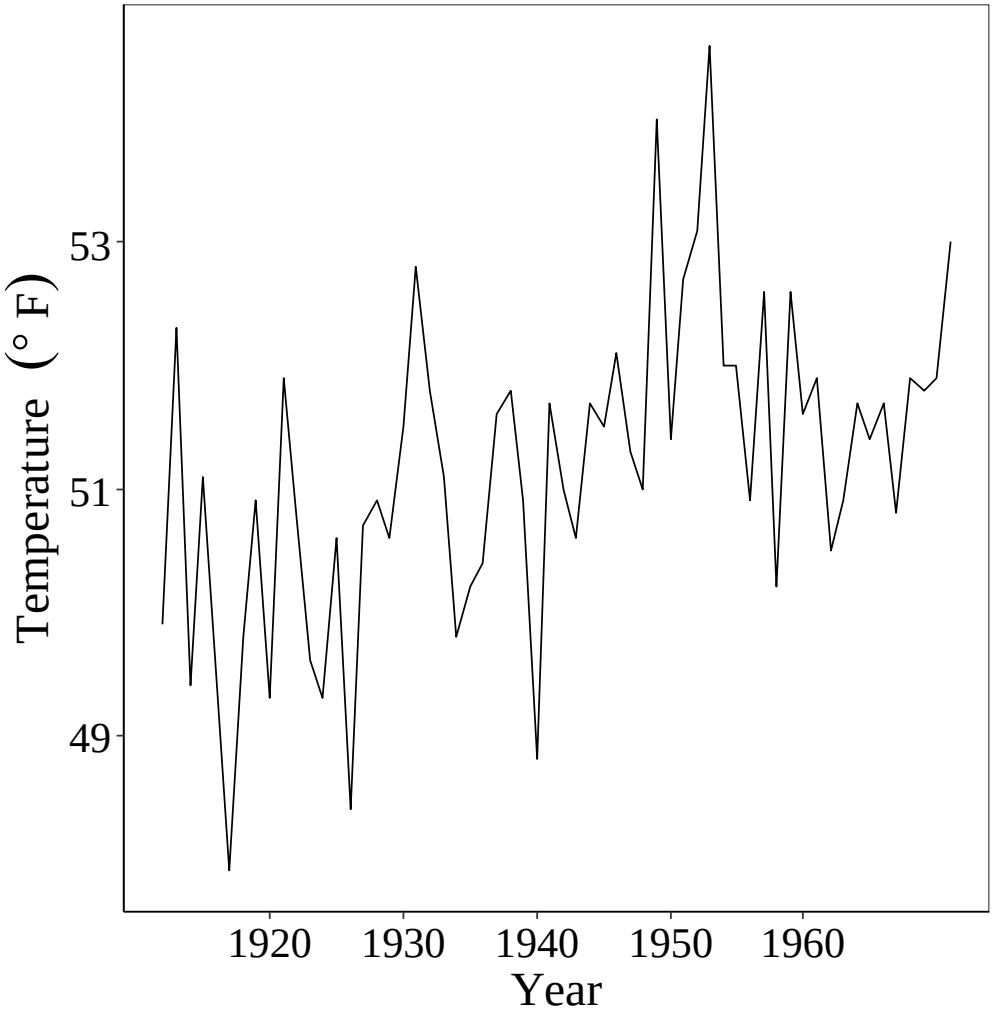
Basic plot using our theme

```
r<-ggplot(tab_temp, aes(x=Year, y=Temperature))+
  geom_line()+
  newtheme_print+
  scale_y_continuous(name="Temperature (F)")+
  scale_x_continuous(name="Year", limits =
    c(1912,1971),breaks=c(1920,1930,1940,1950,1960),
    labels=c("1920","1930","1940","1950","1960") )
r
```



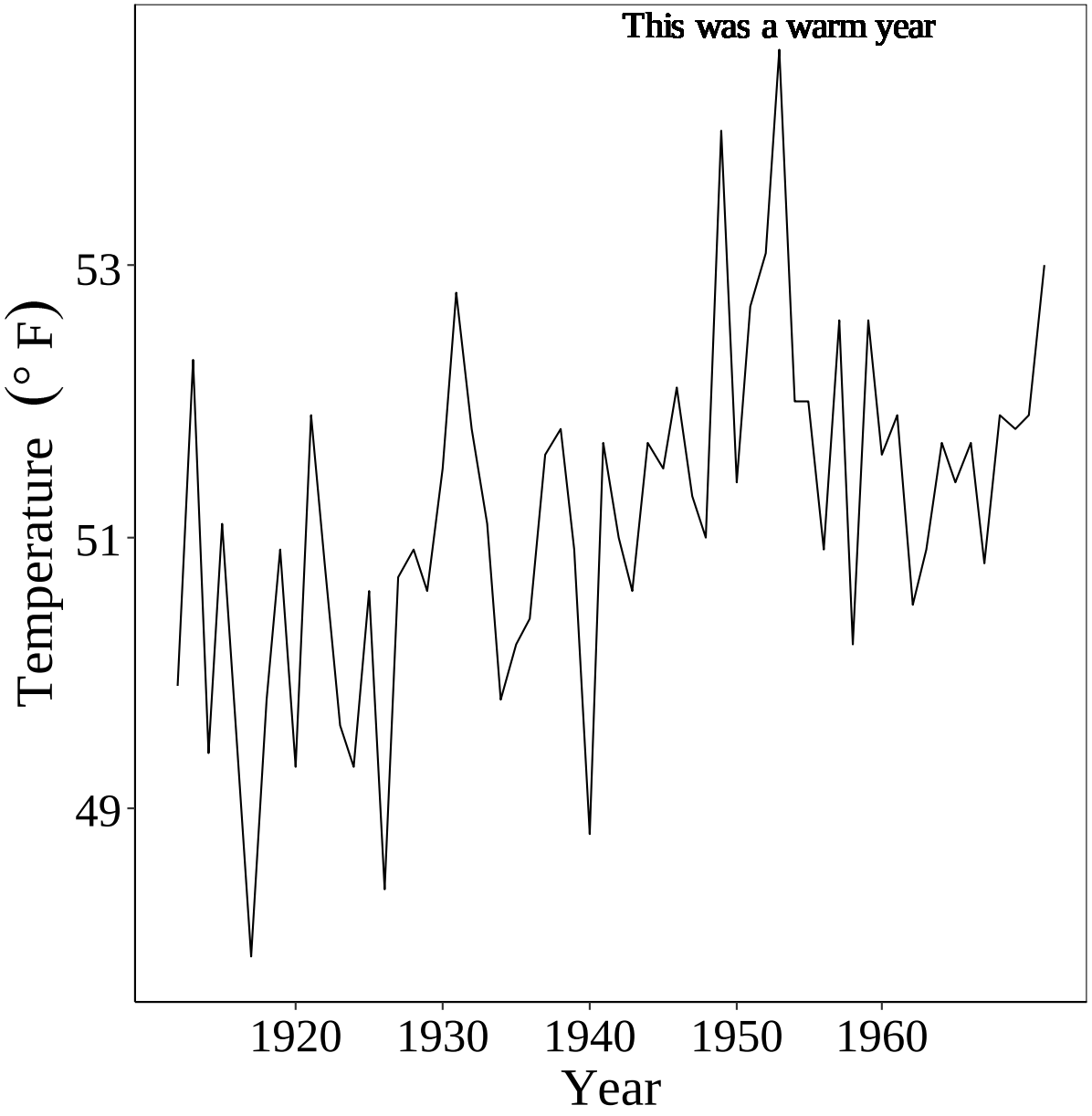
Adding special symbols

```
ylab <- expression("Temperature "~( degree~F))
#you can also add superscripts and greek letters

r<- r +
  scale_y_continuous(name=ylab)

r
```



Adding text

```
r <- r+
  geom_text(x=1953, y=54.8,
    label="This was a warm year",
    size=5, color="black",
    family="serif")
r
```

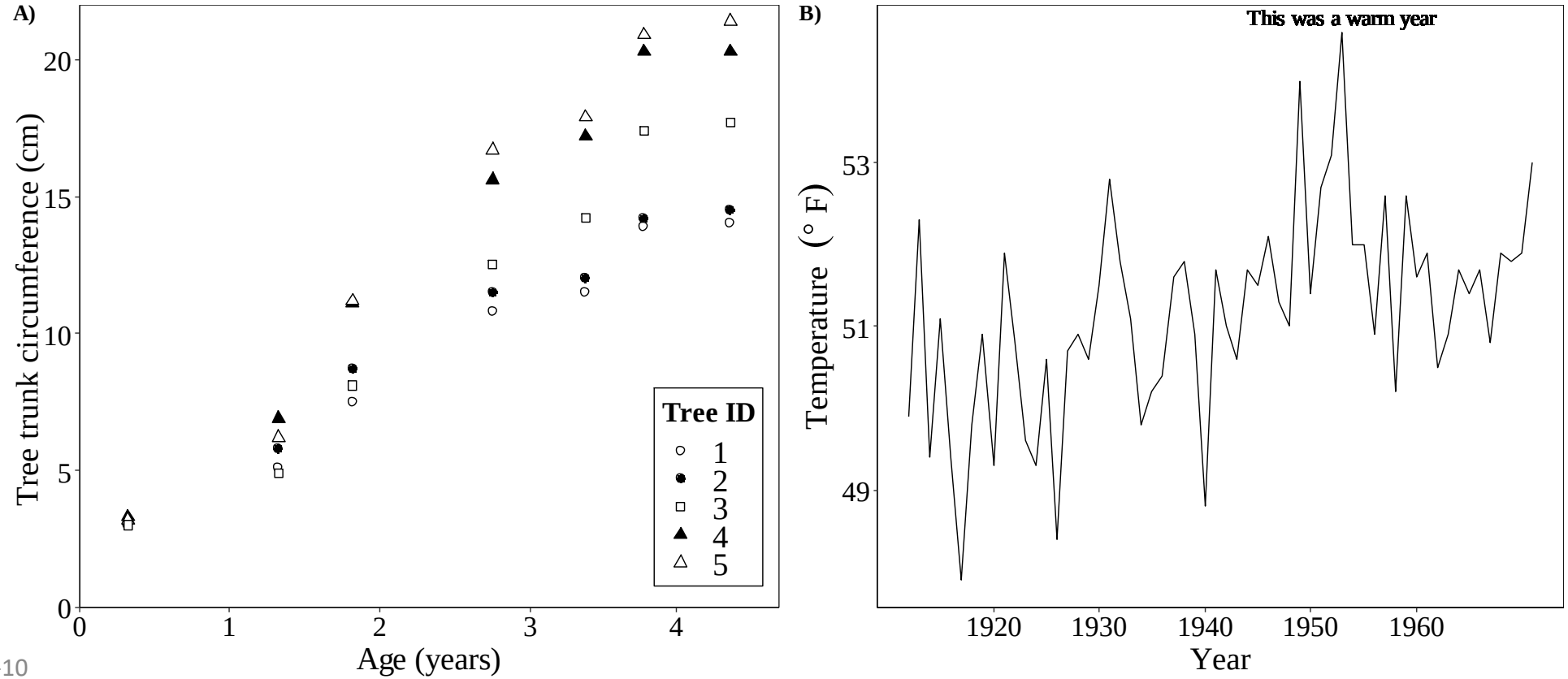


Note: this is good for adding p-values directly only your graph

Exporting both graphs next to one another

```
figure<- ggarrange(p, r, labels = c("A)", "B)"), font.label = list (size=14,family="serif"),
                   ncol = 2, nrow = 1)
```

Can remove y or x-axis titles easily: r + rremove("y.title")



Part 2.3: Plotting averages using ddply

R

```
cdata <- ddply
    (tab, c("age"), summarise,
     N    = sum(!is.na(circumference)),
     mean = mean(circumference, na.rm=T),
     sd   = sd(circumference, na.rm=T),
     se   = sd / sqrt(N)
)
cdata
```

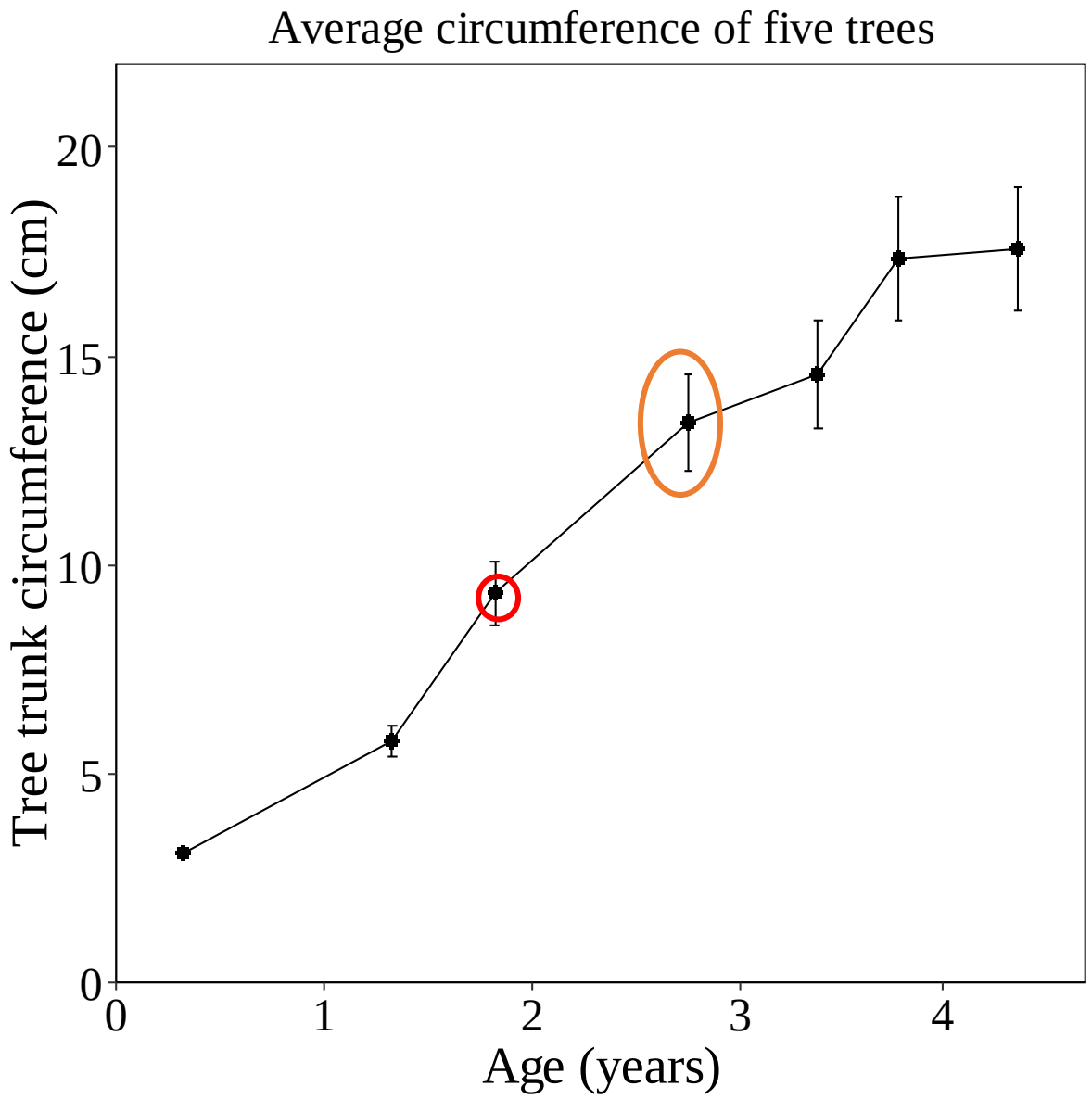
```
 age N  mean        sd         se
1  118 5  31.0  1.414214  0.6324555
2  484 5  57.8  8.167007  3.6523965
3  664 5  93.2 17.239490  7.7097341 ...
```

With the Orange tree data frame = tab

Plotting averages

R

```
p<- ggplot(cdata, aes(x=age, y=mean))+
   geom_line() +
   geom_point(size=2.5,shape=16)+
   geom_errorbar(aes(ymin=mean-se,
     ymax=mean+se), size= 0.3, width=15)+
   ...
p
```



Part 2.4: Fitting a model (i.e. a logistical model)

R

Note: This model can be found within the datasets package for this dataset

```
logistical_mod <- nls(mean ~ SSlogis(age, Asym, xmid, scal), data = cdata)
logistical_mod
```

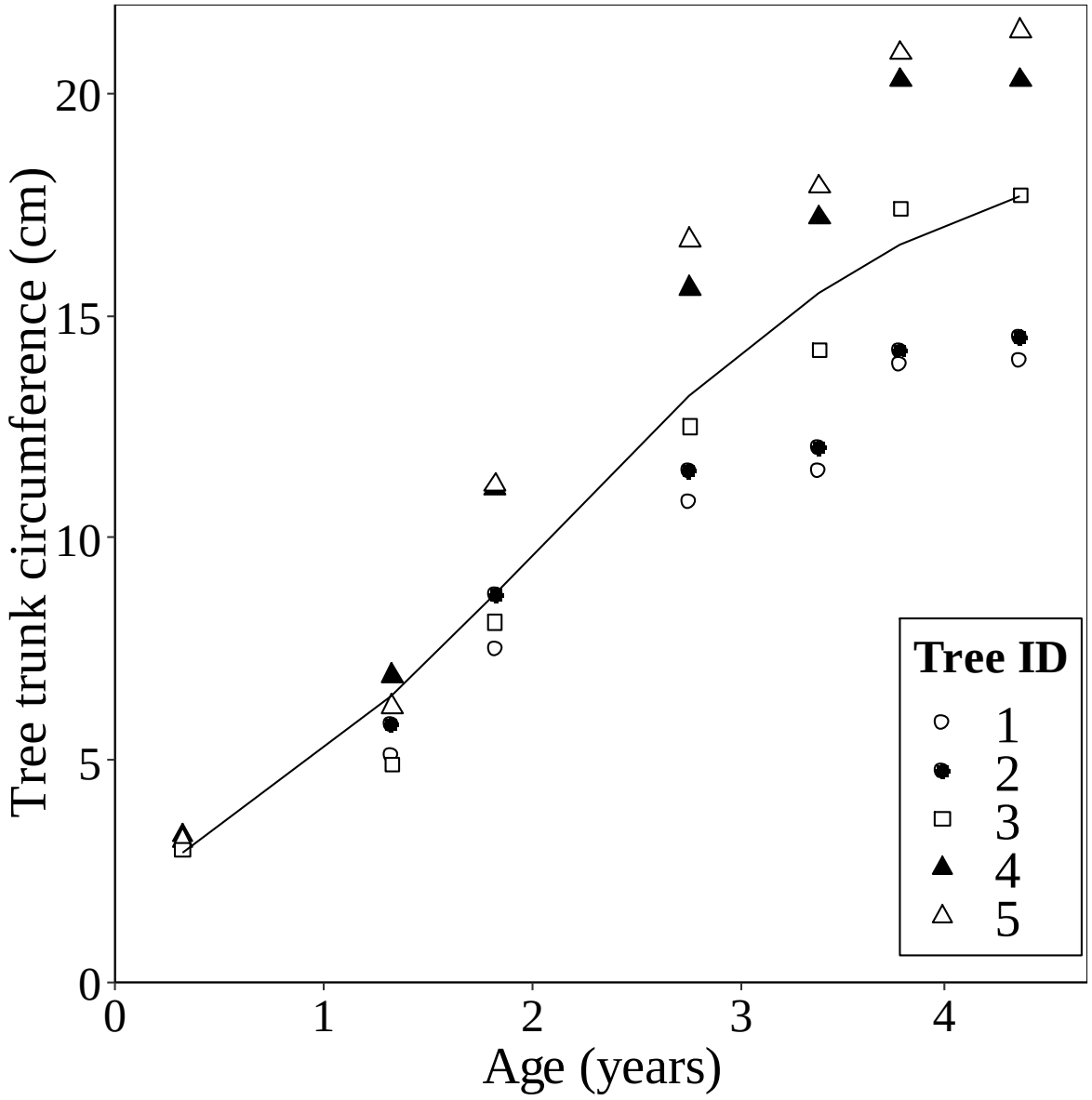
```
mod.predict <- cbind(data=cdata, predict(logistical_mod, interval = 'confidence'))

colnames(mod.predict) <- c("Age","N","mean","sd","se","Predicted_values")

mod.predict
```

Fitting a model

R

```
p<- ggplot(tab, aes(x=age, y=circumference))+
    geom_point(aes(group=Tree,
    shape=Tree,fill=Tree),size=2.5) +
    …
    … +
    geom_line(data=mod.predict, aes(x=Age,
    y=Predicted_values))
p
```

Note: You can import data from different datasets



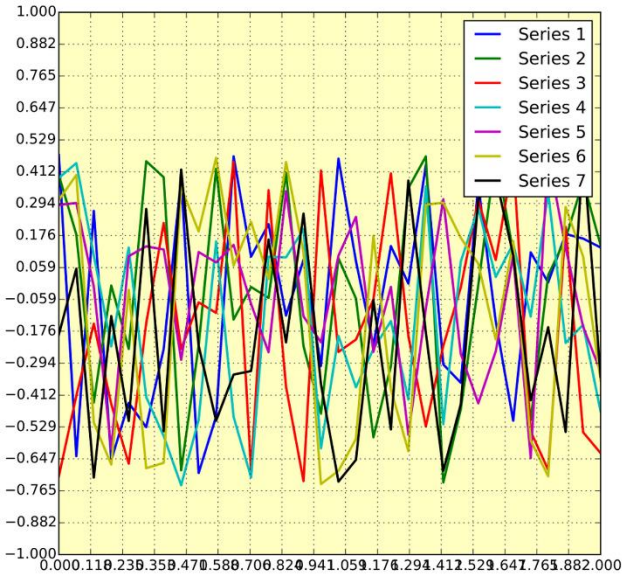
Part 2.5: Saving your figure

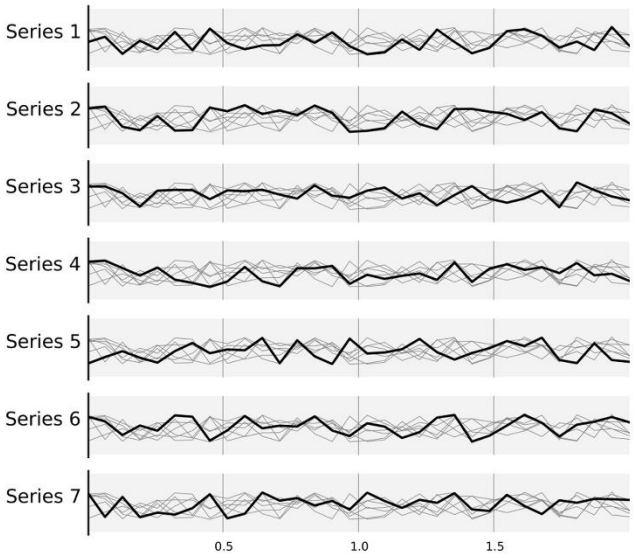
"tiff", "png", "jpeg","bmp" Pixel-based

"pdf", "svg", "eps", "emf" Vector-based


```
ggsave("Your_figure.emf", plot=p, width=6,
height=6, dpi=300, device="emf")
```

8. Avoid ‘chartjunk’



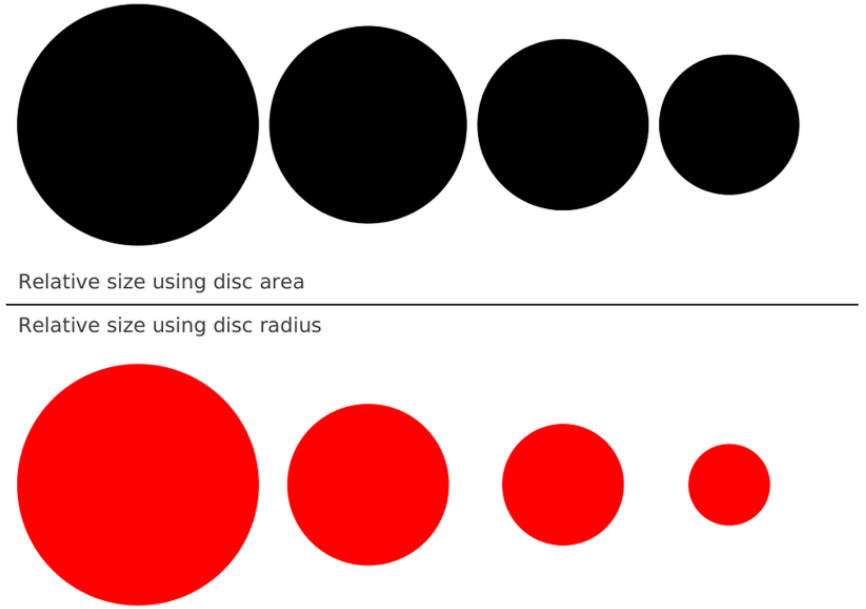


Examples from Rougier et al. 2014

8. Avoid ‘chartjunk’

9. Do not mislead the reader

Series of four values: 30, 20, 15, 10
Upper part: the disc area to represent the value
Lower part: the disc radius.

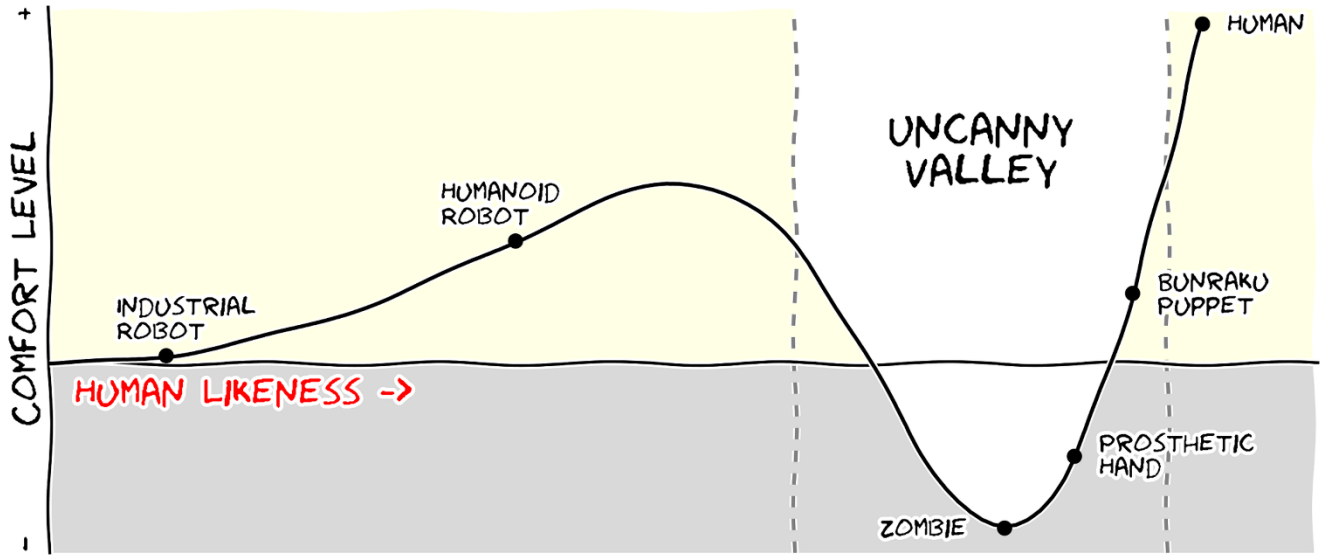


Examples from Rougier et al. 2014

8. Avoid ‘chartjunk’

9. Do not mislead the reader

10. Message over beauty



Example from Rougier et al. 2014

Part 3: Have any tips?

- cowplot() in the package “cowplot” is similar to ggarrange() → It can align multiple plots by their axes
- You can use the package “dplyr” to calculate your mean and standard error directly into your ggplot

→ See: https://rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

Thanks!