dplyr, tidyr, and ggplot2

Intro to the *tidyverse*

Samuel Robinson, Ph.D.

Sep. 15, 2023

->

Part 2: `ggplot2`

Motivation

What is `ggplot2`?

- `ggplot` philosophy

Motivation

What is `ggplot2`?

- `ggplot` philosophy
- Simple plots

Motivation

What is `ggplot2`?

- `ggplot` philosophy
- Simple plots
- Some useful techniques

Motivation

What is `ggplot2`?

- `ggplot` philosophy
- Simple plots
- Some useful techniques
- More complicated plots

What is ggplot2?

- Updated version of `ggplot` (older `R` package)

What is ggplot2?

- Updated version of `ggplot` (older `R` package)
- Implementation of Wilkinson's *grammar of graphics*

What is ggplot2?

- Updated version of `ggplot` (older `R` package)
- Implementation of Wilkinson's *grammar of graphics*
- Elements: data, transformations, elements, scale, guide, coordinates

What is ggplot2?

- Updated version of `ggplot` (older `R` package)
- Implementation of Wilkinson's *grammar of graphics*
- Elements: data, transformations, elements, scale, guide, coordinates
- Describes a layered approach to building graphics beyond formulaic plots (e.g. "boxplot", "scatterplot")

What is ggplot2?

- Updated version of `ggplot` (older `R` package)
- Implementation of Wilkinson's *grammar of graphics*
- Elements: data, transformations, elements, scale, guide, coordinates
- Describes a layered approach to building graphics beyond formulaic plots (e.g. "boxplot", "scatterplot")
- Many different extensions available here

What is ggplot2?

- Updated version of `ggplot` (older `R` package)
- Implementation of Wilkinson's *grammar of graphics*
- Elements: data, transformations, elements, scale, guide, coordinates
- Describes a layered approach to building graphics beyond formulaic plots (e.g. "boxplot", "scatterplot")
- Many different extensions available here

What is ggplot2?

- Updated version of `ggplot` (older `R` package)
- Implementation of Wilkinson's *grammar of graphics*
- Elements: data, transformations, elements, scale, guide, coordinates
- Describes a layered approach to building graphics beyond formulaic plots (e.g. "boxplot", "scatterplot")
- Many different extensions available here

Philosophy:

- Data input centered around around `data.frames` or `tibbles`

What is ggplot2?

- Updated version of `ggplot` (older `R` package)
- Implementation of Wilkinson's *grammar of graphics*
- Elements: data, transformations, elements, scale, guide, coordinates
- Describes a layered approach to building graphics beyond formulaic plots (e.g. "boxplot", "scatterplot")
- Many different extensions available here

Philosophy:

- Data input centered around around `data.frames` or `tibbles`
- Data display centered around `geoms` (geometric objects)

What is ggplot2?

- Updated version of `ggplot` (older `R` package)
- Implementation of Wilkinson's *grammar of graphics*
- Elements: data, transformations, elements, scale, guide, coordinates
- Describes a layered approach to building graphics beyond formulaic plots (e.g. "boxplot", "scatterplot")
- Many different extensions available here

Philosophy:

- Data input centered around around `data.frames` or `tibbles`
- Data display centered around `geoms` (geometric objects)
- Columns from data frames are `mapped` into `geoms` using `aesthetics`

What is ggplot2?

- Updated version of `ggplot` (older `R` package)
- Implementation of Wilkinson's *grammar of graphics*
- Elements: data, transformations, elements, scale, guide, coordinates
- Describes a layered approach to building graphics beyond formulaic plots (e.g. "boxplot", "scatterplot")
- Many different extensions available here

Philosophy:

- Data input centered around around `data.frames` or `tibbles`
- Data display centered around `geoms` (geometric objects)
- Columns from data frames are `mapped` into `geoms` using `aesthetics`
- `geoms` are displayed according to `themes`

Simple example - scatterplot

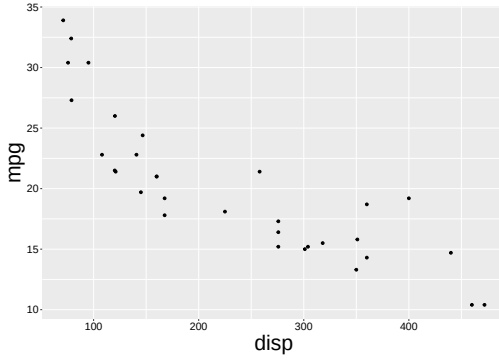
```
data(mtcars) # mtcars dataset (built into R)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

Top line of code says:

- data from `mtcars` dataframe

```
ggplot(data = mtcars, aes(x = disp, y = mpg))+
  geom_point() # Display data using points
```



Simple example - scatterplot

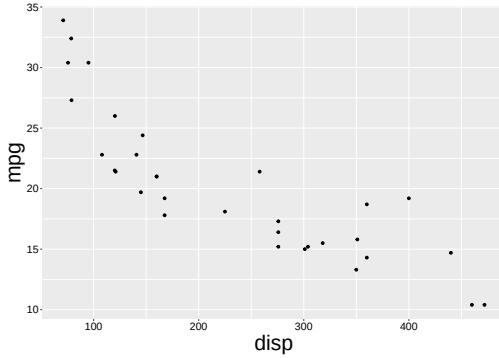
```
data(mtcars) # mtcars dataset (built into R)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

Top line of code says:

- data from `mtcars` dataframe
- `aes` = aesthetics from dataframe

```
ggplot(data = mtcars, aes(x = disp, y = mpg))+
  geom_point() # Display data using points
```



Simple example - scatterplot

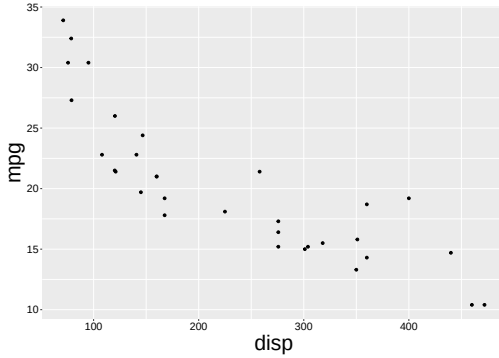
```
data(mtcars) # mtcars dataset (built into R)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

Top line of code says:

- data from `mtcars` dataframe
- `aes` = aesthetics from dataframe
- map `disp` to x-axis, `mpg` to y-axis

```
ggplot(data = mtcars, aes(x = disp, y = mpg))+
  geom_point() # Display data using points
```



Simple example - bar plot

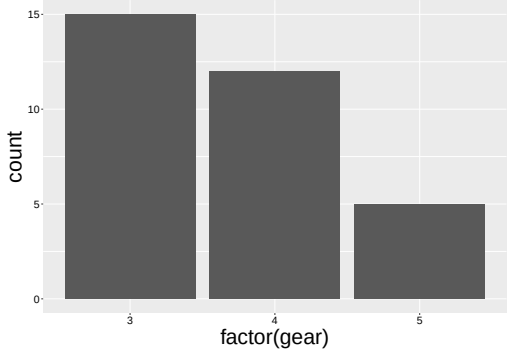
```
data(mtcars) # mtcars dataset (built into R)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0  0    3    2
```

Top line of code says:

- map gear to x-axis (first converted to a `factor`)

```
ggplot(data = mtcars, aes(x = factor(gear)))+
  geom_bar()
# Display number of data points for each
#   factor level
```



Simple example - bar plot

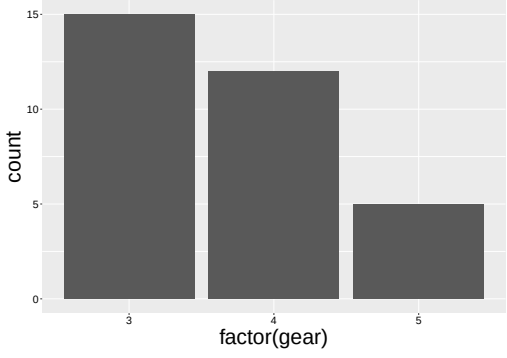
```
data(mtcars) # mtcars dataset (built into R)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

Top line of code says:

- map `gear` to x-axis (first converted to a `factor`)
- Automatically uses `stat='count'` to group data according to factor

```
ggplot(data = mtcars, aes(x = factor(gear)))+
  geom_bar()
# Display number of data points for each
#   factor level
```



Simple example - histogram

```
data(mtcars) # mtcars dataset (built into R)
```

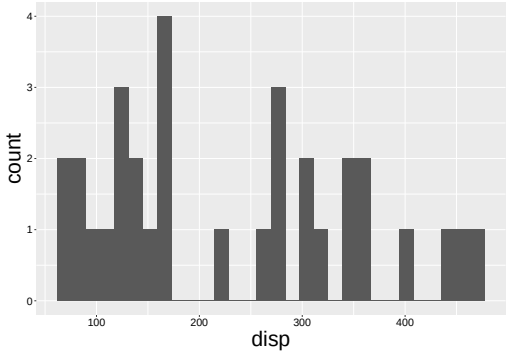
```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

Top line of code says:

- map `disp` to x-axis

```
ggplot(data = mtcars, aes(x = disp))+
  # Group disp into bins, and display
  #   count in each bin
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better valu
```



Simple example - histogram

```
data(mtcars) # mtcars dataset (built into R)
```

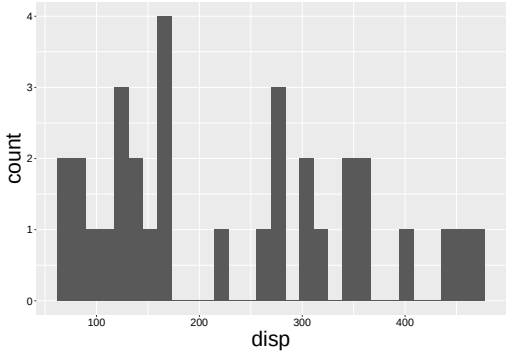
```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

Top line of code says:

- map disp to x-axis
- geom_histogram()

```
ggplot(data = mtcars, aes(x = disp))+
  # Group disp into bins, and display
  #   count in each bin
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better valu
```

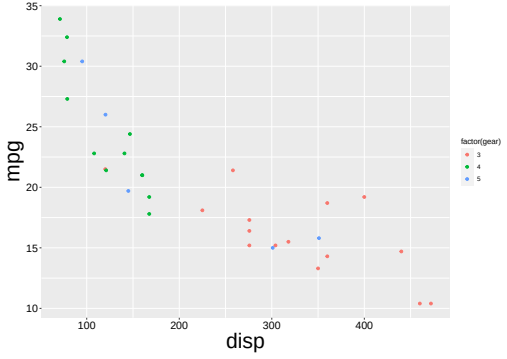


Colours in plots

- Colours can be *mapped* (via aes) or *set* (outside of aes)

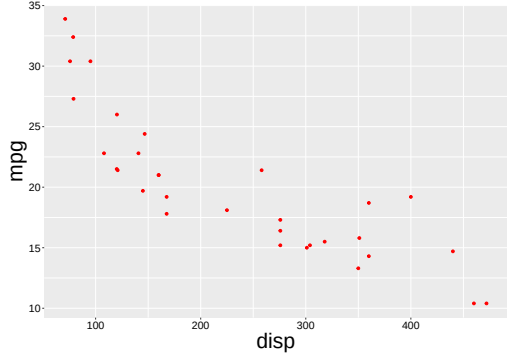
Maps colour to gear

```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(aes(col=factor(gear)))
```



Sets colour as red

```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(colour='red')
```

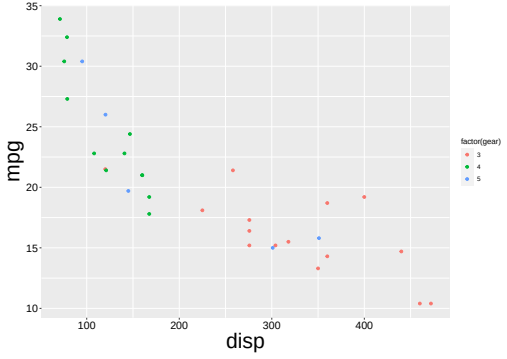


Colours in plots

- Colours can be *mapped* (via aes) or *set* (outside of aes)
- *mapping* associates a variable with a colour scheme, *setting* fixes the colour to a preset value

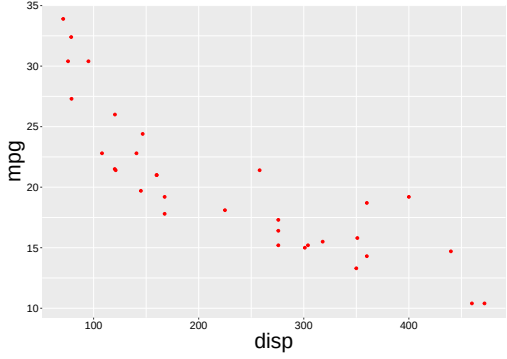
Maps colour to gear

```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(aes(col=factor(gear)))
```



Sets colour as red

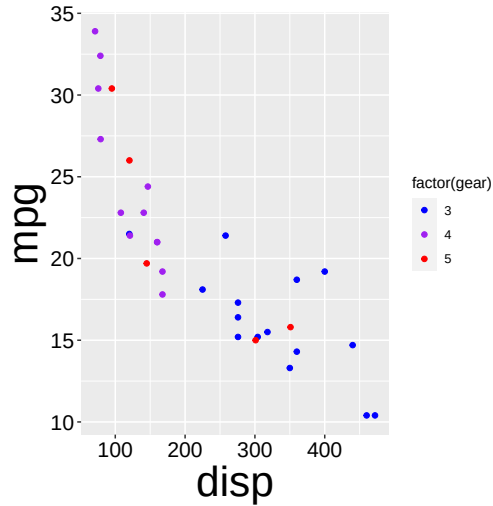
```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(colour='red')
```



What if I want different colours?

- Default colour themes are pretty bad. Change them with `scale_colour_manual` or `scale_fill_manual`

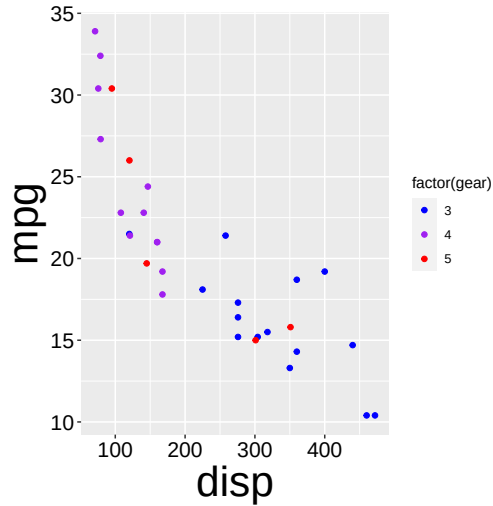
```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(aes(col=factor(gear)))+
  scale_colour_manual(values=c('blue','purple','red
```



What if I want different colours?

- Default colour themes are pretty bad. Change them with `scale_colour_manual` or `scale_fill_manual`
- `scale_colour_brewer` is generally pretty good; see examples here

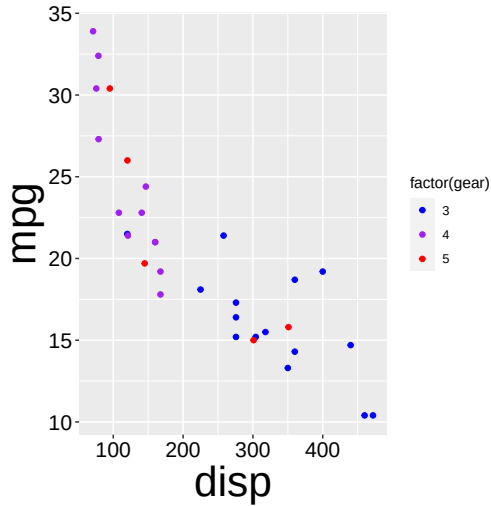
```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(aes(col=factor(gear)))+
  scale_colour_manual(values=c('blue','purple','red
```



What if I want different colours?

- Default colour themes are pretty bad. Change them with `scale_colour_manual` or `scale_fill_manual`
- `scale_colour_brewer` is generally pretty good; see examples here
- ~10% of (European) males are red-green colourblind; see here for some suggested schemes

```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(aes(col=factor(gear)))+
  scale_colour_manual(values=c('blue','purple','red
```

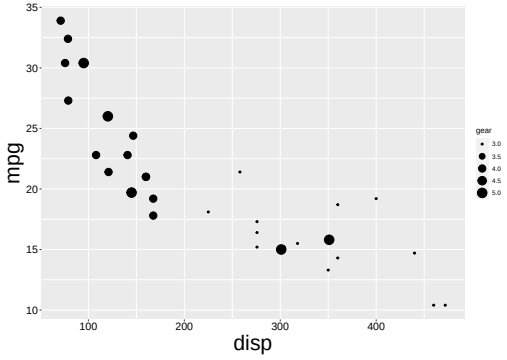


Sizes in plots

- Sizes of things can also be *mapped* (via aes) or *set* (outside of aes), similar to colours

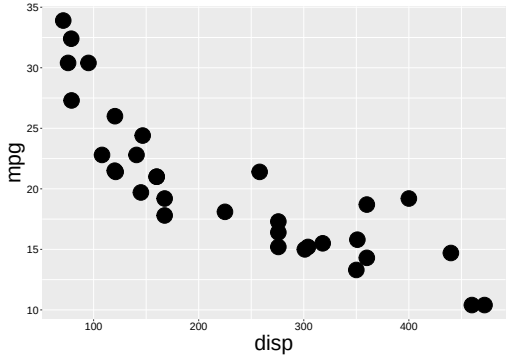
Maps gear to size:

```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(aes(size=gear))
```



Sets size at 10:

```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(size=10)
```

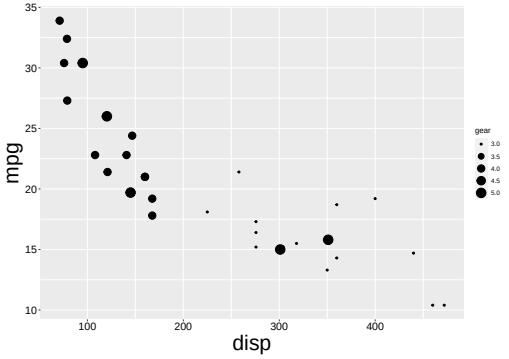


Sizes in plots

- Sizes of things can also be *mapped* (via `aes`) or *set* (outside of `aes`), similar to colours

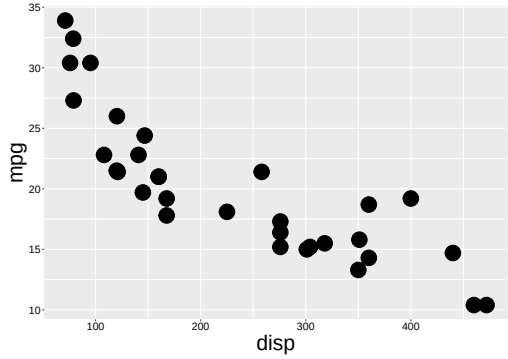
Maps `gear` to `size`:

```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(aes(size=gear))
```



Sets size at 10:

```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  geom_point(size=10)
```

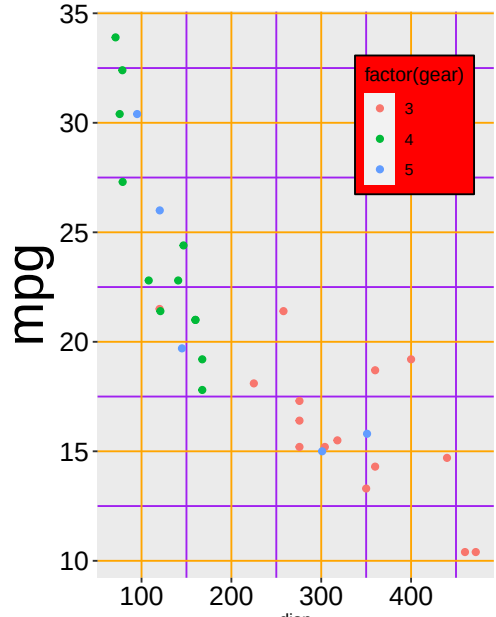


- Similar to colour choices, you can alter mapped sizes using `scale_size`

Change plot theme

- theme controls almost all non-data elements of plots

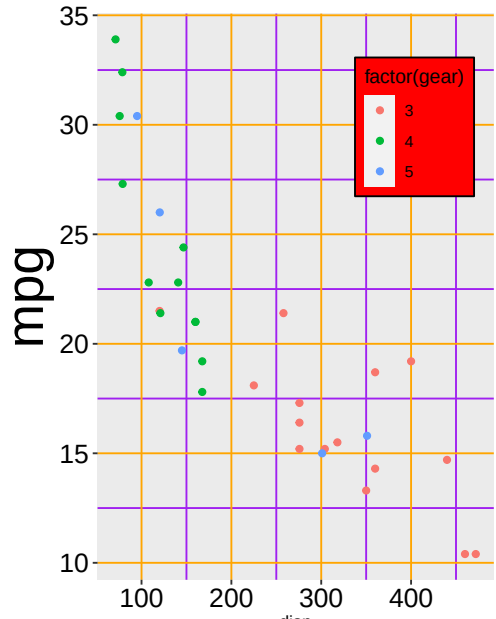
```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  # Maps gear to colour
  geom_point(aes(col=factor(gear))) +
  #Changes plot theme
  theme(axis.title.x=element_text(size=10),
    legend.background=element_rect(fill='red'),
    legend.position=c(0.8,0.8),
    panel.grid.minor=element_line(colour='purple'),
    panel.grid.major=element_line(colour='orange'))
```



Change plot theme

- theme controls almost all non-data elements of plots
- Made up of *elements*: `element_line()`, `element_text()`, `element_rect()`

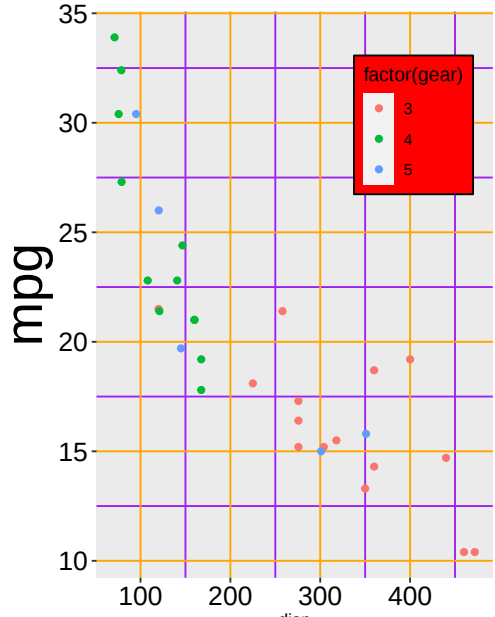
```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  # Maps gear to colour
  geom_point(aes(col=factor(gear))) +
  #Changes plot theme
  theme(axis.title.x=element_text(size=10),
    legend.background=element_rect(fill='red'),
    legend.position=c(0.8,0.8),
    panel.grid.minor=element_line(colour='purple'),
    panel.grid.major=element_line(colour='orange'))
```



Change plot theme

- theme controls almost all non-data elements of plots
- Made up of *elements*: `element_line()`, `element_text()`, `element_rect()`
- Let's make some changes:

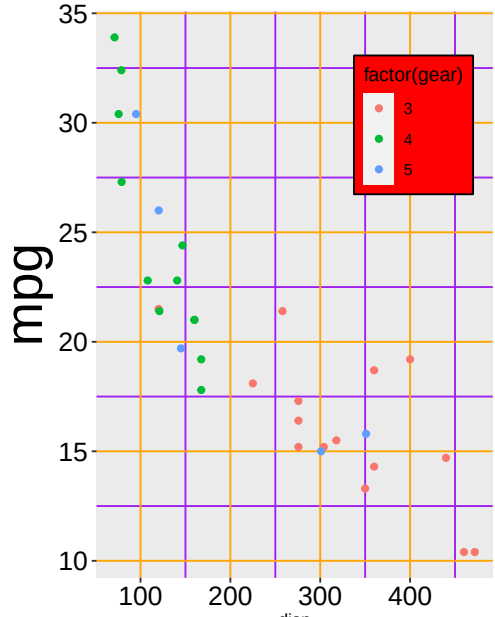
```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  # Maps gear to colour
  geom_point(aes(col=factor(gear))) +
  #Changes plot theme
  theme(axis.title.x=element_text(size=10),
    legend.background=element_rect(fill='red'),
    legend.position=c(0.8,0.8),
    panel.grid.minor=element_line(colour='purple'),
    panel.grid.major=element_line(colour='orange'))
```



Change plot theme

- theme controls almost all non-data elements of plots
- Made up of *elements*: `element_line()`, `element_text()`, `element_rect()`
- Let's make some changes:

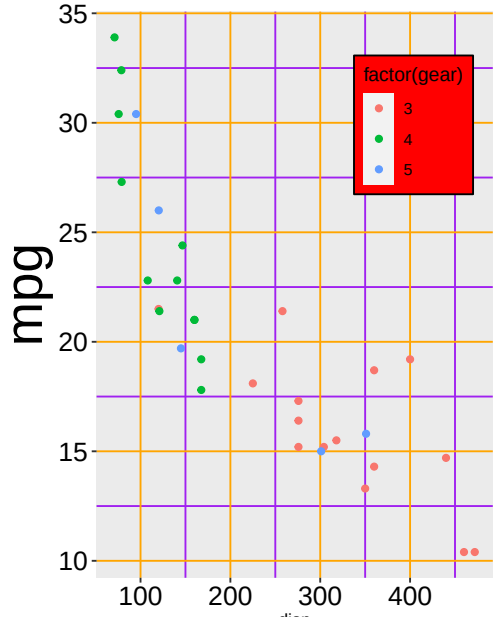
```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  # Maps gear to colour
  geom_point(aes(col=factor(gear))) +
  #Changes plot theme
  theme(axis.title.x=element_text(size=10),
    legend.background=element_rect(fill='red'),
    legend.position=c(0.8,0.8),
    panel.grid.minor=element_line(colour='purple'),
    panel.grid.major=element_line(colour='orange'))
```



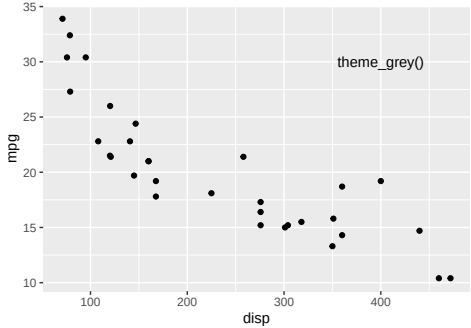
Change plot theme

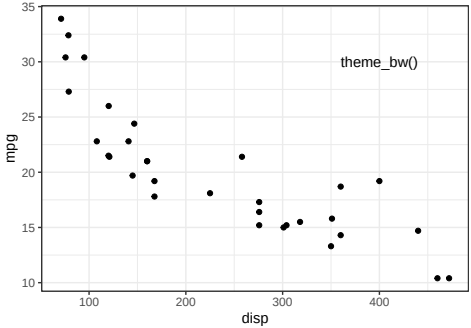
- theme controls almost all non-data elements of plots
- Made up of *elements*: `element_line()`, `element_text()`, `element_rect()`
- Let's make some changes:

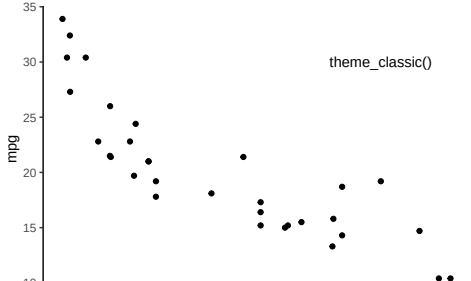
```
ggplot(data=mtcars,aes(x=disp,y=mpg))+
  # Maps gear to colour
  geom_point(aes(col=factor(gear))) +
  #Changes plot theme
  theme(axis.title.x=element_text(size=10),
    legend.background=element_rect(fill='red'),
    legend.position=c(0.8,0.8),
    panel.grid.minor=element_line(colour='purple'),
    panel.grid.major=element_line(colour='orange'))
```

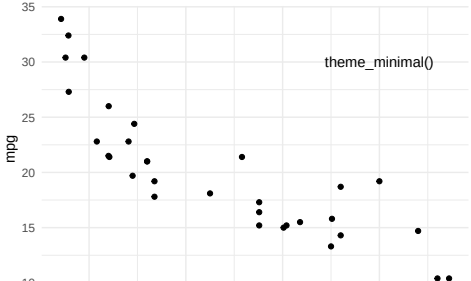


Preset themes





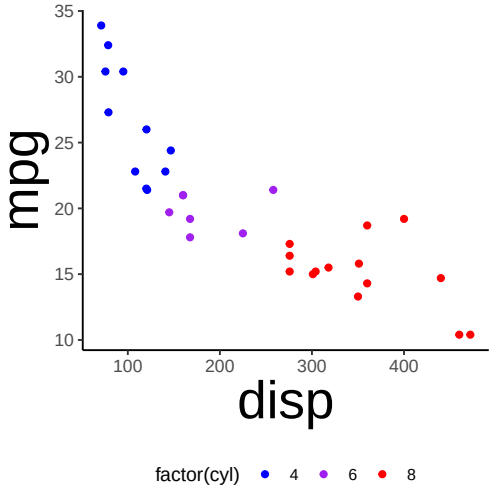




Make your own themes!

- You can modify existing themes in order to create your own

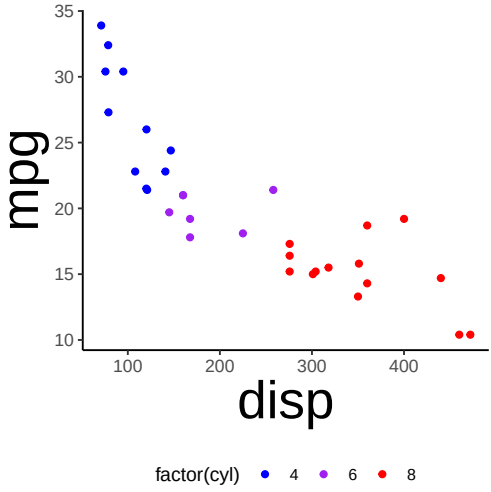
```
myTheme <- theme_classic()+ #Existing theme
  #Makes axis text bigger
  theme(axis.title=element_text(size=30),
        axis.text=element_text(size=10),
        legend.position='bottom')
#Sets up this theme as "default"
theme_set(myTheme)
```



Make your own themes!

- You can modify existing themes in order to create your own
- Try using `theme_set()` at the start of your script to pre-set the theme for the rest of the script

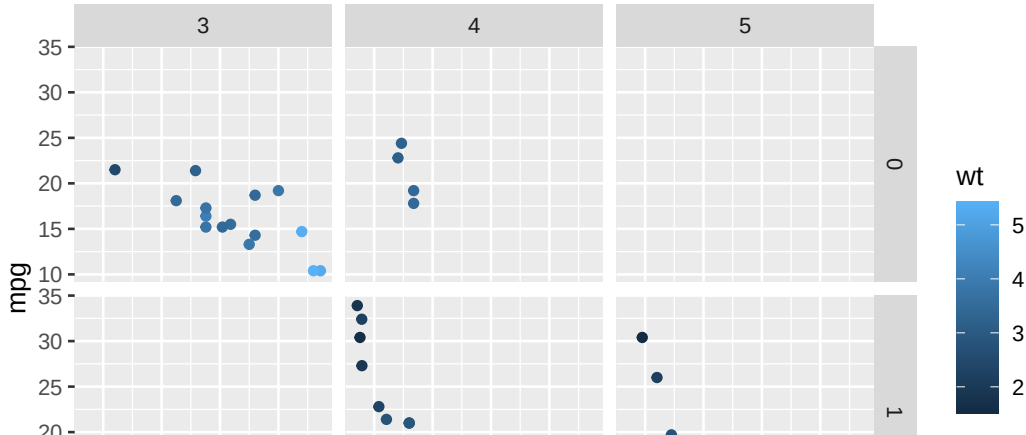
```
myTheme <- theme_classic()+ #Existing theme
  #Makes axis text bigger
  theme(axis.title=element_text(size=30),
        axis.text=element_text(size=10),
        legend.position='bottom')
#Sets up this theme as "default"
theme_set(myTheme)
```



Complex plots - facets

- It is possible to break up the plot into smaller `facets` that are mapped to a given variable

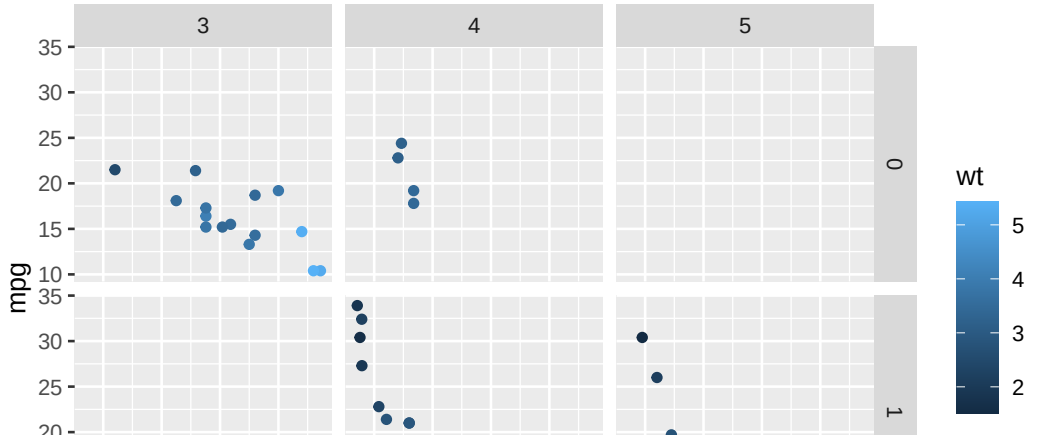
```
ggplot(mtcars,aes(x=disp,y=mpg))+ geom_point(aes(col=wt))+
  facet_grid(factor(am) ~ factor(gear))
```



Complex plots - facets

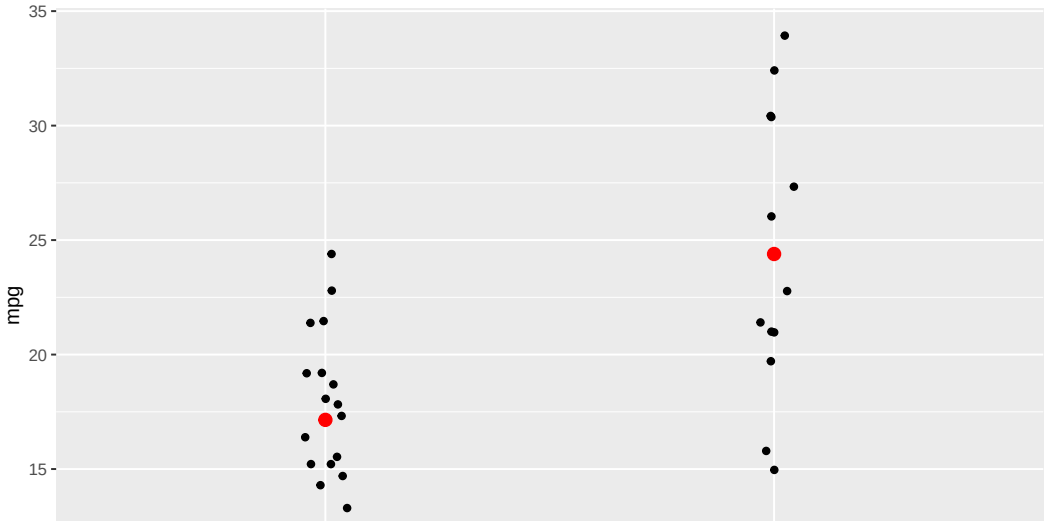
- It is possible to break up the plot into smaller `facets` that are mapped to a given variable
- This can be combined with colour/size mappings

```
ggplot(mtcars,aes(x=disp,y=mpg))+ geom_point(aes(col=wt))+
  facet_grid(factor(am) ~ factor(gear))
```



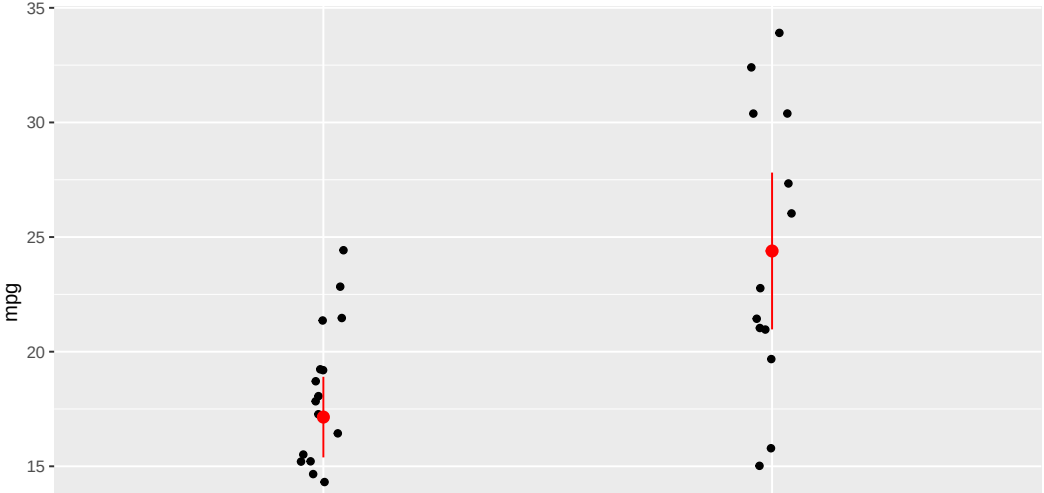
Complex plots - summary statistics (mean)

```
ggplot(mtcars,aes(x=factor(am),y=mpg))+
  geom_point(position=position_jitter(width=0.05))+ #Adds noise to data in x-dimension
  geom_point(stat='summary',fun=mean,col='red',size=3) #Mean only
```



Complex plots - summary statistics (mean + SD)

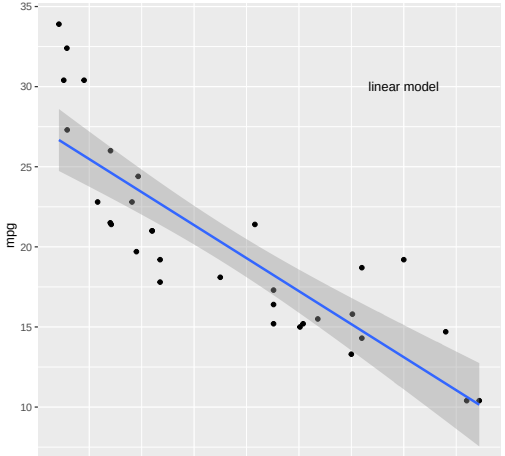
```
ggplot(arrange(mtcars,am,disp),aes(x=factor(am),y=mpg))+
  geom_point(position=position_jitter(width=0.05))+
  geom_pointrange(stat='summary',fun.data=mean_se,
                  fun.args = list(mult = 2),col='red') #Mean + 2 SE
```



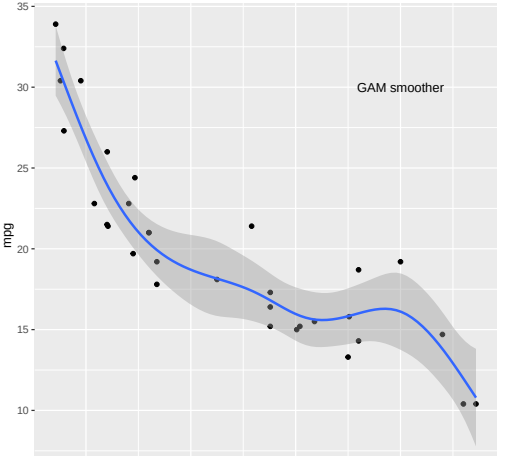
Complex plots - smoothers

- You can add `lm` (or other model) predictions to your plots:

```
ggplot(mtcars,aes(x=disp,y=mpg))+
  geom_point() +
  geom_smooth(method='lm',formula=y~x)
```



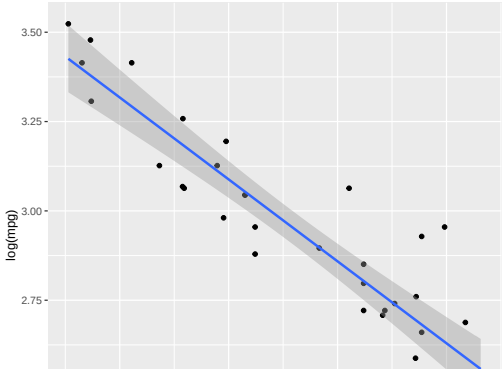

```
ggplot(mtcars,aes(x=disp,y=mpg))+
  geom_point() +
  geom_smooth(method='gam',formula=y~s(x))
```



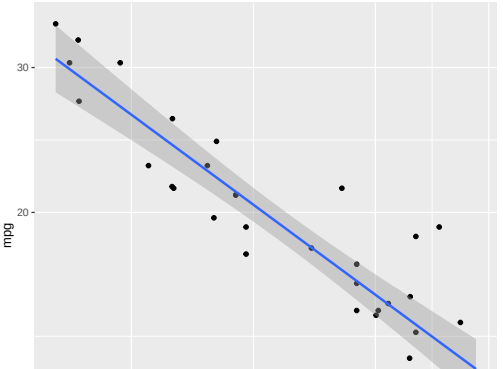
Complex plots - transformations

- You can show transformed data OR you can transform the axes themselves using `scale_*_log10` (x or y axis)

```
ggplot(mtcars,aes(x=log(disp),y=log(mpg)))+
  geom_point() +
  geom_smooth(method='lm',formula=y~x)
# Harder to interpret, because people can't
#     usually do log(x) in their head
```

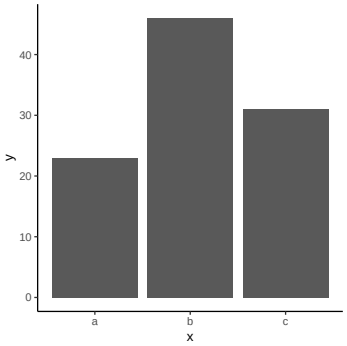


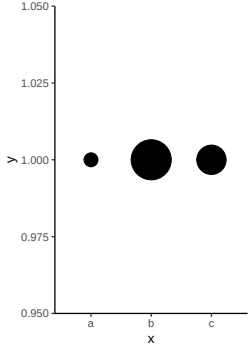

```
ggplot(mtcars,aes(x=disp,y=mpg))+
  geom_point() +
  geom_smooth(method='lm',formula=y~x)+
  scale_x_log10() + scale_y_log10()
# sqrt is also popular
```

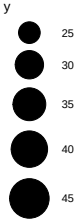


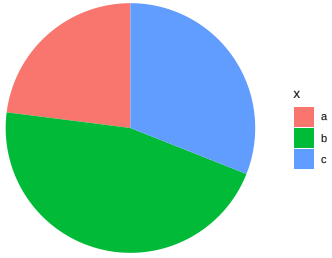
Things to remember:

- Simpler plots are often better. Try to keep it to 3 aesthetics per panel. Avoid 3D plots.



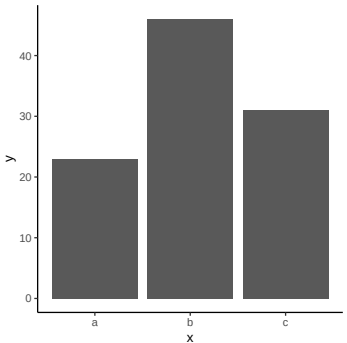


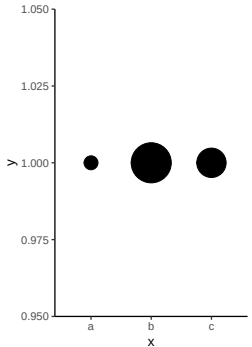


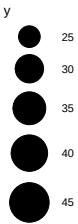


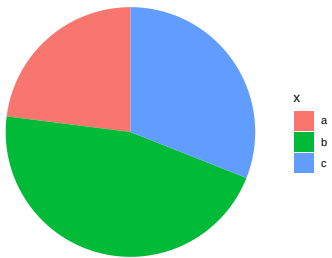
Things to remember:

- Simpler plots are often better. Try to keep it to 3 aesthetics per panel. Avoid 3D plots.
- Making plots is iterative. Make a simple one and tweak it to improve it.



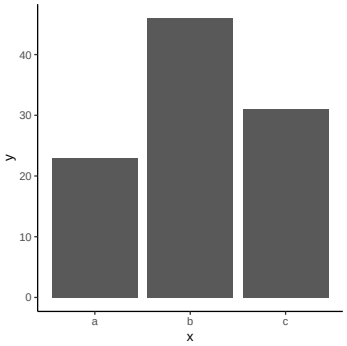


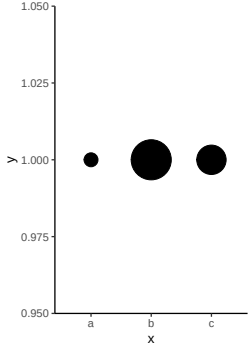


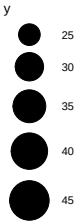


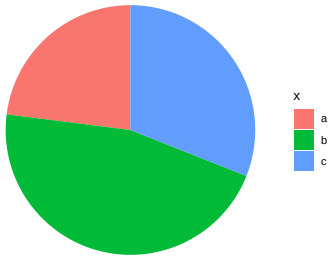
Things to remember:

- Simpler plots are often better. Try to keep it to 3 aesthetics per panel. Avoid 3D plots.
- Making plots is iterative. Make a simple one and tweak it to improve it.
- Avoid "non-data ink" (see Edward Tufte's work)



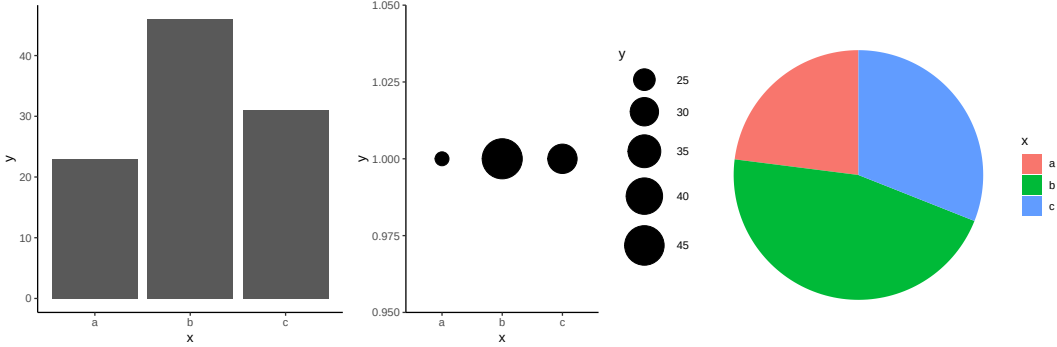






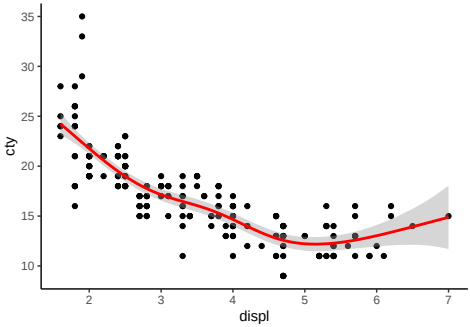
Things to remember:

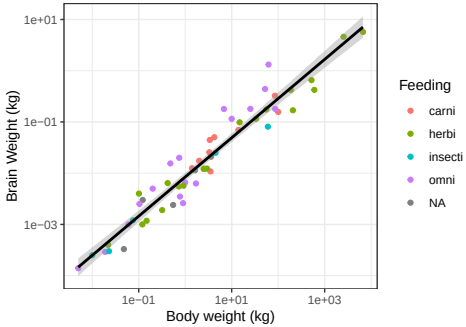
- Simpler plots are often better. Try to keep it to 3 aesthetics per panel. Avoid 3D plots.
- Making plots is iterative. Make a simple one and tweak it to improve it.
- Avoid "non-data ink" (see Edward Tufte's work)
- Our eyes are good at estimating linear positions, but bad at estimating area, volume, colour shading, and angles:

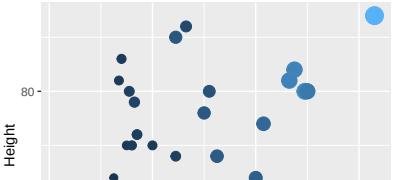


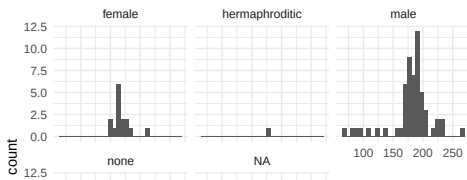
A challenger approaches:

Make these figures! Datasets are found in `mpg`, `msleep`, `trees`, and `starwars` (built into the `ggplot2` and `dplyr` packages)





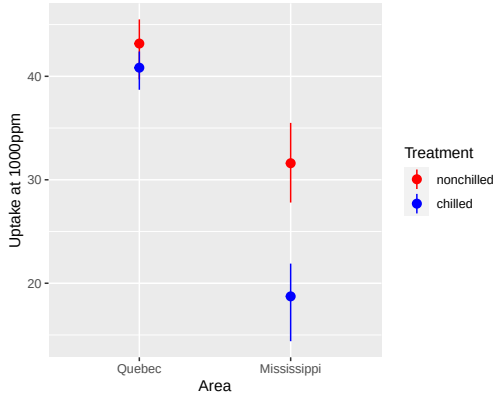




Final remarks

- `dplyr` & `tidyr` work with other parts of the tidyverse, such as `ggplot2`

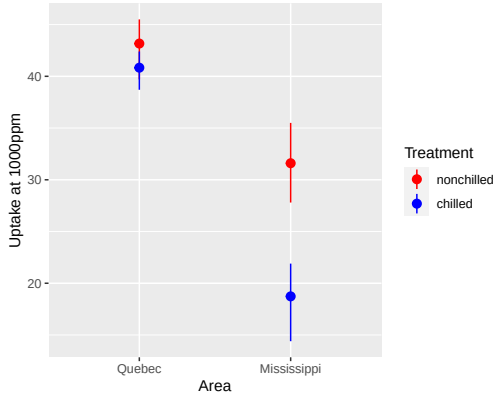
```
library(ggplot2)
#Code for dplyr begins here
CO2 %>% filter(conc==1000) %>%
  group_by(Type,Treatment) %>%
  summarize(meanUp=mean(uptake),
            maxUp=max(uptake),
            minUp=min(uptake)) %>%
  #Code for ggplot begins here
  ggplot(aes(x=Type,col=Treatment))+
  geom_pointrange(aes(y=meanUp,
                      ymax=maxUp,
                      ymin=minUp))+
  labs(x='Area',y='Uptake at 1000ppm')+
  scale_colour_manual(values=c('red','blue'))
```



Final remarks

- `dplyr` & `tidyr` work with other parts of the tidyverse, such as `ggplot2`
- Example: filtered summary plot

```
library(ggplot2)
#Code for dplyr begins here
CO2 %>% filter(conc==1000) %>%
  group_by(Type,Treatment) %>%
  summarize(meanUp=mean(uptake),
            maxUp=max(uptake),
            minUp=min(uptake)) %>%
  #Code for ggplot begins here
  ggplot(aes(x=Type,col=Treatment))+
  geom_pointrange(aes(y=meanUp,
                      ymax=maxUp,
                      ymin=minUp))+
  labs(x='Area',y='Uptake at 1000ppm')+
  scale_colour_manual(values=c('red','blue'))
```

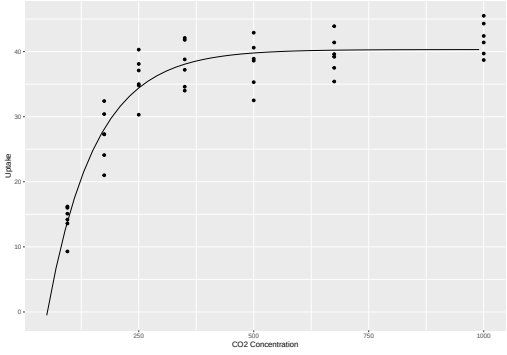


Final remarks

- dplyr & tidyr can pass data frames to and from non-`tidyverse` functions: use '.' operator

```
co2mod <- CO2 %>%
  filter(Type=='Quebec') %>%
  #Code for nls begins here
  nls(uptake~SSasymp(conc,A,B,C),
      start=list(A=30,B=-15,C=-5),data=.)

data.frame(conc=seq(50,1000,20)) %>%
  predict(co2mod,newdata=.) %>%
  data.frame(conc=seq(50,1000,20),predUp=.) %>%
  #Code for ggplot begins here
  ggplot(aes(conc,predUp))+
  geom_line()+
  geom_point(data=filter(CO2,Type=='Quebec'),
             aes(conc,uptake))+
  labs(x='CO2 Concentration',y='Uptake')
```



Final remarks

- dplyr & tidyr can pass data frames to and from non-`tidyverse` functions: use '.' operator
- Example: nonlinear growth model

```
co2mod <- CO2 %>%
  filter(Type=='Quebec') %>%
  #Code for nls begins here
  nls(uptake~SSasymp(conc,A,B,C),
      start=list(A=30,B=-15,C=-5),data=.)

data.frame(conc=seq(50,1000,20)) %>%
  predict(co2mod,newdata=.) %>%
  data.frame(conc=seq(50,1000,20),predUp=.) %>%
  #Code for ggplot begins here
  ggplot(aes(conc,predUp))+
  geom_line()+
  geom_point(data=filter(CO2,Type=='Quebec'),
             aes(conc,uptake))+
  labs(x='CO2 Concentration',y='Uptake')
```