

Introduction to ggplot2

Michael Friendly
Psych 6135

<http://euclid.psych.yorku.ca/www/psy6135/>

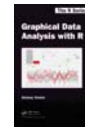
Resources: Books



Hadley Wickham, *ggplot2: Elegant graphics for data analysis*, 2nd Ed.
1st Ed: Online, <http://ggplot2.org/book/>
ggplot2 Quick Reference: <http://sape.inf.usi.ch/quick-reference/ggplot2/>
Complete ggplot2 documentation: <http://docs.ggplot2.org/current/>



Winston Chang, *R Graphics Cookbook: Practical Recipes for Visualizing Data*
Cookbook format, covering common graphing tasks; the main focus is on ggplot2
R code from book: <http://www.cookbook-r.com/Graphs/>
Download from: <http://ase.tufts.edu/bugs/guide/assets/R%20Graphics%20Cookbook.pdf>



Antony Unwin, *Graphical Data Analysis with R*
A gentle introduction to doing visual data analysis, mainly with ggplot2.
R code: <http://www.gradaanwr.net/>



G. Grolemund & H. Wickham, *R for Data Science*
A text on the tidyverse approach to data wrangling and visualization using ggplot2.
Online version: <http://r4ds.had.co.nz/>

Resources: Cheat sheets

- Data visualization with ggplot2:
<https://www.rstudio.com/wp-content/uploads/2016/11/ggplot2-cheatsheet-2.1.pdf>
- Data transformation with dplyr:
<https://github.com/rstudio/cheatsheets/raw/master/source/pdfs/data-transformation-cheatsheet.pdf>



What is ggplot2?

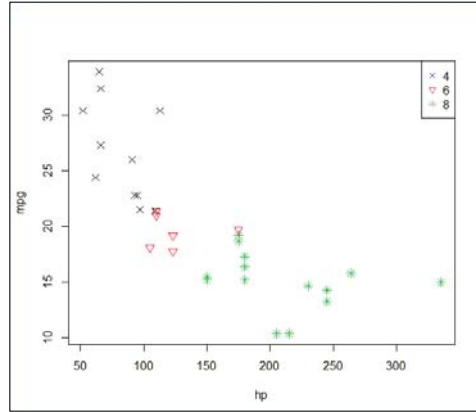
- ggplot2 is Hadley Wickham's R package for producing "elegant graphics for data analysis"
 - It is an implementation of many of the ideas for graphics introduced in Lee Wilkinson's *Grammar of Graphics*
 - These ideas and the syntax of ggplot2 help to think of graphs in a new and more general way
 - Produces pleasing plots, taking care of many of the fiddly details (legends, axes, colors, ...)
 - It is built upon the "grid" graphics system
 - It is open software, with a large number of gg_ extensions.
See: <http://www.ggplot2-exts.org/gallery/>

ggplot2 vs base graphics

Some things that should be simple are harder than you'd like in base graphics

Here, I'm plotting gas mileage (mpg) vs. horsepower and want to use color and shape for different # of cylinders.

But I don't quite get it right!



```
mtcars$cyl <- as.factor(mtcars$cyl)
plot(mpg ~ hp, data=mtcars,
     col=cyl, pch=c(4,6,8)[mtcars$cyl], cex=1.2)
legend("topright", legend=levels(mtcars$cyl),
     pch = c(4,6,8),
     col=levels(mtcars$cyl))
```

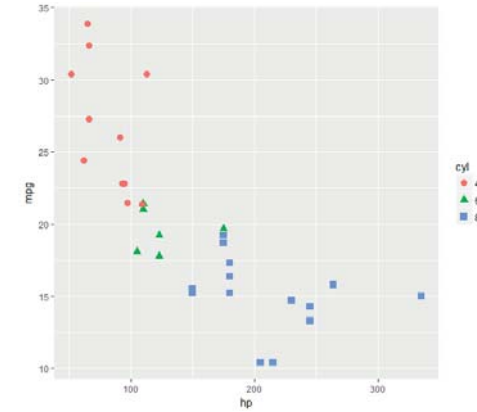
colors and point symbols work differently in plot() and legend()

ggplot2 vs base graphics

In ggplot2, just map the data variables to aesthetic attributes
aes(x, y, shape, color, size, ...)

ggplot() takes care of the rest

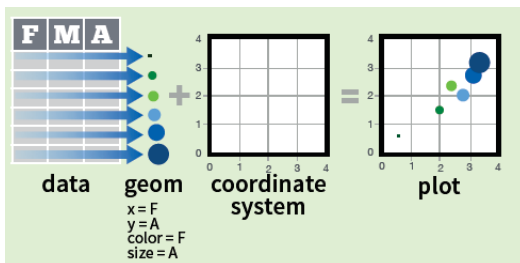
aes() mappings set in the call to ggplot() are passed to geom_point() here



```
library(ggplot2)
ggplot(mtcars, aes(x=hp, y=mpg, color=cyl, shape=cyl)) +
  geom_point(size=3)
```

Grammar of Graphics

- Every graph can be described as a combination of independent building blocks:
 - **data**: a data frame: quantitative, categorical; local or data base query
 - **aesthetic** mapping of variables into visual properties: size, color, x, y
 - **geometric** objects ("geom"): points, lines, areas, arrows, ...
 - **coordinate** system ("coord"): Cartesian, log, polar, map,



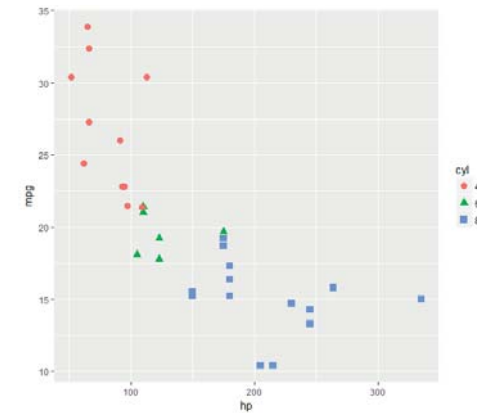
ggplot2: data + geom -> graph

```
ggplot(data=mtcars,
       aes(x=hp, y=mpg,
           color=cyl, shape=cyl)) +
  geom_point(size=3)
```

- 1
- 2
- 3
- 4

In this call,

1. data=mtcars: data frame
 2. aes(x=hp, y=mpg): plot variables
 3. aes(color, shape): attributes
 4. geom_point(): what to plot
- the coordinate system is taken to be the standard Cartesian (x,y)



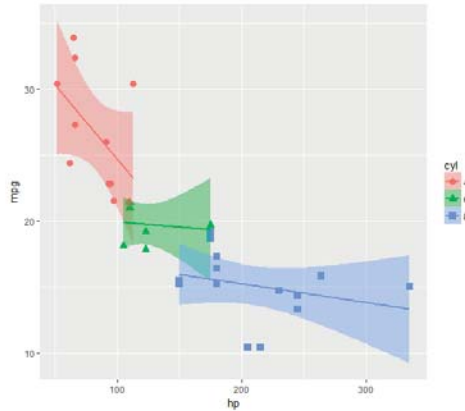
ggplot2: geoms

Wow! I can really see something there.

How can I enhance this visualization?

Easy: add a `geom_smooth()` to fit linear regressions for each level of `cyl`

It is clear that horsepower and # of cylinders are highly related (Duh!)



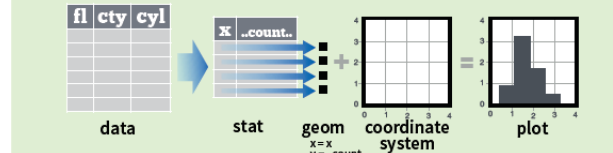
```
ggplot(mtcars, aes(x=hp, y=mpg, color=cyl, shape=cyl)) +
  geom_point(size=3) +
  geom_smooth(method="lm", aes(fill=cyl))
```

Grammar of Graphics

Other GoG building blocks:

- statistical transformations ("stat") -- data summaries: mean, sd, binning & counting, ...
- scales: legends, axes to allow reading data from a plot

A stat builds new variables to plot (e.g., count, prop).



Grammar of Graphics

Other GoG building blocks:

- position adjustments: jitter, dodge, stack, ...
- faceting: small multiples or conditioning to break a plot into subsets.

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

```
s <- ggplot(mpg, aes(fl, fill = drv))
```

- `s + geom_bar(position = "dodge")`
Arrange elements side by side
- `s + geom_bar(position = "fill")`
Stack elements on top of one another, normalize height
- `e + geom_point(position = "jitter")`
Add random noise to X and Y position of each element to avoid overplotting
- `e + geom_label(position = "nudge")`
Nudge labels away from points
- `s + geom_bar(position = "stack")`
Stack elements on top of one another

Each position adjustment can be recast as a function with manual `width` and `height` arguments

```
s + geom_bar(position = position_dodge(width = 1))
```

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

```
t <- ggplot(mpg, aes(cty, hwy)) + geom_point()
```

- `t + facet_grid(~ fl)`
facet into columns based on fl
- `t + facet_grid(year ~ .)`
facet into rows based on year
- `t + facet_grid(year ~ fl)`
facet into both rows and columns
- `t + facet_wrap(~ fl)`
wrap facets into a rectangular layout

ggplot2: GoG -> graphic language

The implementation of GoG ideas in ggplot2 for R created a more expressive language for data graphs

- layers: graph elements combined with "+" (read: "and")

```
ggplot(mtcars, aes(x=hp, y=mpg)) +
  geom_point(aes(color = cyl)) +
  geom_smooth(method = "lm") +
```

- themes: change graphic elements consistently

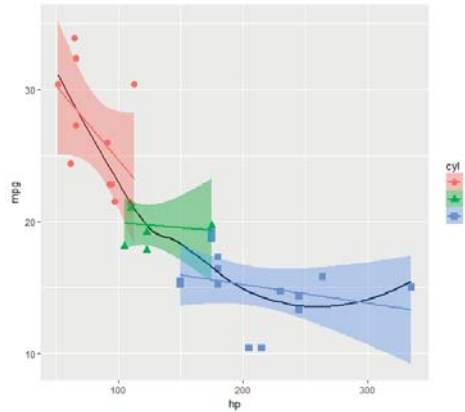
- `r + theme_bw()`
White background with grid lines
- `r + theme_classic()`
White background, no grid
- `r + theme_gray()`
Grey background (default theme)
- `r + theme_linedraw()`
Minimal themes
- `r + theme_minimal()`
Minimal themes
- `r + theme_dark()`
dark for contrast
- `r + theme_void()`
Empty theme

ggplot2: layers & aes()

Aesthetic attributes in the `ggplot()` call are passed to `geom_()` layers

Other attributes can be passed as **constants** (`size=3, color="black"`) or with `aes(color=, ...)` in different layers

This plot adds an overall loess smooth to the previous plot.
`color="black"` overrides the `aes(color=cyl)`



```
ggplot(mtcars, aes(x=hp, y=mpg)) +
  geom_point(size=3, aes(color=cyl, shape=cyl)) +
  geom_smooth(method="lm", aes(color=cyl, fill=cyl)) +
  geom_smooth(method="loess", color="black", se=FALSE)
```

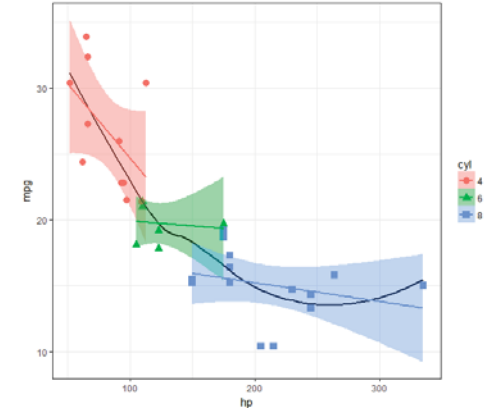
13

ggplot2: themes

All the graphical attributes of ggplot2 are governed by themes – settings for all aspects of a plot

A given plot can be rendered quite differently just by changing the theme

If you haven't saved the ggplot object, `last_plot()` gives you something to work with further



`last_plot() + theme_bw()`

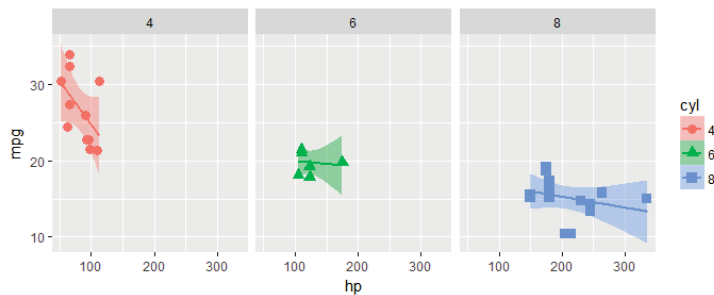
14

ggplot2: facets

Facets divide a plot into separate subplots based on one or more discrete variables

```
plt <-
ggplot(mtcars, aes(x=hp, y=mpg, color=cyl, shape=cyl)) +
  geom_point(size=3) +
  geom_smooth(method="lm", aes(fill=cyl))
```

`plt + facet_wrap(~cyl)`



15

labeling points: geom_text()

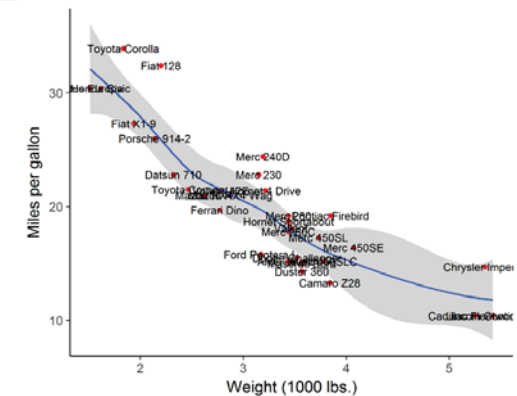
```
plt2 <- ggplot(mtcars, aes(x=wt, y=mpg)) +
  geom_point(color = 'red', size=2) +
  geom_smooth(method="loess") +
  labs(y="Miles per gallon", x="Weight (1000 lbs.)") +
  theme_classic(base_size = 16)
```

Sometimes it is useful to label points to show their identities.

`geom_text()` usually gives messy, overlapping text

`plt2 + geom_text(aes(label = rownames(mtcars)))`

Note the use of `theme_classic()` and better axis labels



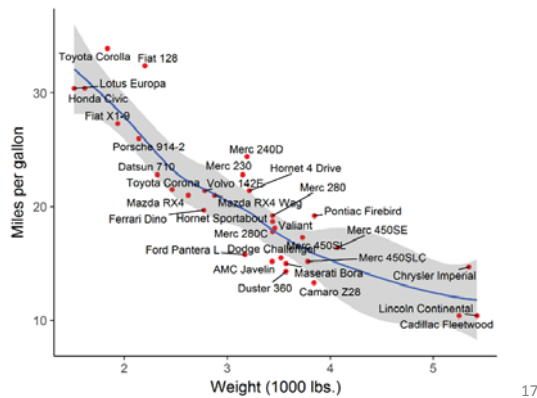
16

labeling points: geom_text_repel()

```
library(ggrepel)
plt2 +
  geom_text_repel(aes(label = rownames(mtcars)))
```

geom_text_repel() in the ggrepel package assigns repulsive forces among points and labels to assure no overlap

Some lines are drawn to make the assignment clearer

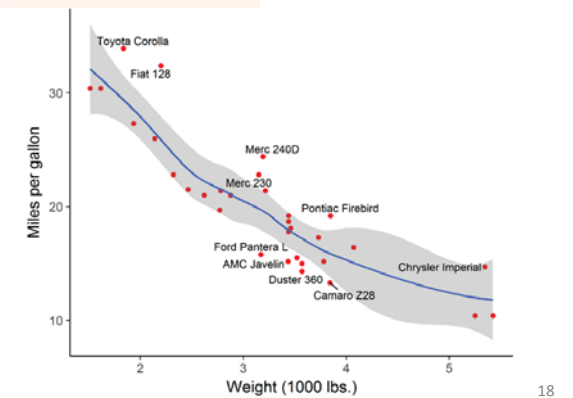


labeling points: selection

It is easy to label points selectively, using some criterion to assign labels to points

```
mod <- loess(mpg ~ wt, data=mtcars)
resids <- residuals(mod)
mtcars$label <- ifelse(abs(resids) > 2.5,
  rownames(mtcars), "")
plt2 + geom_text_repel(aes(label = mtcars$label))
```

- Here, I:
1. fit the smoothed loess curve,
 2. extract residuals, r_i
 3. assign labels where $|r_i| > 2.5$
 4. add the text layer

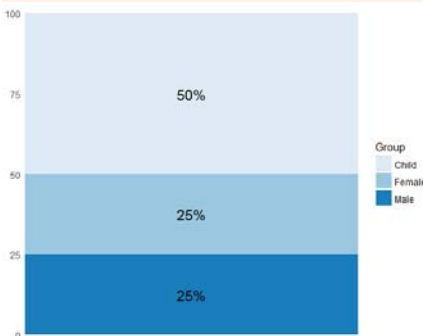


ggplot2: coords

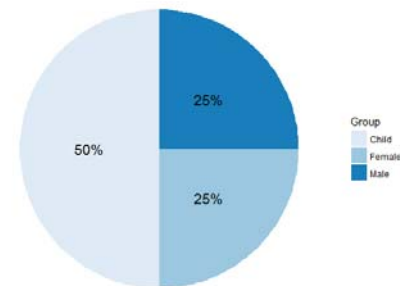
Coordinate systems, coord_*() functions, handle conversion from geometric objects to what you see on a 2D plot.

- A simple bar chart, standard coordinates
- A pie chart is just a bar chart in polar coordinates!

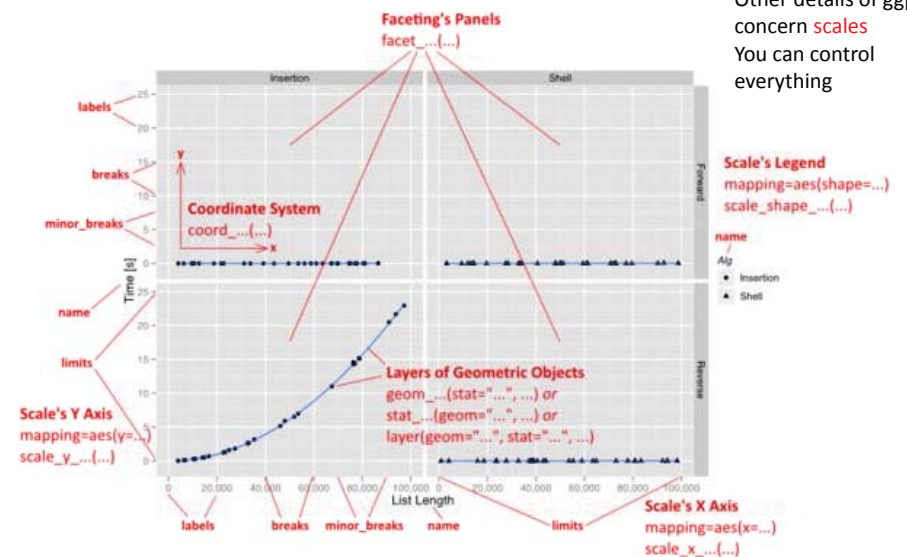
```
p <- ggplot(df, aes(x = "", y = value, fill = group)) +
  geom_bar(stat = "identity")
```



```
p + coord_polar("y", start = 0)
```



Anatomy of a ggplot



ggplot objects

Traditional R graphics just produce graphical output on a device
However, ggplot() produces a "ggplot" object, a list of elements

```
> names(plt)
[1] "data" "layers" "scales" "mapping" "theme" "coordinates"
[7] "facet" "plot_env" "labels"
> class(plt)
[1] "gg" "ggplot"
```

What methods are available?

```
> methods(class="gg")
[1] +
```

The "gg" class provides the "+" method

```
> methods(class="ggplot")
[1] grid.draw plot print summary
```

The "ggplot" class provides other, standard methods

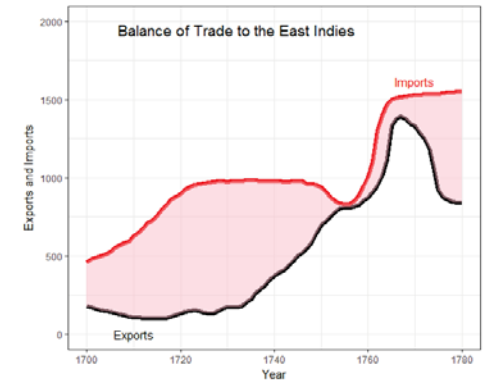
21

Playfair: Balance of trade charts

In the *Commercial and Political Atlas*, William Playfair used charts of imports and exports from England to its trading partners to ask "How are we doing"?

Here is a re-creation of one example, using ggplot2. How was it done?

```
> data(EastIndiesTrade, package="GDAdata")
> head(EastIndiesTrade)
  Year Exports Imports
1 1700   180    460
2 1701   170    480
3 1702   160    490
4 1703   150    500
5 1704   145    510
6 1705   140    525
...   ...   ...
```

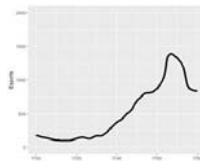


22

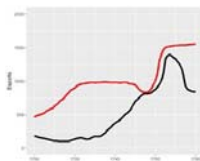
ggplot thinking

I want to plot two time series, & fill the area between them

Start with a line plot of Exports vs. Year: **geom_line()**
Add a layer for the line plot of Imports vs. Year

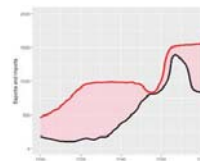


```
c1 <-
ggplot(EastIndiesTrade, aes(x=Year, y=Exports)) +
  ylim(0,2000) +
  geom_line(colour="black", size=2) +
  geom_line(aes(x=Year, y=Imports), colour="red", size=2)
```



Fill the area between the curves: **geom_ribbon()**
change the Y label

```
c1 <- c1 +
  geom_ribbon(aes(ymin=Exports, ymax=Imports), fill="pink") +
  ylab("Exports and Imports")
```



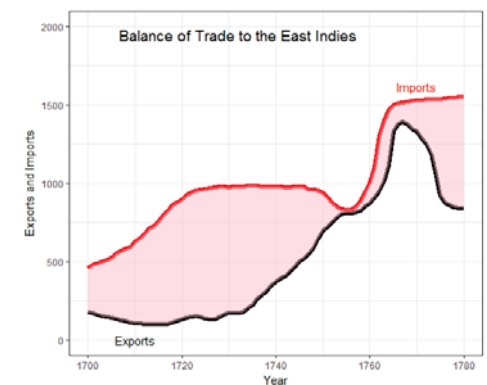
23

That looks pretty good. Add some text labels using `annotate()`

```
c1 <- c1 +
  annotate("text", x = 1710, y = 0, label = "Exports", size=4) +
  annotate("text", x = 1770, y = 1620, label = "Imports", color="red", size=4) +
  annotate("text", x = 1732, y = 1950, label = "Balance of Trade to the East Indies", color="black", size=5)
```

Finally, change the theme to b/w

```
c1 <- c1 + theme_bw()
```



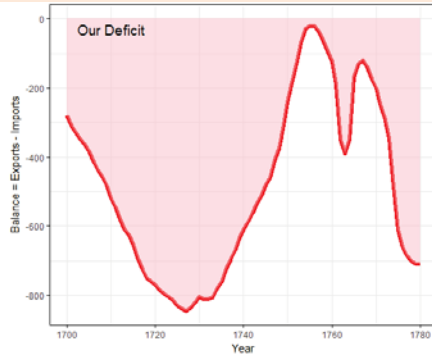
24

Plot what you want to show

Playfair's goal was to show the balance of trade with different countries.
Why not plot Exports – Imports directly?

```
c2 <-
ggplot(EastIndiesTrade, aes(x = Year, y = Exports - Imports)) +
  geom_line(colour="red", size=2) +
  ylab("Balance = Exports - Imports") +
  geom_ribbon(aes(ymin=Exports-Imports, ymax=0), fill="pink",alpha=0.5) +
  annotate("text", x = 1710, y = -30, label = "Our Deficit", color="black", size=5) +
  theme_bw()
```

aes(x=, y=) can use expressions
calculated from data variables



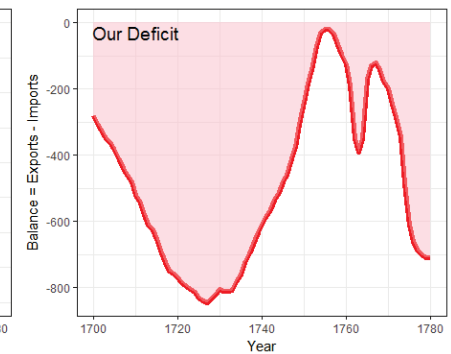
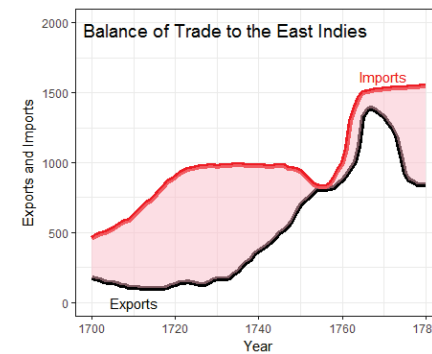
25

Composing several plots

ggplot objects use grid graphics for rendering

The `gridExtra` package has functions for combining or manipulating grid-based graphs

```
library(gridExtra)
grid.arrange(c1, c2, nrow=1)
```



26

Saving plots: ggsave()

- If the plot is on the screen

```
ggsave("path/filename.png") # height=, width=
```

- If you have a plot object

```
ggsave(myplot, file="path/filename.png")
```

- Specify size:

```
ggsave(myplot, "path/filename.png", width=6, height=4)
```

- any plot format (pdf, png, eps, svg, jpg, ...)

```
ggsave(myplot, file="path/filename.jpg")
```

```
ggsave(myplot, file="path/filename.pdf")
```

27

Building a custom graph

Custom graphs can be constructed by adding graphical elements (points, lines, text, arrows, etc.) to a basic ggplot()

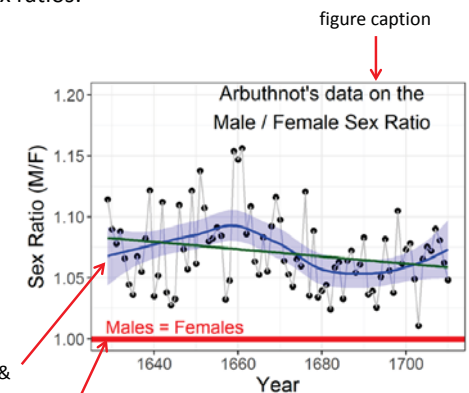
John Arbuthnot: data on male/female sex ratios:

```
> data(Arbuthnot, package="HistData")
> head(Arbuthnot[,c(1:3,6,7)])
Year Males Females Ratio Total
1 1629 5218 4683 1.114 9.901
2 1630 4858 4457 1.090 9.315
3 1631 4422 4102 1.078 8.524
4 1632 4994 4590 1.088 9.584
5 1633 5158 4839 1.066 9.997
6 1634 5035 4820 1.045 9.855
... ..
```

Arbuthnot didn't make a graph. He simply calculated the probability that in 81 years from 1629–1710, the sex ratio would **always** be > 1. The first significance test!

regression line &
loess smooth

reference line

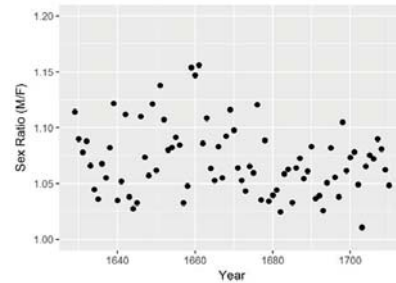


28

Building a custom graph

```
ggplot(Arbuthnot, aes(x=Year, y=Ratio)) +  
  ylim(1, 1.20) +  
  ylab("Sex Ratio (M/F)") +  
  geom_point(pch=16, size=2)
```

Start with a basic scatterplot,
Ratio vs. Year

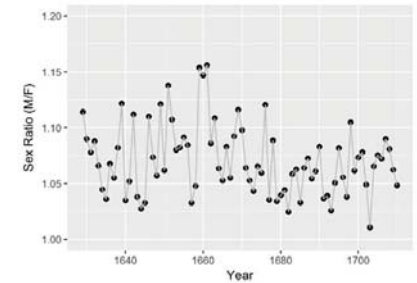


29

Building a custom graph

```
ggplot(Arbuthnot, aes(x=Year, y=Ratio)) +  
  ylim(1, 1.20) +  
  ylab("Sex Ratio (M/F)") +  
  geom_point(pch=16, size=2) +  
  geom_line(color="gray")
```

Connect points with a line



30

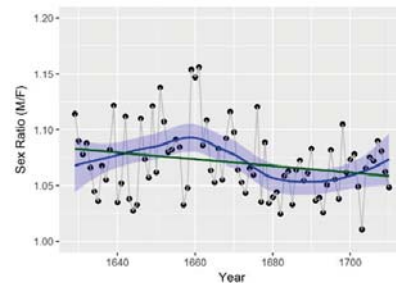
Building a custom graph

```
ggplot(Arbuthnot, aes(x=Year, y=Ratio)) +  
  ylim(1, 1.20) +  
  ylab("Sex Ratio (M/F)") +  
  geom_point(pch=16, size=2) +  
  geom_line(color="gray") +  
  geom_smooth(method="loess", color="blue",  
             fill="blue", alpha=0.2) +  
  geom_smooth(method="lm", color="darkgreen",  
             se=FALSE)
```

Add smooths:

- loess curve
- linear regression line

```
# save what we have so far  
arbuth <- last_plot()
```

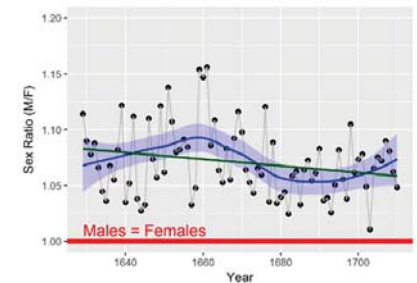


31

Building a custom graph

```
arbuth +  
  geom_hline(yintercept=1, color="red", size=2) +  
  annotate("text", x=1645, y=1.01, label="Males = Females", color="red", size=5)
```

Add horizontal reference line
& text label

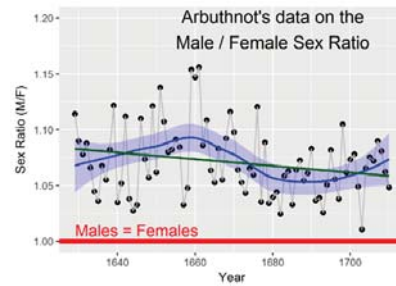


32

Building a custom graph

```
arbuth +  
  geom_hline(yintercept=1, color="red", size=2) +  
  annotate("text", x=1645, y=1.01, label="Males = Females", color="red", size=5) +  
  annotate("text", x=1680, y=1.19,  
         label="Arbutnot's data on the\nMale / Female Sex Ratio", size=5.5)
```

Add figure title



Building a custom graph

```
arbuth +  
  geom_hline(yintercept=1, color="red", size=2) +  
  annotate("text", x=1645, y=1.01, label="Males = Females", color="red", size=5) +  
  annotate("text", x=1680, y=1.19,  
         label="Arbutnot's data on the\nMale / Female Sex Ratio", size=5.5) +  
  theme_bw() + theme(text = element_text(size = 16))
```

Change the theme and font size

