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/
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/

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


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!

```r
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()`

goal of `ggplot2`: this should "just work"

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

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)*

**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,
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!)

```r
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).

**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")
  - themes: change graphic elements consistently

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

```r
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)
```

---

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.

```r
last_plot() + theme_bw()
```

---

ggplot2: facets

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

```r
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)
```

---

labeling points: geom_text()

Sometimes it is useful to label points to show their identities. `geom_text()` usually gives messy, overlapping text. Note the use of `theme_classic()` and better axis labels.

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

But this is still messy: wouldn’t want to publish this.
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.

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

```r
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.

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!

```r
p <- ggplot(df, aes(x = "", y = value, fill = group)) + geom_bar(stat = "identity")
p + coord_polar("y", start = 0)
```

Anatomy of a ggplot

Other details of ggplot concern scales.
You can control everything.
ggplot objects

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

```r
> names(plt)
[1] "data" "layers" "scales" "mapping" "theme" "coordinates"
> class(plt)
[1] "gg" "ggplot"
```

What methods are available?

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

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?

```r
> data(EastIndiesTrade, package="GDAdata")
> head(EastIndiesTrade)
```

<table>
<thead>
<tr>
<th>Year</th>
<th>Exports</th>
<th>Imports</th>
</tr>
</thead>
<tbody>
<tr>
<td>1700</td>
<td>180</td>
<td>460</td>
</tr>
<tr>
<td>1701</td>
<td>170</td>
<td>480</td>
</tr>
<tr>
<td>1702</td>
<td>160</td>
<td>490</td>
</tr>
<tr>
<td>1703</td>
<td>150</td>
<td>500</td>
</tr>
<tr>
<td>1704</td>
<td>145</td>
<td>510</td>
</tr>
<tr>
<td>1705</td>
<td>140</td>
<td>525</td>
</tr>
</tbody>
</table>

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:

```r
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()`

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

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

```r
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:

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

```r
library(EastIndiesTrade)
plot <- 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()
```

Composing several plots

ggplot objects use grid graphics for rendering. The `gridExtra` package has functions for combining or manipulating grid-based graphs:

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

Saving plots: `ggsave()`

- If the plot is on the screen
  ```r
ggsave("path/filename.png")  # height=, width=
```
- If you have a plot object
  ```r
ggsave(myplot, file="path/filename.png")
```
- Specify size:
  ```r
ggsave(myplot, "path/filename.png", width=6, height=4)
```
- any plot format (pdf, png, eps, svg, jpg, …)
  ```r
ggsave(myplot, file="path/filename.jpg")
ggsave(myplot, file="path/filename.pdf")
```

Building a custom graph

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

```r
library(HistData)
head(Arbuthnot[, c(1:3, 6, 7)])
```

John Arbuthnot: data on male/female sex ratios:

```r
# Hartung nicht gemacht, 61 Jahre nach 1629–1700, der prozentuale Anteil 
# von weiblichen Kids wäre also 58.7 %
```
Building a custom graph

```r
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

An R script for this example is available at:

Building a custom graph

```r
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

Building a custom graph

```r
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()
```

Add horizontal reference line & text label

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

Add figure title

```
Arbuthnot's data on the Male / Female Sex Ratio
```

Change the theme and font size

```
theme_bw() + theme(text = element_text(size = 16))
```