

L12 ggplot2 I

EPID 799B

Xiaojuan Li

Fall 2016

Announcement

Two lectures on ggplot2

- Today – architecture + basics
- Monday – advanced topics + tips & tricks

Announcement

- Updated course plan
- Homework 3 assigned, due Monday Oct 10th

Some background on ggplot2



Leland Wilkinson

Grammar of Graphics, 2005

- data visualization approach
- set of rules connecting data to graphics



Hadley Wickham

ggplot2

- excellent for “multi-layered” graphs
- graphs built up from a number of simple elements
- philosophy:
 - make graphics easier
 - use the grammar to facilitate research into new types of display

plyr and reshape2

- great packages for data manipulation
- dplyr and tidyr**

Resources

- Workshop ([link](#))
- ggplot2 documentation ([link](#)) – great reference site for code and examples
- Cookbook for R ([link](#)) – solution based recipes
- Quick-R ([link](#)) – simple examples
- ggplot2: Elegant Graphics for Data Analysis by Hadley Wickham – book on this subject
- ggplot2 mailing list ([link](#))
- Wiki ([link](#))
- Website ([link](#))
- StackOverflow ([link](#))

Components of a plot

- Data
- Geometric objects (geom)
- Statistical transformations (stat)
- Scales
- Coordinate system
- (+ Position adjustment, faceting)

} Layer = data + mapping +
geom + stat + position

The Basic idea of Grammar of Graphics

Independently specify plot building blocks and combine them to create just about any kind of graphic display you want.

Data to be plotted

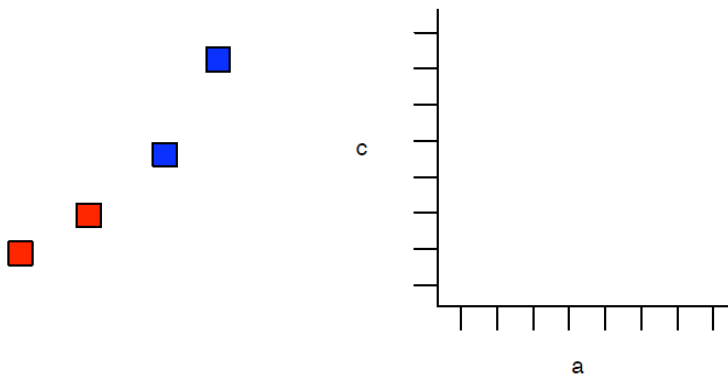
length	width	depth	trt
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b



x	y	colour
2	3	a
1	2	a
4	5	b
9	10	b

Wickham, ggplot2: An implementation of the grammar of graphics, 2007 Vanderbilt

Components of a plot



Geoms

Guides

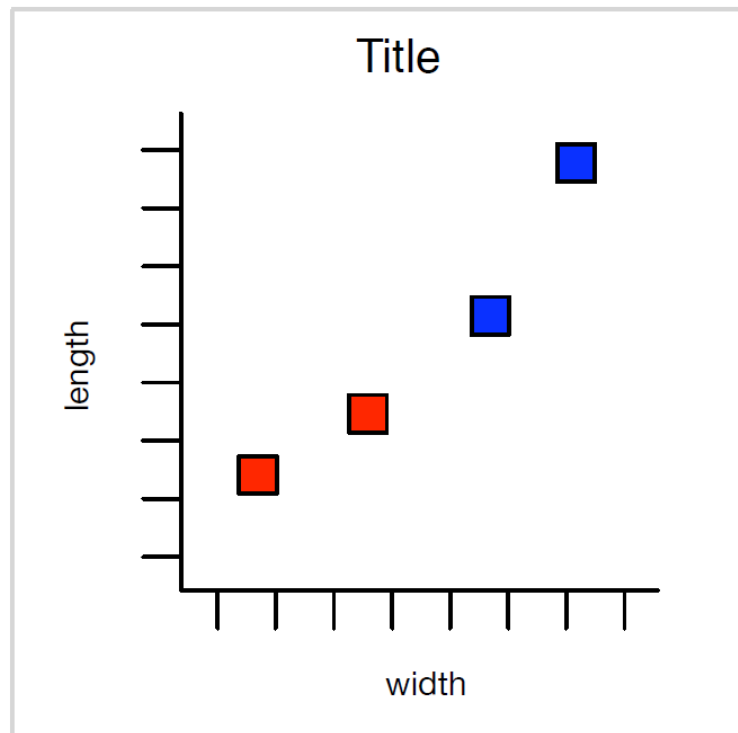
(from scales and
coordinate systems)



Plot

Wickham, ggplot2: An implementation of the grammar of graphics, 2007 Vanderbilt

Components of a plot



Wickham, ggplot2: An implementation of the grammar of graphics, 2007 Vanderbilt

Installation

```
install.packages("ggplot2")
```

do this once

```
library(ggplot2)
```

do this each time you need to use the package

note the 2 here

qplot() – for quick plot

The function can be used to create the most common graph types, though not exposing ggplot's full power.

form

```
qplot(x, y, data=, color=, shape=, size=, alpha=,  
      geom=, method=, formula=, facets=,  
      xlim=, ylim= xlab=, ylab=, main=, sub=)
```

qplot() – for quick plot

The function can be used to create the most common graph types, though not exposing ggplot's full power.

form

```
qplot(x, y, data=, color=, shape=, size=, alpha=,  
      geom=, method=, formula=, facets=,  
      xlim=, ylim=, xlab=, ylab=, main=, sub=)
```

Associates the levels
of variable with
color, shape, or size

qplot() – for quick plot

The function can be used to create the most common graph types, though not exposing ggplot's full power.

form

```
qplot(x, y, data=, color=, shape=, size=, alpha=,  
      geom=, method=, formula=, facets=,  
      xlim=, ylim=, xlab=, ylab=, main=, sub=)
```

Transparency for
overlapping
elements [0, 1]

qplot() – for quick plot

The function can be used to create the most common graph types, though not exposing ggplot's full power.

form

```
qplot(x, y, data=, color=, shape=, size=, alpha=,  
      geom=, method=, formula=, facets=,  
      xlim=, ylim=, xlab=, ylab=, main=, sub=)
```

Creates a trellis graph by specifying conditioning variable

qplot() – for quick plot

The function can be used to create the most common graph types, though not exposing ggplot's full power.

form

```
qplot(x, y, data=, color=, geom=, method=, formula=, facets=,  
      xlim=, ylim=, xlab=, ylab=, main=, sub=)
```

Specifies smoothing algorithm:
method for regression, formula
for the form of the fit

qplot() – for quick plot

The function can be used to create the most common graph types, though not exposing ggplot's full power.

form

Specifies the geometric
objects that define the
graph type

qplot(x, y, data, size=, alpha=,
geom=, method=, formula=, facets=,
xlim=, ylim=, xlab=, ylab=, main=, sub=)

qplot() – for quick plot

The function can be used to create the most common graph types, though not exposing ggplot's full power.

form

```
qplot(x, y, data=, color=, shape=, size=, alpha=,  
      geom=, method=, formula=, facets=,  
      xlim=, ylim= xlab=, ylab=, main=, sub=)
```

Modifying plot attributes:

`_lim`: range of axes

`_lab`: axis labels

Main: title

Sub: subtitle

ggplot

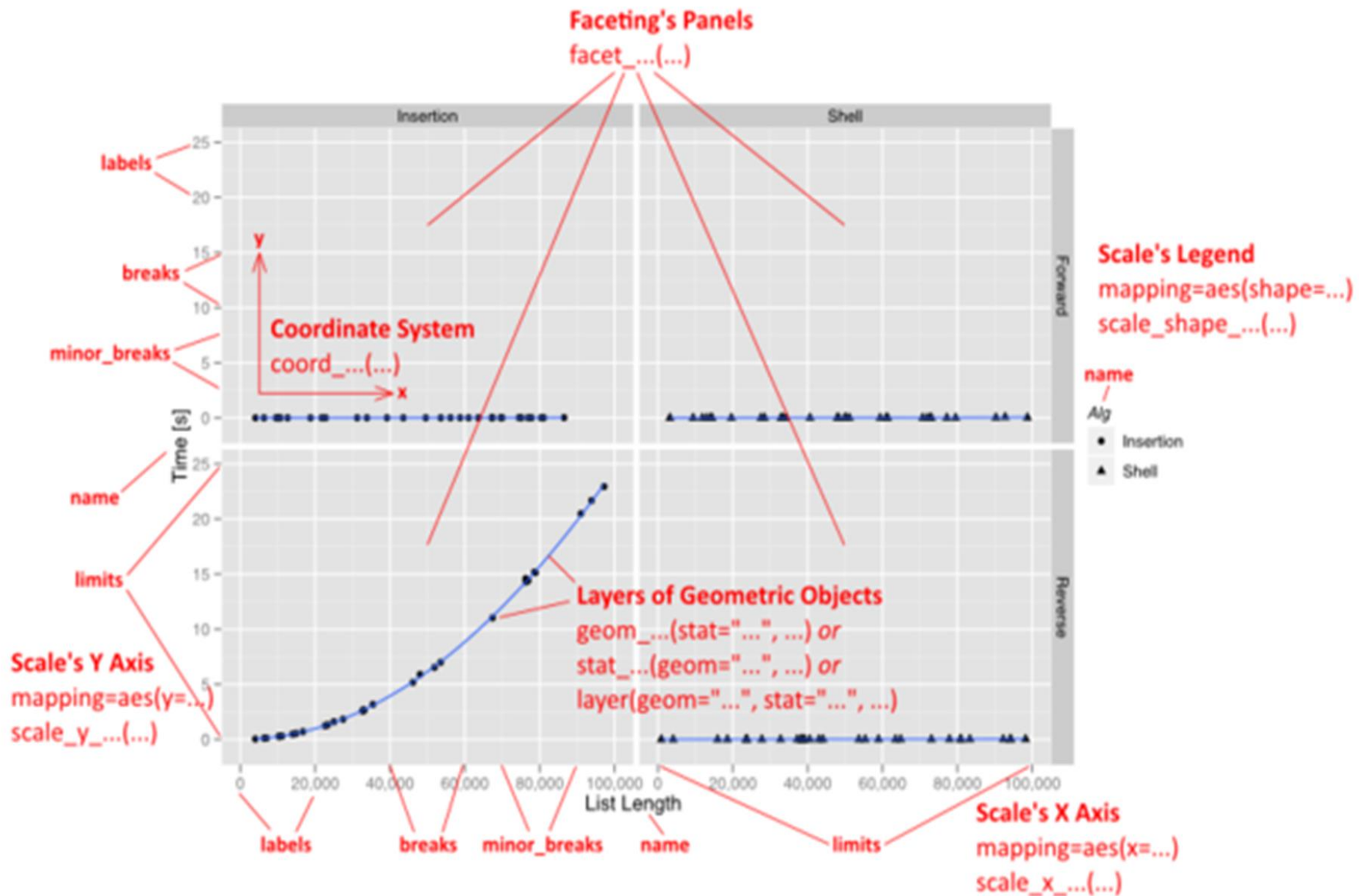
Full specification:

ggplot(data, mapping) +

```
  layer(stat = "",  
        geom= "",  
        position = "",  
        geom_params = list(),  
        stat_params = list(),  
        )
```

Usually won't write out the full specification, but use a shortcut:

```
ggplot() + geom_xxxx() +  
stat_summary() +  
scale_colour_brewer()
```



ggplot architecture – in one page

Aesthetics

- what are x, y?
- how do you want to represent additional variables?
 - can use color, shape, size and transparency

“geoms”

- do you want a scatter, line, bar, density, or other type plot?
- Can add statistics (e.g., regression line) and text

Scales

- for transforming (e.g., log, sq. root).
- also used to set legend – title, breaks, labels

Facets

- break the data down by factor and create separate plots for each

Stats, coordinates

Theme

- Adjust appearance: background, fonts, etc

ggplot architecture

Geoms – geometric objects, describe the type of plot (the basic shape of the elements on the plot) you will produce.

- e.g. `geom_point`, `geom_line`, `geom_histogram`

Statistics – statistical transformation of the data before plotting

- e.g. `stat_boxplot`, `stat_density`
- Some geoms are statistics in disguise:
 - `geom_histogram` = `geom_bar` + `stat_bin`
 - `geom_density` = `geom_ribbon` + `stat_density`

ggplot architecture –cont'd

Scales – controls the mapping between data and aesthetics, and control the display of the matching guide (axis or legend)

- e.g. `scale_x_log10` (plot on non-linear scales)
`scale_linetype` (scale for line patterns)

Faceting – display subsets of the dataset in different panels.

- e.g. `facet_grid` (lay out panels in a grid)

Coordinate systems – adjust the mapping from coordinates to the 2nd plane of the computer screen

- e.g. `coord_flip` (flipped Cartesian coordinates)

ggplot architecture –cont'd

Position adjustments – fine tune positioning of objects to achieve effects like dodging, jittering and stacking.

- e.g. `position_stack` (stack overlapping objects on top of one another)
`position_jitter` (jitter points to avoid overplotting)

Annotation – specialized functions for adding annotations to a plot

- e.g. `annotate`

Fortify – make it possible to use `ggplot2` with other types of objects, not just data frames.

ggplot architecture –cont'd

Themes – control non-data components of the plot


- e.g. `theme_bw` (a theme with white background and black gridlines)

Aesthetics – generate mappings that describe relationship how variables in the data are

- Any aesthetic can also be used as a parameter, which modify appearance of geoms and operation of statistics

Components of a plot

- Data
- Geometric object (geom)
- Statistical transformation (stat)
- Scales
- Coordinate system
- (+ Position adjustment, faceting)



Layer = data + mapping +
geom + stat + position

Let's plot!

Start with something basic

a simple density plot

```
ggplot(data=births1, aes(x=mage)) + geom_density(na.rm=TRUE)
```

a simple histogram plot

```
ggplot(births1, aes(mage)) + geom_histogram()
```

```
ggplot(births1, aes(mage)) + geom_histogram(fill="blue",  
na.rm=TRUE)
```

change aesthetics of the bars

```
ggplot(births1, aes(mage)) + geom_histogram(fill="blue",  
alpha=0.3, binwidth=0.5, na.rm=TRUE)
```

Aside: Bar and line graphs

x axis is	Height of bar represents	Common name
<i>Continuous</i>	<i>Count</i>	Histogram
<i>Discrete</i>	<i>Count</i>	Bar graph
<i>Continuous</i>	<i>Value</i>	Bar graph
<i>Discrete</i>	<i>Value</i>	Bar graph

Stratified by a group variable

```
# create separate densities for each cigdur group
ggplot(na.omit(births1), aes(mage, fill=cigdur)) +
  geom_density(alpha=0.5)
```

```
# separate histograms
ggplot(births1, aes(mage, fill=as.factor(sex))) +
  geom_histogram(alpha=0.5, binwidth=0.5, position = "dodge")
```

```
# a common error message
# Error in unit(tic_pos.c, "mm") : 'x' and 'units' must have length > 0
# Problem: fill= variable is not factor
# Solution: use as.factor() around the fill variable, e.g.
# fill=as.factor(sex)
```

Paneled plots

Faceting - Conditioning, trellising, and latticing

```
# paneled by another variable sex  
ggplot(na.omit(births1), aes(mage, fill=cigdur)) +  
  geom_histogram(alpha=0.8, binwidth=0.6, position = "dodge") +  
  facet_wrap(~sex)
```

Overlay plots

fitting a line to a scatterplot

```
a <- ggplot(births1, aes(mdif, wksgest)) + geom_point(color="blue",  
position="jitter")
```

```
a + geom_smooth(method="lm", col="red") +  
  ggtitle("Weeks gestation vs. month PNC began") +  
  xlab("month PNC began") +  
  ylab("gestational age at birth") +  
  theme_bw()
```

could save a plot object to a variable

Change shape, size and line type

```
library(splines)
library(MASS)
library(scales)
ggplot(births1, aes(mdif, wksgest, shape=as.factor(sex))) +
  geom_point(size=2.5, alpha=0.8, na.rm=TRUE, position="jitter") +
  scale_shape_manual(values=c(1,4)) +
  geom_smooth(aes(color=as.factor(sex), linetype=as.factor(sex)),
  method = "lm", formula = y ~ ns(x, 4), na.rm=TRUE) +
  scale_colour_brewer(palette="Set1") +
  scale_x_continuous(breaks=pretty_breaks()) +
  theme_bw()
```

more on shapes and line types

http://www.cookbook-r.com/Graphs/Shapes_and_line_types/

Add another layer

```
# scatter plot, a natural spline to the plot, grouped by sex, paneled
by cigdur
ggplot(births1, aes(mdif, wksgest, shape=as.factor(sex))) +
  geom_point(size=2.5, alpha=0.8, na.rm=TRUE, position="jitter") +
  scale_shape_manual(values=c(1,4)) +
  geom_smooth(aes(color=as.factor(sex), linetype=as.factor(sex)),
  method = "lm", formula = y ~ ns(x, 4), na.rm=TRUE) +
  scale_colour_brewer(palette="Set1") +
  scale_x_continuous(breaks=pretty_breaks()) +
  theme_bw() +
  facet_wrap(~cigdur)
```


Smoothing layer - stats

```
b<- ggplot(births1, aes(mdif, wksgest, shape=as.factor(sex))) +  
  geom_point(size=2.5, alpha=0.8, na.rm=TRUE, position="jitter") +  
  scale_shape_manual(values=c(1,4)) +  
  scale_colour_brewer(palette="Set1") +  
  scale_x_continuous(breaks=pretty_breaks())  
  
# adding a smoothing line to the scatter plot  
b + geom_smooth(aes(color=as.factor(sex)))  
  
# specifying method  
b + geom_smooth(aes(color=as.factor(sex), method = 'lm'))  
  
# changing method, add a natural spline to the plot  
b + geom_smooth(aes(color=as.factor(sex), linetype=as.factor(sex)),  
method = "lm", formula = y ~ ns(x, 4), na.rm=TRUE)
```

Change color

```
ggplot(births1, aes(mdif, wksgest, shape=as.factor(sex))) +  
  geom_point(size=2.5, alpha=0.8, na.rm=TRUE, position="jitter") +  
  scale_shape_manual(values=c(1,4)) +  
  geom_smooth(aes(color=as.factor(sex), linetype=as.factor(sex)),  
    method = "lm", formula = y ~ ns(x, 4), na.rm=TRUE) +  
  scale_x_continuous(breaks=pretty_breaks()) +  
  scale_colour_brewer(palette="Set1")
```

Change theme

```
# changing to a black & white theme  
ggplot(births1, aes(mdif, wksgest)) + geom_point(color="blue",  
  position="jitter") + geom_smooth(method="lm", col="red") +  
  theme_bw()
```

Annotation

```
# adding title, axis labels
ggplot(births1, aes(mdif, wksgest)) + geom_point(color="blue",
  position="jitter") + geom_smooth(method="lm", col="red") +
  ggtitle("Weeks gestation vs. month PNC began") +
  xlab("month PNC began") +
  ylab("gestational age at birth") +
  theme_bw()
```

Annotation – cont'd

```
# changing key labels for factor variable in the legend
ggplot(births1, aes(mdif, wksgest, shape=as.factor(sex))) +
  geom_point(size=2.5, alpha=0.8, na.rm=TRUE, position="jitter") +
  scale_shape_manual("Sex",
                    values=c(1,4), labels=c("Male", "Female")) +
  scale_x_continuous(breaks=pretty_breaks()) +
  ggtitle("Weeks gestation vs. month PNC began") +
  xlab("month PNC began") +
  ylab("gestational age at birth") +
  theme_bw()
```