

basic ggplot2

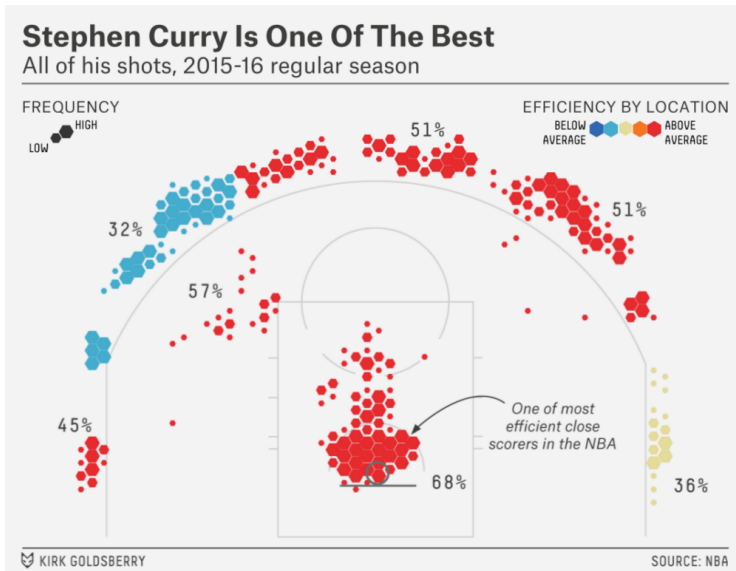
Zhenke Wu

Seminar Course: *Visualization for Individualized Health*
Johns Hopkins University

<http://zhenkewu.com>
Figures and R code from online

2016-02-23

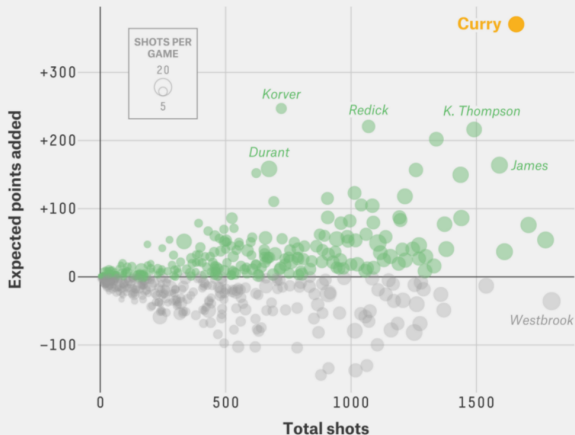
Statistical Graphic?



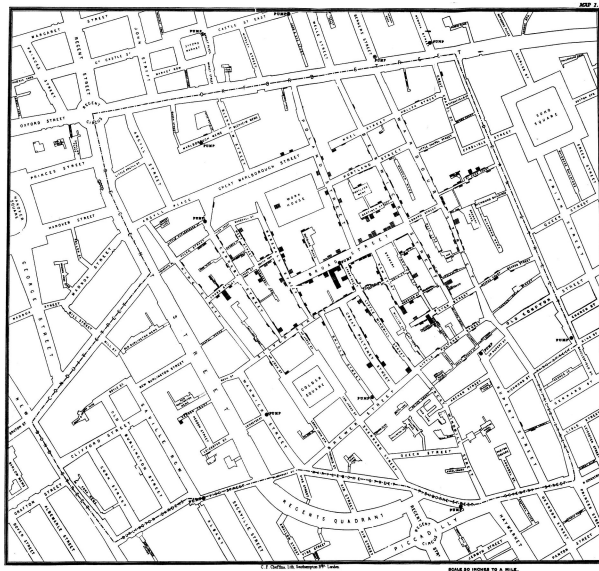
Statistical Graphic?

Curry Is The Most Valuable Shooter (By A Lot)

Shooting value added (based on distance, shot clock and defender distance) vs. shots, by player; last season through Nov. 28, 2015



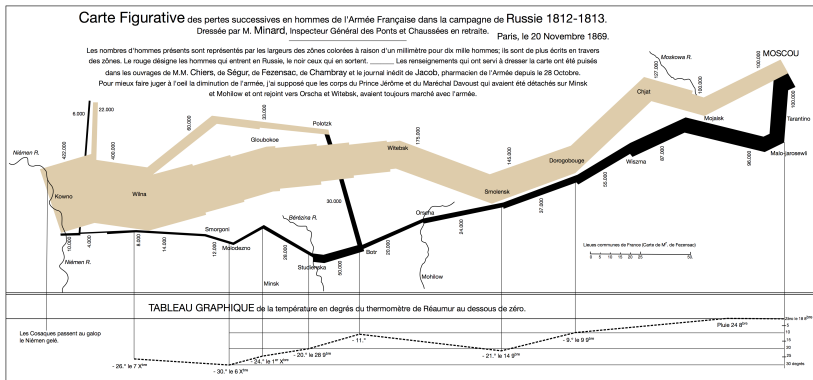
Statistical Graphic?



- John Snow's

Cholera map in dot style; dots represent deaths from cholera in London in 1854 to detect the source of the disease

Statistical Graphic?



Charles Joseph Minard (1869), *Napoleon's March to Moscow - The War of 1812*

Important questions for statistical graphics

- ▶ What is a graphic?
- ▶ How can we succinctly describe a graphic?
- ▶ How can we create the graphic that we have described?

One approach: develop a grammar!

- ▶ Grammar: *the fundamental principles or rules of an art or science* (Oxford English Dictionary; Item 6)
 - ▶ Allows us to gain insights into the composition of complicated graphics
 - ▶ Reveals unexpected connections for understanding a diverse range of graphics
 - ▶ Guides us to produce sensible and well-formed graphics
- ▶ Analogy to the English language: good grammar is just the first step in creating a good sentence.

Existing R graphics tools

- ▶ base graphics (Ross Ihaka)
 - ▶ pen on paper model; cannot modify or delete existing content
 - ▶ no representation of the graphics, apart from their appearance on the screen
 - ▶ fast but with limited scope
- ▶ grid (Paul Murrell, 2000)
 - ▶ a much richer system of graphical primitives (only primitives; no tools for producing statistical graphics)
 - ▶ graph objects represented independently of the plot and can be modified later
 - ▶ a system of viewports to lay out complex graphics
- ▶ lattice package (Deepayan Sarkar, 2008)
 - ▶ uses grid to implement the trellis graphics system of Cleveland
 - ▶ can easily produce conditioned plots and some details (e.g., legends) are automatically taken care of
 - ▶ lacks a formal model; hard to extend

ggplot2: a framework for producing statistical graphics

- ▶ takes the good things from base and lattice graphics
- ▶ uses a strong underlying model with several principles (details to follow)

What we get:

- ▶ a compact syntax to describe a wide range of graphics
- ▶ independent components that are easily extensible

ggplot2 Scatterplot Example: data

create a simple data set with 4 variables in the columns

```
dat0 <- as.data.frame(list(A = c(2,1,4,9),  
                          B = c(3,2,5,10),  
                          C = c(4,1,15,80),  
                          D = c("a", "a", "b", "b")))
```

dat0

```
##   A  B  C D  
## 1 2  3  4 a  
## 2 1  2  1 a  
## 3 4  5 15 b  
## 4 9 10 80 b
```

ggplot2 Scatterplot Example: *geom* aesthetics and mapping

- ▶ Scatterplot:
 - ▶ a point for each observation
 - ▶ position the point horizontally according to the value of A, vertically according to C
 - ▶ Here, we will also map categorical variable D to the shape of the points
- ▶ Aesthetics:
 - ▶ x-position: A
 - ▶ y-position: C
 - ▶ shape: D

```
##   x  y Shape
##  1  2  4     a
##  2  1  1     a
##  3  4 15     b
##  4  9 80     b
```

Example: mapping from data space to aesthetic space (controlled by **scale**)

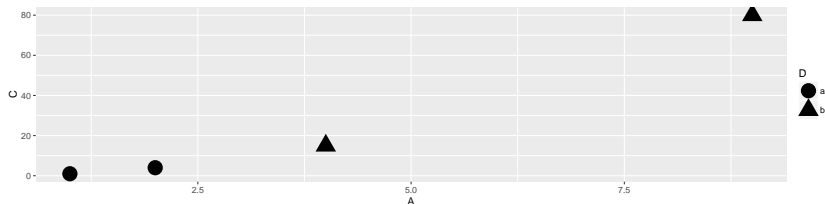
```
# ggplot2 creates (internally) a simple data set with  
# 3 variables in the columns:  
dat_internal <- as.data.frame(list(x = c(25,0,75,200),  
                                  y = c(11,20,53,300),  
                                  Shape = c("circle","circle","square",  
print(dat_internal)
```

```
##      x    y Shape  
## 1  25   11 circle  
## 2   0   20 circle  
## 3  75   53 square  
## 4 200  300 square
```

Example: Plot the Geometric objects (**geom**)

```
library(ggplot2)
example <- ggplot(data = dat0,
                 mapping = aes(x=A,y=C,shape=D))+
  # Set aesthetics to fixed value
  geom_point(size = 7)
```

example



```
# run `?geom_point` geom_point understands the
# following aesthetics (required aesthetics are in bold).
```

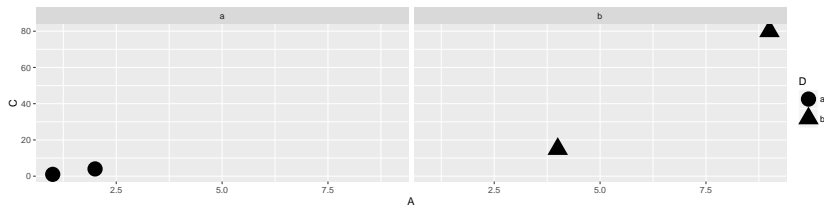
Details about geometric objects, or **geom**

- ▶ Controls the type of the plot you create (a point **geom** creates a scatterplot; a line **geom** creates a line plot, etc.)
 - ▶ 0d: point, text,
 - ▶ 1d: path, line (ordered path),
 - ▶ 2d: polygon, interval.
- ▶ Are abstract and can be rendered in different ways (e.g., intervals).
- ▶ Require outputs from a statistic (e.g., x,y-positions in scatterplot; edges in boxplots)
- ▶ **Every geom has a default statistic, and every statistic a default geom.**
 - ▶ For example, the bin **statistic** defaults to using the bar **geom** to produce a histogram.
- ▶ Each **geom** can only display certain aesthetics.
 - ▶ Try ?geom_point
 - ▶ Different parameterizations may be useful (e.g., polar coordinate system).

Example: Faceting (**facet**)

```
library(ggplot2)
example <- ggplot(data = dat0,
                  mapping = aes(x=A,y=C,shape=D))+
  # Set aesthetics to fixed value
  geom_point(size = 7)+facet_grid(~D)
```

example



What are the components in the previous example? layered grammar of graphics (Wickham, 2009)

- ▶ **data** and **mappings** (describe how variables in the data are mapped to aesthetic attributes that you can perceive)
- ▶ geometric objects (**geom**); what you actually see on the plot, e.g., points, lines, polygons, etc.
- ▶ statistical transformations, **stat**; summarize data in useful ways, e.g., binning and counting to create a histogram
- ▶ **scale**: maps values in the data space to values in an aesthetic space; **scale** draws a legend or axes to make it possible to read the original data values from the graph (**inverse mapping**: what does this mean?)
- ▶ A coordinate system: **coord**
- ▶ A **faceting** specification: describes how to break up data into subsets and how to display them as small multiples; also known as *conditioning* or *latticing/trellising*.

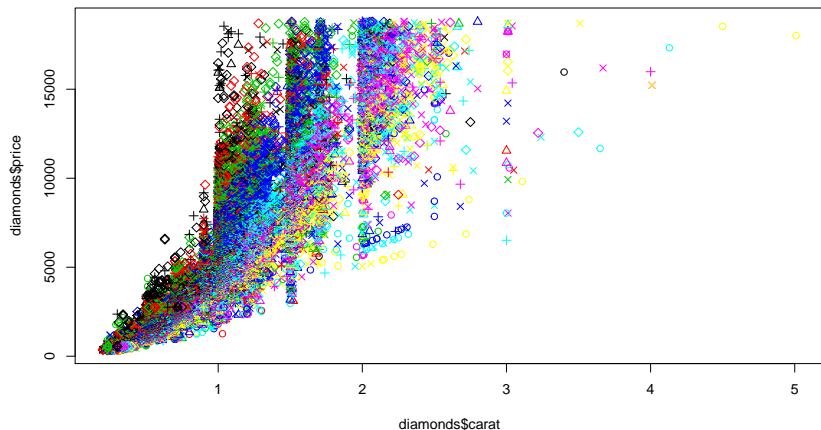
Diamond data

```
# just getting some data  
library(ggplot2)  
head(diamonds)
```

```
##   carat      cut color clarity depth table price     x  
## 1  0.23    Ideal     E    SI2   61.5    55   326 3.95 3  
## 2  0.21  Premium     E    SI1   59.8    61   326 3.89 3  
## 3  0.23     Good     E    VS1   56.9    65   327 4.05 4  
## 4  0.29  Premium     I    VS2   62.4    58   334 4.20 4  
## 5  0.31     Good     J    SI2   63.3    58   335 4.34 4  
## 6  0.24 Very Good     J   VVS2   62.8    57   336 3.94 3
```

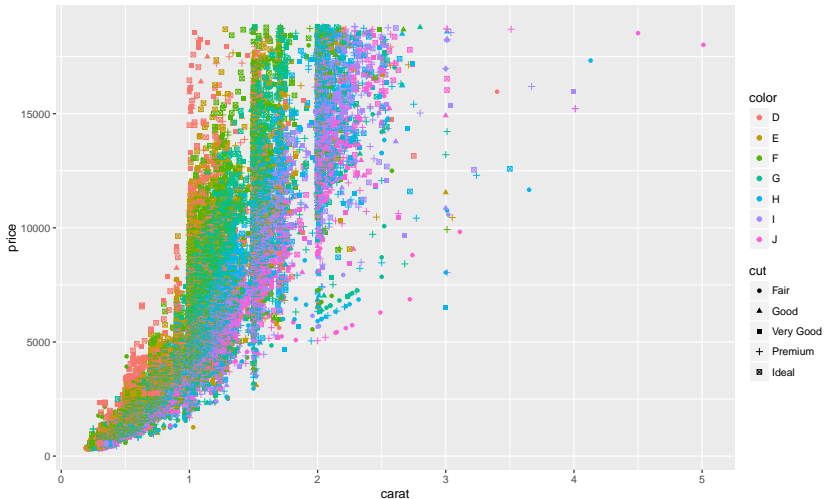

Diamond data plotted by base graphics

```
plot(diamonds$carat, diamonds$price, col = diamonds$color,  
     pch = as.numeric(diamonds$cut))
```



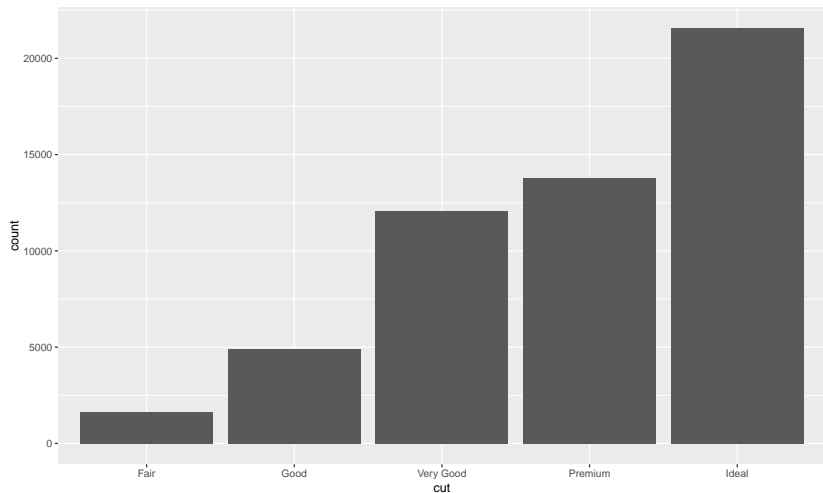
Diamond data plotted by ggplot2

```
ggplot(diamonds, aes(carat, price, col = color,  
                      shape = cut)) +  
  geom_point()
```



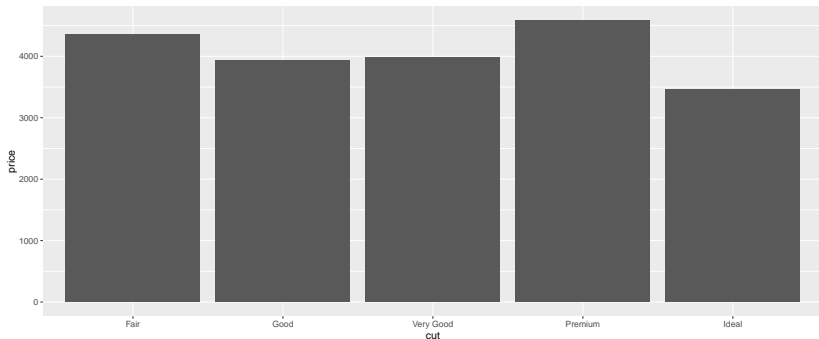
Diamond Example: count within each cut category

```
d <- ggplot(diamonds, aes(cut))  
d + geom_bar()
```



Diamond Example: average prices within each cut category

```
d + stat_summary_bin(aes(y = price), fun.y = "mean",  
                    geom = "bar")
```



Example: Napoleon's March to Moscow

```
library(HistData)
head(Minard.troops)
```

```
##   long  lat survivors direction group
## 1 24.0 54.9   340000          A      1
## 2 24.5 55.0   340000          A      1
## 3 25.5 54.5   340000          A      1
## 4 26.0 54.7   320000          A      1
## 5 27.0 54.8   300000          A      1
## 6 28.0 54.9   280000          A      1
```

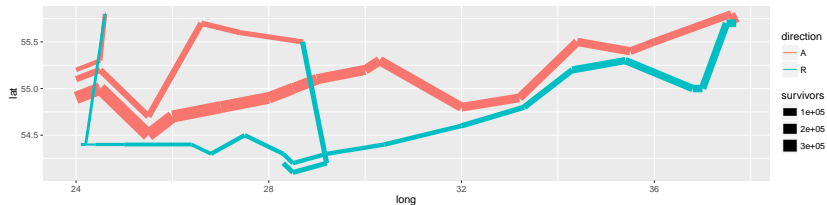
```
head(Minard.cities)
```

```
##   long  lat      city
## 1 24.0 55.0     Kowno
## 2 25.3 54.7     Wilna
## 3 26.4 54.4 Smorgoni
## 4 26.8 54.3 Moiodexno
```

Example: Napoleon's March to Moscow

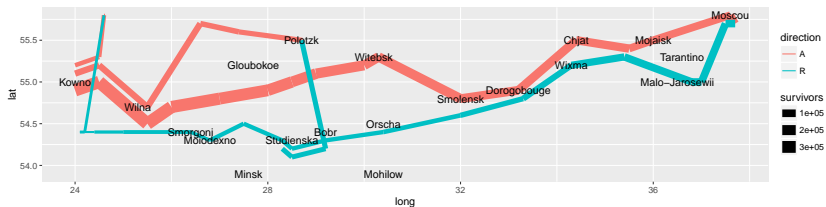
```
data(Minard.troops); data(Minard.cities)
library(ggplot2)
plot_troops <- ggplot(Minard.troops, aes(long, lat)) +
  geom_path(aes(size = survivors,
               colour = direction, group = group))
```

```
plot_troops
```



Example: Napoleon's March to Moscow

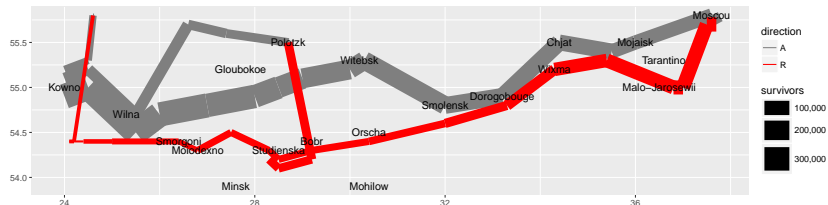
```
plot_both <- plot_troops +  
  geom_text(aes(label = city), size = 4,  
            data = Minard.cities)  
plot_both
```



Example: Napoleon's March to Moscow

```
plot_polished <- plot_both +  
  scale_size(range = c(1, 12),  
            breaks = c(1, 2, 3) * 10^5,  
            labels = scales::comma(c(1, 2, 3) * 10^5)) +  
  scale_colour_manual(values = c("grey50", "red")) +  
  xlab(NULL) +  
  ylab(NULL)
```

plot_polished



What grammar of graphics doesn't do

- ▶ It doesn't suggest what graphics you should use to answer the questions you are interested in.
 - ▶ `ggplot2` focuses on how to produce the plots you want, not knowing what plots to produce.
- ▶ Grammar doesn't specify what a graphic should look like and how to make a plot attractive.
 - ▶ Finer details, e.g., font size, background color are not specified by the grammar.
 - ▶ `ggplot2` uses its theming system
- ▶ No real-time interaction; other dynamic and interactive graphics packages exist:
 - ▶ `rCharts`: <http://rcharts.io/>
 - ▶ `clickme`: <https://github.com/nachocab/clickme>
 - ▶ `D3: Data-Driven Documents`: <https://d3js.org/>

“Ins and Outs”

- ▶ Data manipulation (get your data into the form required by `ggplot2`). You will shortly encounter at least these two R packages written by the same author of `ggplot2`:
 - ▶ `reshape2`
 - ▶ `plyr`
- ▶ Make figures publishable
 - ▶ comprehensive theming system in `ggplot2`

References

- ▶ Wickham H(2009). A Layered Grammar of Graphics. *Journal of Computational and Graphical Statistics*

Optional:

- ▶ Examples of statistical graphics used in sport analytics:
 - ▶ Stephen Curry's Bombs Are Too Good To Be True - I mean, they have to be, right? (FiveThirtyEight.com, 2015)
 - ▶ Lionel Messi Is Impossible (FiveThirtyEight.com, 2014)
- ▶ Notes by Wickham himself:
 - ▶ ggplot2 short courses by Wickham:
<http://courses.had.co.nz/11-rice/>
 - ▶ ggplot2 cheatsheet
- ▶ A book-length introduction:
 - ▶ Wickham (2010) ggplot2: *Elegant Graphics for Data Analysis (Use R!)*