The Grammar of Graphics: An Introduction to ggplot2

Jefferson Davis

Research Analytics

Historical background

- In 1970s John Chambers, Rick Becker, and Allan Wilks develop S and S+ at Bell Labs
- Bell System monopoly was broken up in 1982
- Late 80s some attempt to commercialize S/S+ but already too many non-commercial implementations
- Ross Ihaka and Robert Gentleman produce R in early 1990s
- Main developement now by Comprehensive R Archive Network (CRAN)

R: some useful tidbits

- Use left arrow for variable assignment

```
x <- 5
```

- Can concatenate with c()

```
c(4,5)
[1] 4 5
c("baby","elephant”)
[1] "baby"     "elephant"
```

- Download packages with the install() command

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

- Make packages available with library()

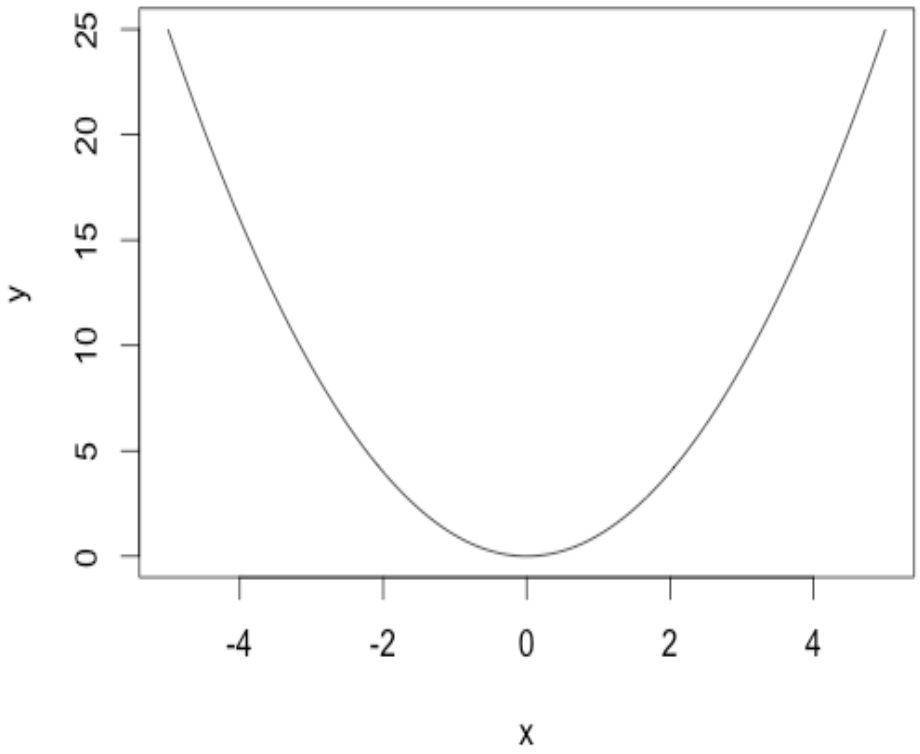
```
library(ggplot2)
library(ggthemes)
```

- Get help with ?

```
? ggplot
```

Statistical Graphics

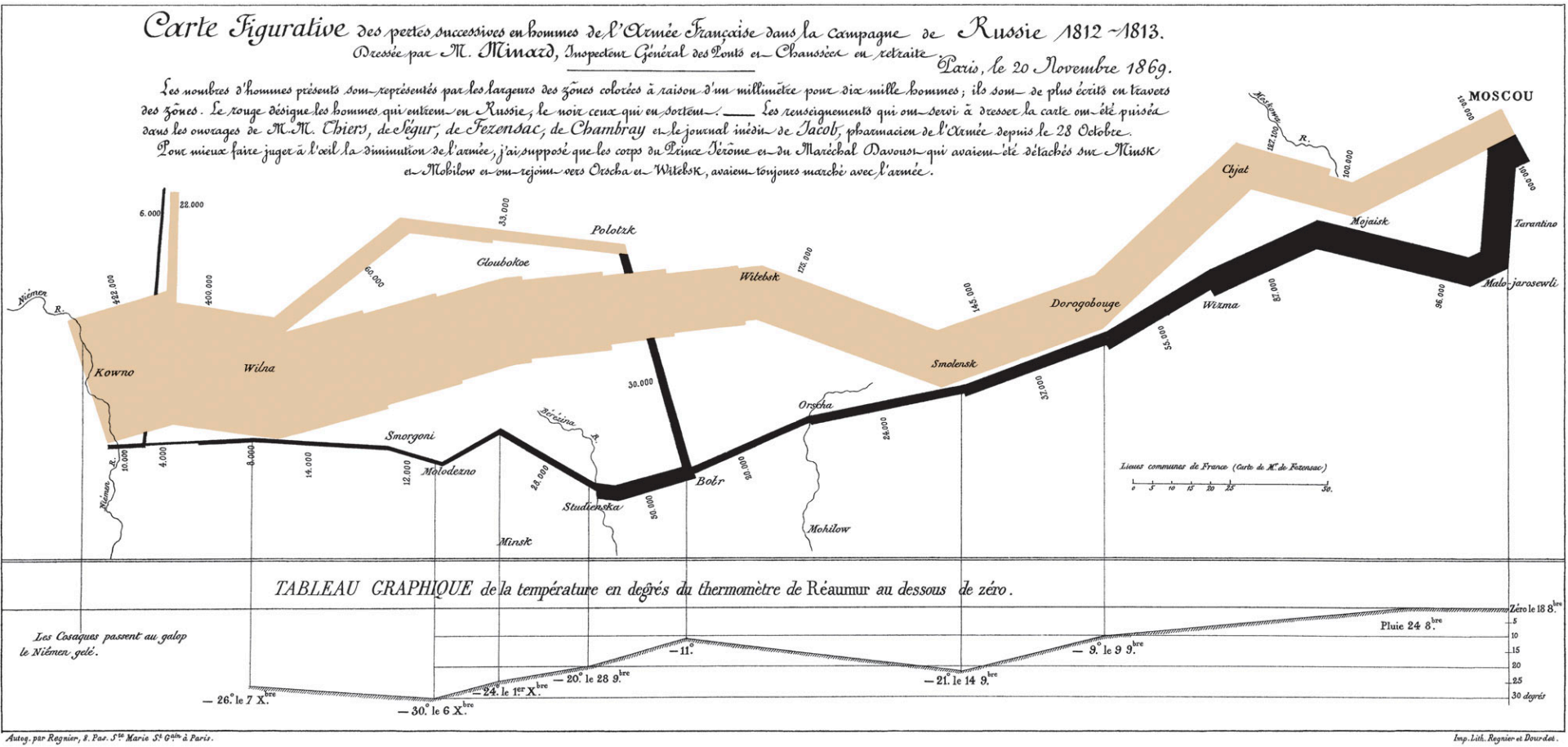
x	$y=x^2$
0.0	0.00
0.1	0.01
0.2	0.04
0.3	0.09
0.4	0.16
0.5	0.25
0.6	0.36
0.7	0.49
0.8	0.64
0.9	0.81
1.0	1.00
1.1	1.21
1.2	1.44
1.3	1.69



Statistical Graphics

Graphics can convey meaning without displaying any particular quantatative data.

Statistical Graphics



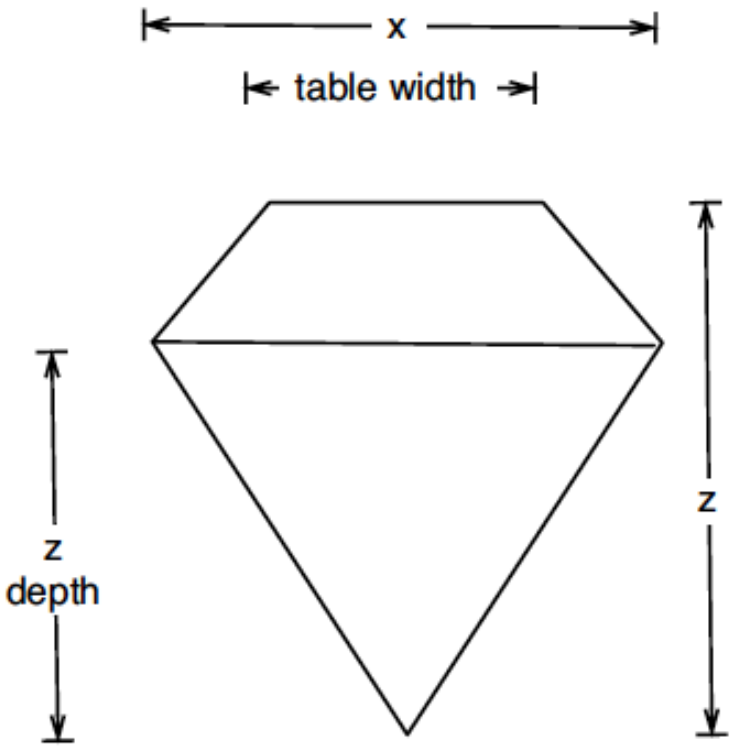
Statistical Graphics

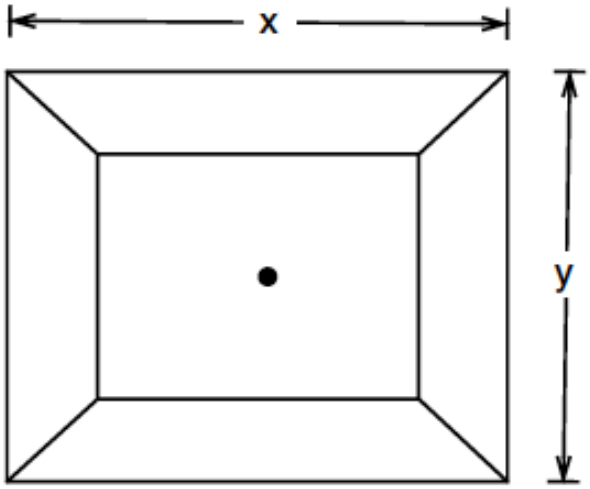
The dataset

The diamond dataset

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

The dataset





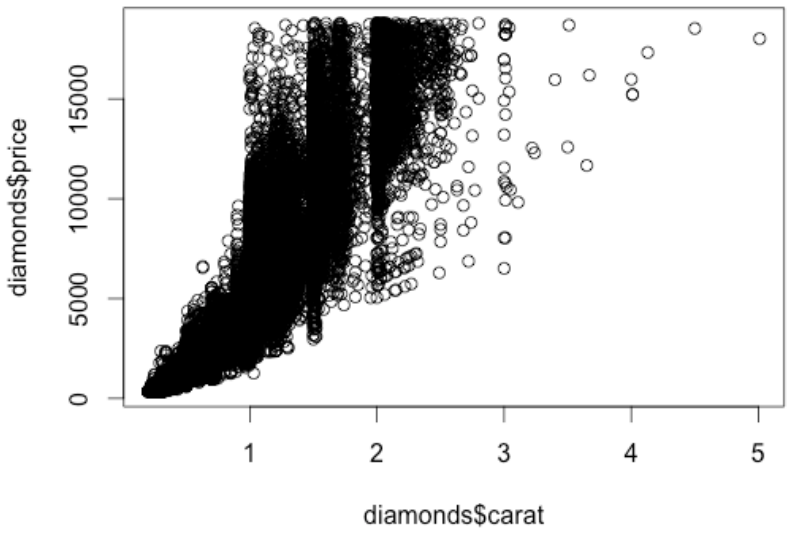
depth = z depth / z * 100
table = table width / x * 100

The dataset

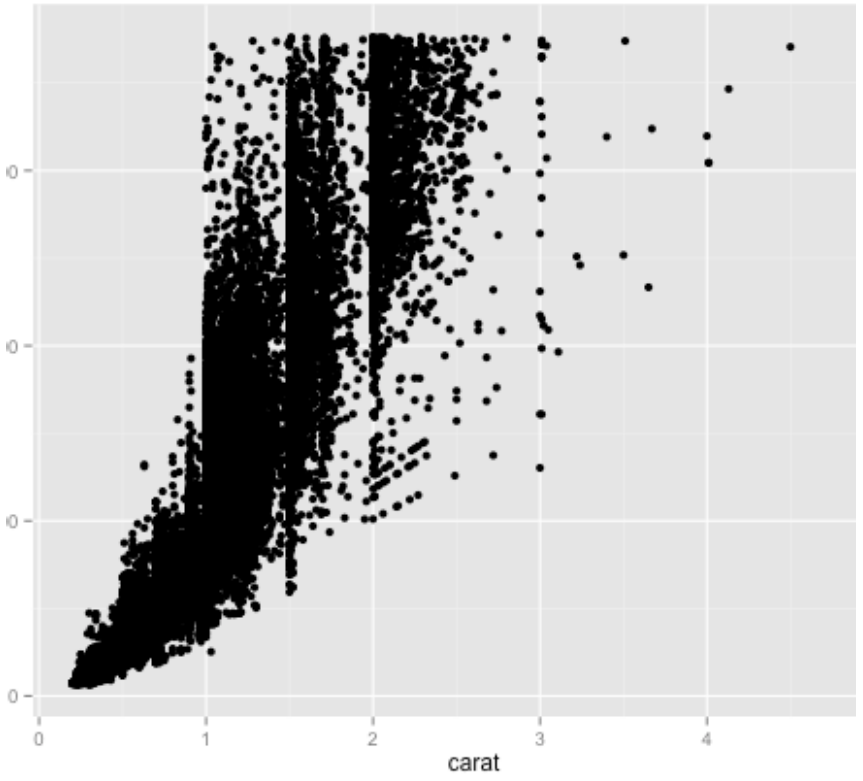
```
library(ggplot2)
head(diamonds)[,1:4]
# A tibble: 6 × 4
  carat       cut color clarity
  <dbl>     <ord> <ord>   <ord>
1  0.23     Ideal     E     SI2
2  0.21   Premium     E     SI1
3  0.23      Good     E     VS1
4  0.29   Premium     I     VS2
5  0.31      Good     J     SI2
6  0.24 Very Good     J    VVS2

#If working with diamonds and your machine is slow
#diamonds <- diamonds[seq(1,nrow(diamonds),10), ]
```

qplot()

```
plot(diamonds$carat,
     diamonds$price)
```

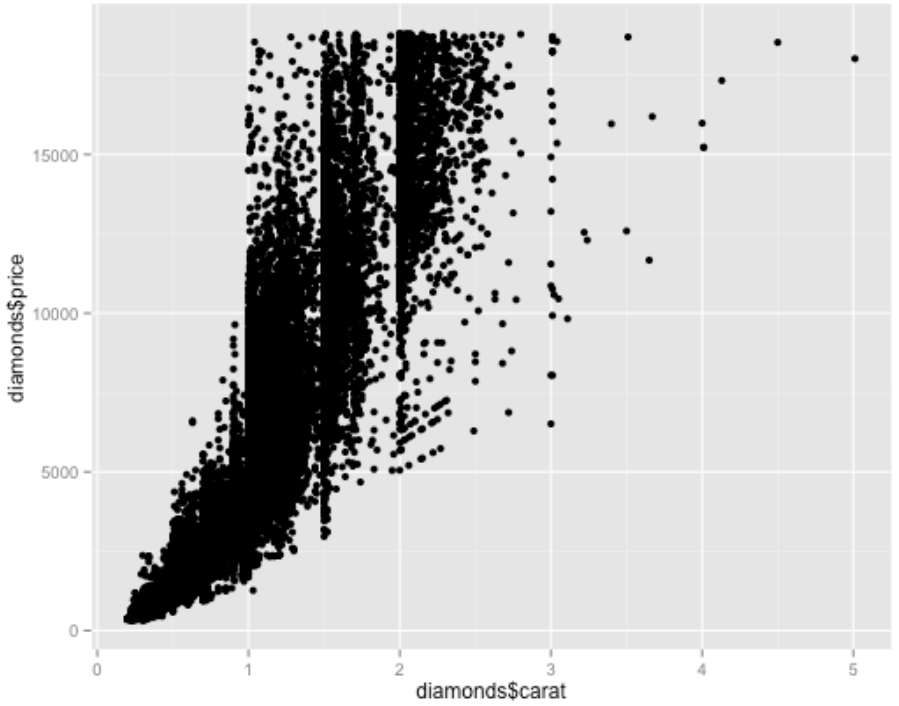



```
qplot(carat, price,
      data = diamonds)
```



qplot()

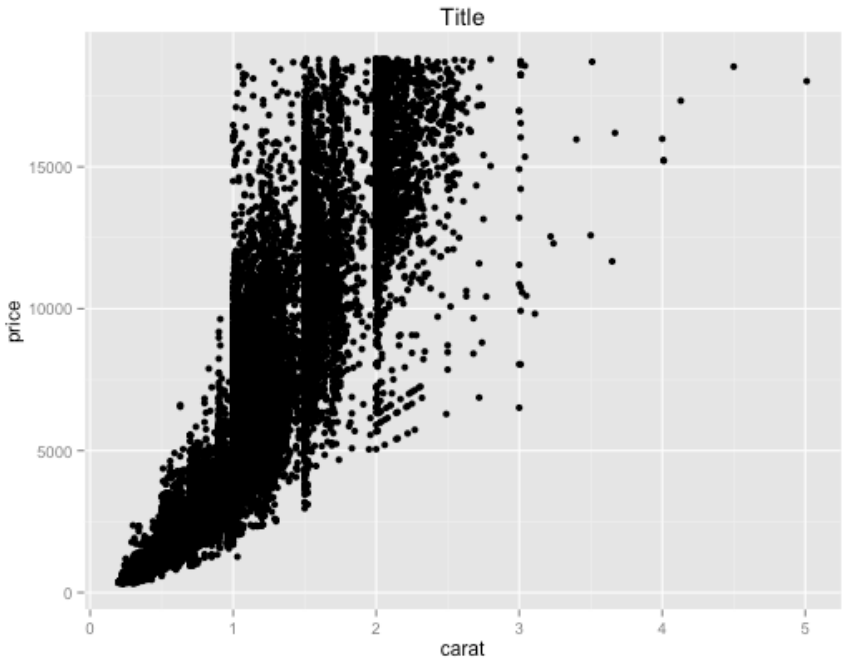
```
qplot(carat,
price,
data=diamonds)
```



qplot()

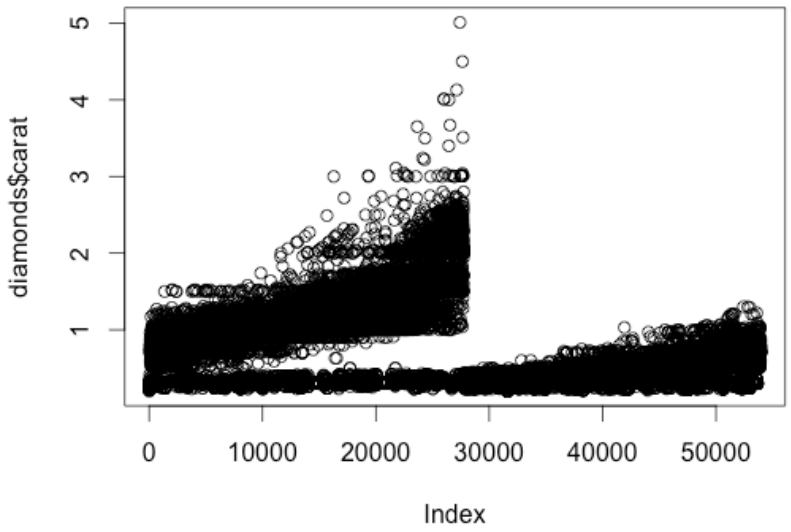
If we add a title we will redraw

```
qplot(carat,
price,
data=diamonds) +
labs(title = "Title")
```

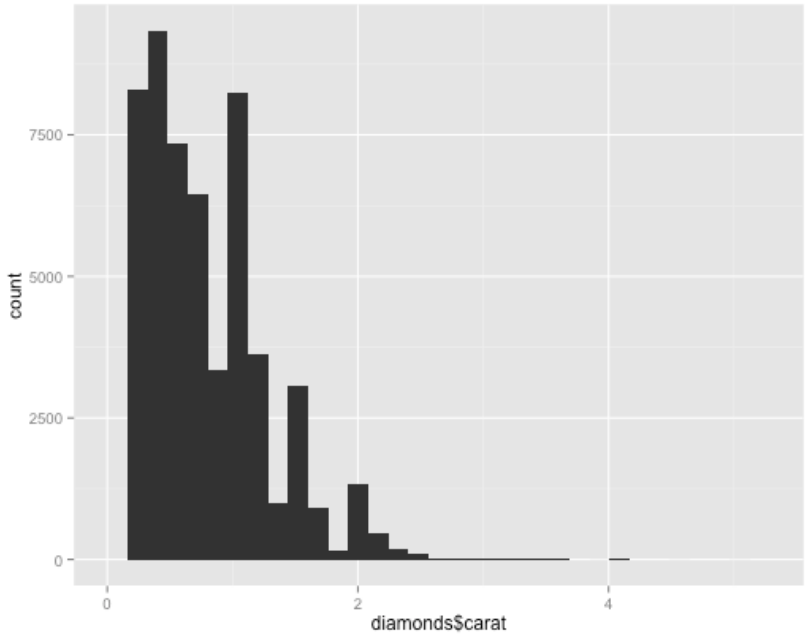


qplot()

```
plot(diamonds$carat)
```

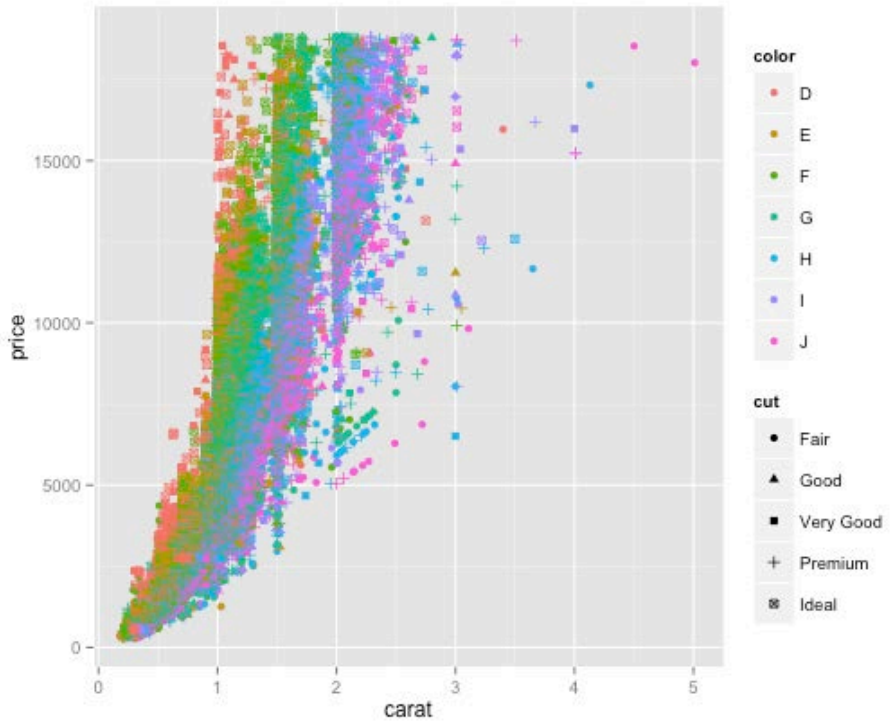



```
qplot(diamonds$carat)
```



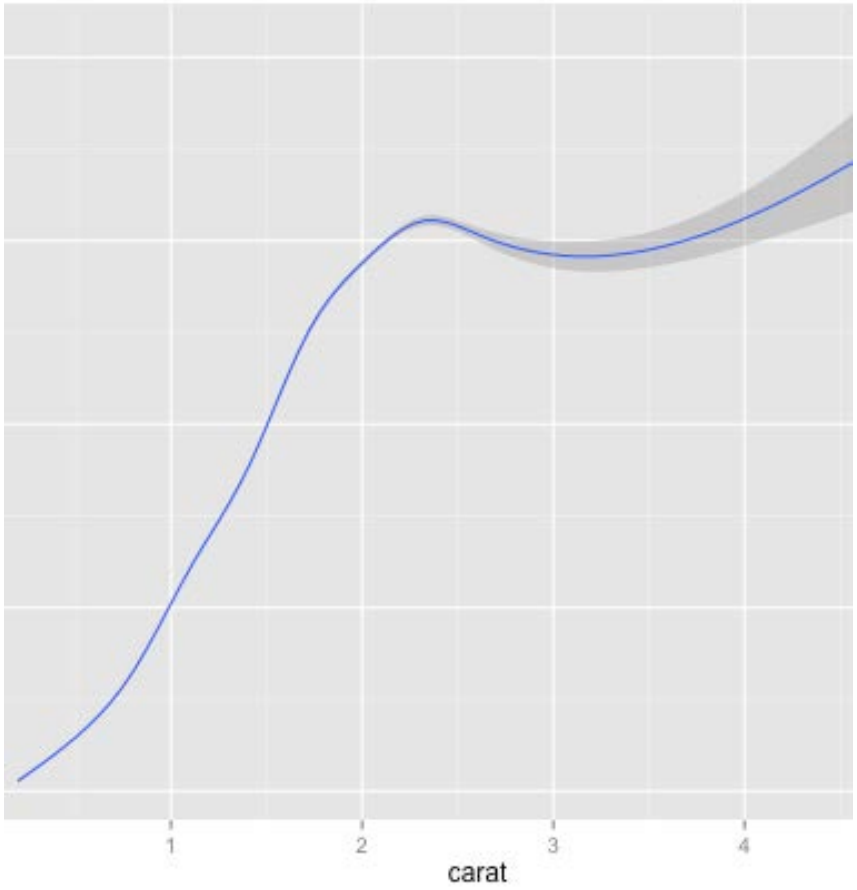
qplot()

```
qplot(carat, price,
data = diamonds,
shape = cut,
color = color)
```



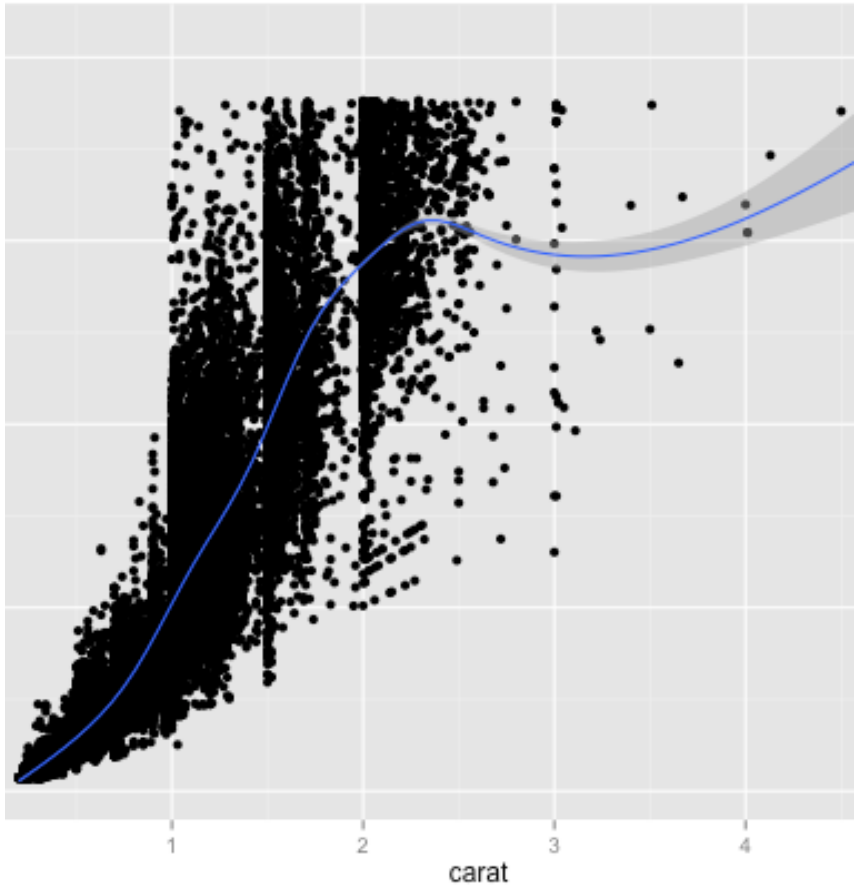
qplot()

```
qplot(carat, price,
data = diamonds,
geom = c("smooth"))
```



Qplot

```
qplot(carat, price,
data = diamonds,
geom =
c("point", "smooth"))

qplot(carat, price,
data = diamonds) +
geom_smooth()
```



Basic ggplot2

Any layer will have the following:

- Data
- Aesthetic mappings e.g. diamond cut-> shape
- Geometrical objects. Points, bars, polygons, etc.
- Scales
- Coordinate system
- Faceting system

The ggplot2 syntax

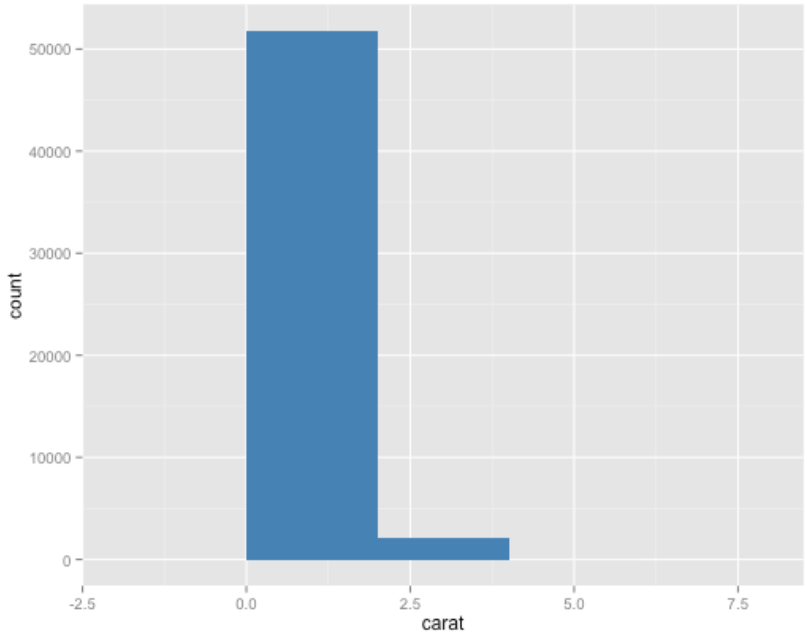
We'll start by assigning a ggplot to a variable

```
p <- ggplot(data = diamonds, aes(x = carat))
```

```
p <- ggplot(diamonds, aes(carat))
```

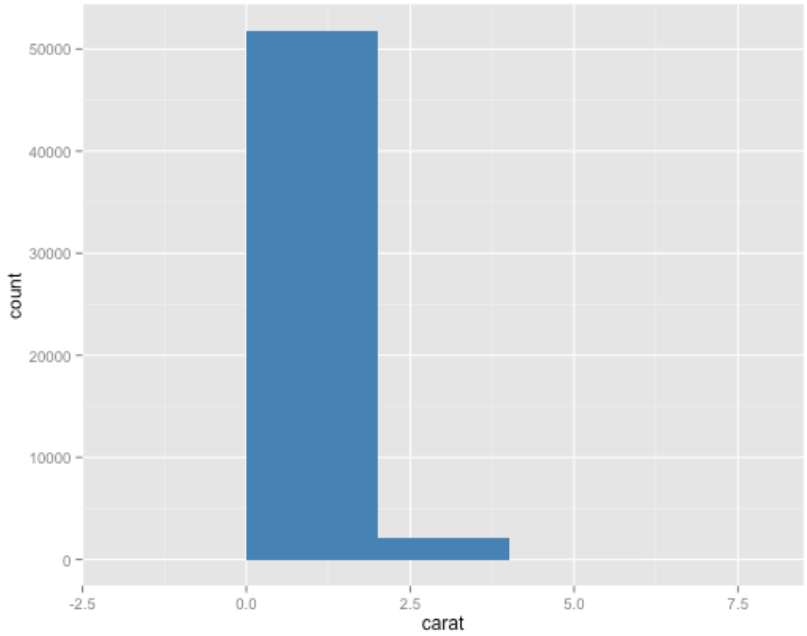
The ggplot2 syntax

```
p +
layer(geom = "bar",
stat = "bin",
position = "fill",
params =
list(fill = "steelblue",
binwidth = 2))
```



The ggplot2 syntax

```
p +
geom_bar(
  fill = "steelblue",
  binwidth = 2)
```



The ggplot2 syntax

geom	default stat	Aesthetics (required in **bold**)
blank	identity	no parameters
text	identity	**x, y, label**, size, color, alpha, hjust, vjust, parse
point	identity	**x, y**, size, shape, color, fill, alpha, na.rm
bar	bin	**x**, y, size, linetype, color, fill, alpha

Scales

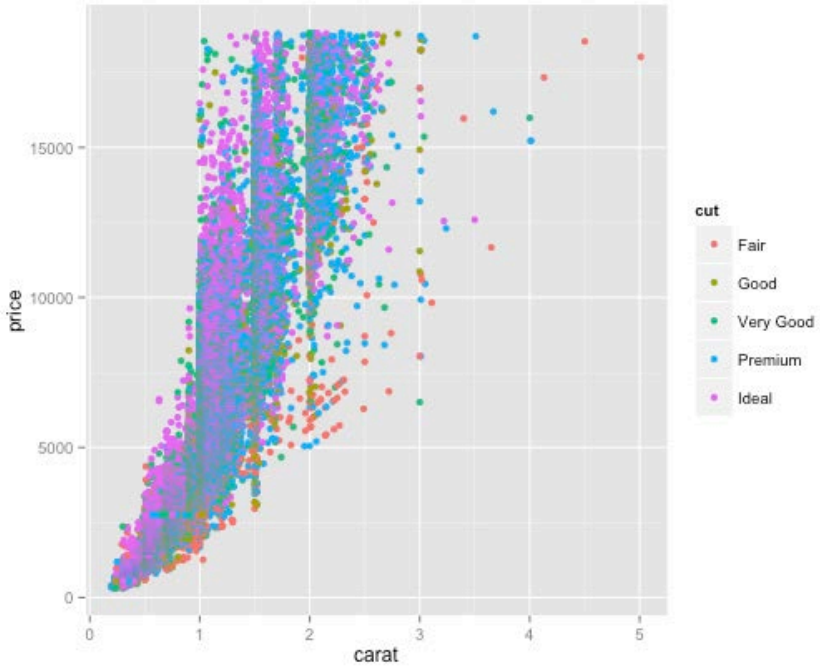
I’ve mentioned aesthetic mappings but I’ve been a bit loosey goosey with how the mappings work.

In ggplot2 scales control the aesthetic mapping.

Wilkinson distinguishes between scales and guides, but for us they are as one.

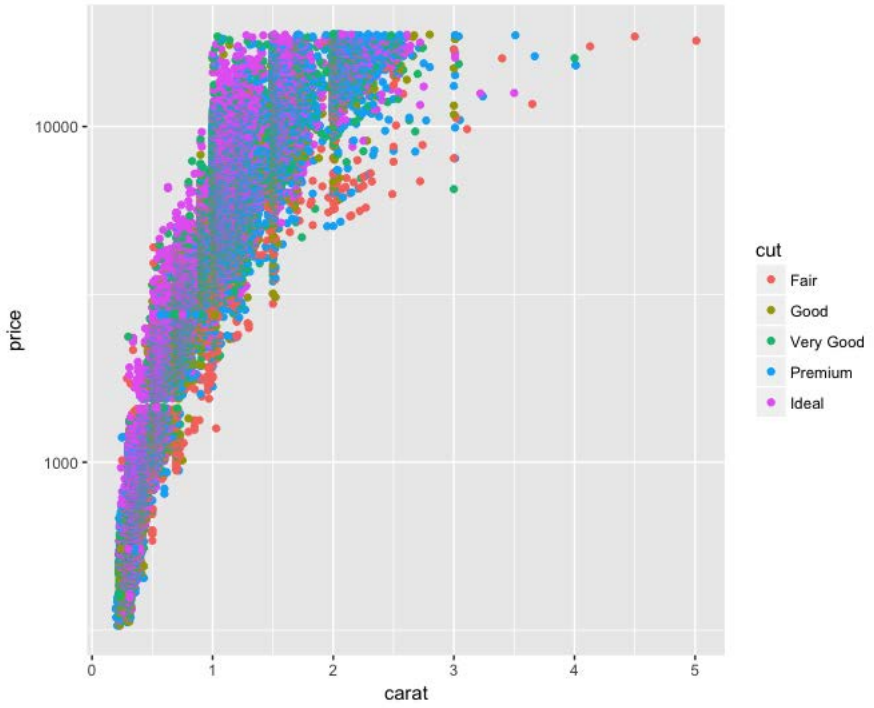
Scales

```
p <- ggplot(diamonds, aes(carat, price,
     color = cut))
   + geom_point()
p
```



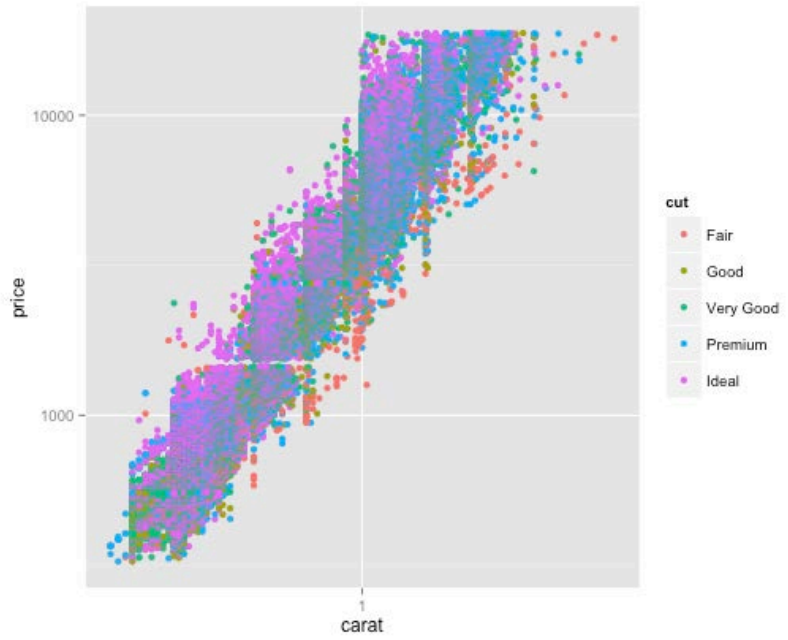
Scales

```
p + scale_y_log10()
```



Scales

```
p + scale_x_log10() + scale_y_log10()
```



Scales

We should consider color as well. The color space in ggplot is hcl. In this space we have the following

Hue: an angle between 0 and 360 for color

Chroma: the “purity” of the color. At 0 you have grey. The maximum depends on luminance

Luminance: brightness. Black at 0

and white at 1.

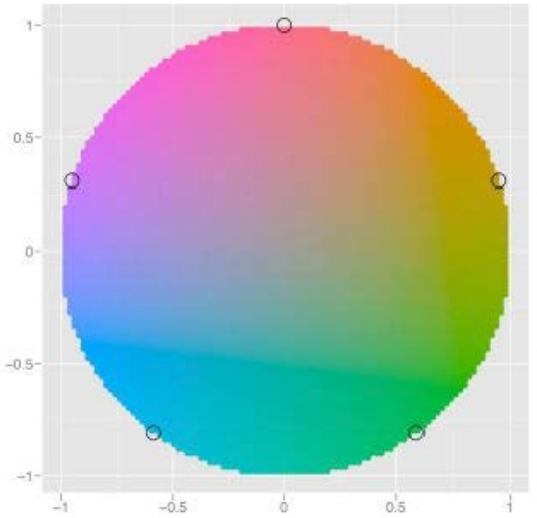
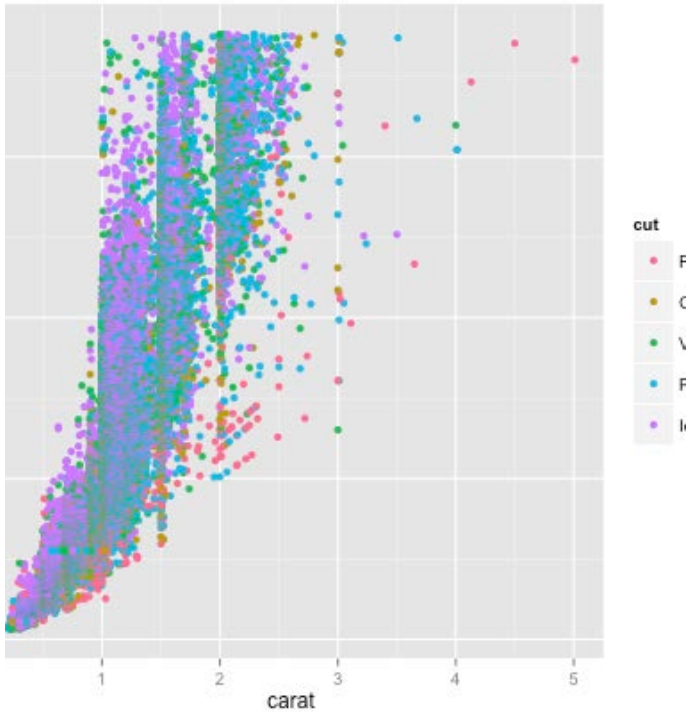


Fig. 3.4: A colour wheel illustrating the choice of five equally spaced colours. This is the default scale for discrete variables.

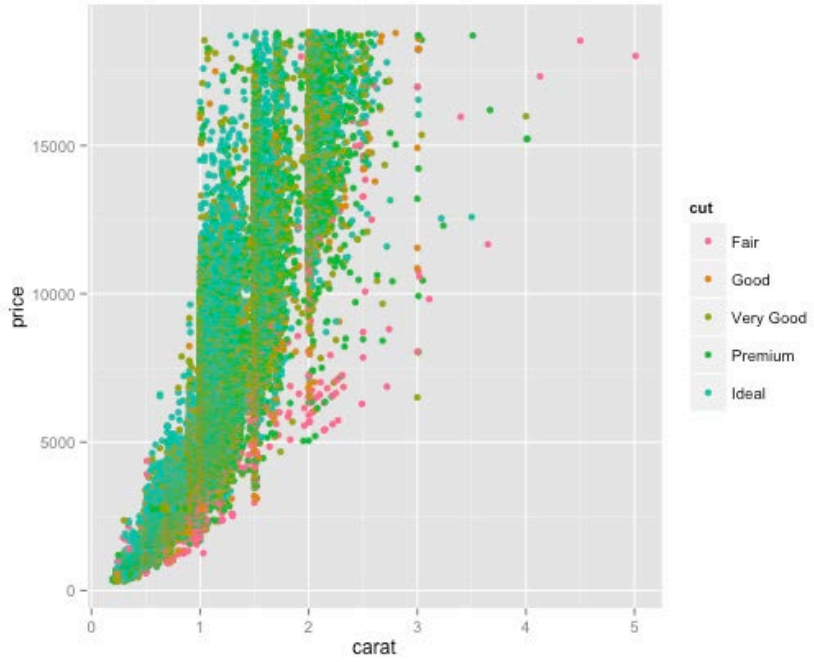
Scales

```
p + scale_color_discrete(h=c(0,360))
```



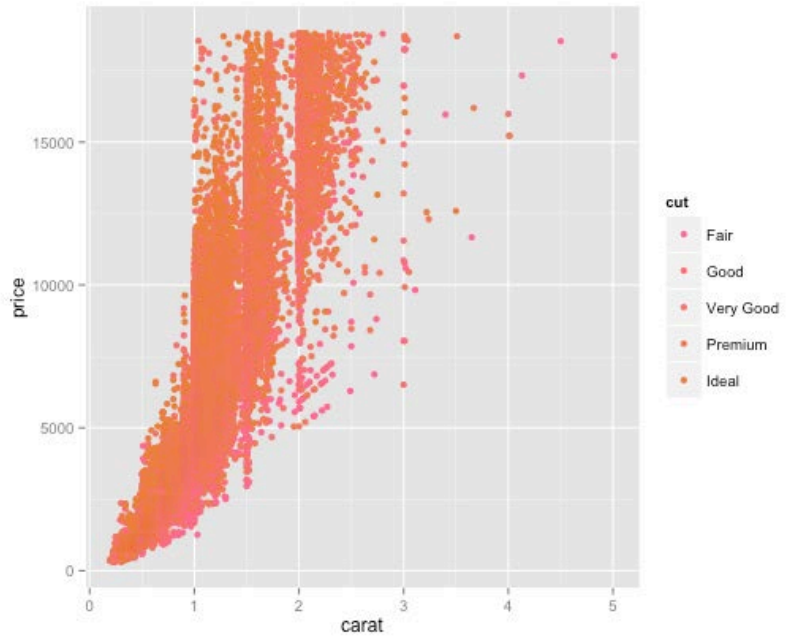
Scales

```
p + scale_color_discrete(h = c(0,180))
```



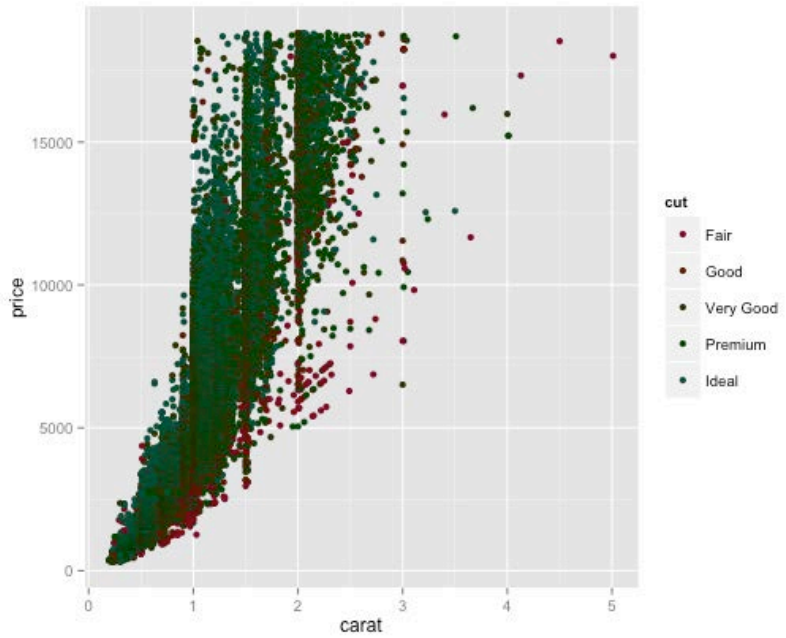
Scales

```
p + scale_color_discrete(h = c(0, 30))
```



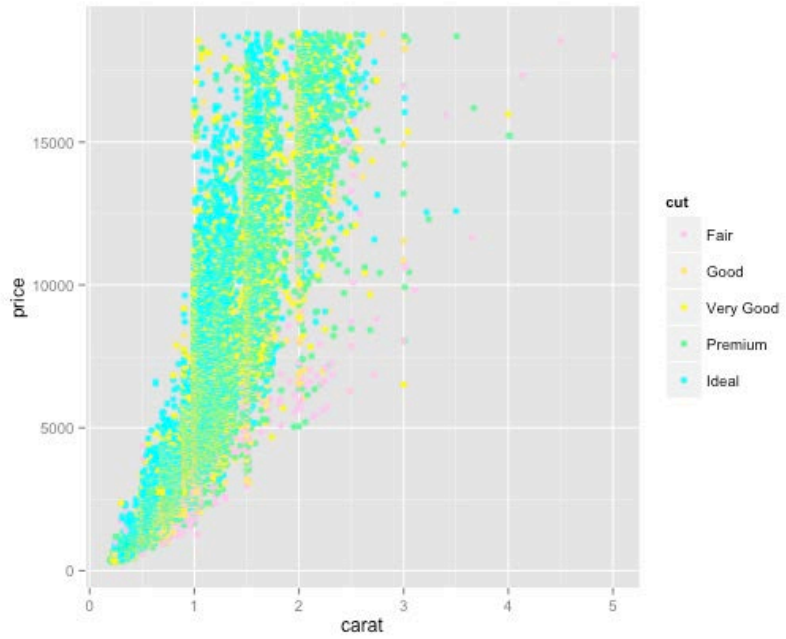
Scales

```
p + scale_color_discrete(h = c(0, 180) ,
    c = 100,l = 20)
```



Scales

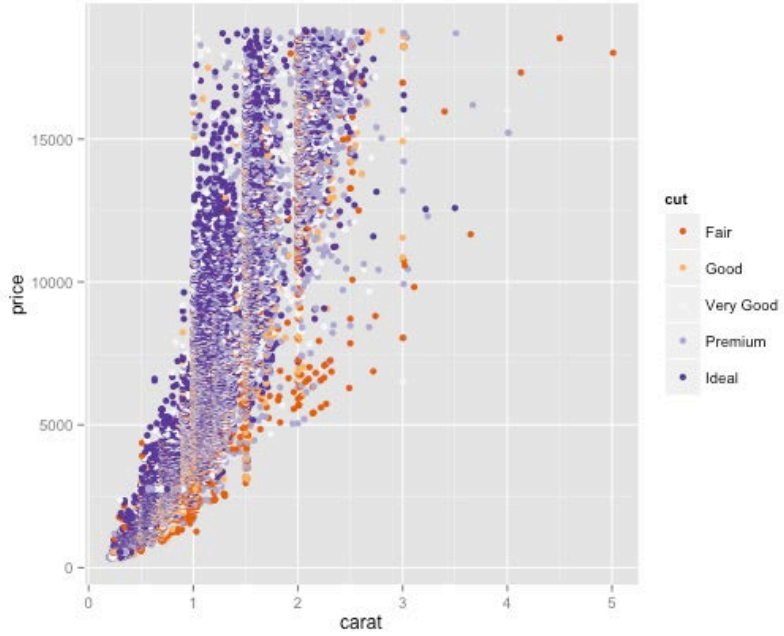
```
p + scale_color_discrete(h = c(0, 180),
c = 100, l = 100)
```



Scales

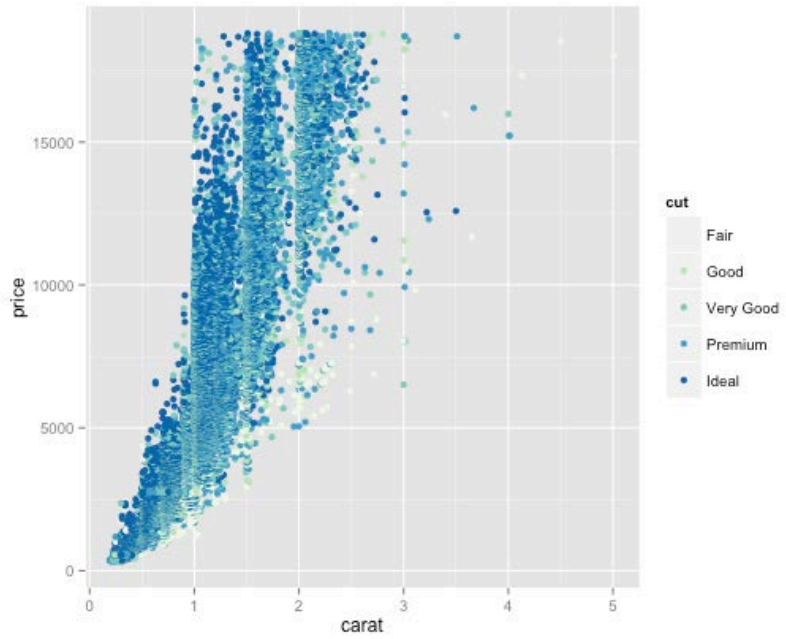
Cynthia Brewer's color palettes are available.

```
p + scale_color_brewer(type = "div",
palette = 4)
```



Scales

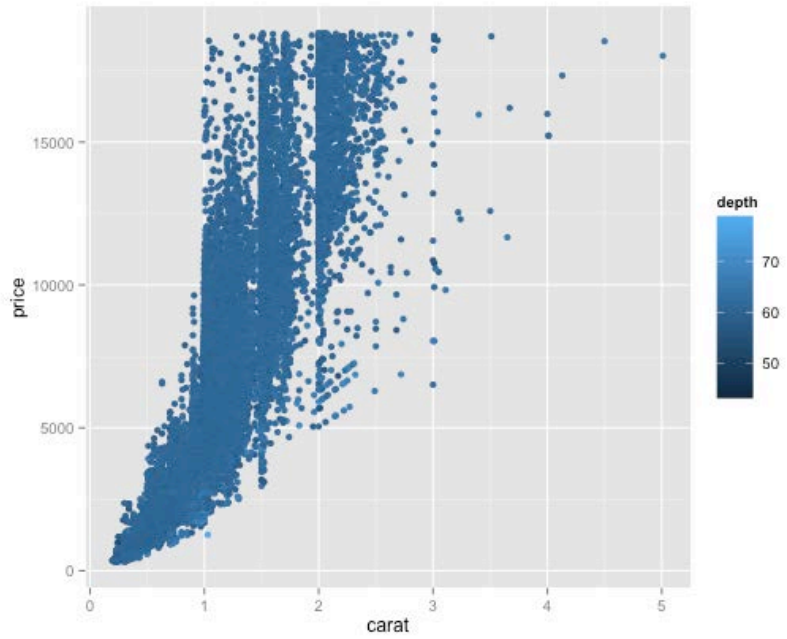
```
p + scale_color_brewer(type = "seq",
      palette = 4)
```



Scales

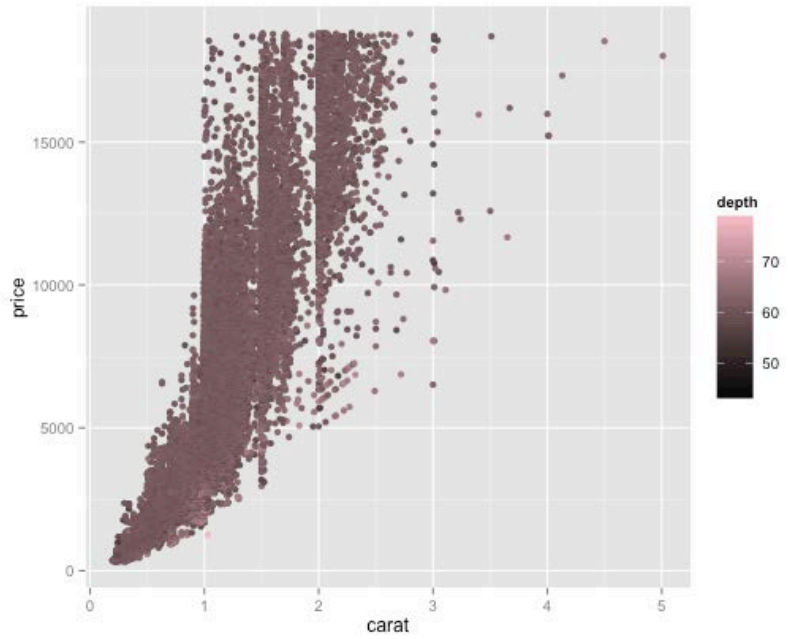
What about color for continuous values?

```
q <- ggplot(diamonds, aes(carat, price,
  color = depth))
 + geom_point()

q
```



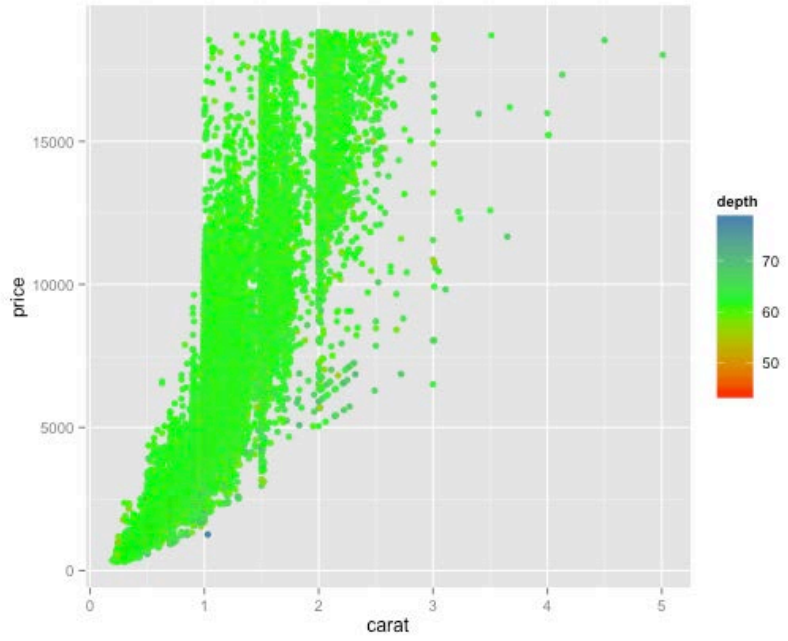
Scales

```
q + scale_color_continuous(low = "black",
    high = "pink"))
```



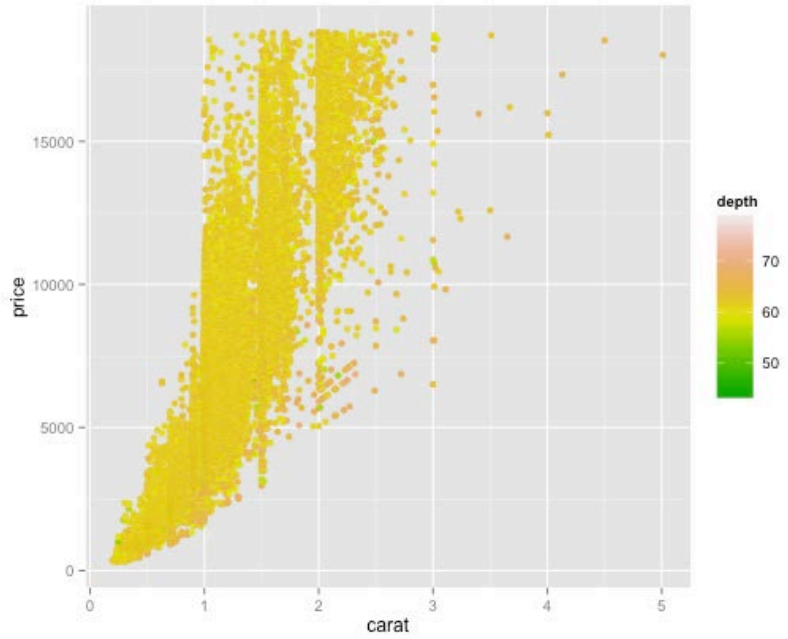
Scales

```
q + scale_color_gradientn(colors = c("red",
  "green", "steel blue"))
```



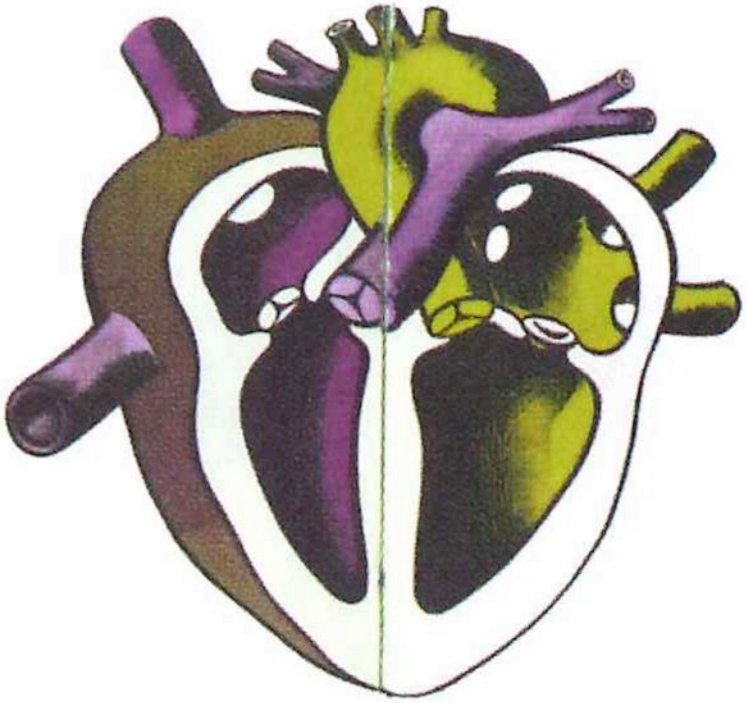
Scales

```
q + scale_color_gradientn(colors =
terrain.colors(10))
```



Scales

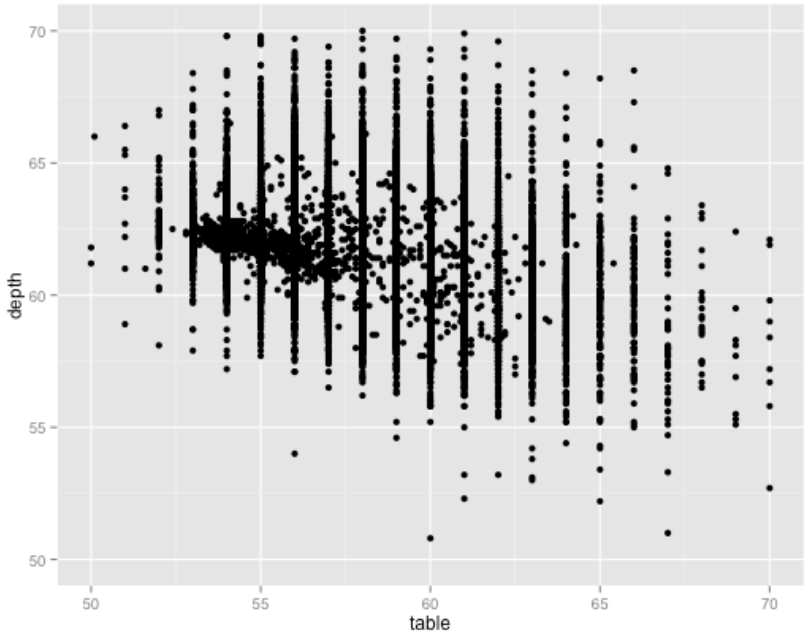
Try not to be too cute with the colors.



Overplotting

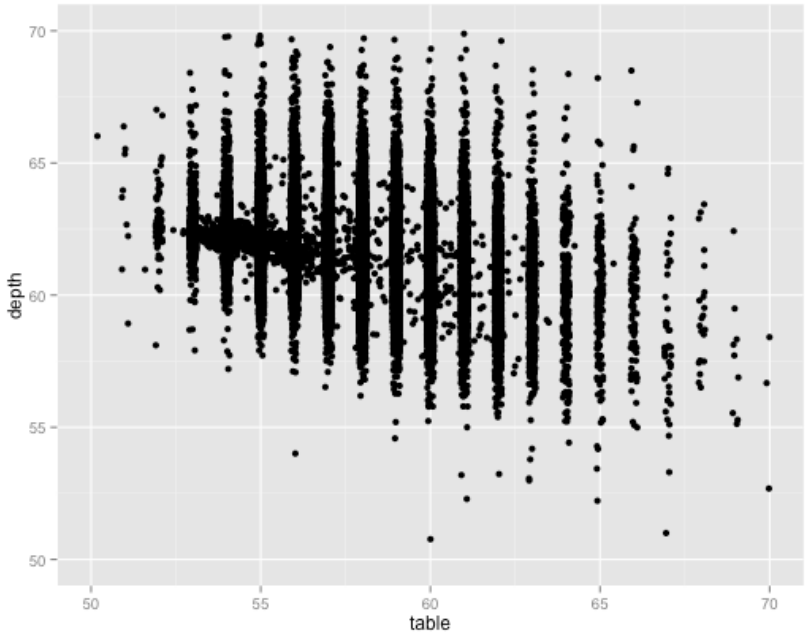
How to deal with overplotting?

```
td <- ggplot(diamonds, aes(table, depth))
+ xlim(50, 70)
+ ylim(50, 70)

td + geom_point()
```



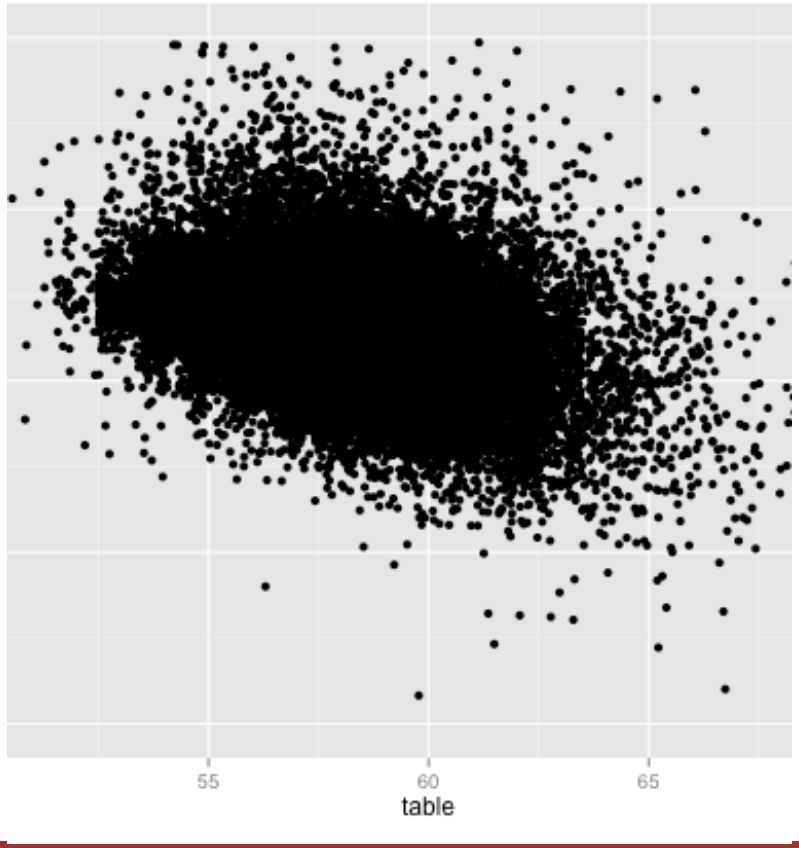
Overplotting

```
td + geom_jitter(position =
      position_jitter(width = .1))
```



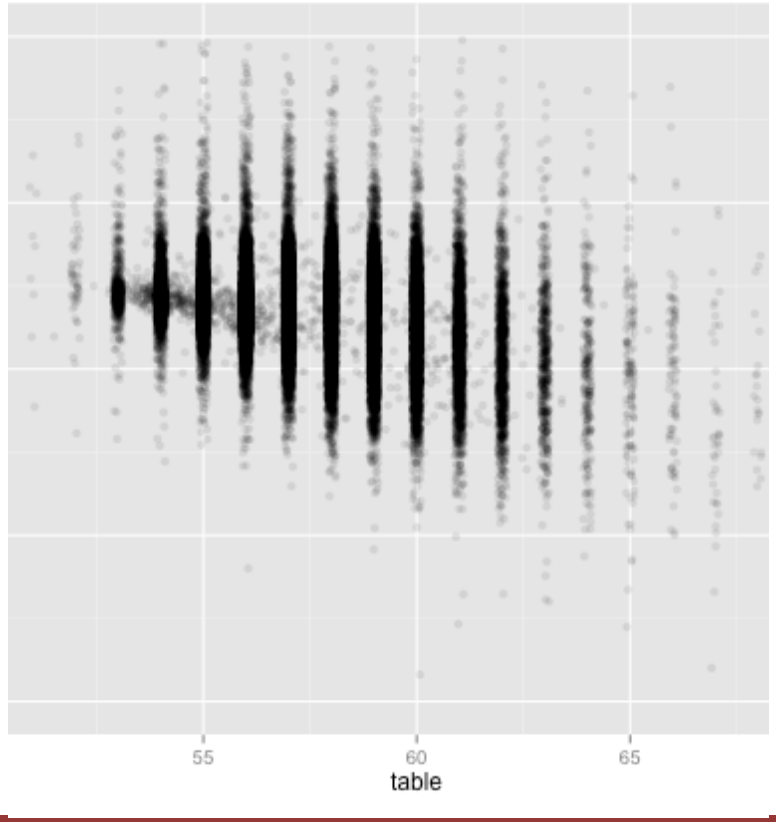
Overplotting

```
td + geom_jitter(position =
position_jitter(width = .5))
```



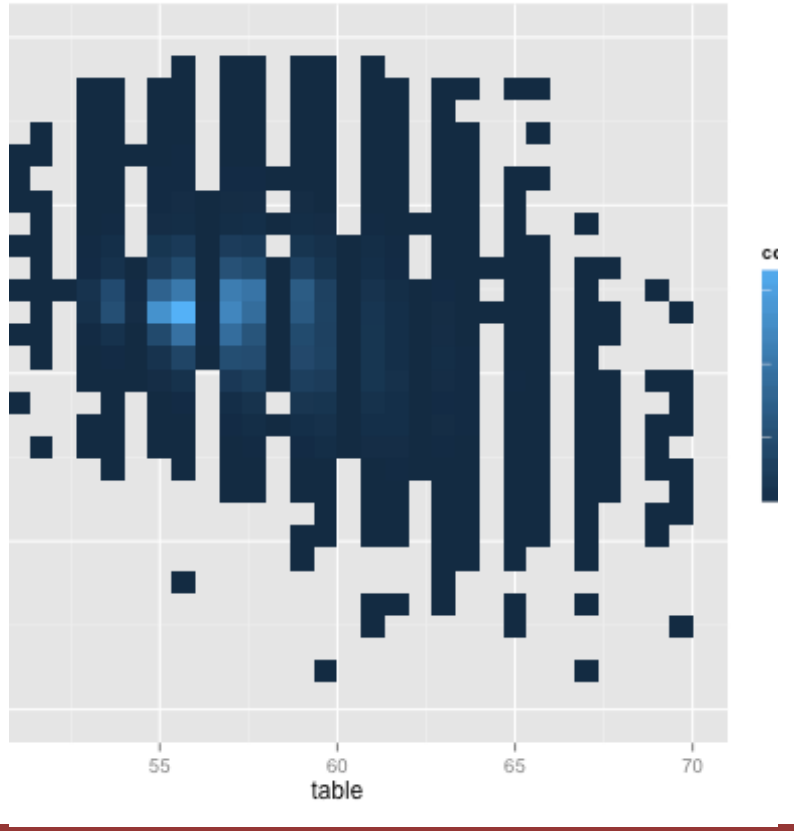
Overplotting

```
td + geom_jitter(position = position_jitter(
width = .1),
alpha =.1 )
```



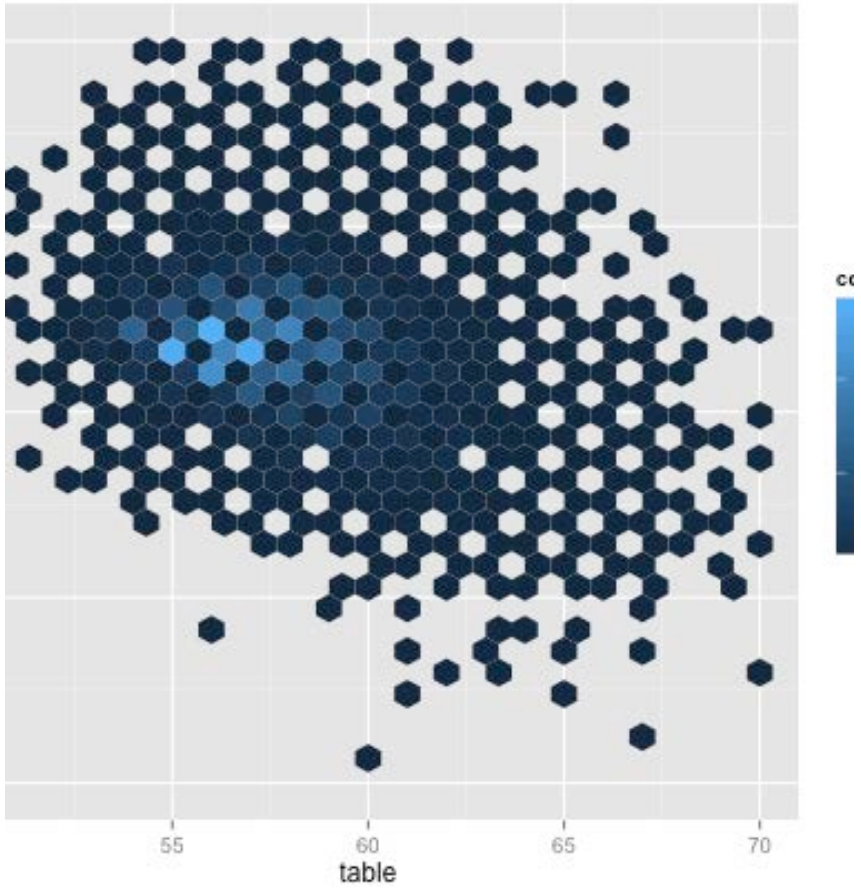
Overplotting

```
td + stat_bin2d()
#This is the same as
#td + geom_bin2d()
```



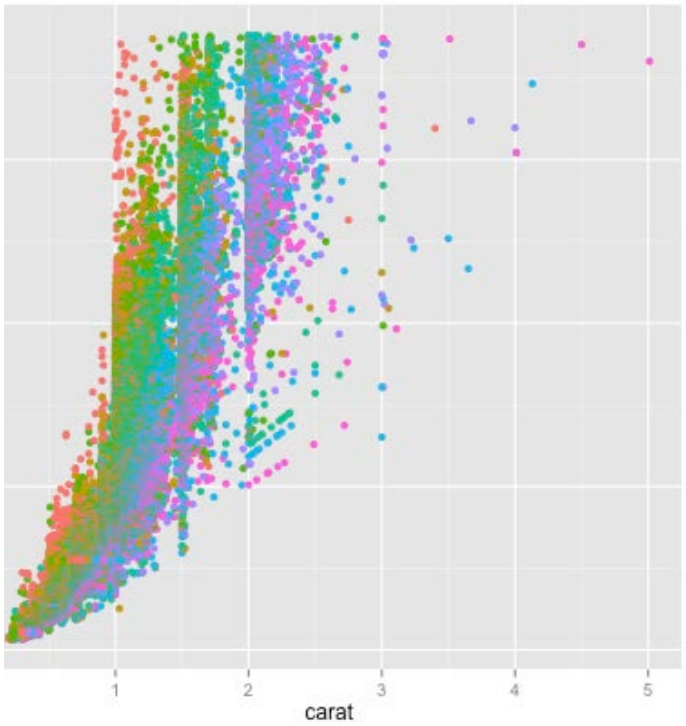
Overplotting

```
td + stat_binhex()
```



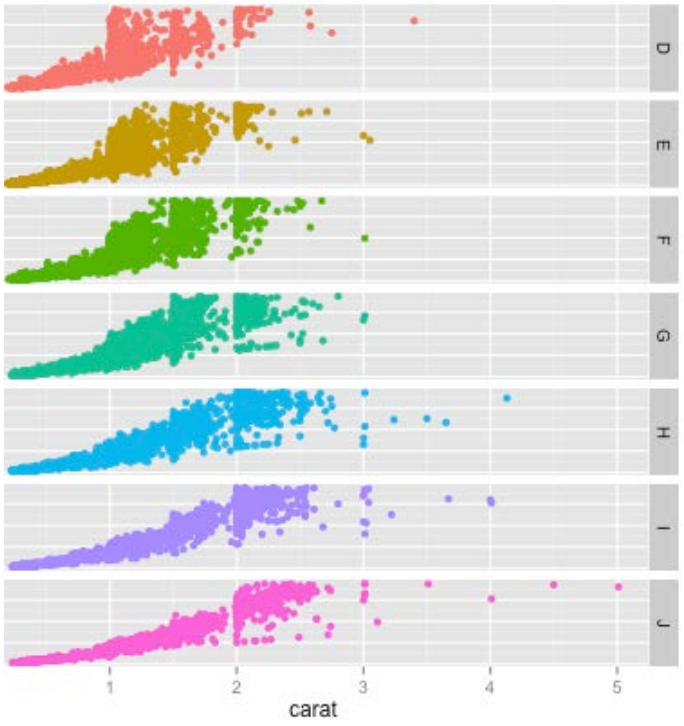
Overplotting/Faceting

```
qplot(carat, price, data = diamonds,
      color = color)
```



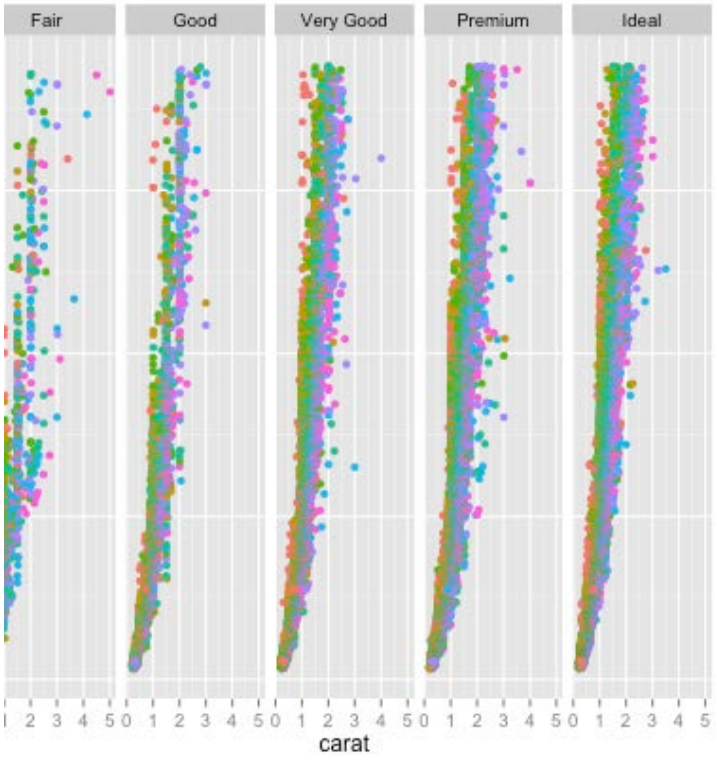
Overplotting/Faceting

```
qplot(carat, price,
data = diamonds,
color = color) + facet_grid(color~.)
```



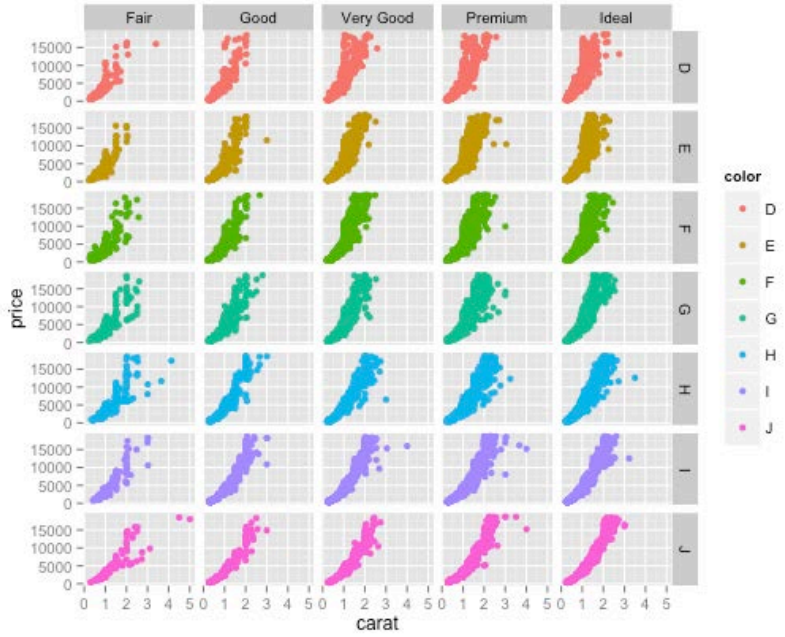
Overplotting/Faceting

```
qplot(carat, price,
data = diamonds,
color = color) + facet_grid(. ~ cut)
```



Overplotting/Faceting

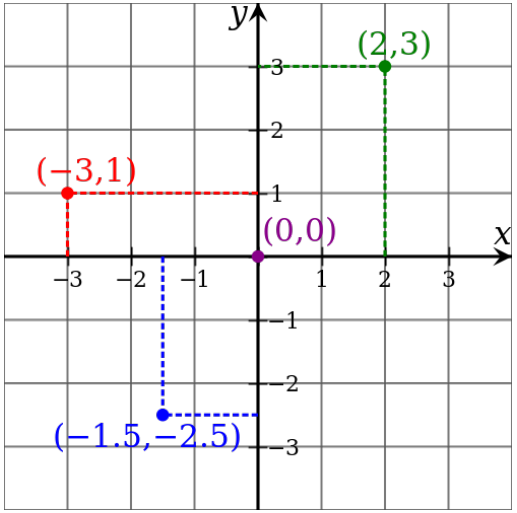
```
qplot(carat, price,
data = diamonds,
color = color) + facet_grid(color ~ cut)
```

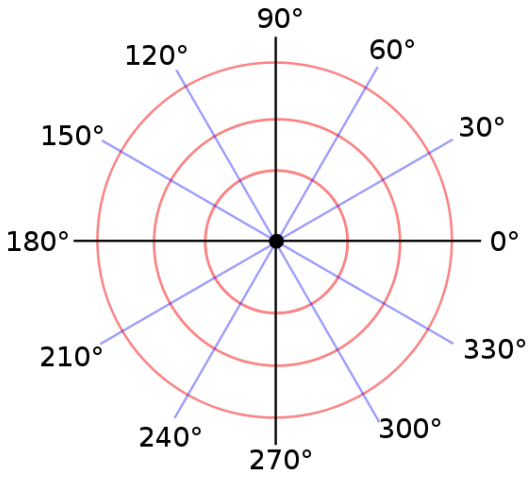


Coordinate systems

The Cartesian coordinate system will get you through just about everything you need.

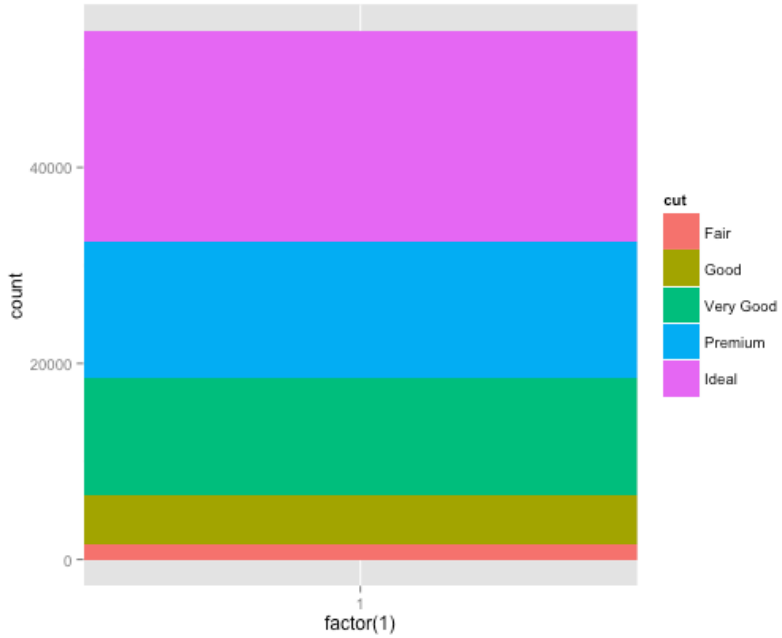
“Once we understand that a pie is a divided bar in polar coordinates, we can construct other polar graphics that are less well known” --Wilkinson





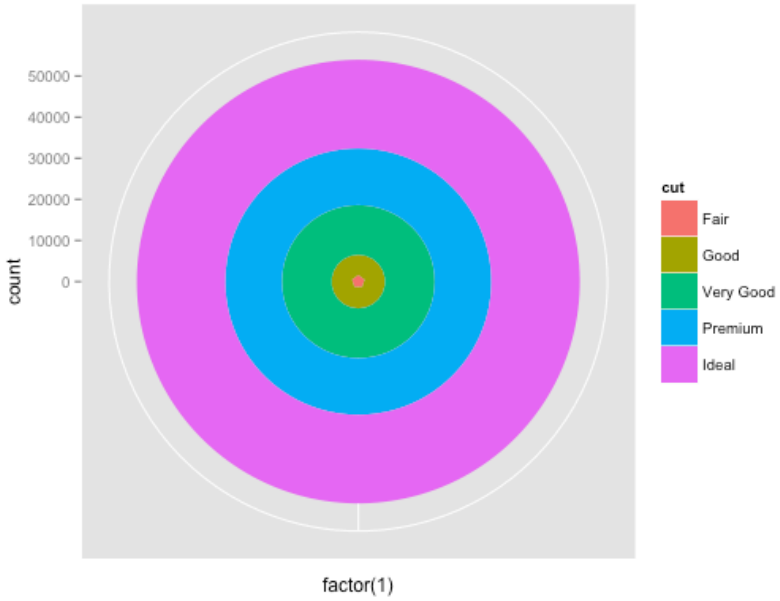
Coordinate systems

```
gplot(diamonds,
   aes(x = factor(1), fill = cut)) +
   geom_bar(width = 1)
```



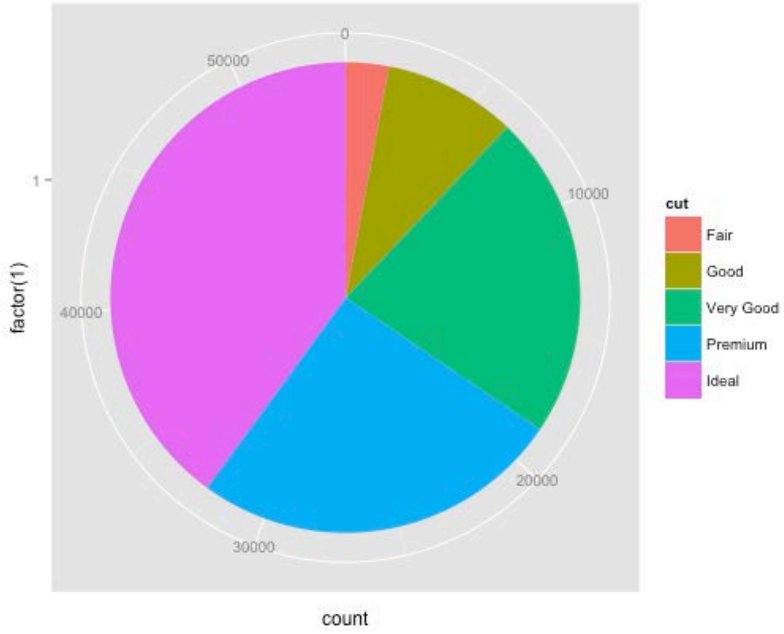
Coordinate systems

```
ggplot(diamonds,
   aes(x = factor(1), fill = cut))  +
   geom_bar(width = 1) +
   coord_polar()
```



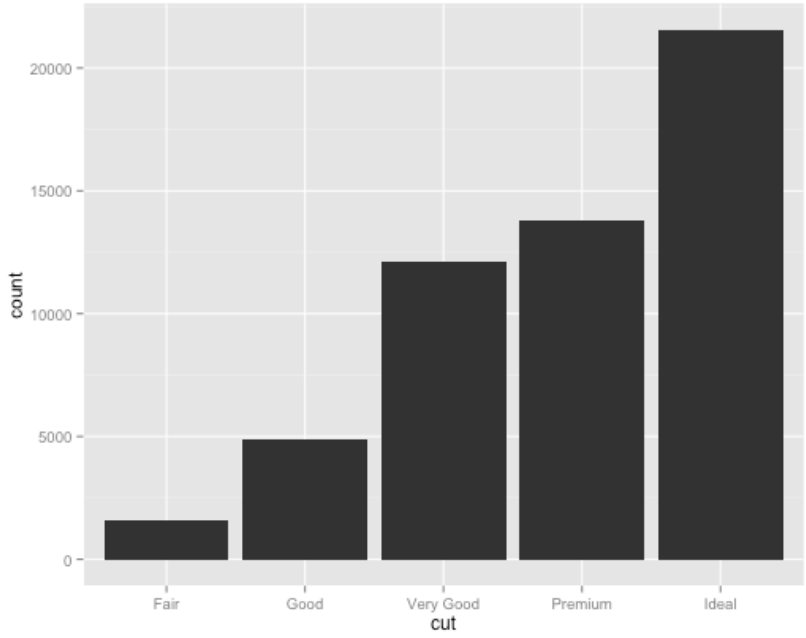
Coordinate systems

```
ggplot(diamonds,
 aes(x = factor(1), fill = cut)) +
 geom_bar(width = 1) +
 coord_polar(theta ="y")
```



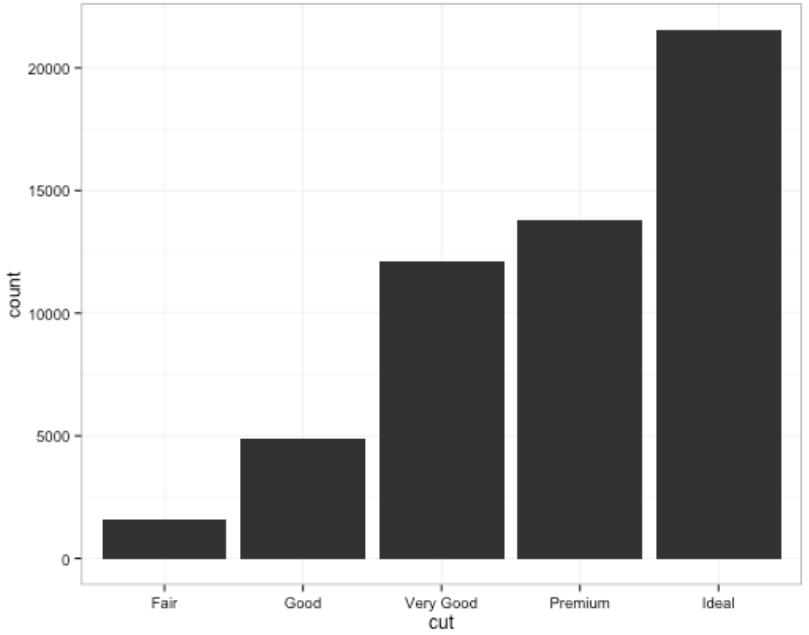
Themes

```
hgram <- qplot(cut, data = diamonds)

hgram
```



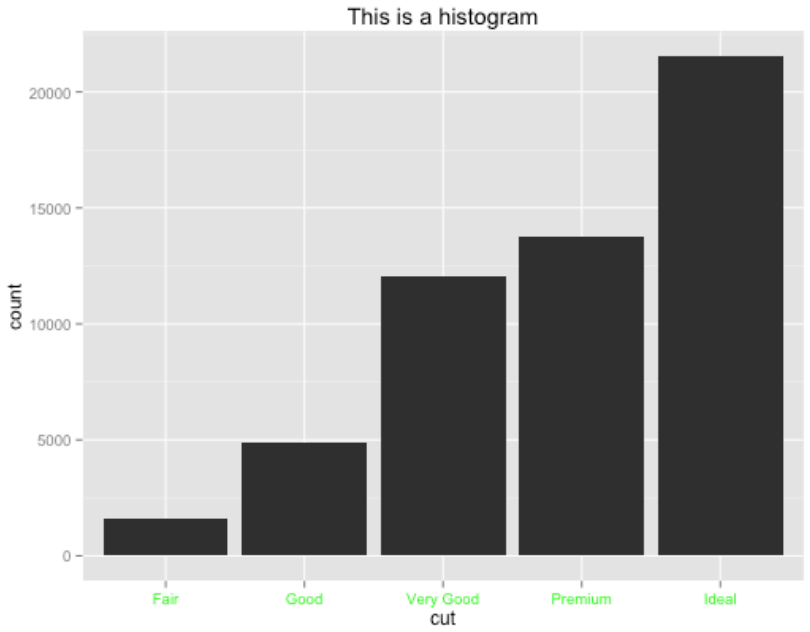
Themes

```
theme_set(theme_bw())
hgram

#Restore default
theme_set(theme_gray())
```



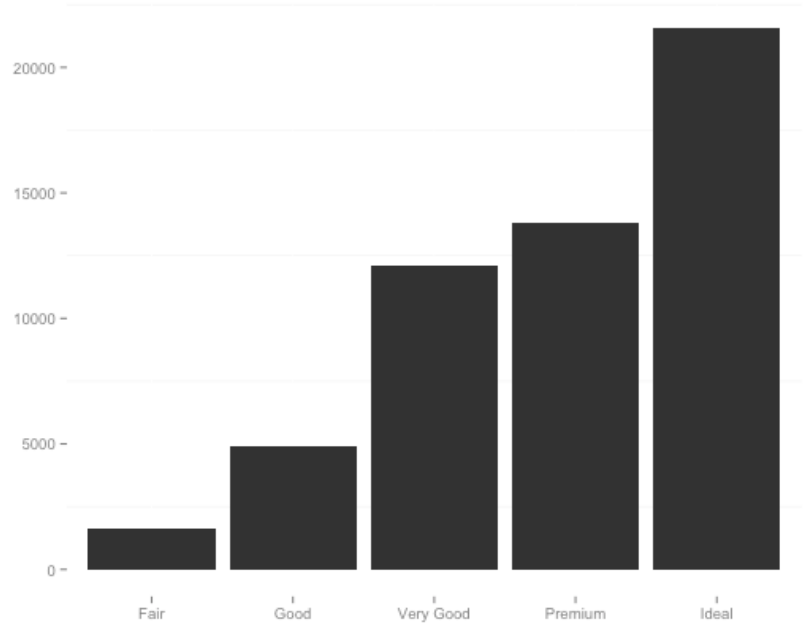
Themes

```
hgram + labs(title = "This is a histogram")
+ theme(axis.text.x =
              element_text(color = "green"))
```



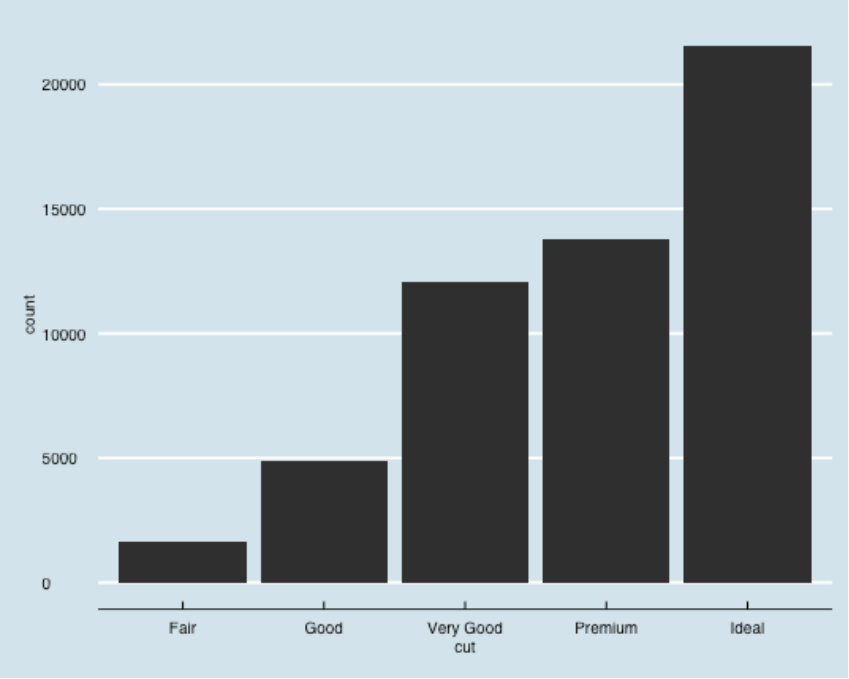
Themes

```
hgram + theme(axis.title.x = element_blank(),
              axis.title.y = element_blank())
    + theme(panel.background = element_blank())
```



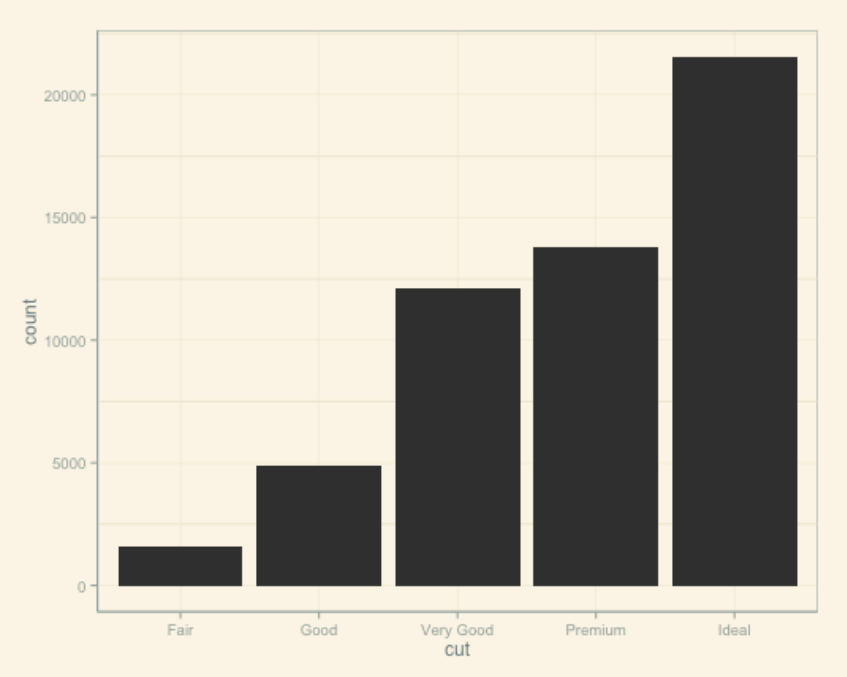
Themes

```
library(ggthemes)

hgram + theme_economist()
```



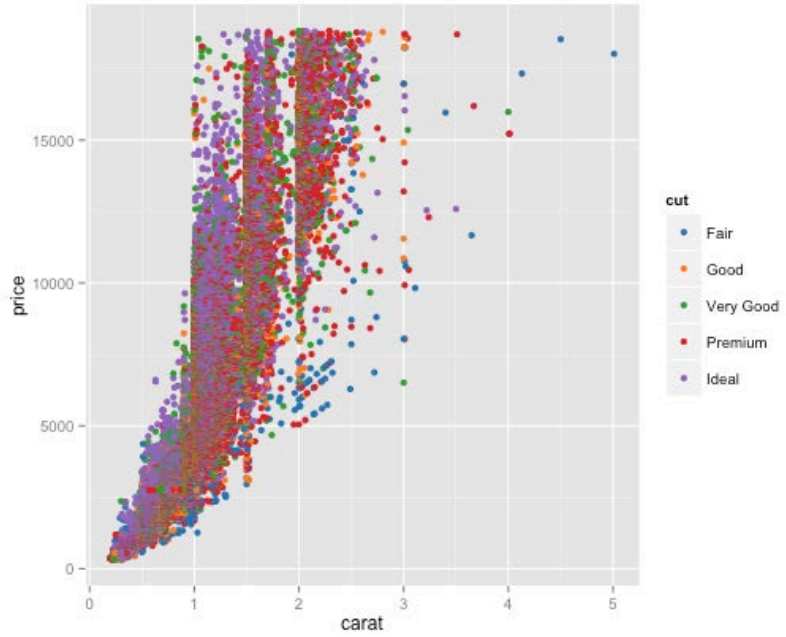
Themes

```
library(ggthemes)

hgram + theme_solarized()
```



Themes

```
library(ggthemes)
p <- ggplot(diamonds, aes(carat, price,
color = cut)) +
     geom_point()

p + scale_colour_tableau()
```



Revisiting Minard

While Tufte say Minard's Napoleon graph tracks six dimensions, Wilkinson points out there is a grouping variable so we have seven values with aesthetic mappings.

```
troops <- read.table("troops.txt", header=TRUE)
cities <- read.table("cities.txt", header=TRUE)
```

Revisiting Minard

```
head(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

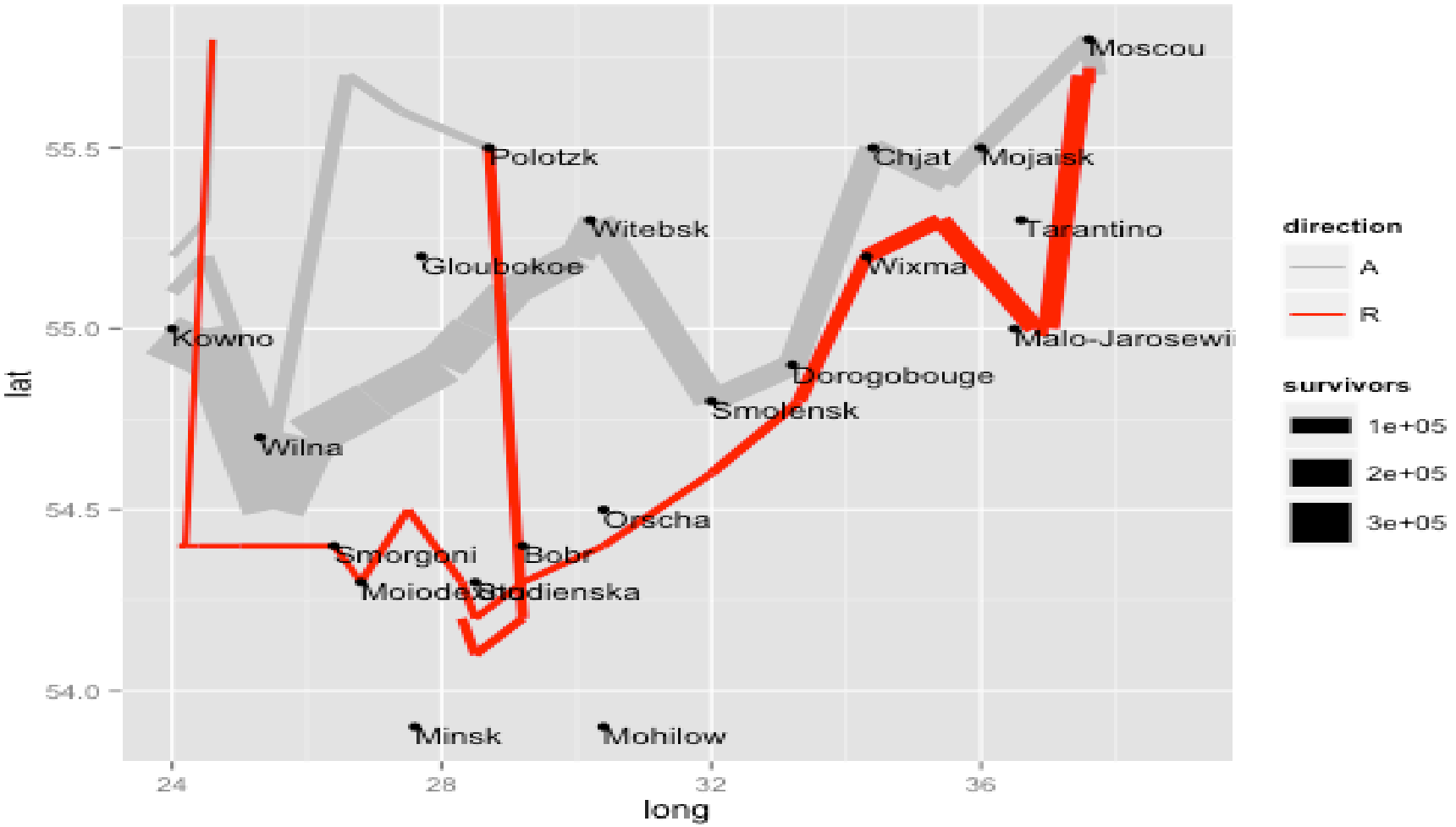
Revisiting Minard

```
head(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
5 27.7 55.2 Gloubokoe
6 27.6 53.9     Minsk
```

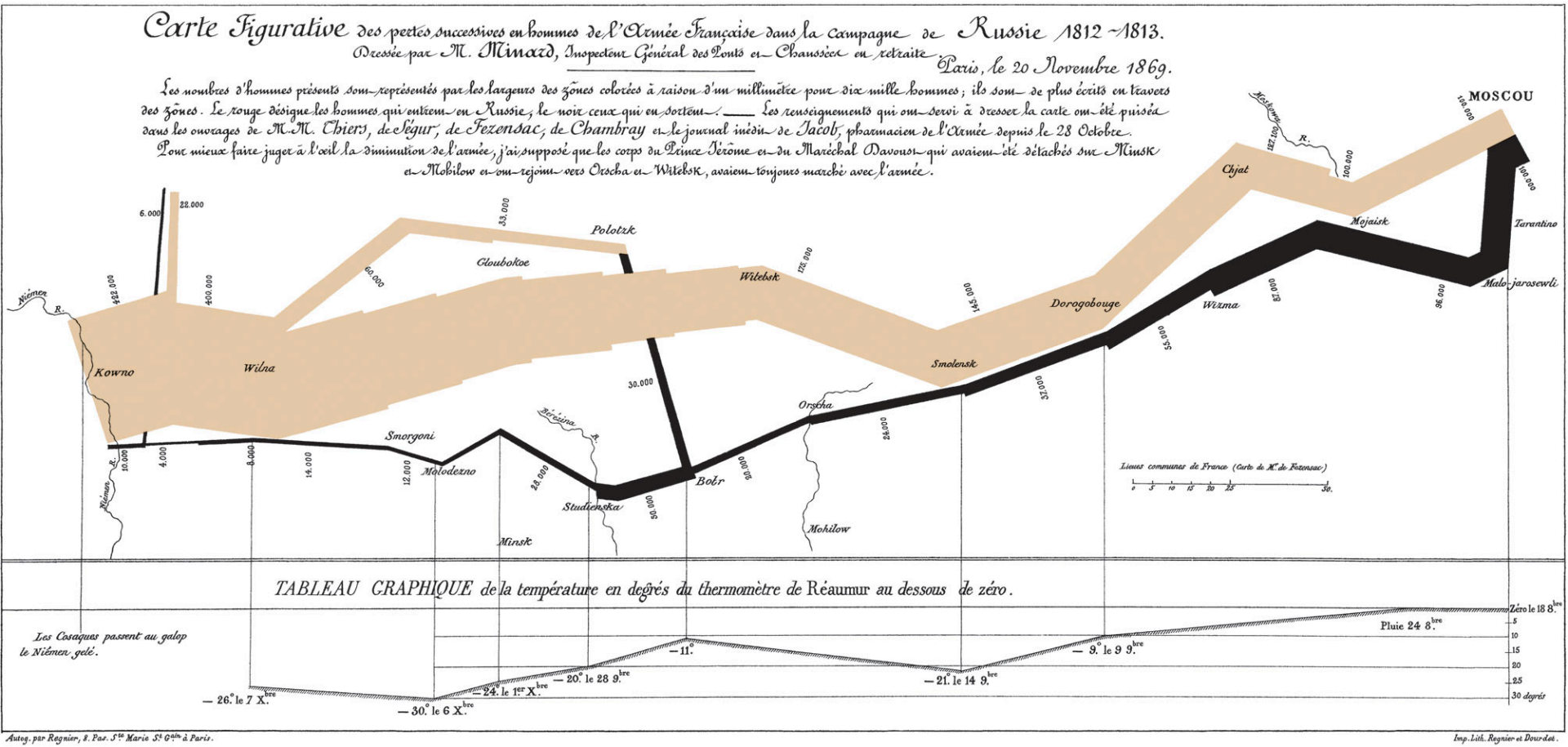
Revisiting Minard

```
xlim <- scale_x_continuous(limits = c(24, 39))
ggplot(cities, aes(x = long, y = lat)) +
    geom_path(
        aes(size = survivors, color = direction, group =
group),
        data = troops
    ) +
    geom_point() +
    geom_text(aes(label = city), hjust=0, vjust=1, size=4)
+
    scale_size(range = c(1, 10)) +
    scale_colour_manual(values = c("grey","red"))  +
xlim
```

Revisiting Minard



Revisiting Minard



Further Reading

Bertin, J. (1983). Semiology of Graphics. Madison, Wisconsin: University of Wisconsin Press.

D. B. Carr, et al. (1987). Scatterplot Matrix Techniques for Large N. Journal of the American Statistical Association, 82(398), 424-436.

Katz, J. (2012). Designing information. Hoboken, New Jersey: John Wiley & Sons, Inc.

Wickham, H. (2009). ggplot2. New York, New York: Springer Science+Business Media.

Wilkinson, L. (2005). The Grammar of Graphics. New York, New York: Springer Science+Business Media.