Data visualization with R

~~Ron Dotsch~~
Bernd Figner

Data visualization with R

- R has awesome data visualization capabilities
- Exploring vs. publication grade quality
- The best package: **ggplot2**
- `install.packages('ggplot2')`

ggplot2

- Grammar of graphics (Wilkinson, 2005)
- Simple set of core principles
- Layered
- Elegant

Grammar of graphics

Statistical graphic is mapping **data** onto **aesthetic** attributes (colour, shape, size, positions) of **geometric** objects (points, lines, bars)

Concepts

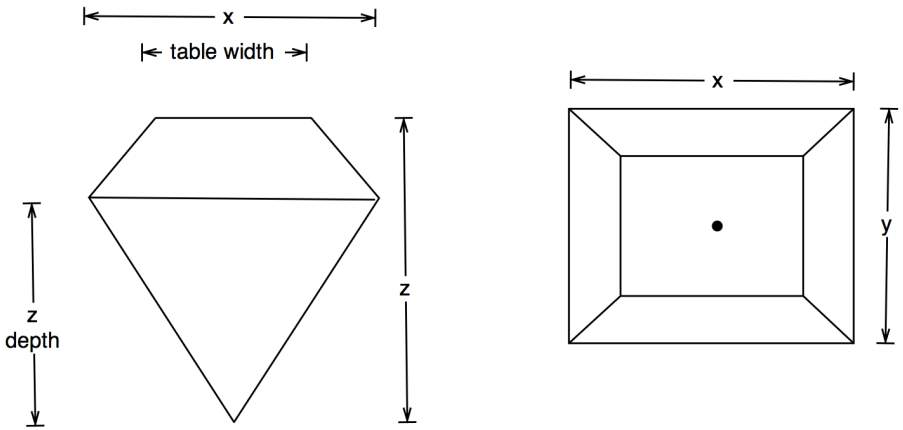
Concept	Definition
aesthetic mapping	how variables in data affect visual components
geom	objects that you actually see in plot (points, lines)
stat	statistical transformation (to summarize data) - optional
facet	how to break up data into subsets
theme	customize the looks of your graphic

Starting easy

`qplot()`

aka

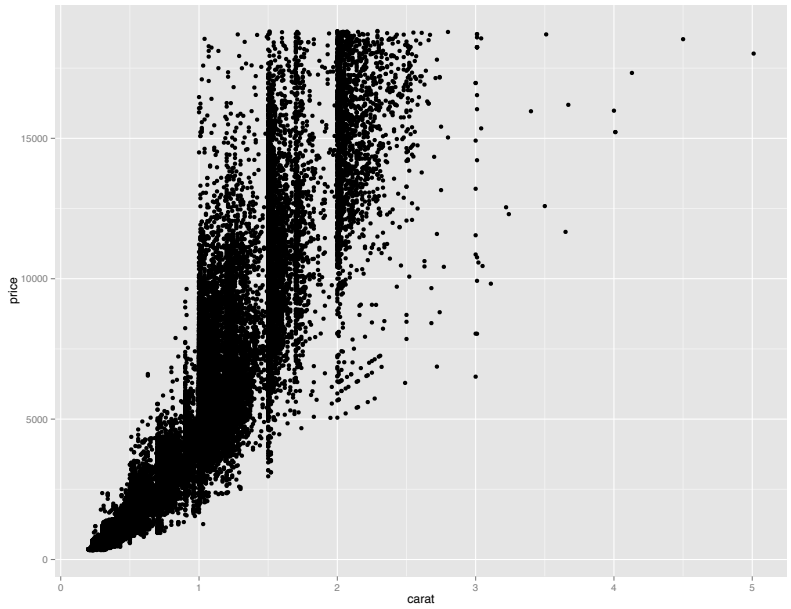
`quickplot()`



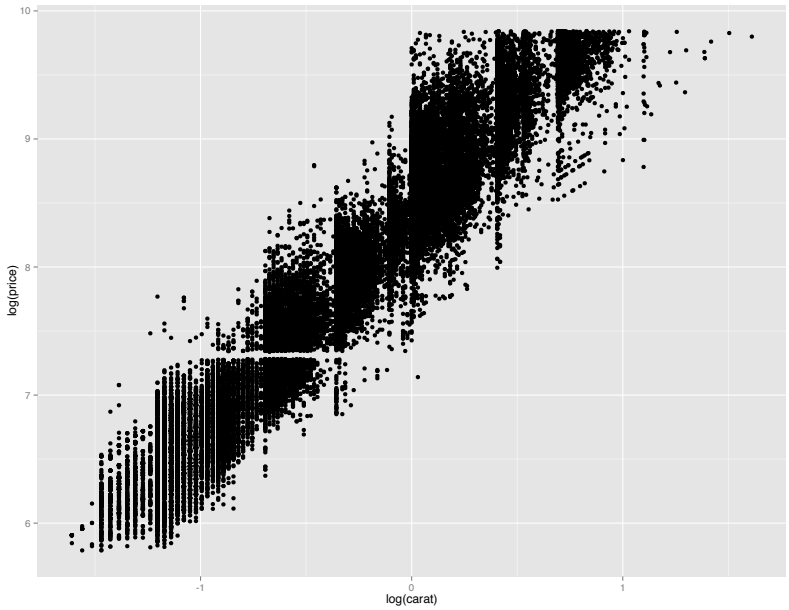
Dataset `diamonds` (example ggplot2 package)

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63
5	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11	2.53
9	0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49
10	0.23	Very Good	H	VS1	59.4	61	338	4	4.05	2.39

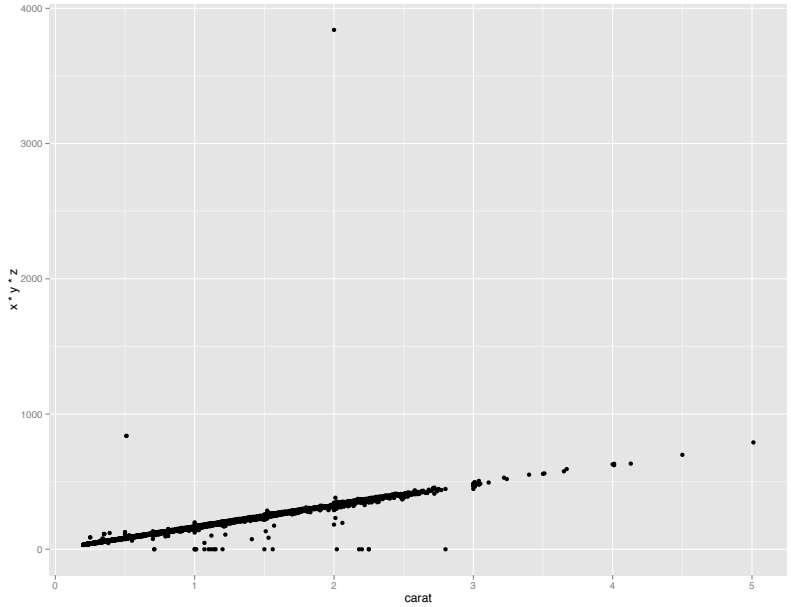
```
qplot(carat, price, data=diamonds)
```




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



```
qplot(carat, x*y*z, data=diamonds)
```

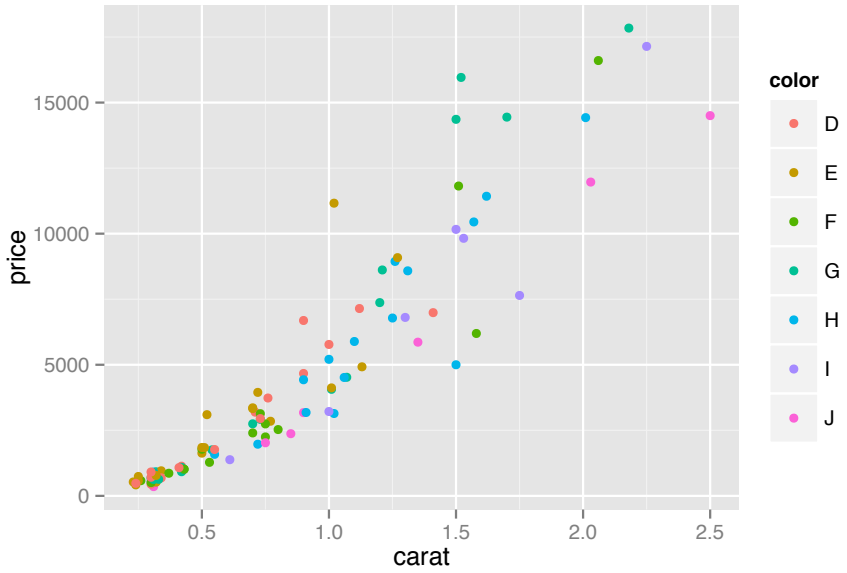


Random sample from data

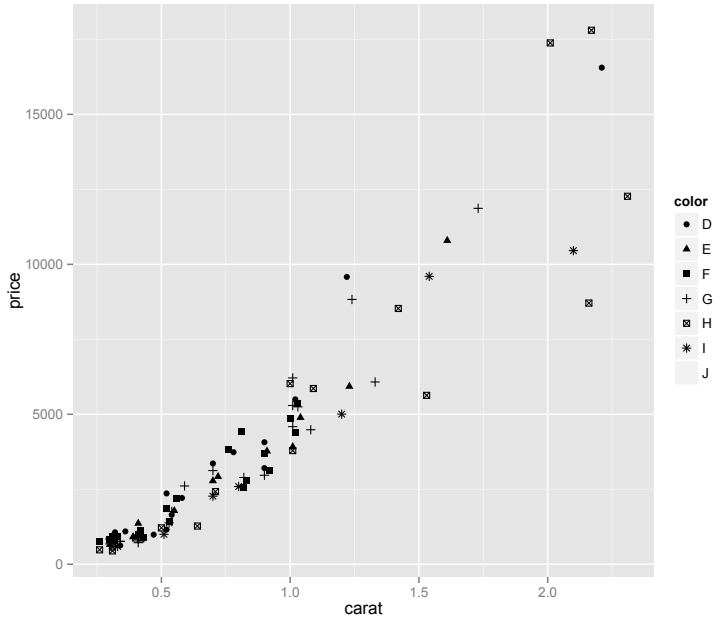
1. Initialize random number generator
2. Draw random sample of 100 rows

```
> set.seed(100)
> dsmall <- diamonds[sample(nrow(diamonds), 100), ]
```

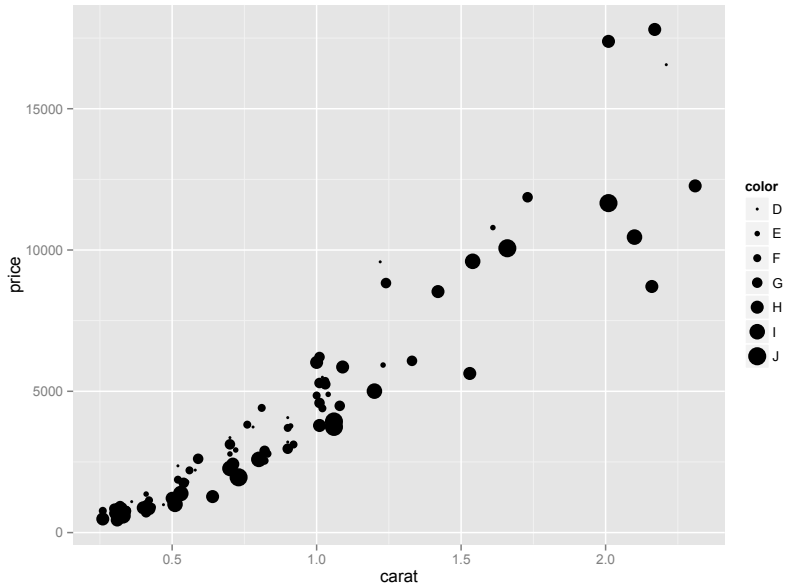
```
qplot(carat, price, data = dsmall, color = color)
```



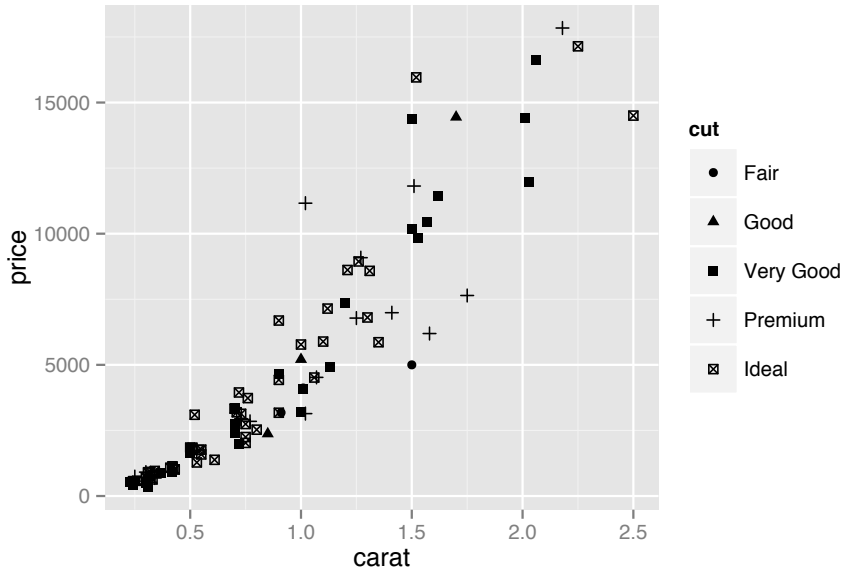

```
qplot(carat, price, data = dsmall, shape = color)
```



```
qplot(carat, price, data = dsmall, size = color)
```




```
qplot(carat, price, data = dsmall, shape = cut)
```

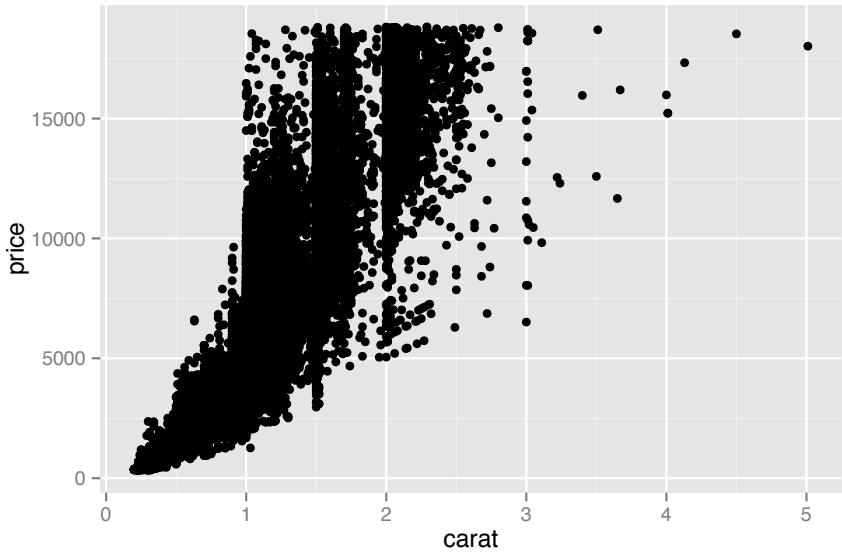


Aesthetics

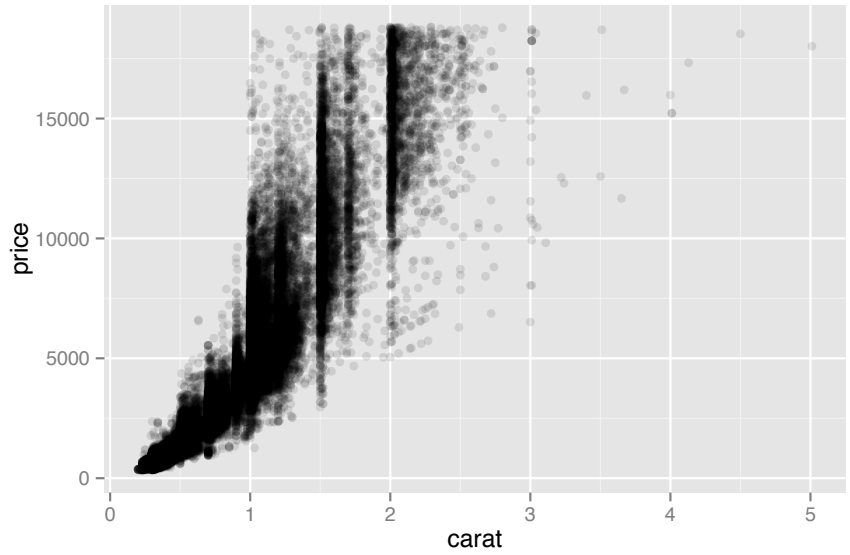
- Color, shape, size
- **Scale**: maps values of variable onto value of aesthetic
  - Size works well for continuous variables
  - Color and shape work well with categorical variables
- Manual setting with `I()`, for example: `size=I(2)`

more info, for example:
http://sape.inf.usi.ch/quick-reference/ggplot2/scale

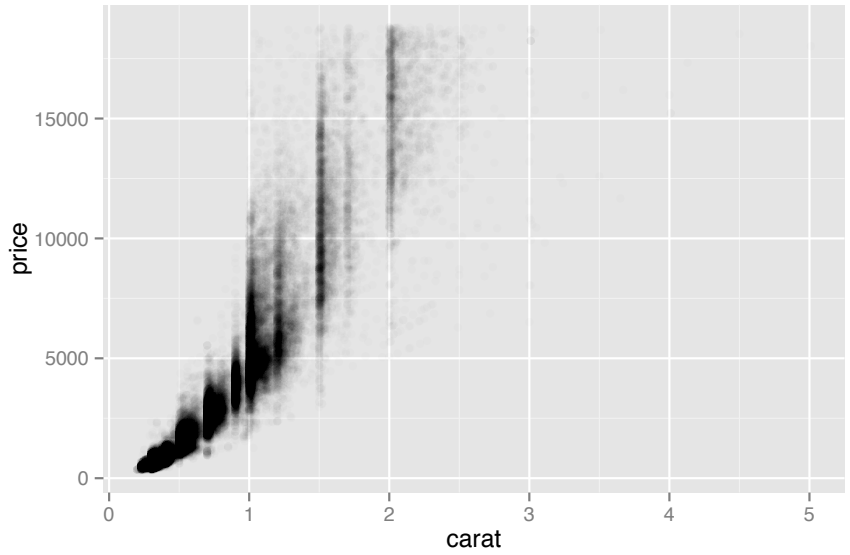

```
qplot(carat, price, data = diamonds)
```



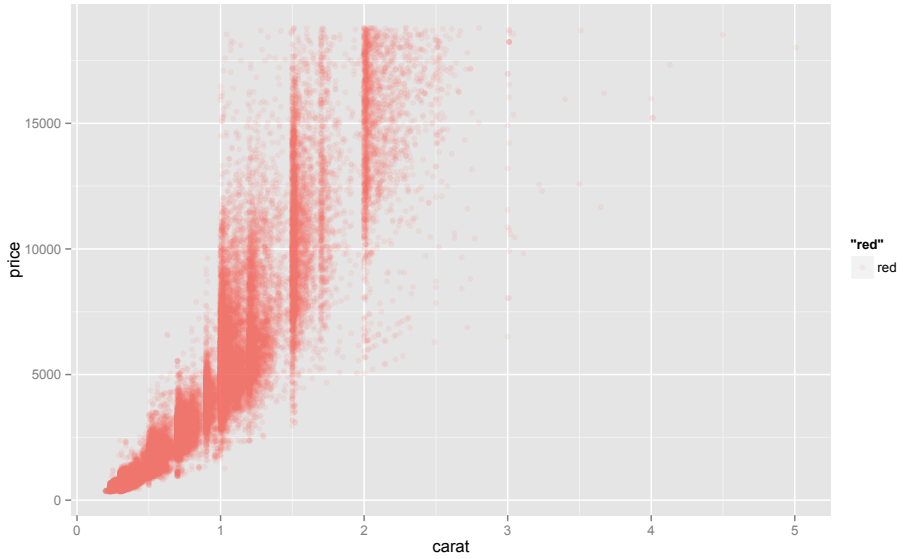
```
qplot(carat, price, data = diamonds, alpha = I(1/10))
```




```
qplot(carat, price, data = diamonds, alpha = I(1/100))
```



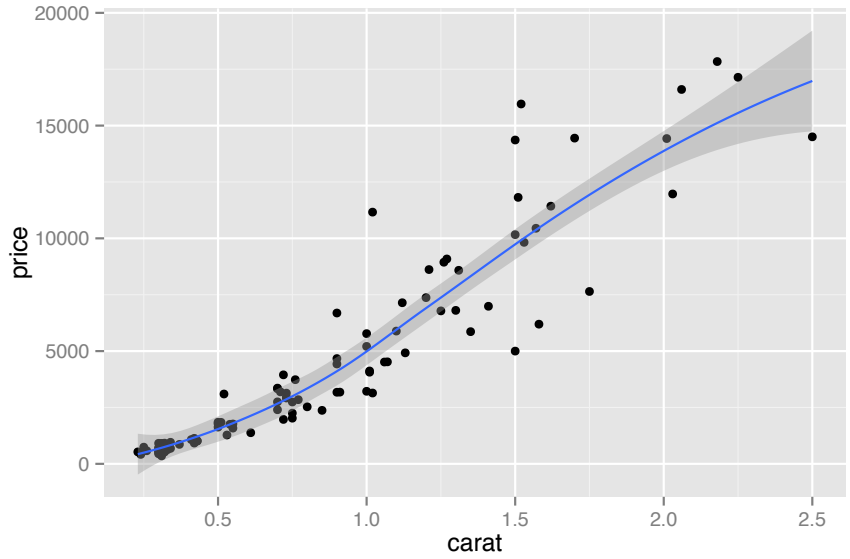
```
qplot(carat, price, data = diamonds, alpha = I(1/10), color = "red")
```



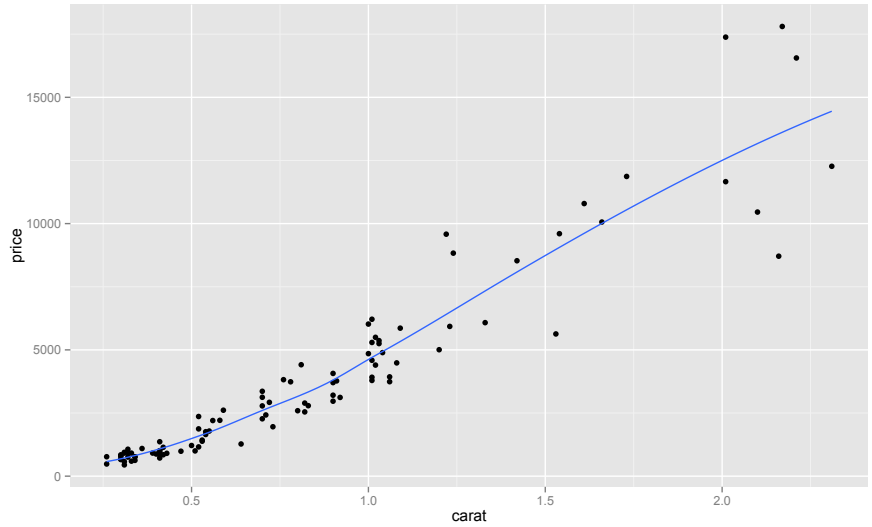
Not just scatterplots

- `qplot` can produce any kind of plot by varying **`geom`** parameter
- Geom describes type of object to display data
- Common geoms:
  - `point, smooth, boxplot, path, line, histogram, freqpoly, density, bar`

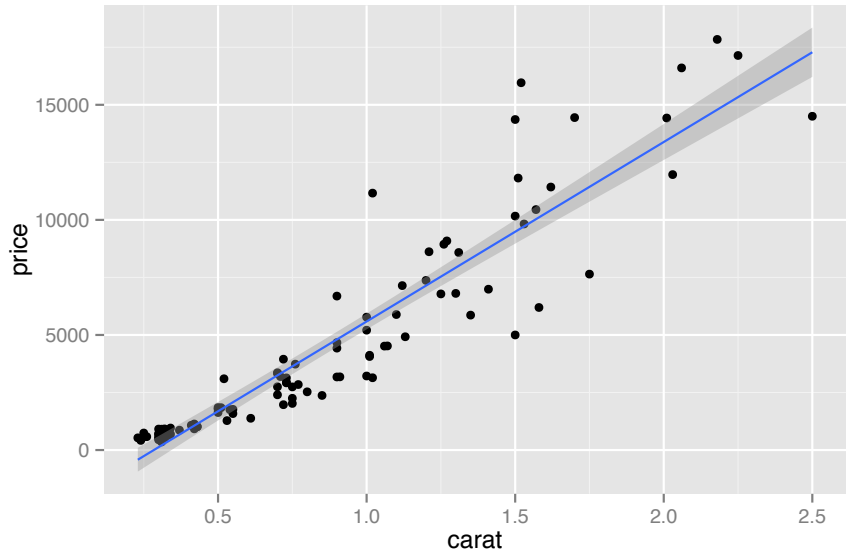
```
qplot(carat, price, data=dsmall, geom=c("point", "smooth"))
```



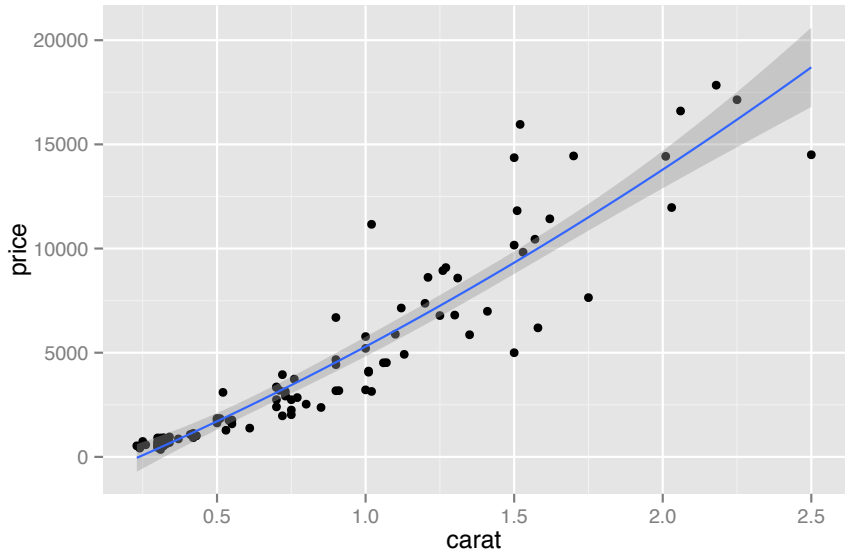

```
qplot(carat, price, data=dsmall, geom=c("point", “smooth”), se = FALSE)
```



```
qplot(carat, price, data=dsmall, geom=c("point", "smooth"),
      method="lm")
```



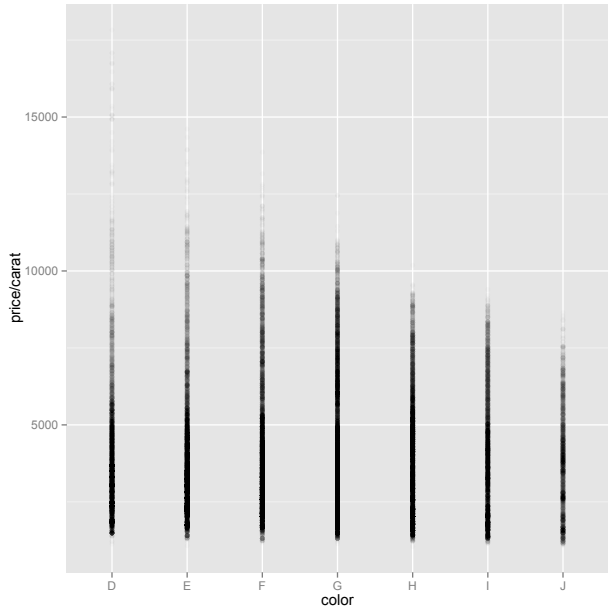

```
qplot(carat, price, data=dsmall, geom=c("point", "smooth"),
      method="lm", formula=y~poly(x,2))
```



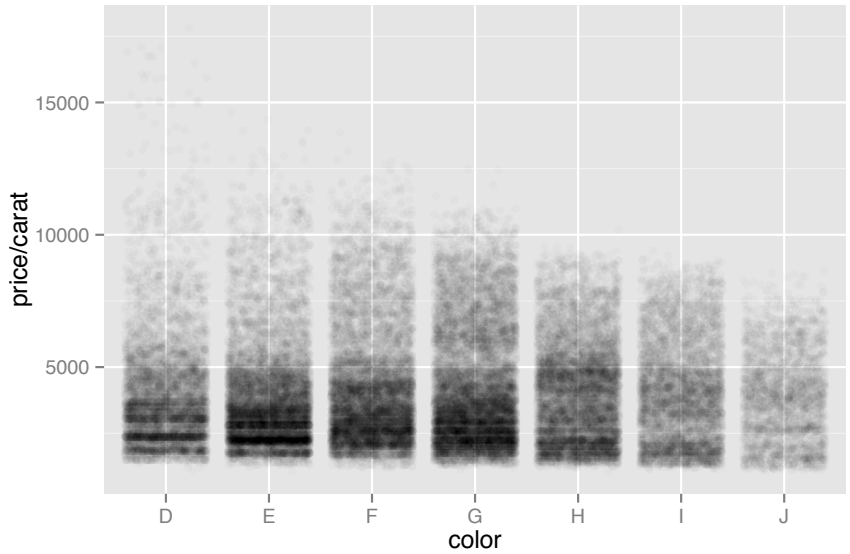
Data by category

- different colors in scatter plot ✔
- jittered points
- box plot
- different colored lines
- facets

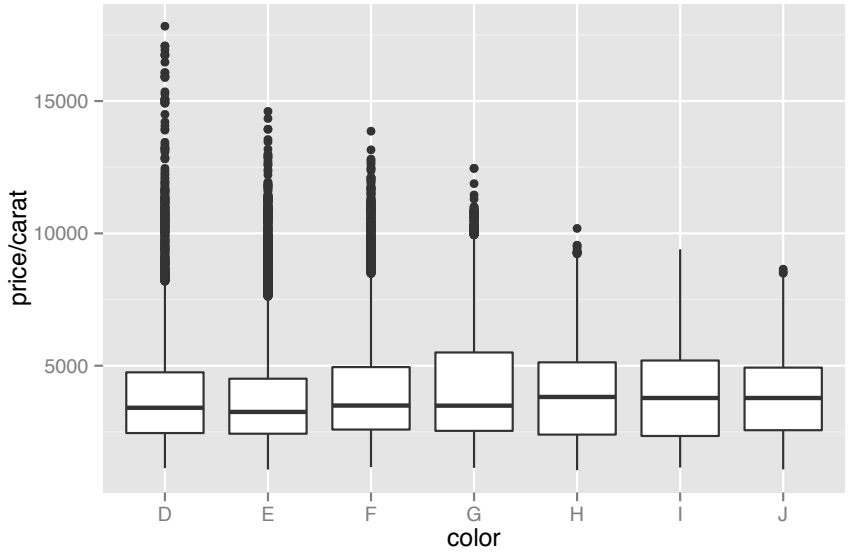

```
qplot(color, price/carat, data=diamonds, alpha=I(1/50))
```



```
qplot(color, price/carat, data=diamonds, geom="jitter",
      alpha=I(1/50))
```



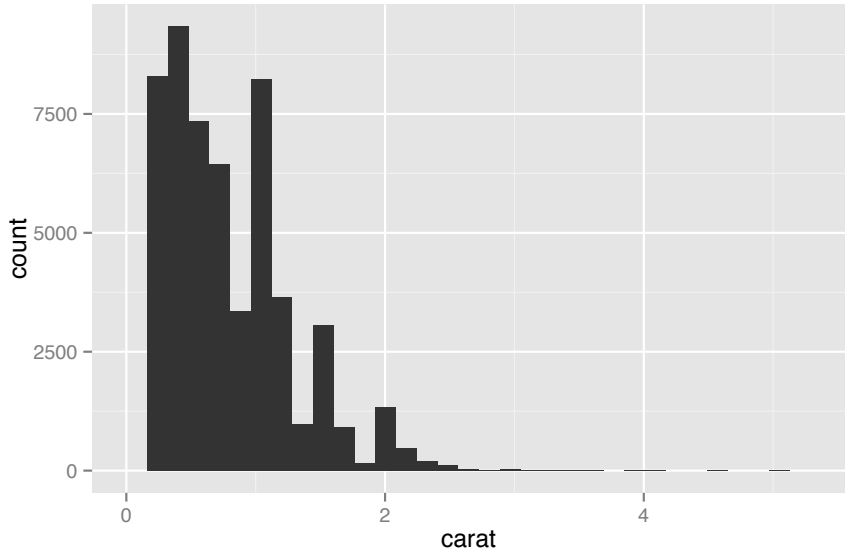

```
qplot(color, price/carat, data=diamonds,
      geom="boxplot")
```



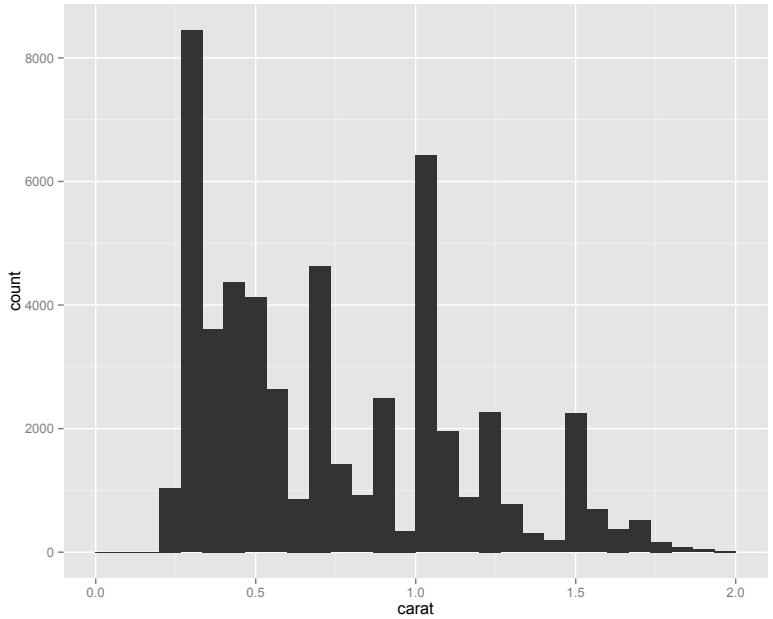
Distributions

- Histogram
- Density plot
- Bar chart (on discrete data)

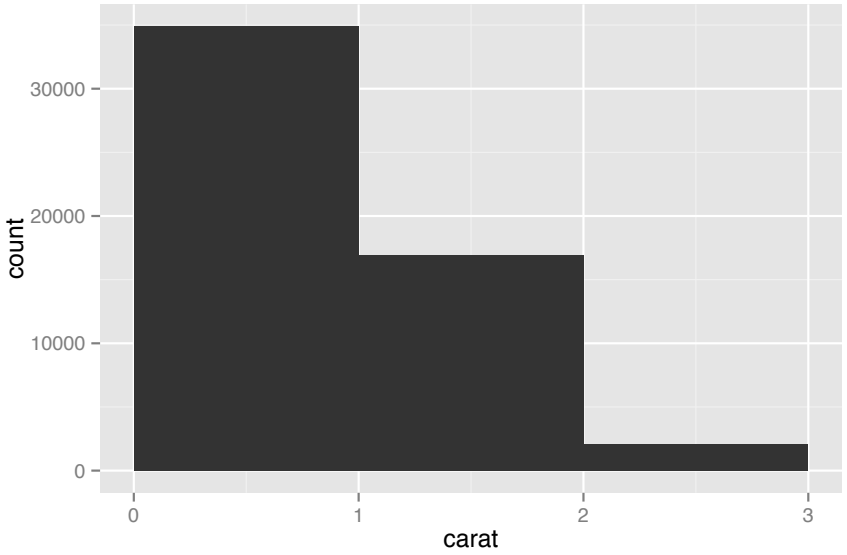

```
qplot(carat, data=diamonds, geom="histogram")
```



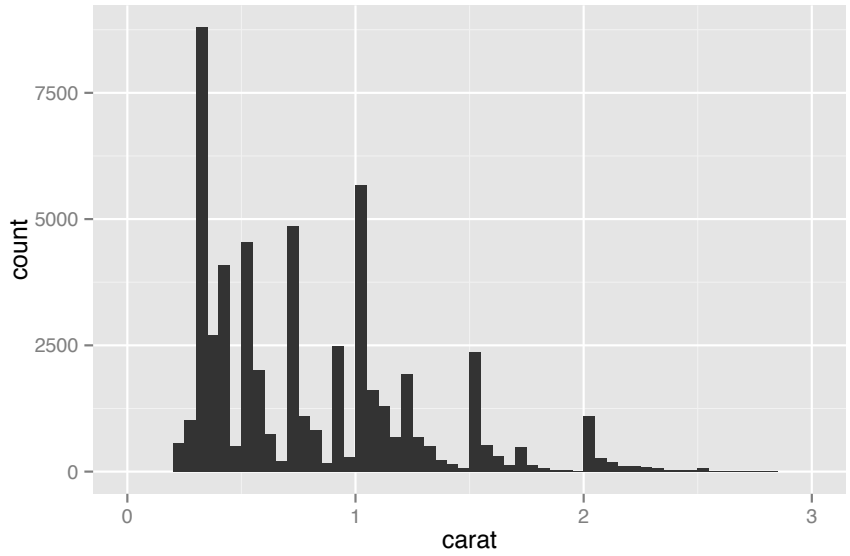
```
qplot(carat, data=diamonds, geom="histogram", xlim = c(0,2))
```



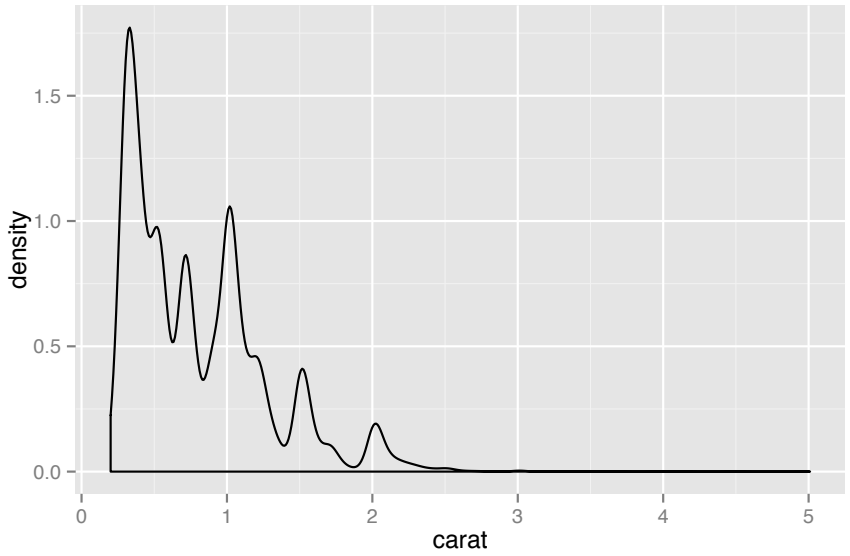

```
qplot(carat, data=diamonds, geom="histogram", binwidth = 1,
xlim=c(0,3))
```



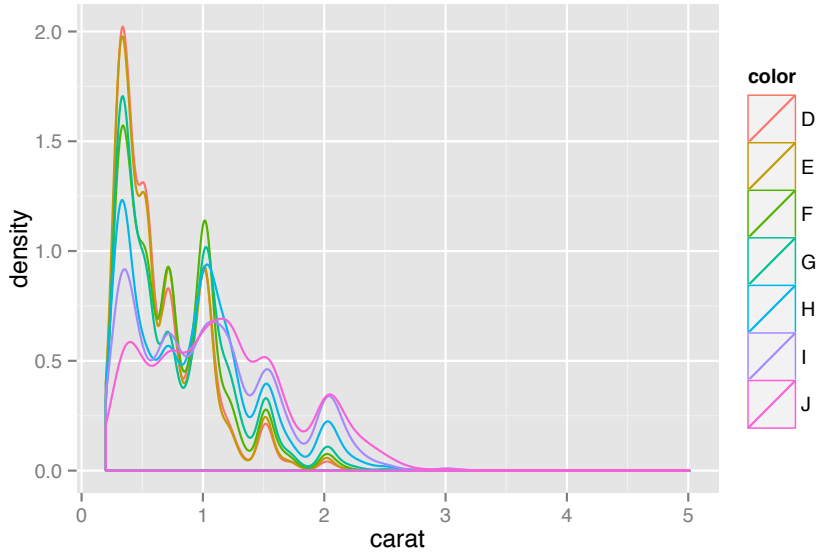
```
qplot(carat, data=diamonds, geom="histogram", binwidth =
                    0.1, xlim=c(0,3))
```



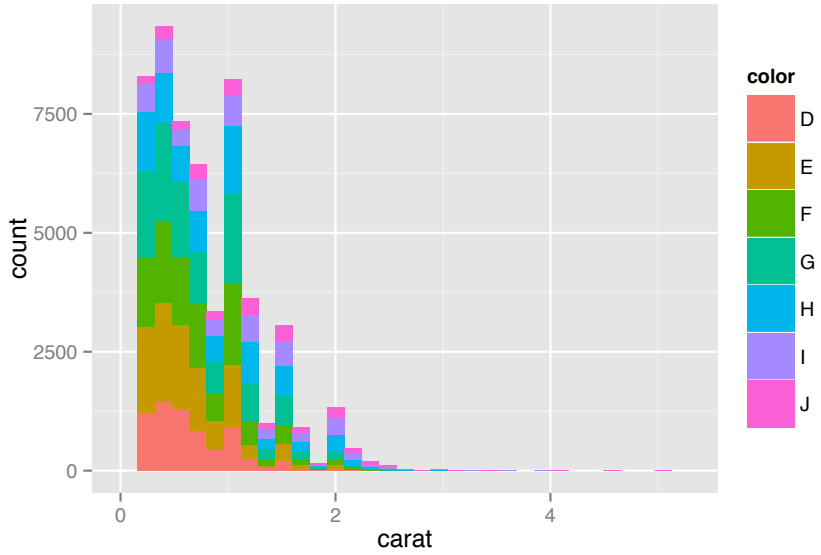

```
qplot(carat, data=diamonds, geom="density")
```



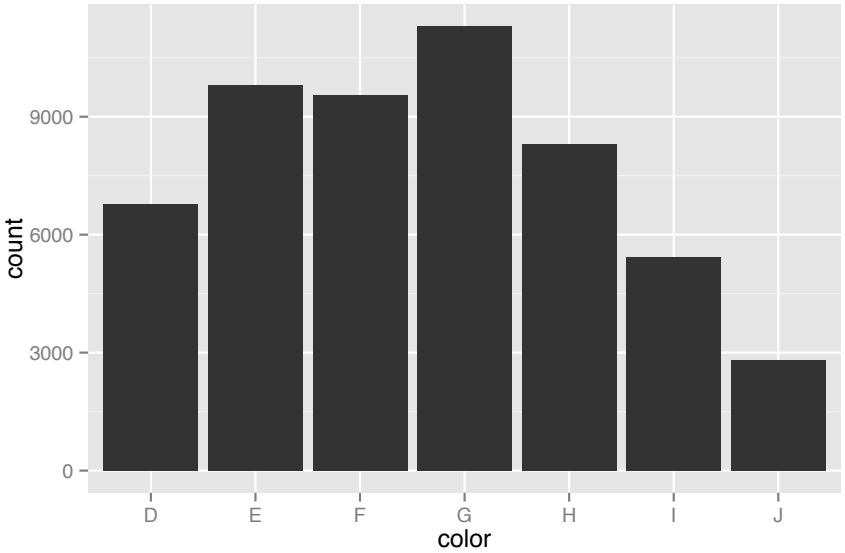
```
qplot(carat, data=diamonds, geom="density", color=color)
```




```
qplot(carat, data=diamonds, geom="histogram", fill=color)
```



```
qplot(color, data=diamonds, geom="bar")
```



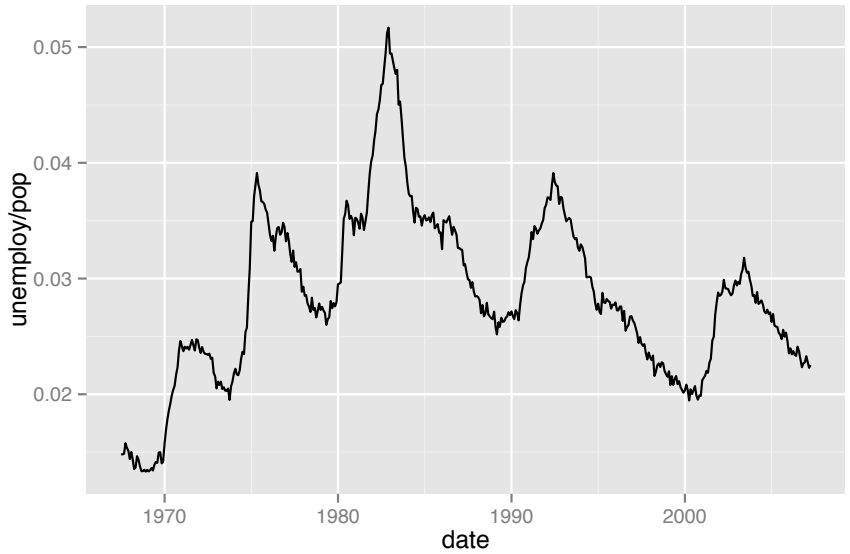
Time series data

- Line plots
  - join points from left to right, e.g., time
- Path plots
  - join points in order of appearance
  - usually to show how two variables simultaneously change over time (e.g., a walk path)

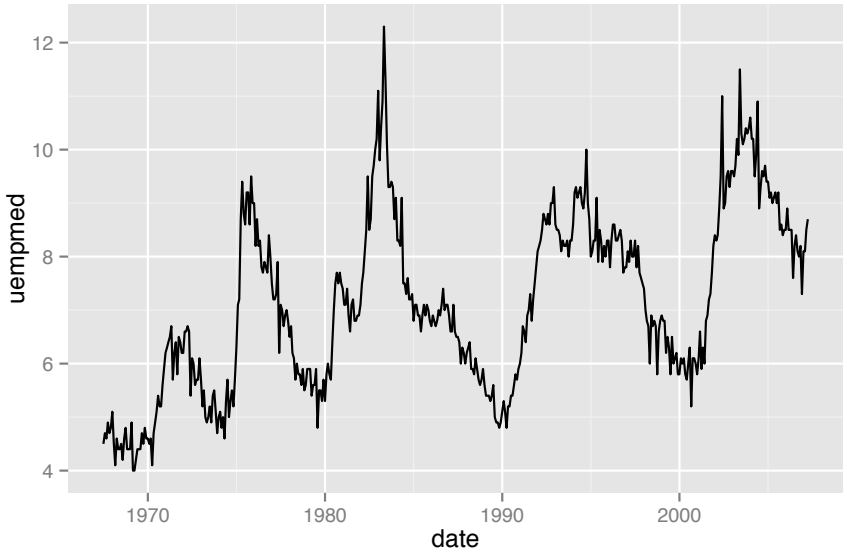
Dataset `economics` (example ggplot2 package)

	date	pce	pop	psavert	uempmed	unemploy
1	1967-06-30	507.8	198712	9.8	4.5	2944
2	1967-07-31	510.9	198911	9.8	4.7	2945
3	1967-08-31	516.7	199113	9	4.6	2958
4	1967-09-30	513.3	199311	9.8	4.9	3143
5	1967-10-31	518.5	199498	9.7	4.7	3066
6	1967-11-30	526.2	199657	9.4	4.8	3018
7	1967-12-31	532	199808	9	5.1	2878
8	1968-01-31	534.7	199920	9.5	4.5	3001
9	1968-02-29	545.4	200056	8.9	4.1	2877
10	1968-03-31	545.1	200208	9.6	4.6	2709

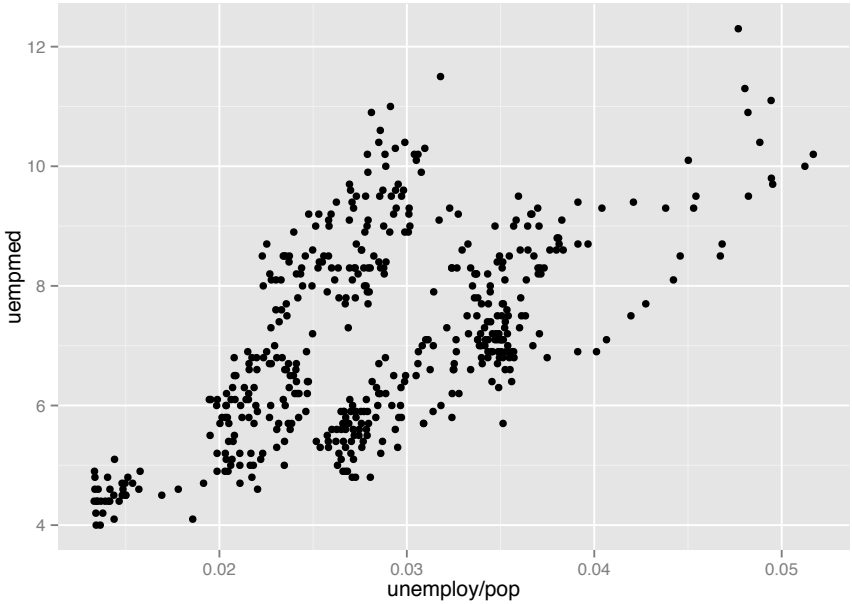

```
qplot(date, unemploy/pop, data=economics, geom="line")
```



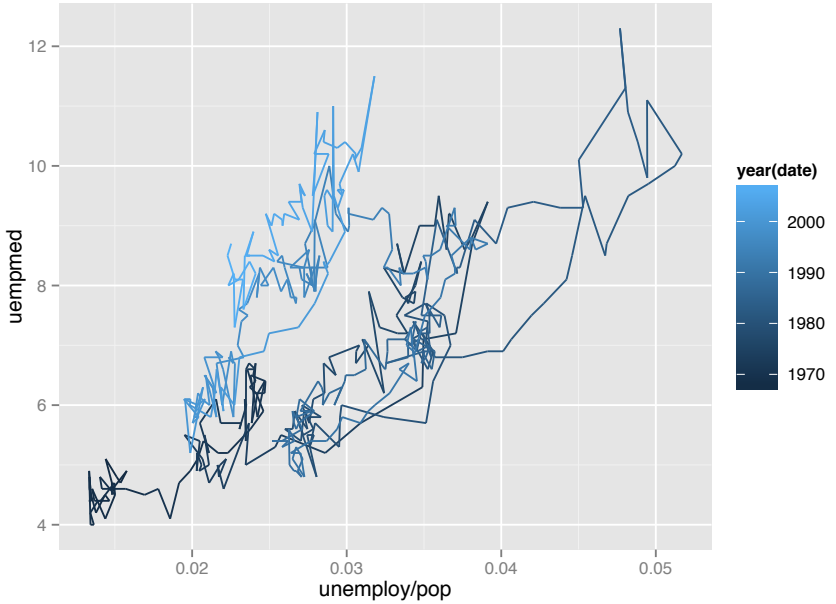
```
qplot(date, uempmed, data=economics, geom="line")
```




```
qplot(unemploy/pop, uempmed, data=economics, geom="point")
```



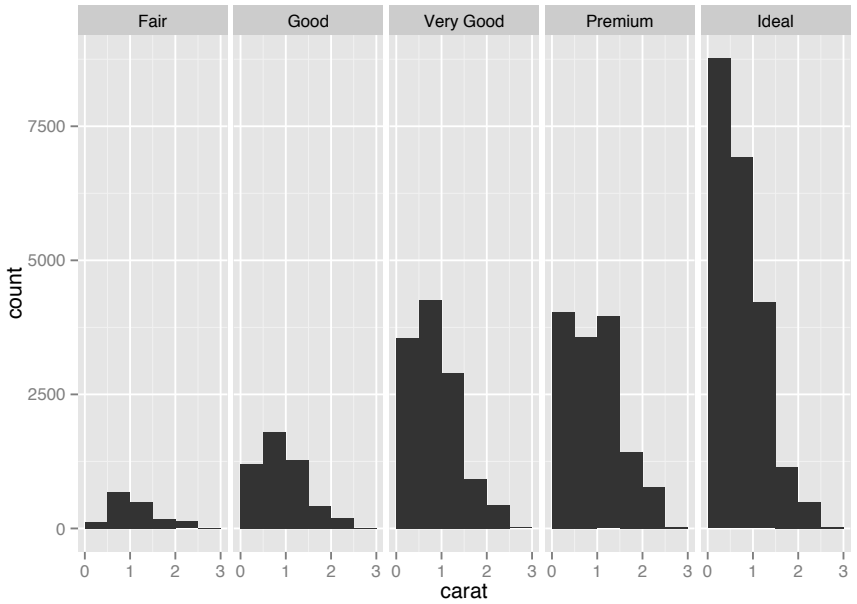
```
> library(lubridate) # for function year()
> qplot(unemploy/pop, uempmed, data=economics, geom="path",
color=year(date))
```



Faceting

- Alternative approach to visualizing data for separate groups or conditions
- Faceting splits data in subsets
- Visualizes subset in a grid of plots by faceting formula: `row_var ~ col_var`
- Only create single column: `row_var ~ .`
- Only create single row: `. ~ col_var`

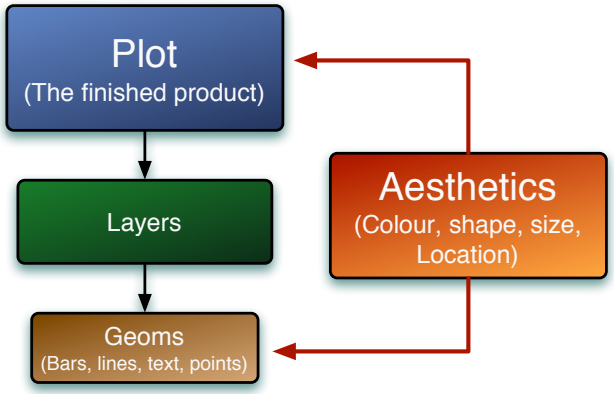
```
qplot(carat, data=diamonds, facets=.~cut, geom="histogram",
                binwidth=0.5, xlim=c(0,3))
```



Some other options

Parameter	Function
`xlim, ylim`	set limits for either axis, e.g., `c(0, 20)`
`main`	main title for plot, e.g., `main="cool plot"`
`xlab, ylab`	set labels for either axis, e.g., xlab=`"cool x axis"`

So what about this grammar?



Building a plot

```
> p <- ggplot(diamonds, aes(x=carat, y=price))
```

Adding a layer

```
> p <- ggplot(diamonds, aes(x=carat, y=price))

> p <- p + geom_point(aes(color=cut))
```

Displaying/saving the plot

```
> p <- ggplot(diamonds, aes(x=carat, y=price))

> p <- p + geom_point(aes(color=cut))

show graph window: > p or > print(p)

> ggsave("plot.pdf", p)
```

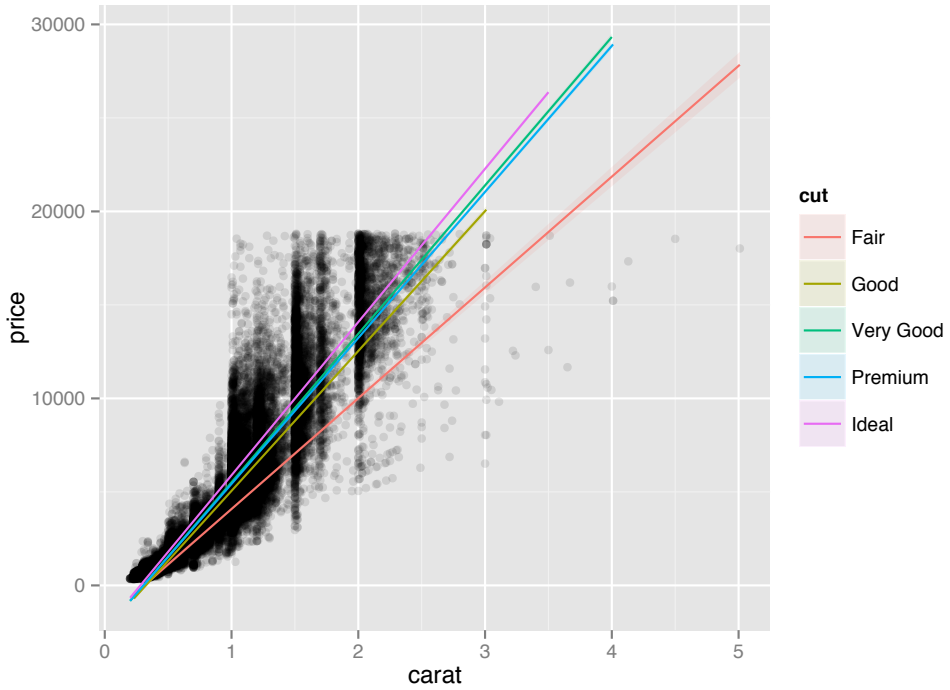
ggsave()

- Parameters are optional (default: save last plot in current working directory)
- Extension of file name defines type of image (pdf/ png/jpg, etc)
- `?ggsave`

Adding another layer

```
> p <- ggplot(diamonds, aes(x=carat, y=price))

> p <- p + geom_point(alpha=.1)

> p <- p + geom_smooth(method='lm',
aes(color=cut, fill=cut), alpha=.1)
```

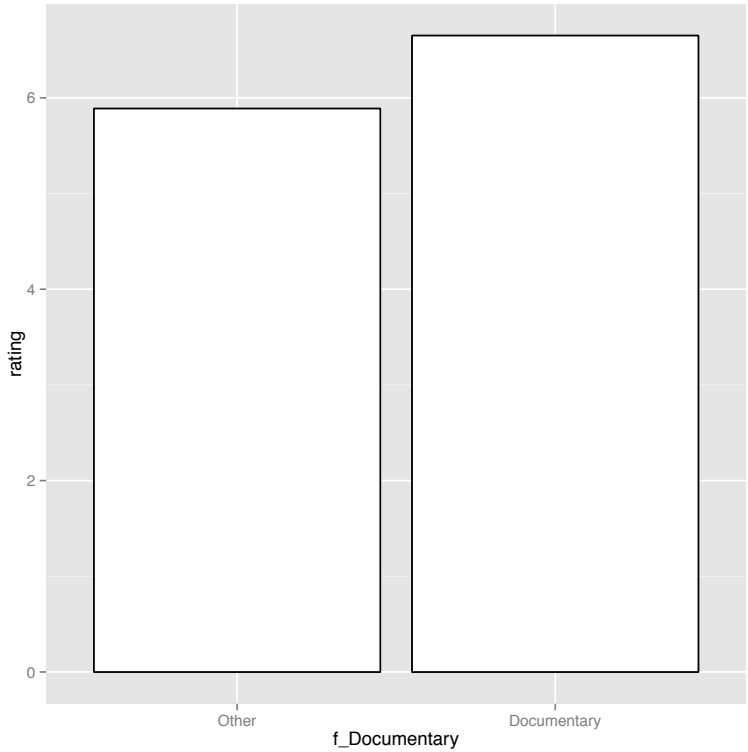
–> How does this graph look?



Bar chart

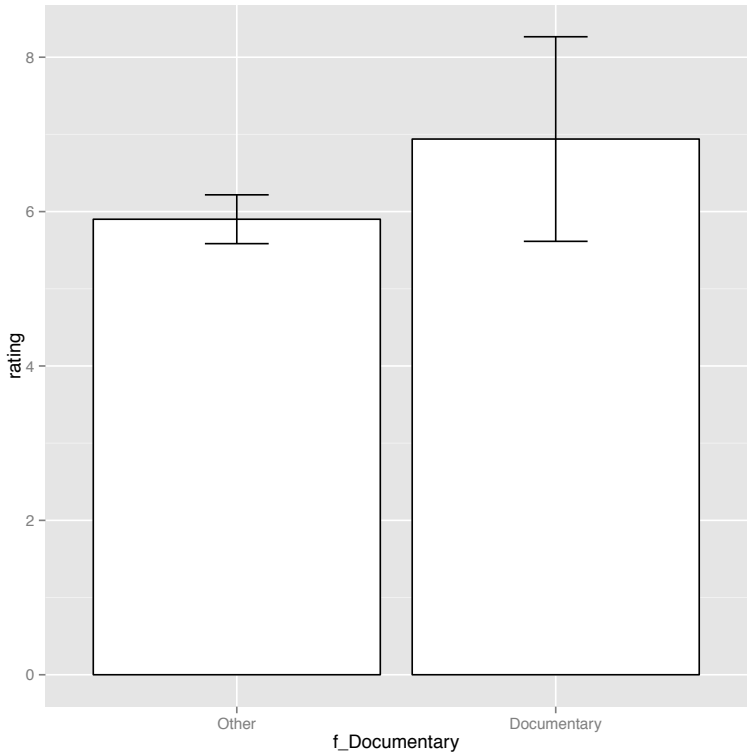
Bar chart summarizes data, therefore we need to tell ggplot how:

```
> movies$f_Documentary <- factor(movies
$Documentary, labels = c('Other', 'Documentary'))

> p <- ggplot(movies, aes(f_Documentary, rating))

> p <- p + stat_summary(fun.y = mean, geom = "bar",
fill = 'white', color = 'black')

> p
```



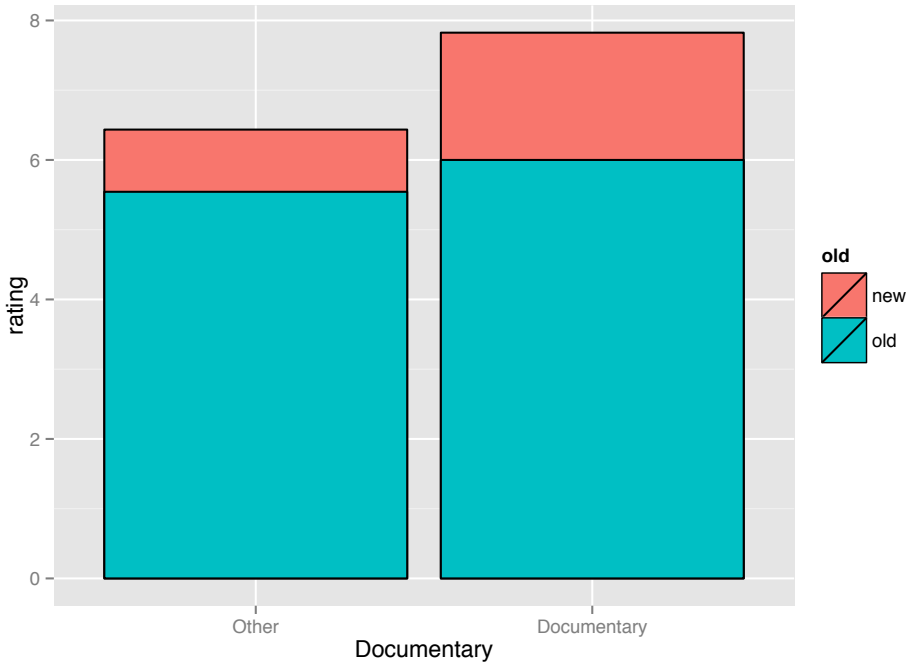
Add error bars

```
> msmall <- movies[sample(nrow(movies), 100),]

> msmall$f_Documentary <- factor(msmall$Documentary,
labels=c('Other', 'Documentary'))

> p2 <- ggplot(msmall, aes(f_Documentary, rating)) +
stat_summary(fun.y = mean, geom = "bar", fill = 'white',
color = 'black')

> p2 <- p2 + stat_summary(fun.data = mean_cl_normal, geom =
"errorbar", width = 0.2)

> p2
```



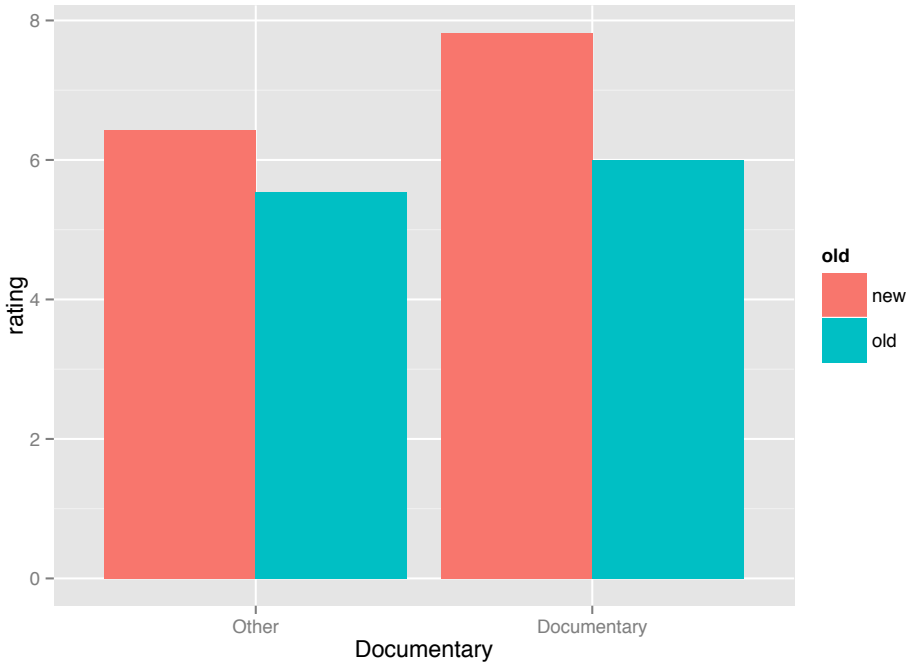
2 x 2

```
> msmall$old <- ifelse(msmall$year < 2000,
"old", "new")

> p3 <- ggplot(msmall, aes(f_Documentary,
rating)) + stat_summary(fun.y = mean, geom =
"bar", aes(fill = old), color = 'black')

> p3
```



2 x 2

```
> msmall$old <- ifelse(msmall$year < 2000,
"old", "new")

> p3 <- ggplot(msmall, aes(f_Documentary,
rating)) + stat_summary(fun.y=mean,
geom="bar", aes(fill=old), position='dodge')

> p3
```



2 x 2 + error bars

```
> msmall$old <- ifelse(msmall$year < 2000, "old",
"new")

> p3 <- ggplot(msmall, aes(f_Documentary, rating,
fill=old)) + stat_summary(fun.y=mean, geom="bar",
position='dodge')

> p3 <- p3 + stat_summary(fun.data=mean_cl_normal,
geom="errorbar",
position=position_dodge(width=0.9), width=.1)

> p3
```

2 x 2 + error bars

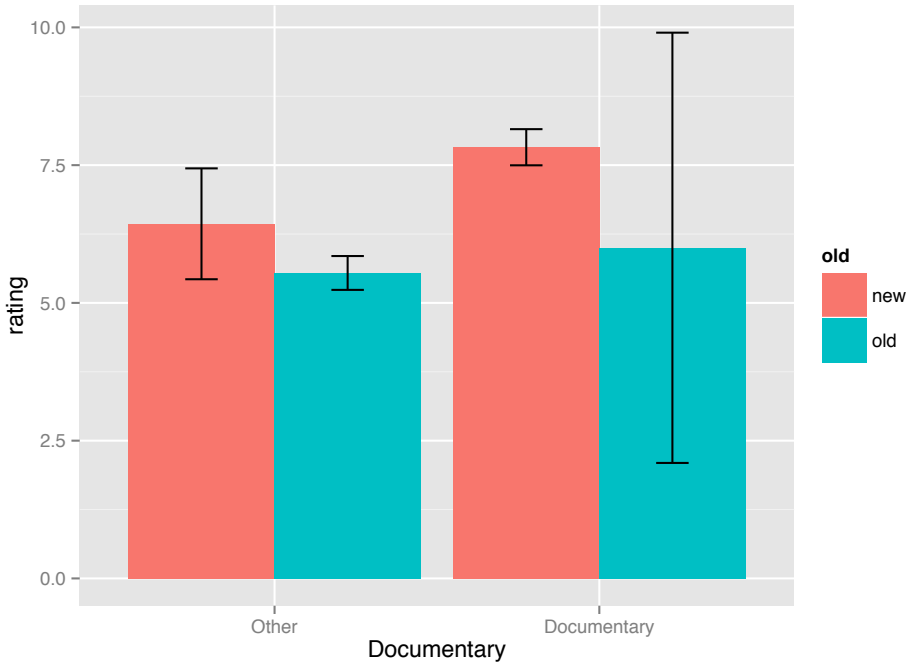
This time, fill is in the main ggplot command
—> error bar stat_summary is aware of grouping

```
msmall$year < 2000, "old",

> p3 <- ggplot(msmall, aes(f_Documentary, rating,
fill=old)) + stat_summary(fun.y=mean, geom="bar",
position='dodge')

> p3 <- p3 + stat_summary(fun.data=mean_cl_normal,
geom="errorbar",
position=position_dodge(width=0.9), width=.1)

> p3
```

ror bars

Positioning the error bars is more complex, because they don't know about the position of the respective bars. 0.9 is always the correct value!

```
l$year < 2000, "old",

> p3 <- gplot(msmall, aes(f_Documentary, rating,
fill=old) + stat_summary(fun.y=mean, geom="bar",
position 'dodge')

> p3 <- 3 + stat_summary(fun.data=mean_cl_normal,
geom="errorbar",
position=position_dodge(width=0.9), width=.1)

> p3
```

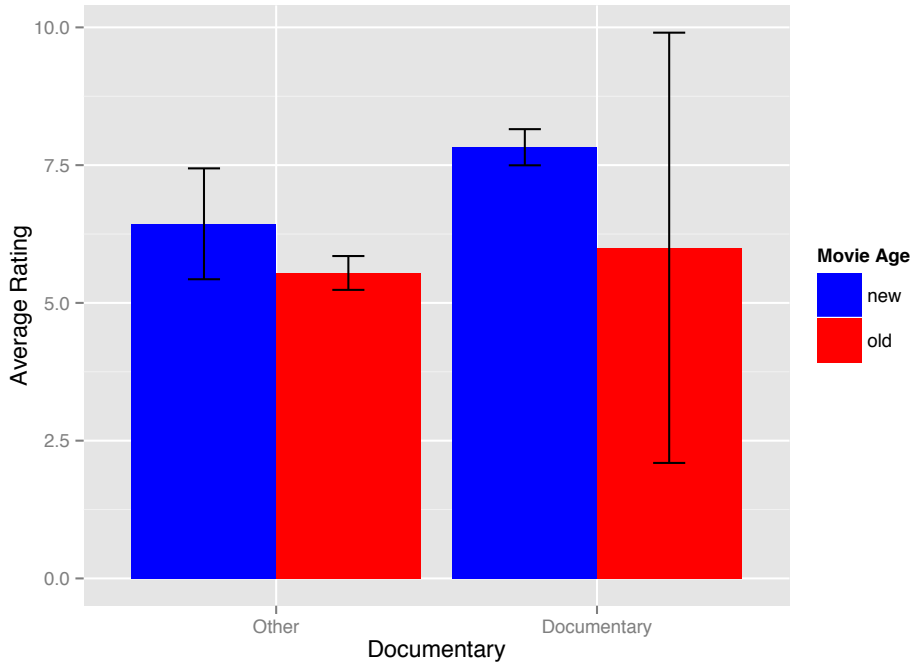


Finishing touches

- Labels
- Colors
- Themes

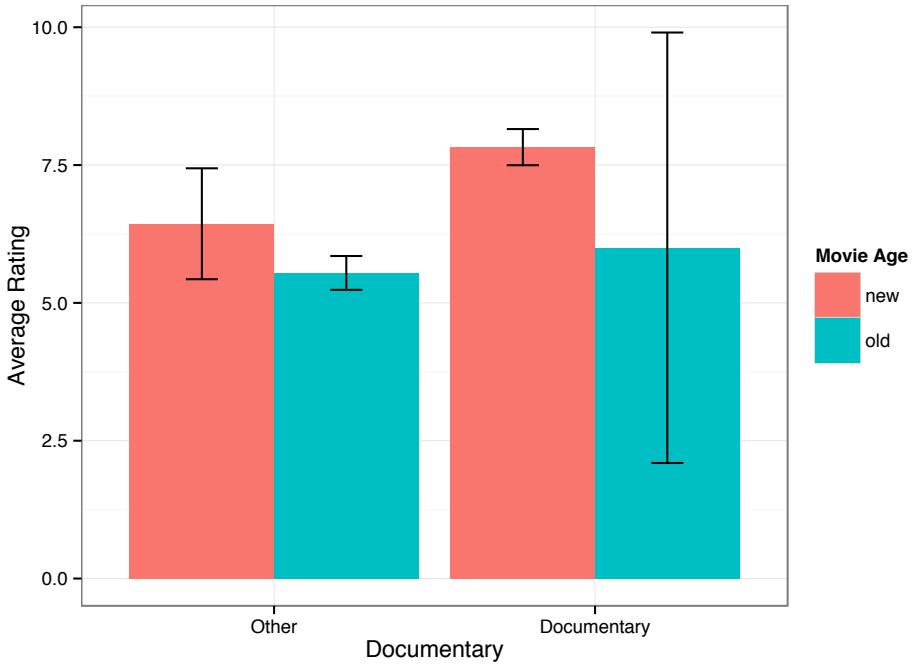
Improving labels and colors

```
> p3 <- p3 + labs(x="Documentary", y="Average Rating", fill="Movie Age")

> p3 <- p3 + scale_fill_manual(values=c("red", "blue"))

> p3
```



Adding a theme

```
> p3 <- p3 + theme_bw()

> p3

> ?ggtheme
```



Importing even more themes

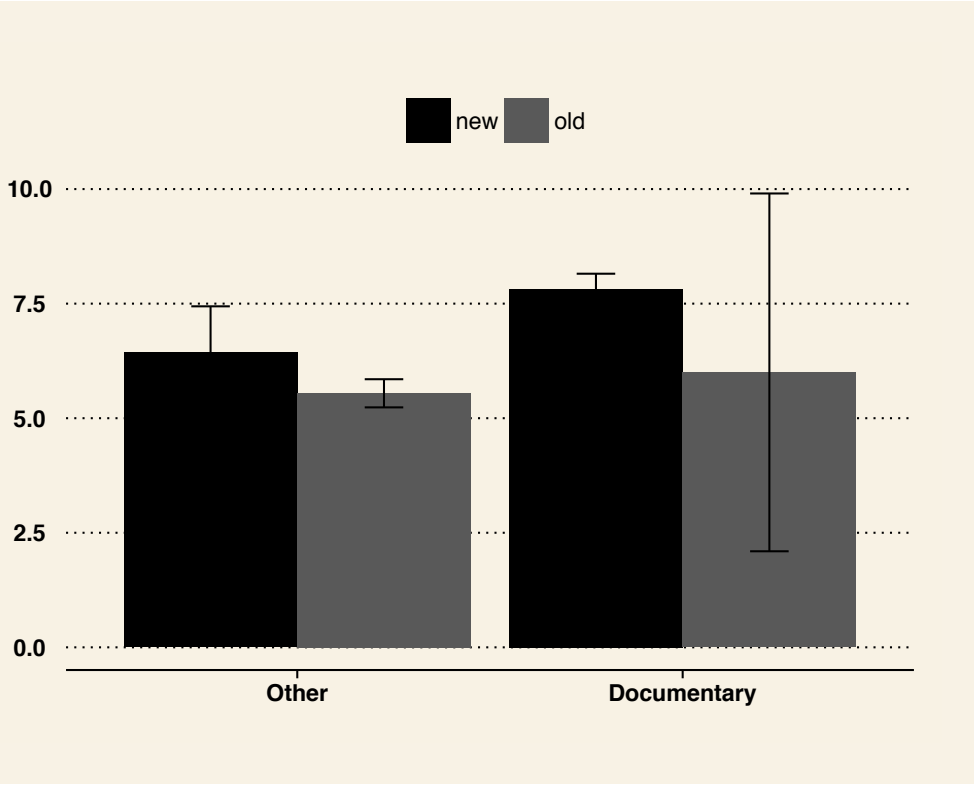
```
> install.packages('ggthemes')

> library(ggthemes)

> p3 <- p3 + scale_fill_wsj("black_green", "")
+ theme_wsj()

> p3
```

Importing even more themes

```
> install.packages(

> library(ggthemes)

> p3 <- p3 + scale_fill_wsj("black_green", "")
+ theme_wsj()

> p3
```

How bars are filled:
?theme_wsj

How the rest of the graph looks



ggthemes

- Available themes and palettes: `> ?ggthemes_data`
- http://cran.r-project.org/web/packages/ggthemes/ggthemes.pdf

All grammar components

http://docs.ggplot2.org/current/

Considerations

- `qplot` for easy graphs with a single layer
- `ggplot` for more complex graphs with multiple layers
- layout parameters go in `aes()` if they change with values in data (for constants use `I()` in `qplot`)
- put `aes` parameters in `ggplot()` command if possible
- Use `stat_summary()` to visualize geoms that summarize data (e.g., means, error bars)

Homework

- Instructions on BB (Week 3): Do a series of plots and interpret them
- Reading: FMF chapters 6 and 7 (+ 5, if you haven't read it for today)

—> Next week: linear regression

Enjoy ggplot2!