SI 618
Week 2:
Data frames, summary statistics, Basics of ggplot2 and plyr

Instructor: Dr. Chris Teplovs (cteplovs@umich.edu)
Lead Developer, Digital Innovation Greenhouse, Office of Academic Innovation
Adjunct Lecturer of Information, School of Information
GSI: SungJin Nam (sjnam@umich.edu)

Class updates

- No CT office hours Wednesday 11/9
- Moved to TUESDAY 11/8 3:00-5:00pm

SI 618 Data Exploration: Class Schedule

(Some curriculum details subject to change)

Date	Topic	Assignments Due
Week 1	Course introduction Basics of Programming with R	
Week 2	Basic analysis and visualization using ggplot2: qplot() Manipulating data frames using plyr	Homework 1
Week 3	Smoothing and Trend-finding. Building ggplot Layer by Layer	Homework 2
Week 4	Finding relationships between variables Time series and autocorrelation	Homework 3
Week 5	Clustering and Finding Outliers	Homework 4
Week 6	Factor Analysis Methods (PCA, EFA)	Homework 5
Week 7	Advanced topics **Project Presentations**	Project Due

Class Outline

- Introduction to basic data analysis
- How to use qplot() in ggplot2
  - Scatterplot
  - Faceting
  - Boxplots
  - Histograms
- Subsetting, transforming, summarizing data
  - Slicing and dicing data with the plyr package

Data frames are a powerful way to keep related variables together in a package

- Data frames can combine vectors of different types
  - e.g. numeric and string (unlike cbind, rbind)

```
> df <- data.frame(a = c(1:5),
+                  b = c("cat", "dog", "emu", "fish", "giraffe"))
> df
  a       b
1 1     cat
2 2     dog
3 3     emu
4 4    fish
5 5 giraffe
> df$a
[1] 1 2 3 4 5
> df$b
[1] cat     dog     emu     fish    giraffe
Levels: cat dog emu fish giraffe
```

Exciting? Real?

- Let's generate our own data set
- This will require an index card and something to write with

Timer

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15

Repeat with non-dominant hand

Timer

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15

Do you have a data set?

- Yes and no
- Share your data
  - Times for dominant and non-dominant hand
  - Generate some other categorical (level/label) data:
    - Hair color
    - Glasses
    - Program?
    - Be imaginative, but sensitive

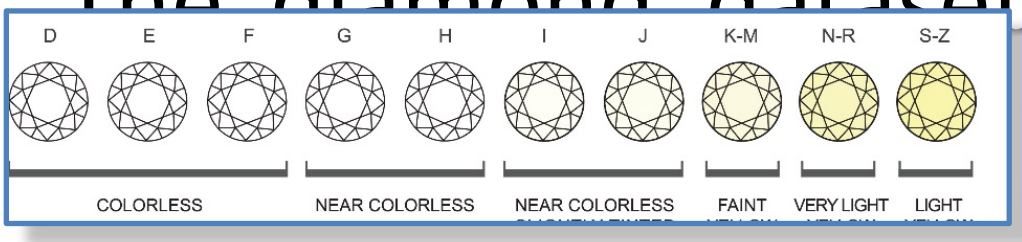
And again… Data frames are a powerful way to keep related variables together in a package

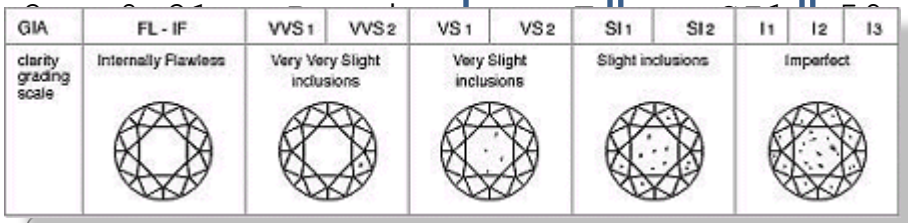
```
> df <- data.frame(dom = c(3,3,2,5,2,8),
nondom = c(10,3,4,15,4,9),
+ hair = c(”brown", ”brown", ”blond",
”red", ”purple"))
> df
```

The ‘diamond’ dataset (from ggplot2)

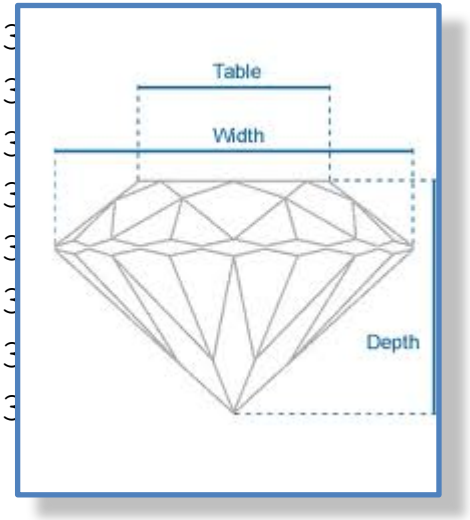
~54000 diamonds

se.info






```
> head(diamonds,15)
    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 4.20 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 4.00 4.05 2.39
11   0.30      Good     J     SI1  64.0    55   339 4.25 4.28 2.73
12   0.23     Ideal     J     VS1  62.8    56   340 3.93 3.90 2.46
13   0.22   Premium     F     SI1  60.4    61   342 3.88 3.84 2.33
14   0.31     Ideal     J     SI2  62.2    54   344 4.35 4.37 2.71
15   0.20   Premium     E     SI2  60.2    62   345 3.79 3.75 2.27
```



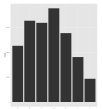
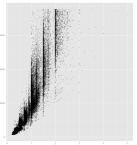
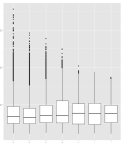
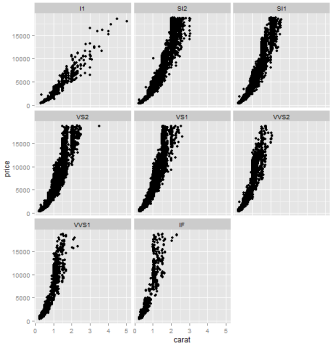
What are some questions we could answer with this data?

```
> head(diamonds,15)
   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 4.20 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
```

Some of my questions:

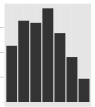
- How is price distributed?
- How is diamond price related to size (carats)? clarity? Color?
- Are some diamond colors more rare than others?
- Are the biggest diamonds usually high or low clarity?

What are some questions we could answer with this data?

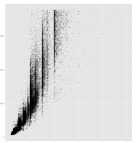
- How one variable is distributed
  - Histogram or density chart 
- How two variables are related
  - Scatterplot 
  - Boxplot 
- How two variables are related conditioned on other variables ("faceting") 

What are some questions we could answer with this data?

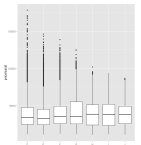
- How one variable is distributed
  - Which colors are more rare?



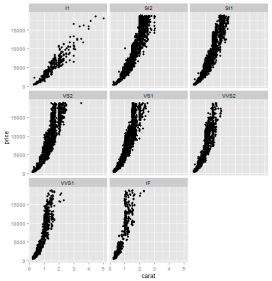
- How two variables are related
  - How are carat weight and price related?



  - How are color and price/carat related?



- How two variables are related conditioned on other variables
  - How are carat and price related, for different clarity levels?



Someone hands you a big dataset.. Now what?

Overall:

– What are the variables?

– What are the variable types and ranges?

- Categorical
- Ordinal (Ordered category)
- Continuous/Discrete numerical values

Someone hands you a big dataset.. Now what?

For each single variable:

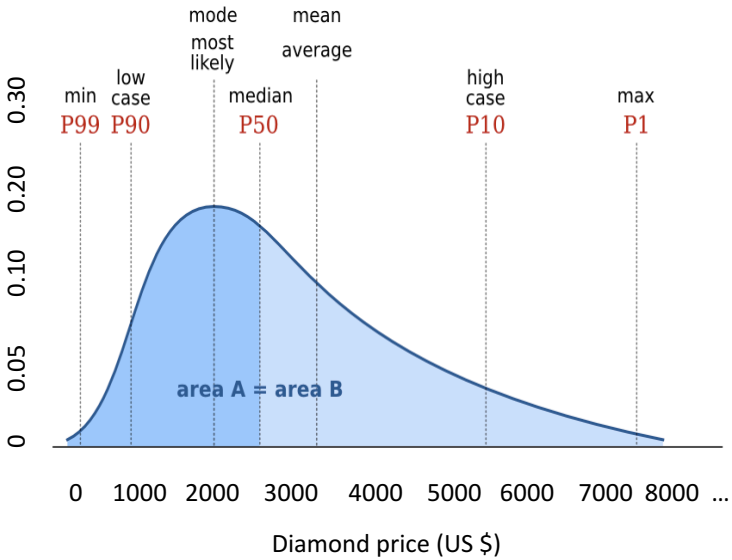
- How do values separate into groups (clusters)?
- Do the values have an asymmetric trailing off more in one direction than another?
- Unexpectedly popular or unpopular values?
- Where are the values centered?
- How widely the values are spread?

Two ways to inspect a dataset

```
> head(diamonds)
  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 4.20 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

> str(diamonds)
'data.frame':       53940 obs. of  10 variables:
 $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
 $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

How are a single variable's values distributed? What are the key properties of that distribution?

- For each variable:
  - Median (middle)
  - Mean
  - Mode
  - First, second, third, fourth quartiles
    - (bottom 25%, … , top 25%)
  - Variance
  - Extreme values
    - High cases (top 10%)
    - Low cases (bottom 90%)
    - (low, high) and *range*



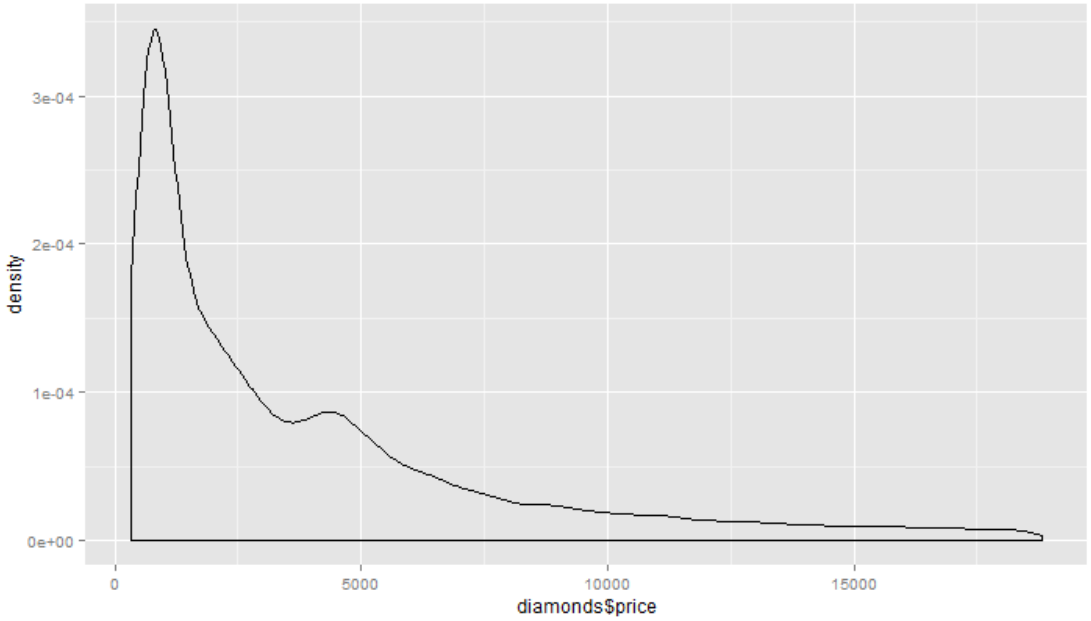
Median = $2401
Mode = $2090
Mean = $3933

Extremes: $326 (low)
$18823 (high)

Source: http://www.agilegeoscience.com/journal/tag/probability

Basic stats with summary()

```
> summary(diamonds)
     carat               cut        color        clarity          depth           table
 Min.   :0.2000   Fair      : 1610   D: 6775   SI1     :13065   Min.   :43.00   Min.   :43.00
 1st Qu.:0.4000   Good      : 4906   E: 9797   VS2     :12258   1st Qu.:61.00   1st Qu.:56.00
 Median :0.7000   Very Good:12082   F: 9542   SI2     : 9194   Median :61.80   Median :57.00
 Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171   Mean   :61.75   Mean   :57.46
 3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2    : 5066   3rd Qu.:62.50   3rd Qu.:59.00
 Max.   :5.0100                     I: 5422   VVS1    : 3655   Max.   :79.00   Max.   :95.00
                                    J: 2808   (Other): 2531

     price
 Min.   :  326
 1st Qu.:  950
 Median : 2401
 Mean   : 3933
 3rd Qu.: 5324
 Max.   :18823
```



Density
geom= "density"

The area under a probability density curve = 1

Visualization with ggplot2

Starting with qplot

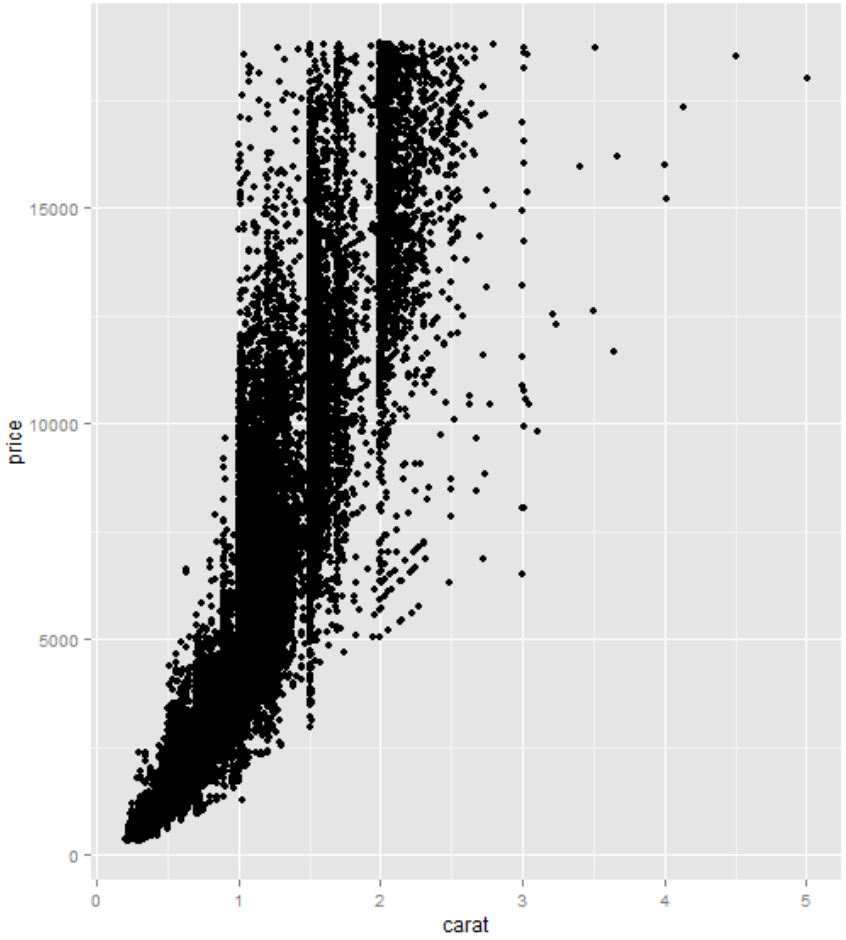
Examples: qplot.R in Files/Lectures/Week 2

examples-week2.zip

Scatterplot: shows how two continuous variables are related

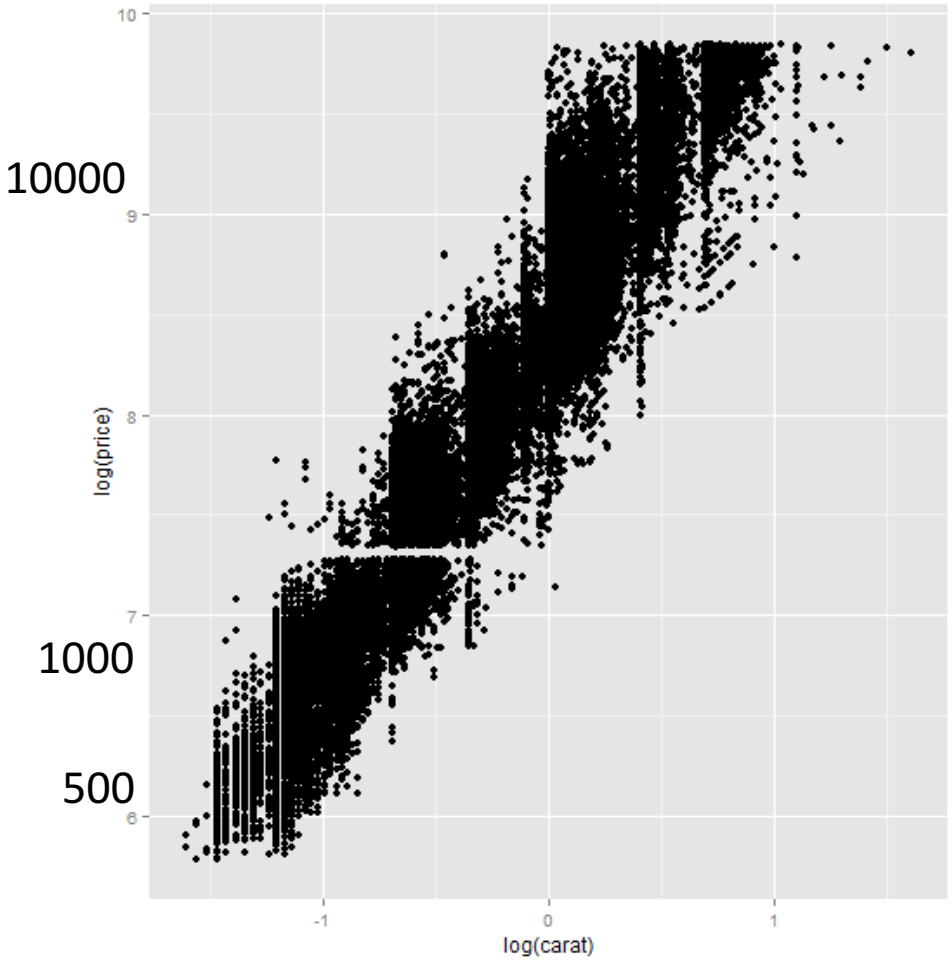
```
qplot(carat, price, data = diamonds)
```

How is diamond price related to size (carats)?



Change of scale can zoom to reveal interesting details

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



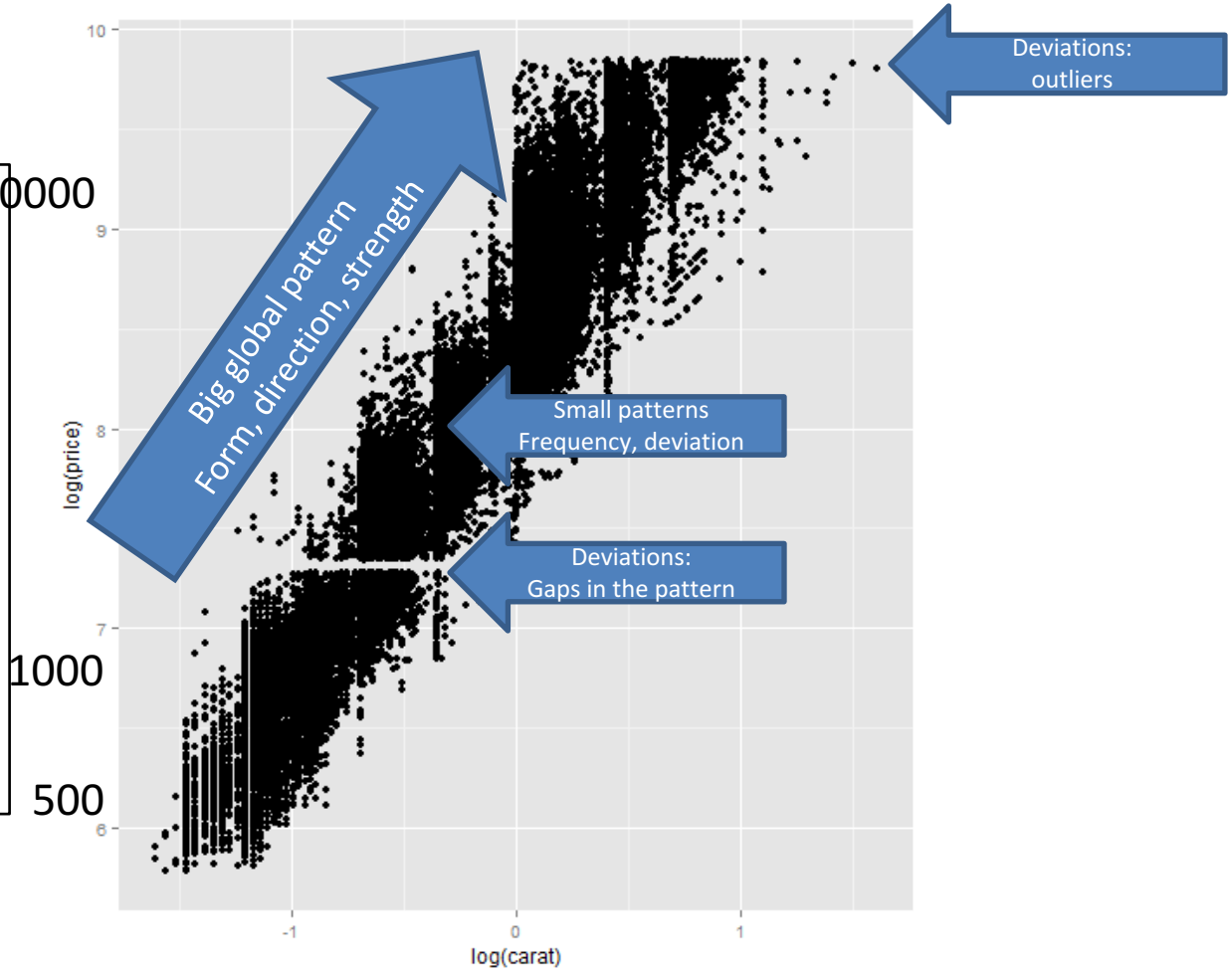
Interpreting a scatterplot

Form:
Is there a global trend?
- Linear? Curved?
- Multi-part trend?

Strength:
Is there lots of noise or variation?

Direction:
Increasing/decreasing?

Deviations:
Outliers and/or gaps?

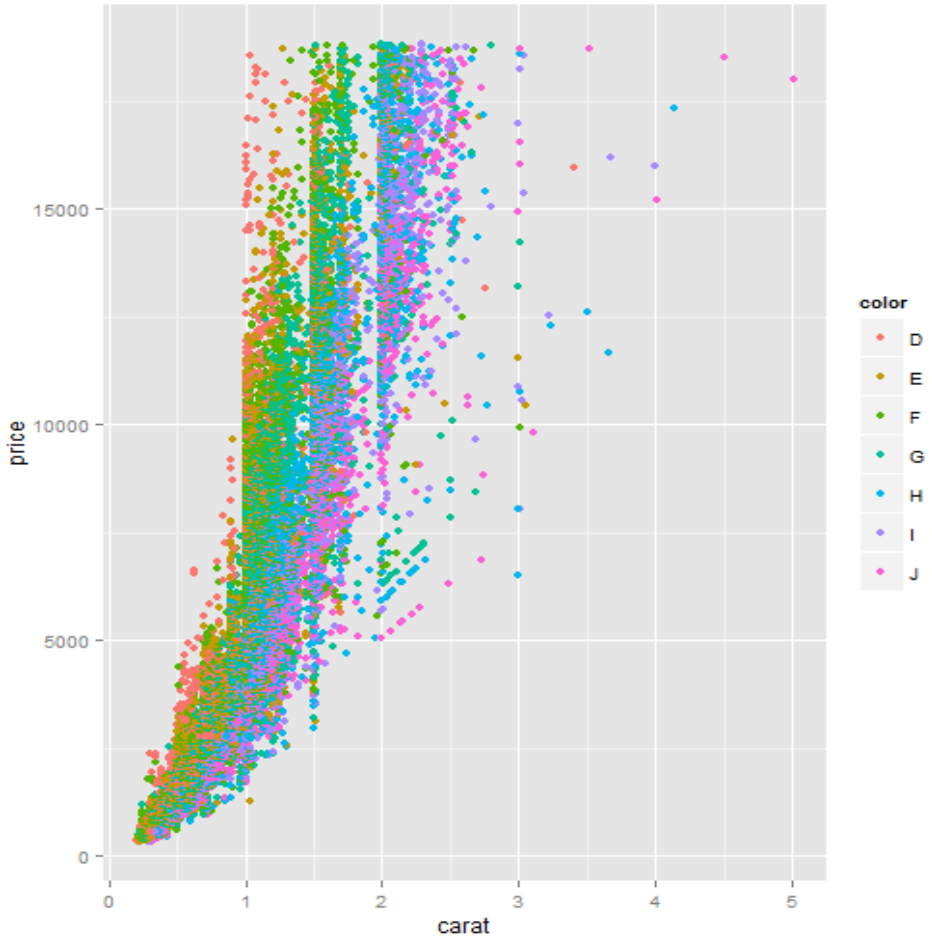


Aesthetic attributes of a plot: Mapping variables to color, size, shape, alpha…

- Visual properties that affect how data are displayed
- Each aesthetic can be mapped to a variable
  - Color and shape aesthetics work well with categorical variables
  - Size aesthetics work better for continuous variables
- Every aesthetic attribute has a *scale*
  - A scale maps data values to aesthetic values
  - ggplot will add a legend automatically if needed

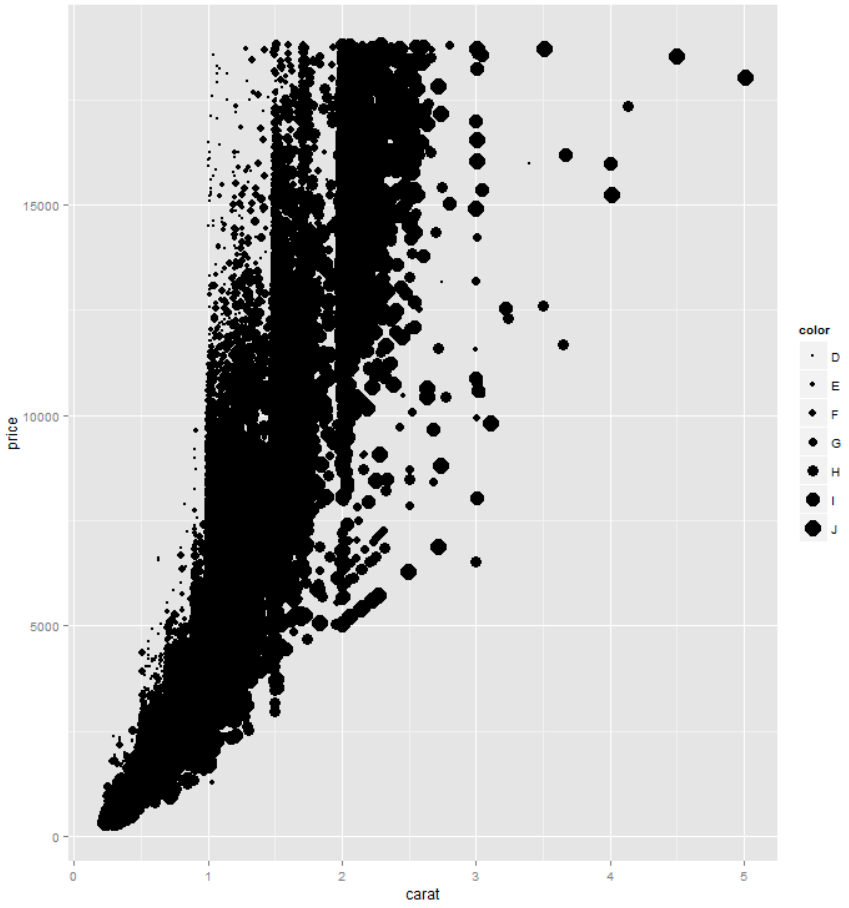
Mapping the diamond color value to the plot colour aesthetic

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



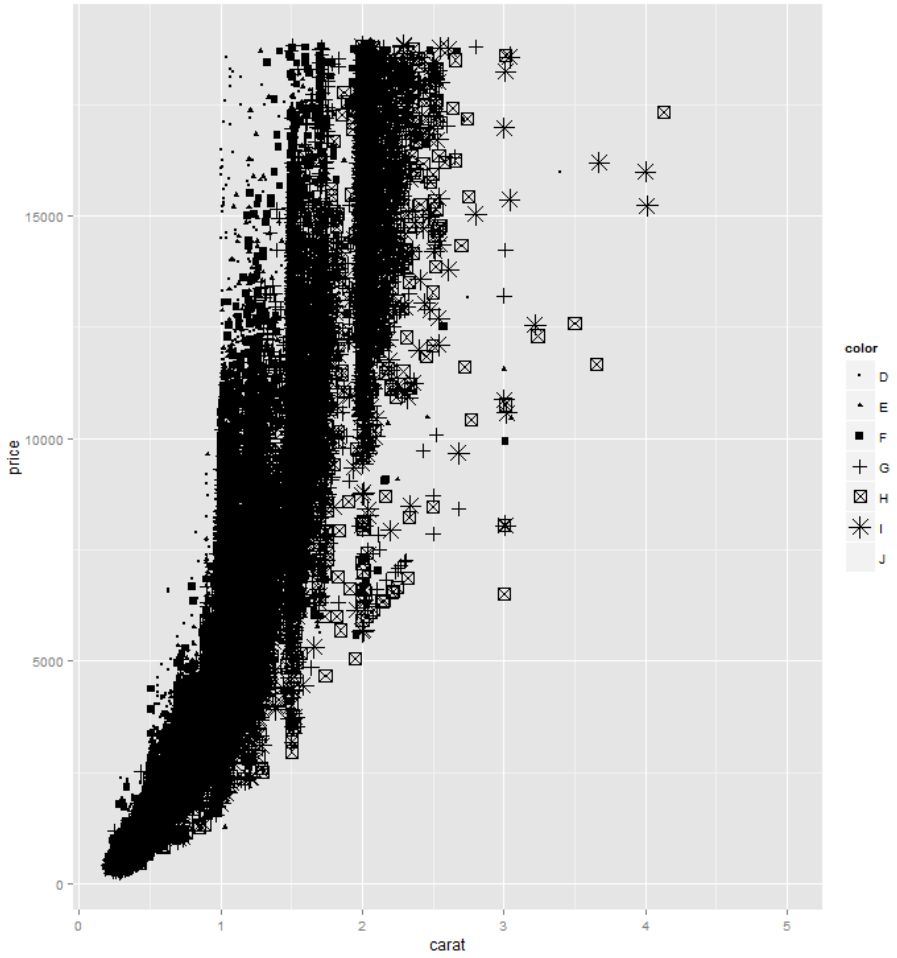
Mapping the diamond color value to the plot size aesthetic

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



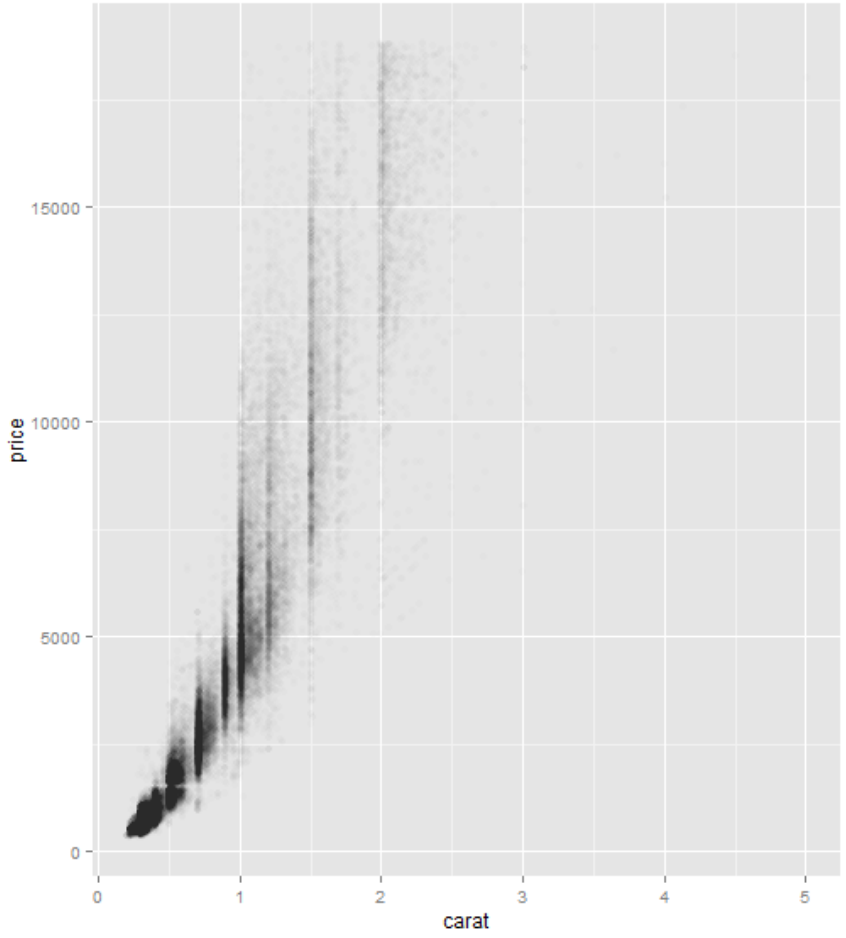
Mapping the diamond color value to the plot size and shape aesthetic

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



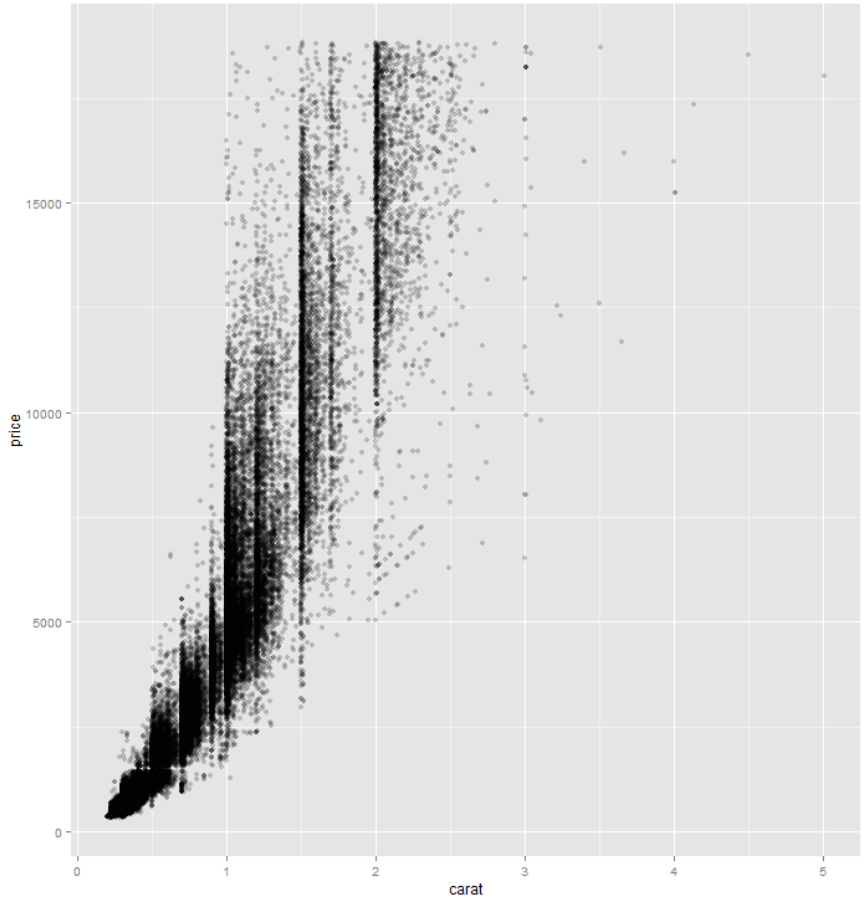
Setting the alpha aesthetic (transparency)

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



Setting the alpha aesthetic (transparency)

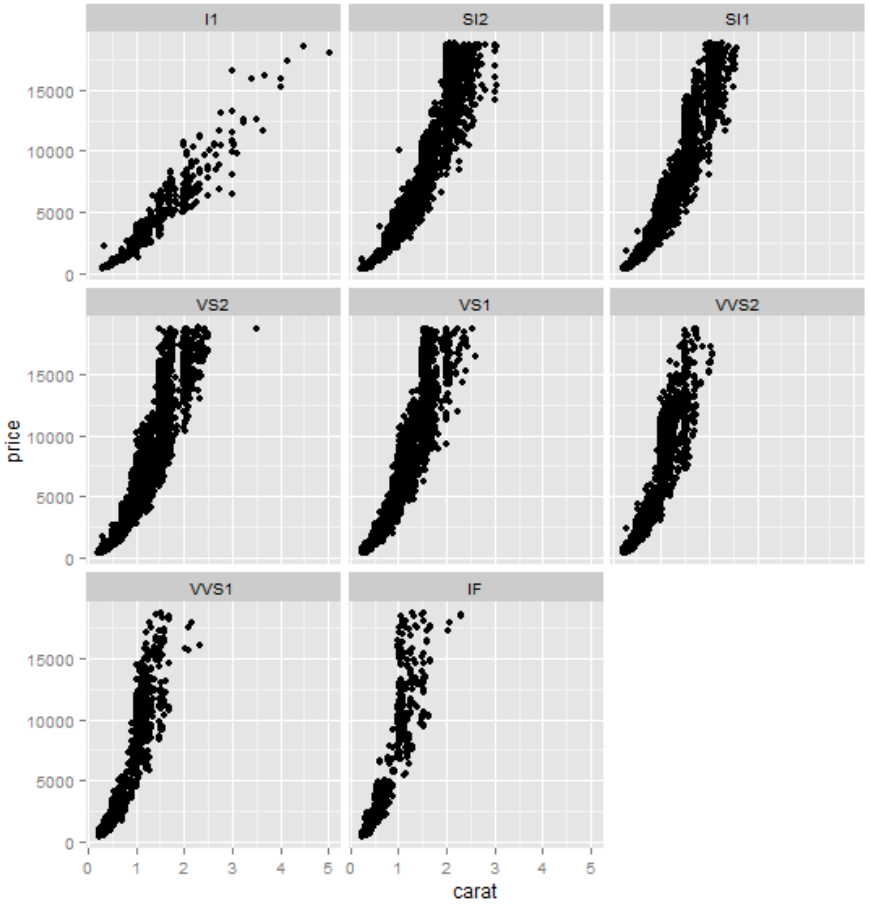
```
qplot(carat, price, data = diamonds, alpha = I(1/5))
```



Facets show the same plot for different subsets of the data

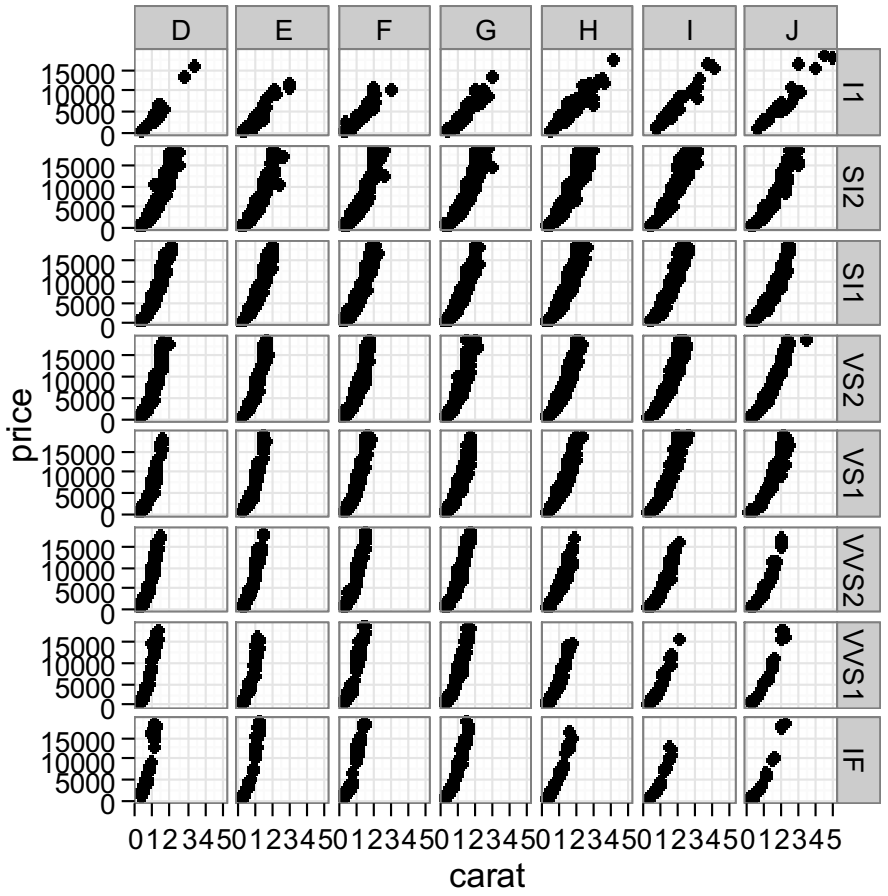
```
qplot(carat, price, data=diamonds, facets=.~clarity)
```

One facet:
clarity

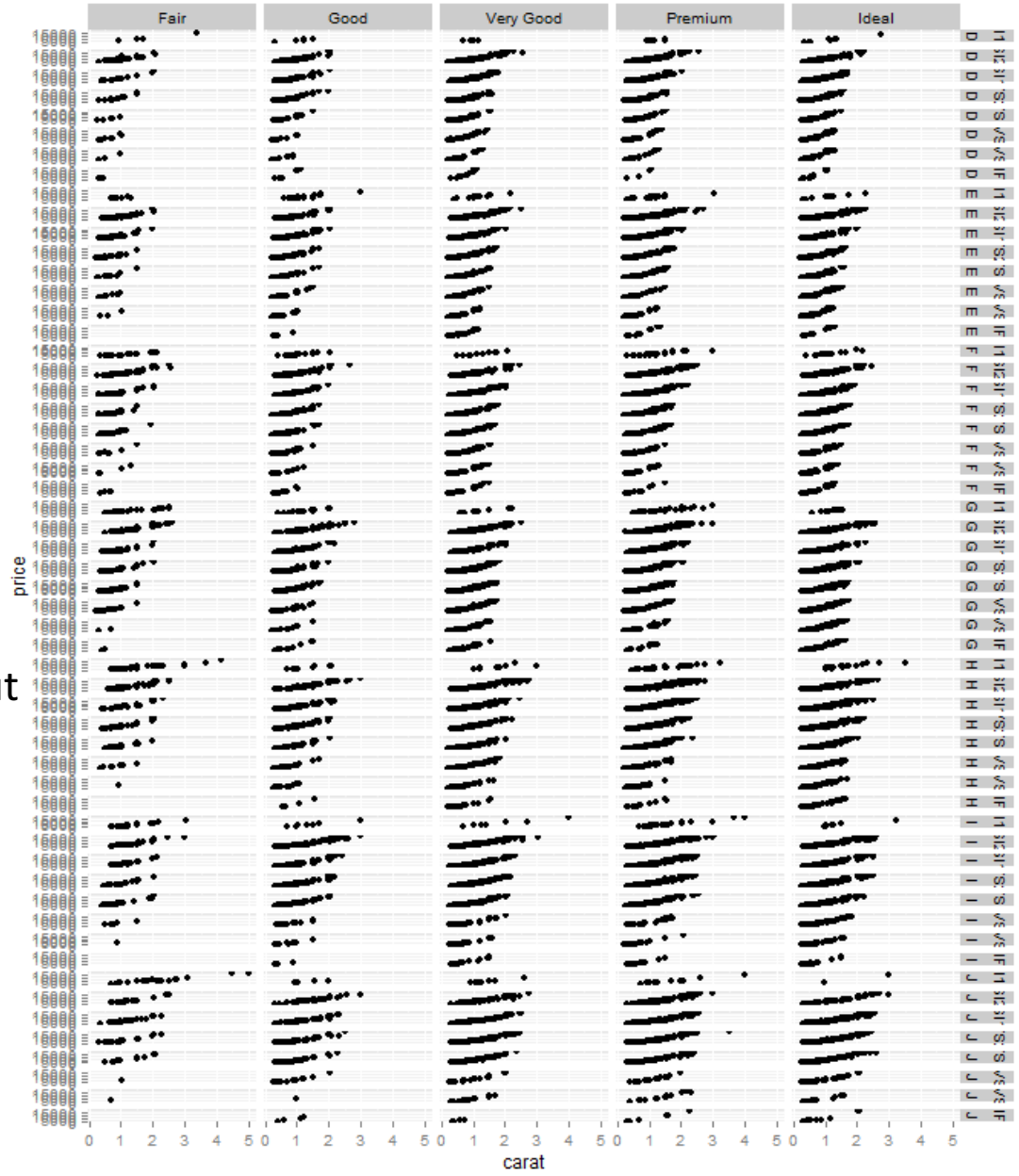


```
qplot(carat, price, data=diamonds,
      facets=clarity~color)
```

Two facets:
clarity (rows)
color (columns)

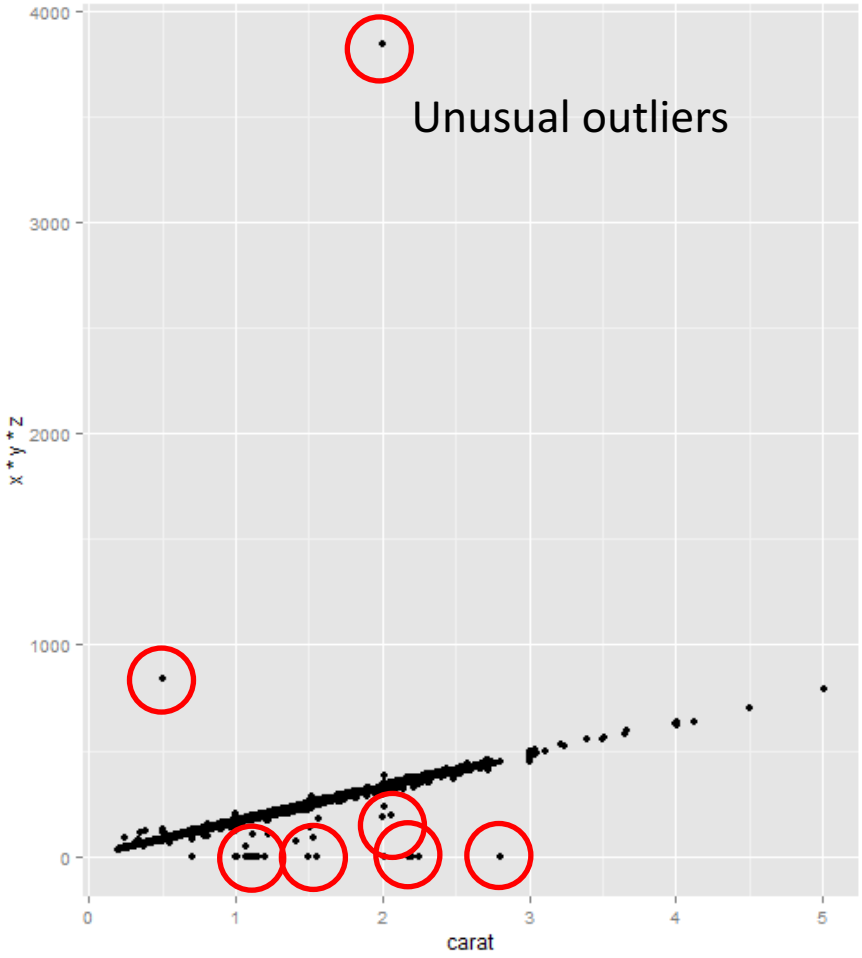


Three facets:
color, clarity, cut



What should this graphic look like?

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



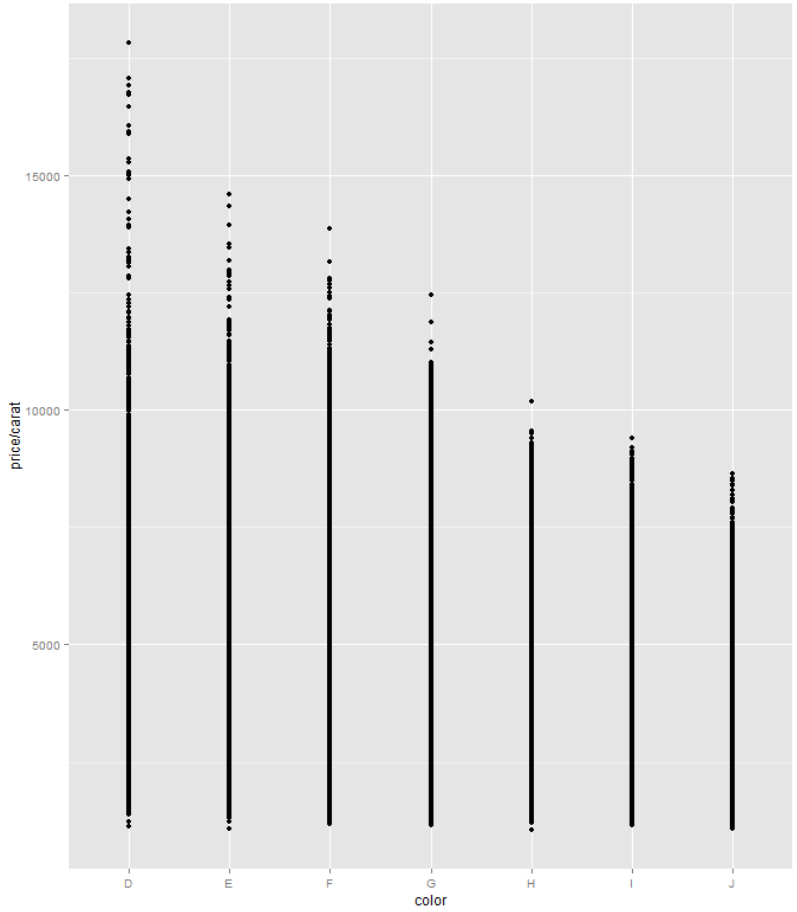
Geometric objects in ggplot: `geom="..."`

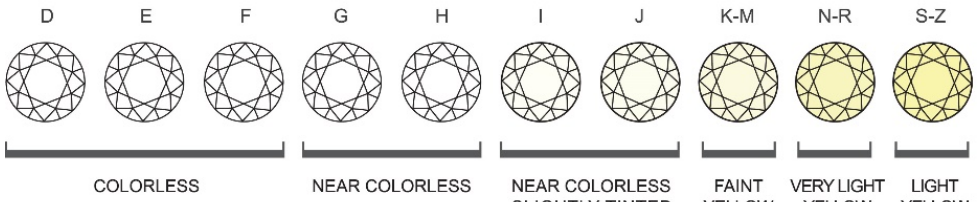
The "geom" type controls the plot you see. We can produce different plots by varying the "geom" geometry type. Three basic "geom":

- point
- jitter
- boxplot

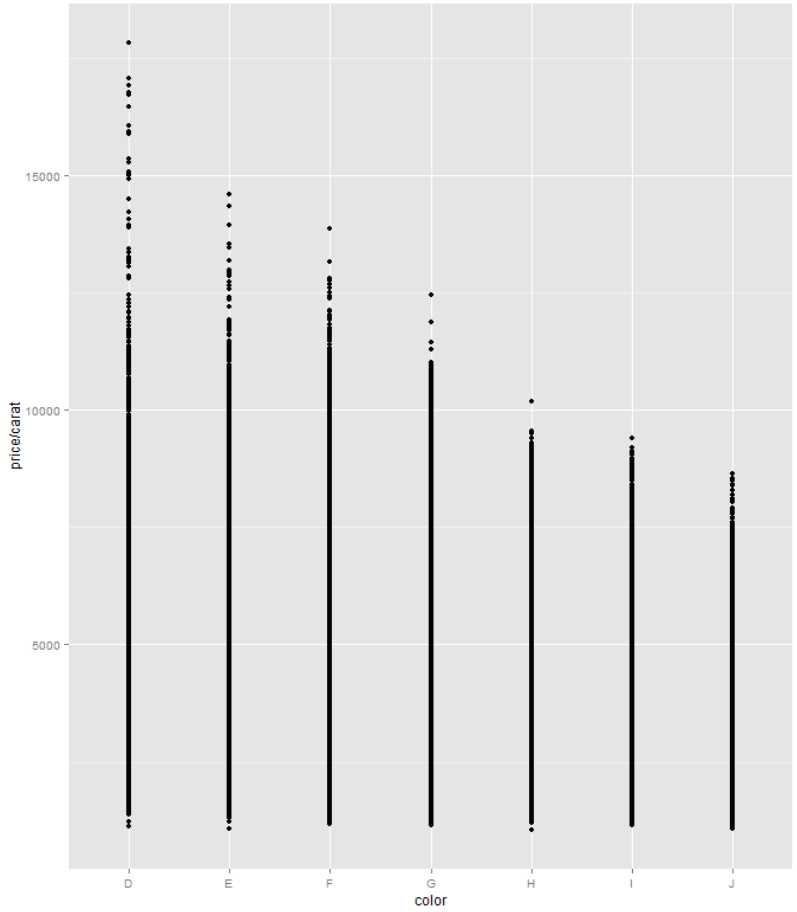
How are diamonds' value ($/carat) related to color on average?

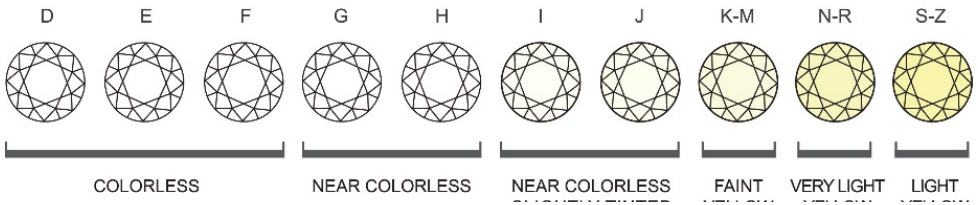
```
qplot(color, price / carat, data = diamonds,
      geom = "point")
```



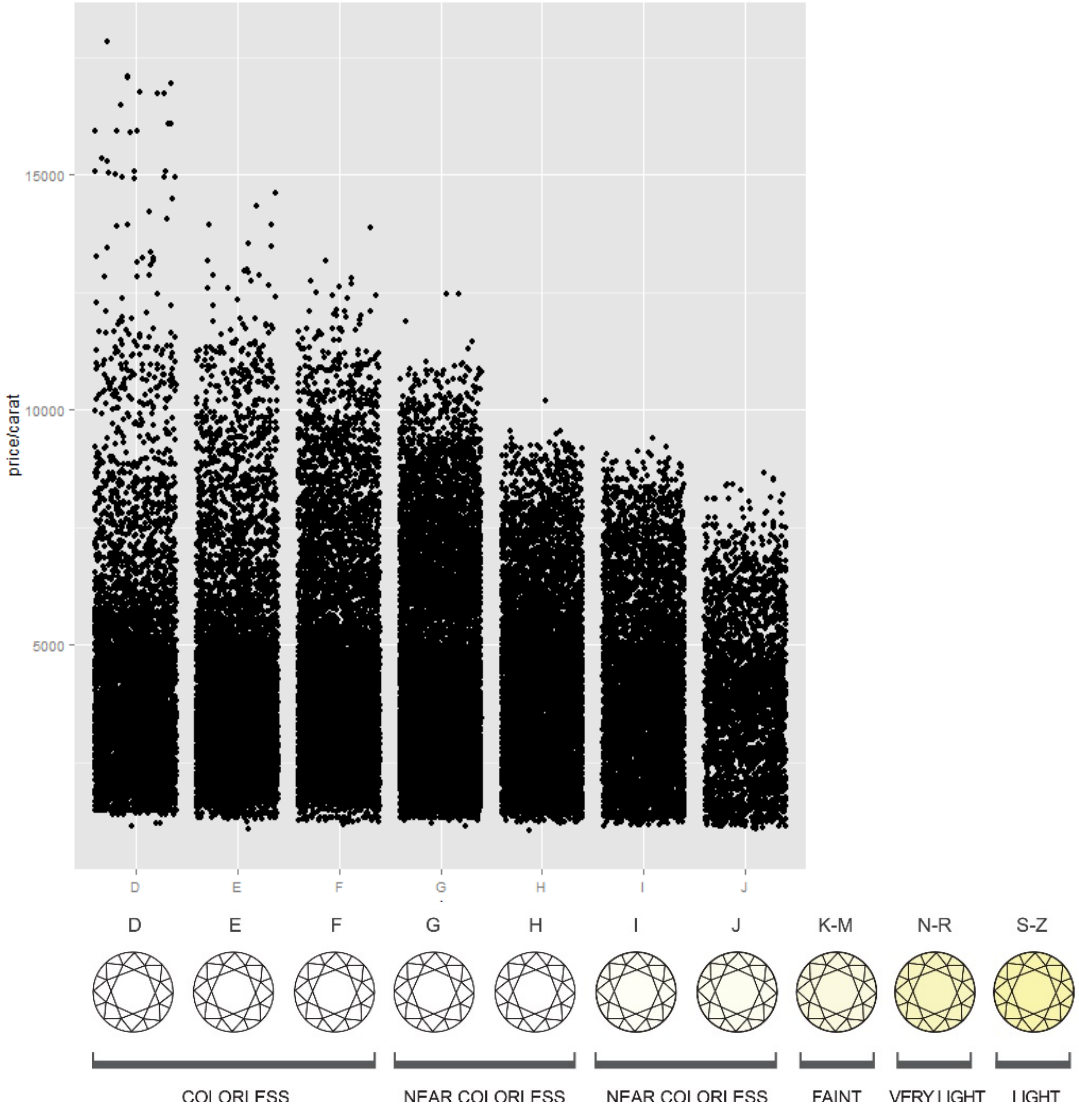


```
qplot(color, price / carat, data = diamonds,
      geom = "point")
```



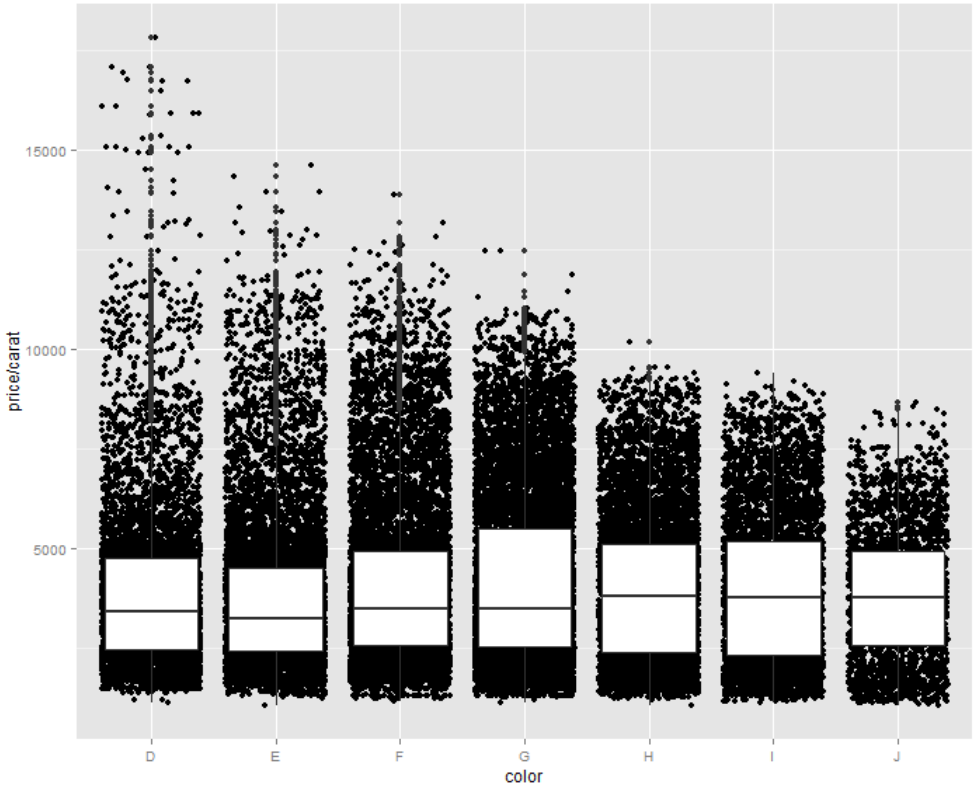


```
qplot(color, price / carat, data = diamonds,
      geom = "jitter")
```

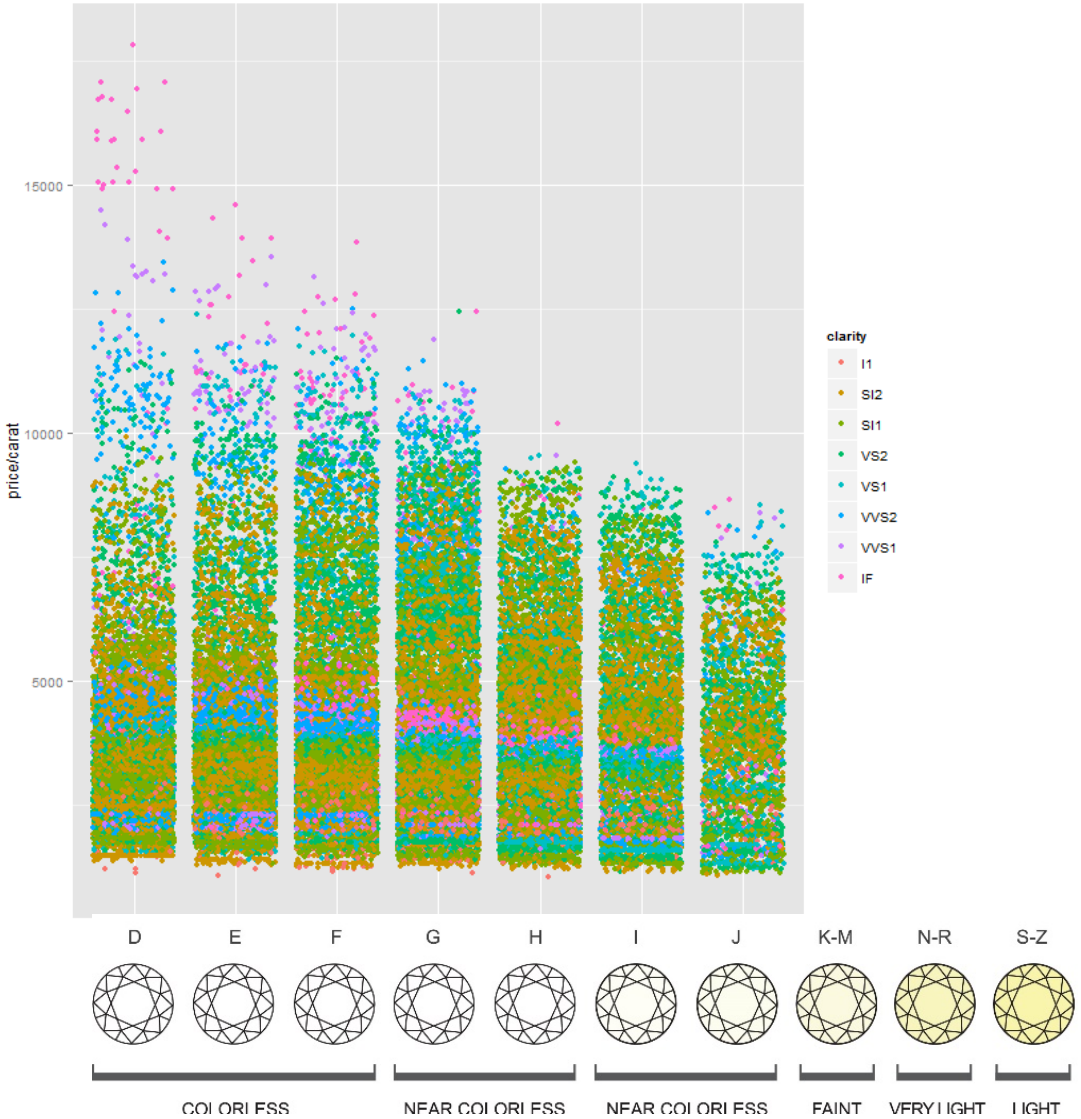


You can add extra layers by passing a vector of geom names

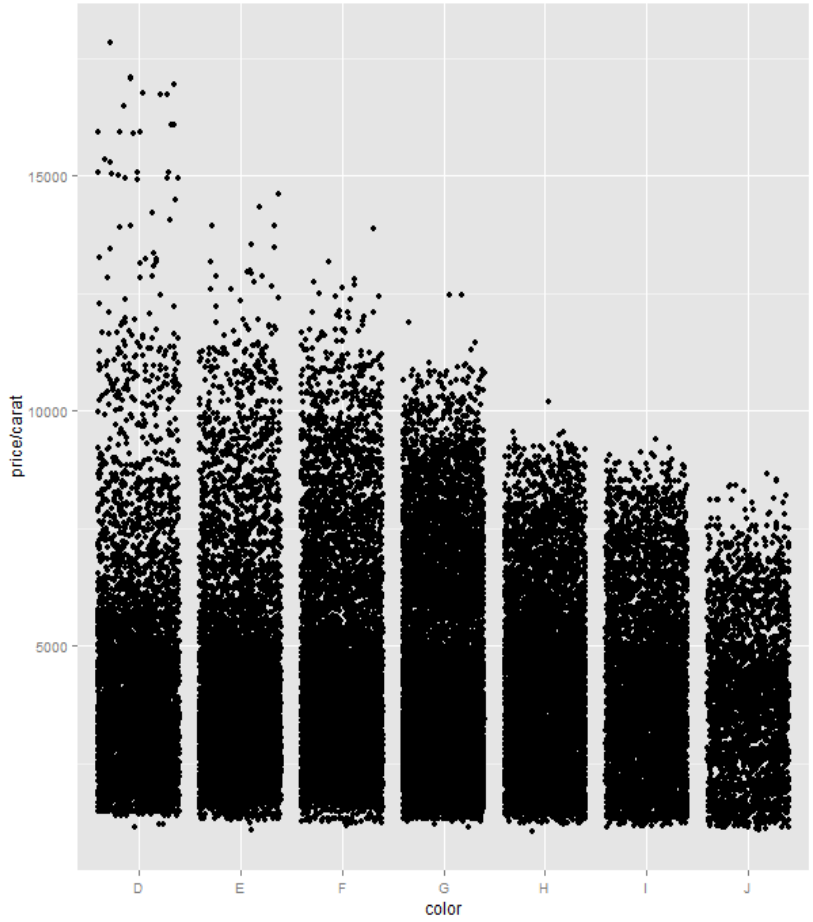
```
qplot(color, price / carat, data = diamonds,
      geom = c("jitter", "boxplot"))
```



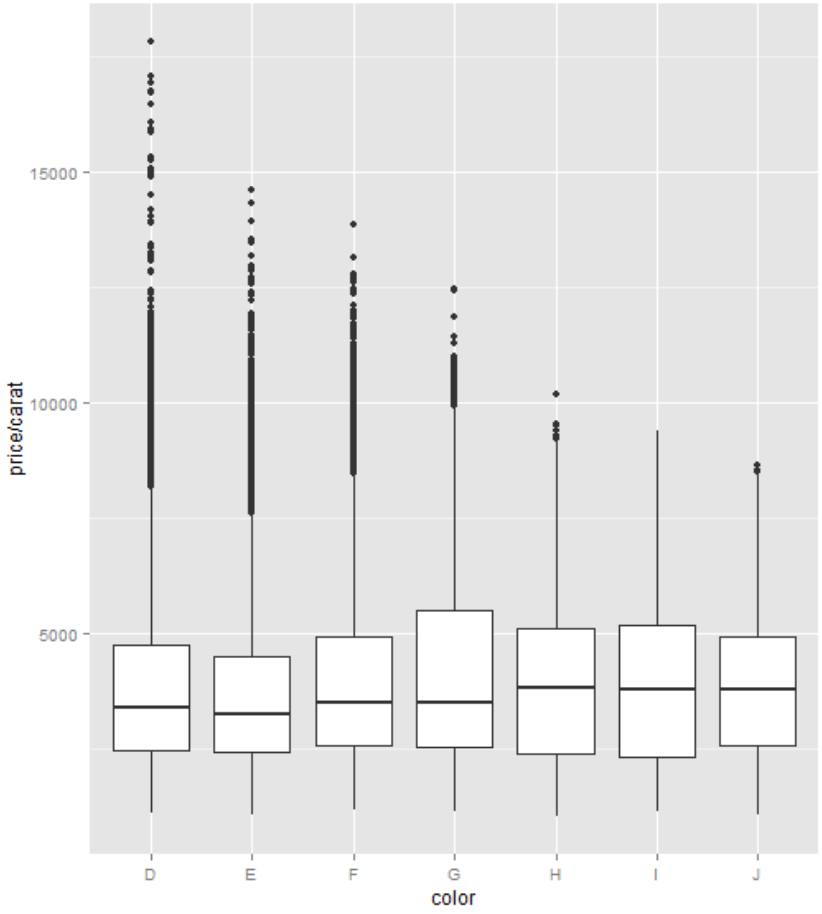
```
qplot(color, price / carat, data = diamonds,
      geom = "jitter", color = clarity)
```



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



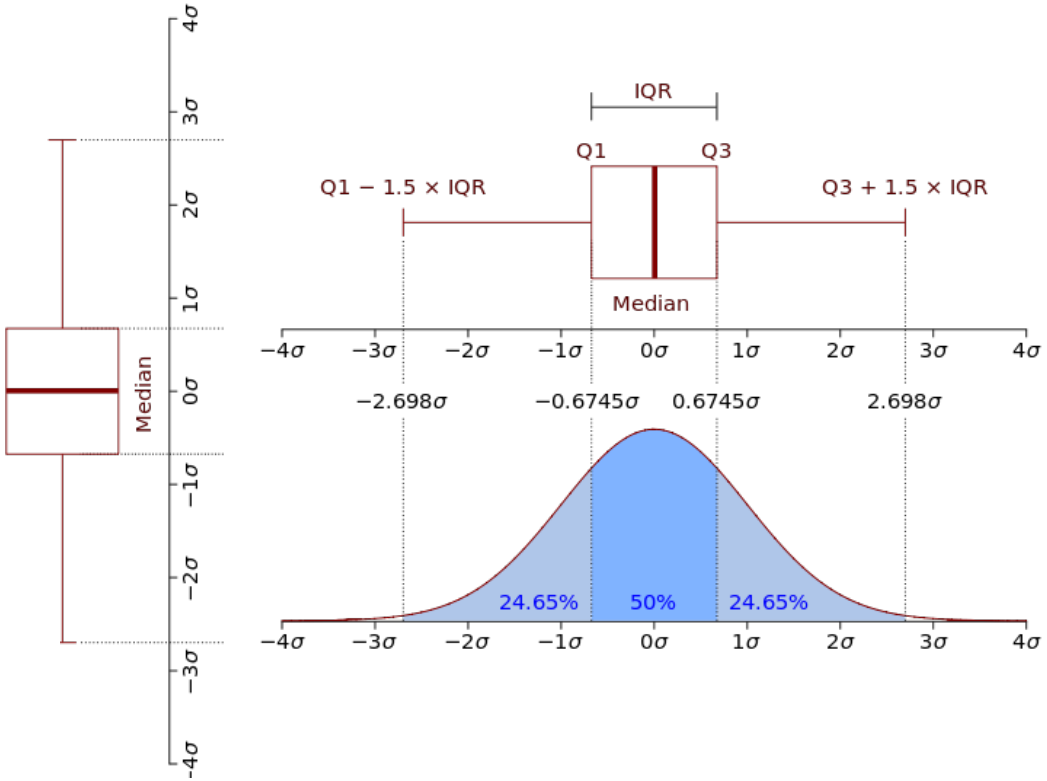

```
qplot(color, price / carat,
      data = diamonds,
      geom = "boxplot")
```



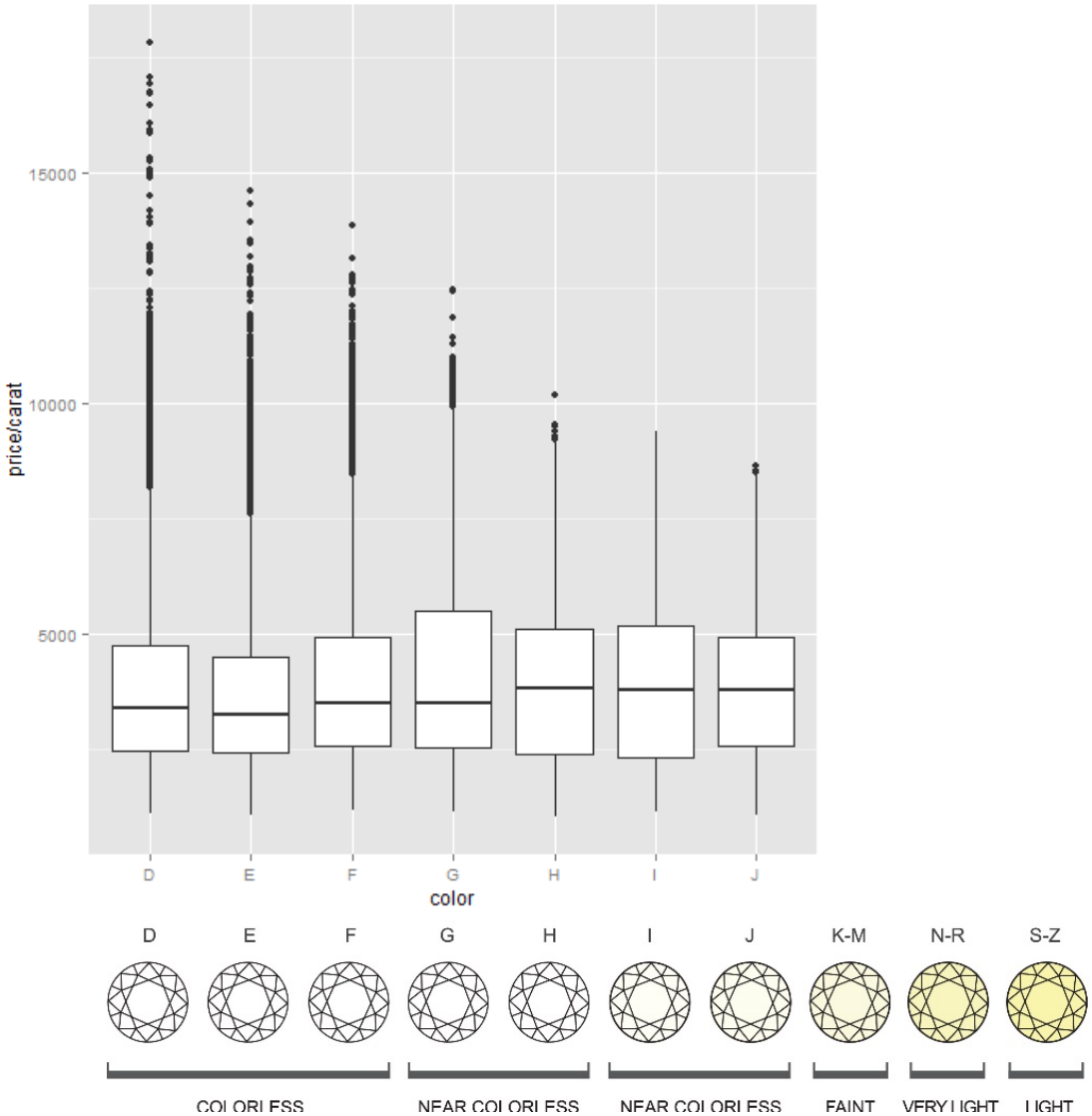
Boxplots summarize distributions and help find outliers

Box plots present summary statistics in one place

- They summarize 5 key values:
  - Bar at median
  - Low – High extreme points (could identify multiple extreme individuals).
  - Hinge: halfway from extreme to the median

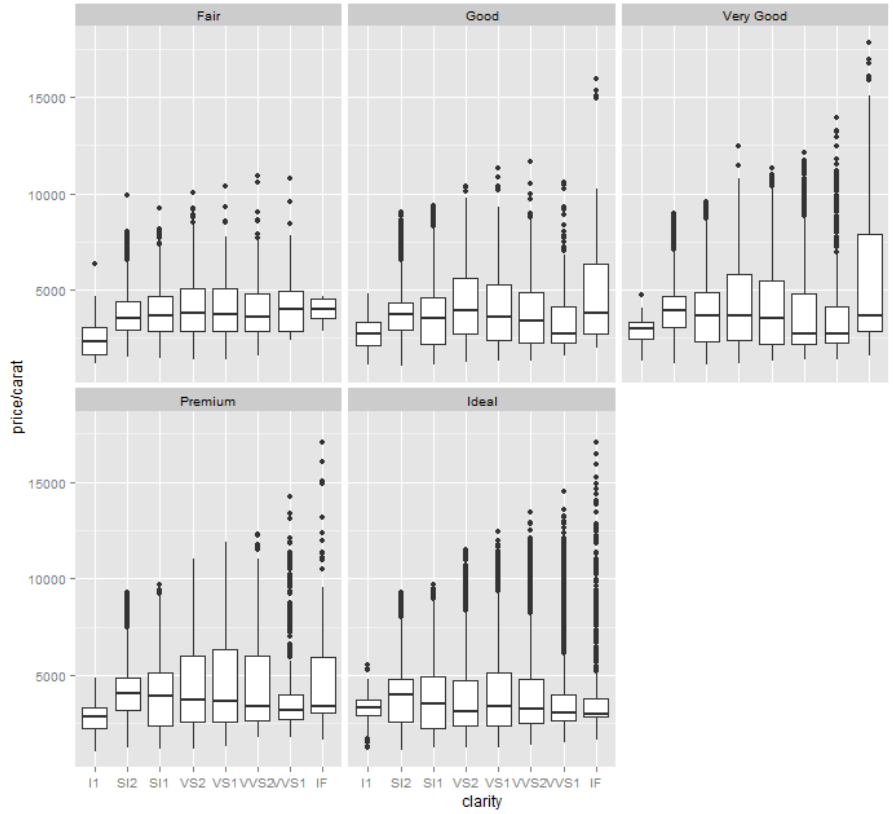


```
qplot(color, price / carat, data = diamonds,
      geom = "boxplot")
```



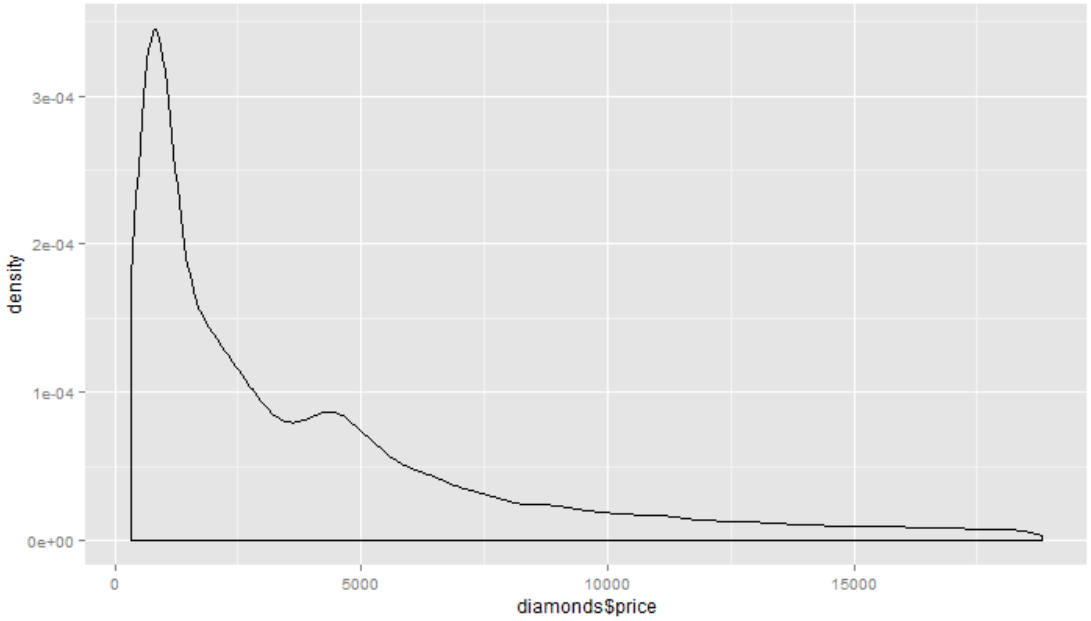
How is price/carat affected by clarity, for each cut type?

```
> qplot(clarity, price/carat, data = diamonds, geom =
"boxplot", facets=.~cut)
```



Basic stats with summary()

```
> summary(diamonds)
     carat               cut        color        clarity          depth           table
 Min.   :0.2000   Fair      : 1610   D: 6775   SI1     :13065   Min.   :43.00   Min.   :43.00
 1st Qu.:0.4000   Good      : 4906   E: 9797   VS2     :12258   1st Qu.:61.00   1st Qu.:56.00
 Median :0.7000   Very Good:12082   F: 9542   SI2     : 9194   Median :61.80   Median :57.00
 Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171   Mean   :61.75   Mean   :57.46
 3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2    : 5066   3rd Qu.:62.50   3rd Qu.:59.00
 Max.   :5.0100                     I: 5422   VVS1    : 3655   Max.   :79.00   Max.   :95.00
                                    J: 2808   (Other): 2531

     price
 Min.   :  326
 1st Qu.:  950
 Median : 2401
 Mean   : 3933
 3rd Qu.: 5324
 Max.   :18823
```

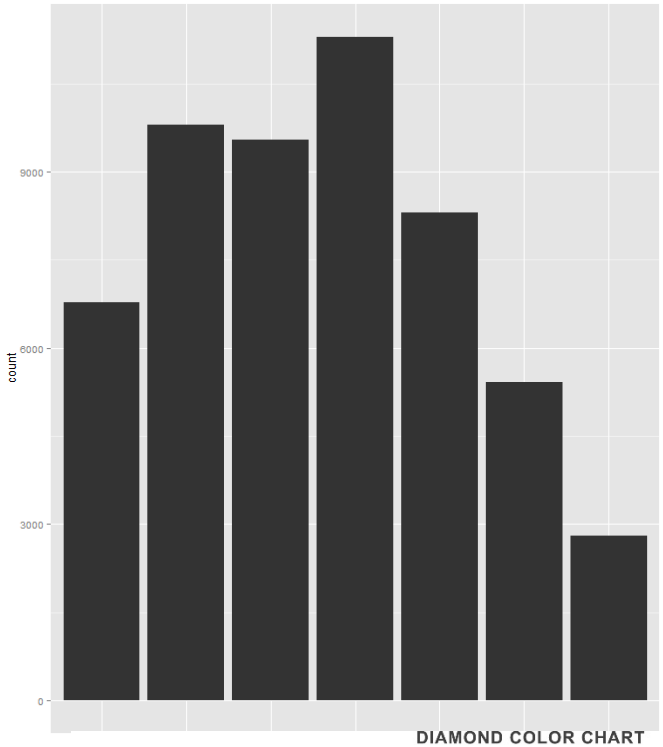


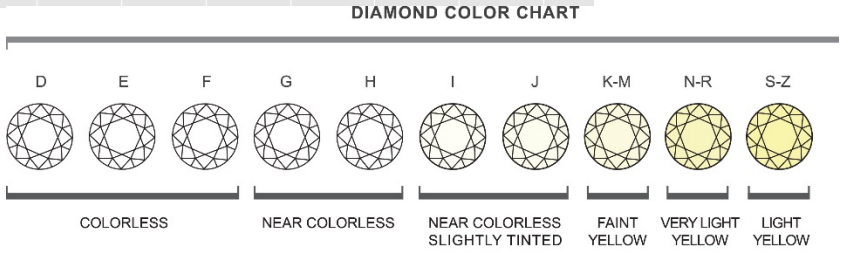
Density
geom= "density"

The area under a probability density curve = 1

Are some diamond colors more rare than others?

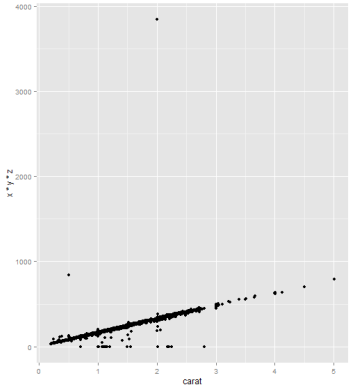
```
qplot(color, data = diamonds, geom = "histogram")
```





Data Manipulation

- Subsets: To explore the relationship between two variables we may want to
  - remove obviously 'bad' values
  - 'zoom in' on a region of interest
- Transforming
  - Adding new columns of features
- Summarizing
  - Statistics over *groups* in the data



Selecting subsets using indices

```
# logical comparisons:  < > <=  >=  !=  == %in%
# multiple conditional statements allowed

#subsetting rows using indices
> head(diamonds[diamonds$color %in% c("E", "F"),])
   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
9   0.22    Fair     E     VS2  65.1    61   337 3.87 3.78 2.49
13  0.22 Premium     F     SI1  60.4    61   342 3.88 3.84 2.33
15  0.20 Premium     E     SI2  60.2    62   345 3.79 3.75 2.27

# subsetting columns using indices
> head(diamonds[, 2:4])
        cut color clarity
1     Ideal     E     SI2
2   Premium     E     SI1
3      Good     E     VS1
4   Premium     I     VS2
5      Good     J     SI2
6 Very Good     J    VVS2

#subsetting both rows and columns using indices
> head(diamonds[diamonds$color %in% c("E", "F"), 2:4])
       cut color clarity
1    Ideal     E     SI2
2  Premium     E     SI1
3     Good     E     VS1
9     Fair     E     VS2
13 Premium     F     SI1
15 Premium     E     SI2
```

Using **subset** to select rows and columns of data frames

[Like SQL SELECT]

```
> head(subset(diamonds, price > 400 & clarity == "IF"))
    carat       cut color clarity depth table price    x    y    z
230  0.52     Ideal     F      IF  62.2    55  2783 5.14 5.18 3.21
251  0.55     Ideal     G      IF  60.9    57  2789 5.28 5.30 3.22
257  0.64     Ideal     G      IF  61.3    56  2790 5.54 5.58 3.41
282  0.72   Premium     I      IF  63.0    57  2795 5.72 5.70 3.60
305  0.60 Very Good     G      IF  61.6    56  2800 5.43 5.46 3.35
314  0.61     Ideal     G      IF  62.3    56  2800 5.43 5.45 3.39

#subsetting both rows and columns
> head(subset(diamonds, price > 400 & clarity == "IF",
          select=c(carat, cut)))
    carat       cut
230  0.52     Ideal
251  0.55     Ideal
257  0.64     Ideal
282  0.72   Premium
305  0.60 Very Good
314  0.61     Ideal
```

Selecting subsets using regular expressions (very handy!)

```
> head(subset(diamonds, grepl('VV', clarity)))
   carat       cut color clarity depth table price    x    y    z
6   0.24 Very Good     J    VVS2  62.8  57.0   336 3.94 3.96 2.48
7   0.24 Very Good     I    VVS1  62.3  57.0   336 3.95 3.98 2.47
26  0.23 Very Good     G    VVS2  60.4  58.0   354 3.97 4.01 2.41
66  0.28     Ideal     G    VVS2  61.4  56.0   553 4.19 4.22 2.58
67  0.32     Ideal     I    VVS1  62.0  55.3   553 4.39 4.42 2.73
70  0.24   Premium     E    VVS1  60.7  58.0   553 4.01 4.03 2.44
```

Slicing and dicing data frames with plyr ('plier')

- The plyr package allows you to easily perform
  - Transformations
  - Summarizations

  Scope:
  - Whole-dataset
  - Group-wise
- Example: bnames-explore.R
  - See zip file in Files/Lectures/Week 2

Baby name dataset: 258,000 records
Top 1000 male & female U.S. baby names from 1880-2008

Variables: year, name, sex, percent

```
> head(bnames,15)
   year    name  percent sex
1  1880    John 0.081541 boy
2  1880 William 0.080511 boy
3  1880   James 0.050057 boy
4  1880 Charles 0.045167 boy
5  1880  George 0.043292 boy
6  1880   Frank 0.027380 boy
7  1880  Joseph 0.022229 boy
8  1880  Thomas 0.021401 boy
9  1880   Henry 0.020641 boy
10 1880  Robert 0.020404 boy
11 1880  Edward 0.019965 boy
12 1880   Harry 0.018175 boy
13 1880  Walter 0.014822 boy
14 1880  Arthur 0.013504 boy
15 1880    Fred 0.013251 boy
```

```
> tail(bnames,15)
       year     name  percent  sex
257986 2008   Neveah 0.000130 girl
257987 2008   Amaris 0.000129 girl
257988 2008 Hadassah 0.000129 girl
257989 2008    Dania 0.000129 girl
257990 2008   Hailie 0.000129 girl
257991 2008   Jamiya 0.000129 girl
257992 2008    Kathy 0.000129 girl
257993 2008   Laylah 0.000129 girl
257994 2008     Riya 0.000129 girl
257995 2008     Diya 0.000128 girl
257996 2008 Carleigh 0.000128 girl
257997 2008    Iyana 0.000128 girl
257998 2008   Kenley 0.000127 girl
257999 2008   Sloane 0.000127 girl
258000 2008  Elianna 0.000127 girl
```

Source: http://plyr.had.co.nz/09-user/

Brainstorming

- What variables and summaries might you want to generate from this data?
- What questions would you like to be able to answer about this data?

Source: http://plyr.had.co.nz/09-user/

Some exploration ideas

- Break down by first/last letter
- Length
- Proportion of vowels
- Tail of "unusual" names: babies with top-2, top-3, top-5, top-100 names
- Celebrity names

Source: http://plyr.had.co.nz/09-user/

plyr implements functions that allow
split-apply-combine similar to MapReduce, SQL Group By

- **Split-apply-combine** analysis strategy
  - take a data frame
  - split it up
  - do something to the parts and/or whole
  - put the pieces back together
  - return a data frame
- **Transform** modifies an existing data frame

```
transform(df, var1 = expr1, …)
```

- **Summarise** creates a new data frame

```
summarise(df, var1 = expr1, …)
```

Source: http://plyr.had.co.nz/09-user/

Group-wise transforms and summaries

- Like SQL GROUP BY statement
- Like map-reduce
- Example:
  - Rank of a baby name within given year and sex

```
one <- subset(bnames, sex == "boy" & year == 2008)
one$rank <- rank(-one$percent, ties.method = "first")

one <- transform(one, rank = rank(-percent, ties.method = "first"))
head(one)
```

- But how to do this for every combination of sex, year values?

Source: http://plyr.had.co.nz/09-user/

Enter ddply

```
# Conceptually if we want to perform this same task
for every sex in every
# year, we need to split up the data, apply
transformation to every piece
# and then join the pieces back together

# This is what ddply does
bnames <- ddply(bnames, c("sex", "year"), transform,
  rank = rank(-percent, ties.method = "first"))
```

Input data

How to split input

Function to apply to each piece

2nd arg to transform

Source: http://plyr.had.co.nz/09-user/

Example: interesting whole-dataset transforms and summaries

```
vowels <- function(x) {
  nchar(gsub("[^AEIOUaeiou]", "", x))     # nchar counts # of each letter
}                                         # gsub does global string replace
                                            using regular expression
letter <- function(x, n=1) {
  if (n < 0) {                            # gets the n-th letter in string x
    nc <- nchar(x)                        # (negative index counts from end)
    n <- nc + n + 1
  }
  tolower(substr(x, n, n))
}
bnames <- transform(bnames,
         first = letter(name,1),
         last = letter(name, -1),
         length = nchar(name),
         vowels = vowels(name))

summarise(bnames,
         max_perc = max(percent),
         min_perc = min(percent))
```

```
> head(bnames)
  year    name  percent sex first last length vowels
1 1880    John 0.081541 boy     j    n      4      1
2 1880 William 0.080511 boy     w    m      7      3
3 1880   James 0.050057 boy     j    s      5      2
4 1880 Charles 0.045167 boy     c    s      7      2
5 1880  George 0.043292 boy     g    e      6      3
6 1880   Frank 0.027380 boy     f    k      5      1
```

```
  max_perc min_perc
1 0.081541  2.6e-05
```

Source: http://plyr.had.co.nz/09-user/

Top baby boy names of the 1880s…

```
> bnames.ddply <- ddply(bnames, c("sex", "year"), transform,
                rank = rank(-percent, ties.method = "first"))
> head(subset(bnames.ddply, rank <= 5), 20)
 year    name  percent sex rank
1    1880    John 0.081541 boy    1
2    1880 William 0.080511 boy    2
3    1880   James 0.050057 boy    3
4    1880 Charles 0.045167 boy    4
5    1880  George 0.043292 boy    5
1001 1881    John 0.080975 boy    1
1002 1881 William 0.078712 boy    2
1003 1881   James 0.050253 boy    3
1004 1881  George 0.043068 boy    4
1005 1881 Charles 0.042828 boy    5
2001 1882    John 0.078314 boy    1
2002 1882 William 0.076191 boy    2
2003 1882   James 0.048281 boy    3
2004 1882  George 0.042553 boy    4
2005 1882 Charles 0.041726 boy    5
3001 1883    John 0.079066 boy    1
3002 1883 William 0.074558 boy    2
3003 1883   James 0.046449 boy    3
3004 1883 Charles 0.042911 boy    4
3005 1883  George 0.042102 boy    5
```

… and 2000s

```
year    name  percent sex rank
126001 2006       Jacob 0.011331 boy    1
126002 2006     Michael 0.010317 boy    2
126003 2006      Joshua 0.010172 boy    3
126004 2006       Ethan 0.009370 boy    4
126005 2006     Matthew 0.009274 boy    5
127001 2007       Jacob 0.010948 boy    1
127002 2007     Michael 0.009911 boy    2
127003 2007       Ethan 0.009511 boy    3
127004 2007      Joshua 0.009309 boy    4
127005 2007      Daniel 0.009135 boy    5
128001 2008       Jacob 0.010355 boy    1
128002 2008     Michael 0.009437 boy    2
128003 2008       Ethan 0.009301 boy    3
128004 2008      Joshua 0.008799 boy    4
128005 2008      Daniel 0.008702 boy    5
```

ddply summaries

```
> ddply(diamonds, c("color"), summarise,
      min.price  = min(price),
      mean.price = mean(price),
      max.price  = max(price))
  color min.price mean.price max.price
1     D       357   3169.954     18693
2     E       326   3076.752     18731
3     F       342   3724.886     18791
4     G       354   3999.136     18818
5     H       337   4486.669     18803
6     I       334   5091.875     18823
7     J       335   5323.818     18710
```

ddply is part of a family of plyr functions

Input \ *Output*	Array	Data frame	List
Array	`aaply`	`adply`	`alply`
Data frame	`daply`	`ddply`	`dlply`
List	`laply`	`ldply`	`llply`

- ddply: take a data frame, split it up, do something to it, return a data frame.
- ldply: take a list, split it up, do something to it, return a data frame.

Sources:
http://www.jstatsoft.org/v40/i01/paper
http://seananderson.ca/2013/12/01/plyr.html

Dealing with missing data

Strategy: List-wise deletion

Recall that missing values in R are represented by the symbol **NA**

```
is.na(x) # returns TRUE of x is missing
y <- c(1,2,3,NA)
is.na(y) # returns a vector (F F F T)
```

To check for rows with missing values use the complete() function

```
# list rows of data that have missing values
mydata[!complete.cases(mydata),]
```

If the data use a special value for missing data, like: 999
Re-code 999 to NA:

```
data <- ifelse(orig.data == 999, NA, orig.data)
```

Once the data use NA for missing data...

Delete any records that contain NA field(s):

```
data <- na.omit(data)
```

Source: http://en.wikipedia.org/wiki/Listwise_deletion

What you should know

- Basic use of qplot
- What facets are and how to plot them
- Basics of summary statistics, scatterplots and boxplots
- Subsetting data
- How to transform and summarize with plyr
- Listwise deletion to deal with missing values
- Exploratory tools:
  - Finding outliers w.r.t expected fixed relationship
  - Finding outliers using boxplots
  - Zooming in/out by changing axis scale
  - Finding differences across facets

Lab time and review materials

- Review of R and ggplot functions
- Starting on homework 2
- Review materials:
  - Wickham, ggplot2, Chapter 2.
  - examples.zip in Resources/Week 2
    - Data frame manipulation
    - qplot examples
    - Baby names data and R code, with plyr

Extra reading:

The hidden biases in big data:

http://blogs.hbr.org/2013/04/the-hidden-biases-in-big-data/

By Kate Crawford, Microsoft Research

Advanced methods for dealing with missing data

http://www.stat.columbia.edu/~gelman/arm/missing.pdf

Supplemental slides

What built-in data frames are available?

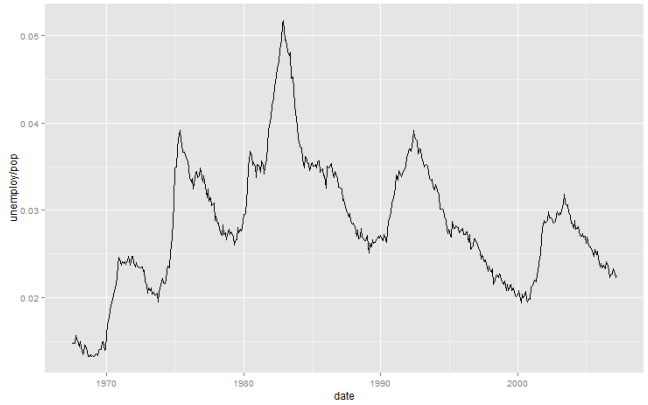
```
library(ggplot2)
data()
[...main dataset list omitted...]
Data sets in package 'ggplot2':

diamonds                Prices of 50,000 round cut diamonds
economics               US economic time series.
midwest                 Midwest demographics.
movies                  Movie information and user ratings from
                        IMDB.com.
mpg                     Fuel economy data from 1999 and 2008 for 38
                        popular models of car
msleep                  An updated and expanded version of the mammals
                        sleep dataset.
presidential            Terms of 10 presidents from Eisenhower to Bush
                        W.
seals                   Vector field of seal movements.
```

More "geom" types

- line: line graph
- path: (*x*, *y*) time series, adjacent points joined
- smooth: smoothed curve-fit
- text: text labels

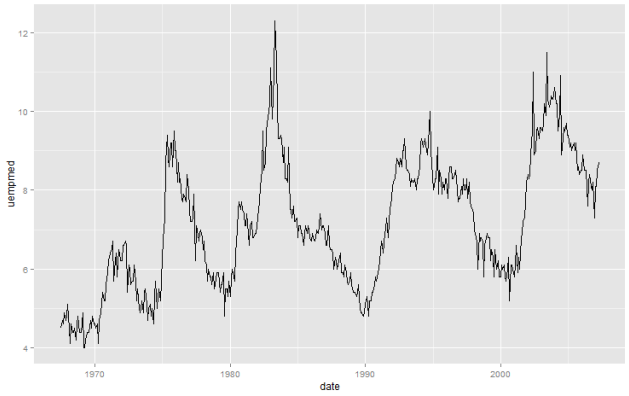
Examples of `geom="line"` and `geom="path"` with "economics" dataset

Unemployment rate



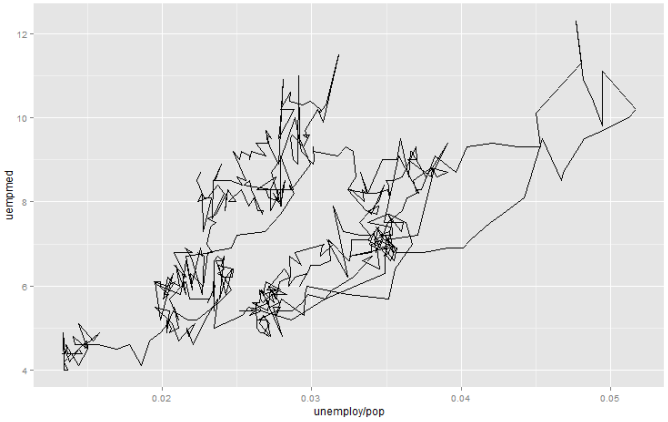

```
qplot(date, unemploy/pop,
data=economics, geom="line")
```

Median weeks unemployed




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

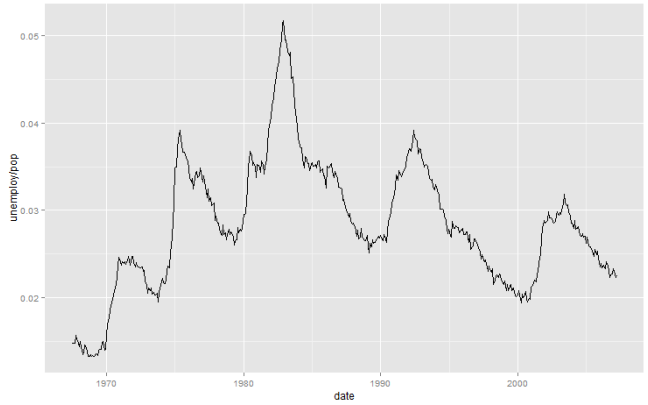
```
qplot(unemploy/pop, uempmed,
data=economics, geom = "path")
```



Two-variable path plot

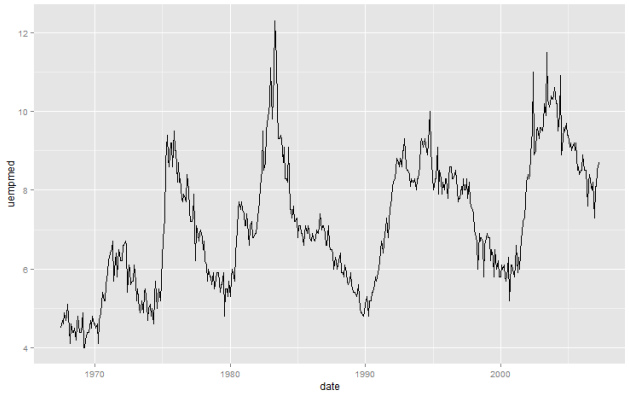
Examples of `geom="line"` and `geom="path"` with "economics" dataset

Unemployment rate



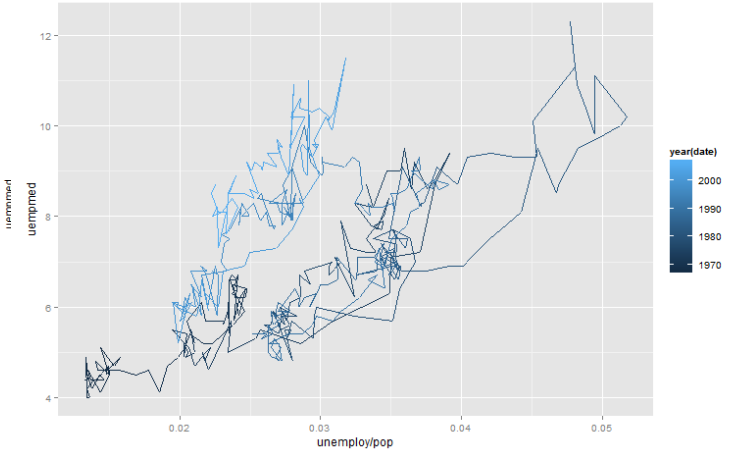

```
qplot(date, unemploy/pop,
data=economics, geom="line")
```

Median weeks unemployed




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

```
qplot(unemploy/pop, uempmed,
data=economics, geom = "path",
colour = year(date))
```



Two-variable path plot with color

(Wickham p. 22)

Selecting a random subset of rows with **sample**

```
> dsmall <- diamonds[sample(1:nrow(diamonds), 100, replace=FALSE),]
> head(dsmall)
      carat       cut color clarity depth table price    x    y    z
48154  0.59     Ideal     E     VS2  60.2    59  1941 5.47 5.49 3.30
38296  0.30     Ideal     G     SI1  60.6    57   487 4.34 4.37 2.63
24145  1.51 Very Good     D     VS2  63.2    57 12311 7.27 7.25 4.59
30315  0.43   Premium     I     SI2  59.8    59   726 4.94 4.90 2.94
22425  1.19     Ideal     F     VS1  60.5    57 10449 6.82 6.88 4.15
10989  1.02   Premium     D     SI2  59.5    62  4912 6.56 6.52 3.89
```

Studying a random sample is often a good way to start exploring a very large dataset.

Sorting rows in a data frame

[Like SQL ORDER BY]

```
# sort by descending price
> head(diamonds[order(-diamonds$price),])
      carat       cut color clarity depth table price    x    y    z
27750  2.29   Premium     I     VS2  60.8    60 18823 8.50 8.47 5.16
27749  2.00 Very Good     G     SI1  63.5    56 18818 7.90 7.97 5.04
27748  1.51     Ideal     G      IF  61.7    55 18806 7.37 7.41 4.56
27747  2.07     Ideal     G     SI2  62.5    55 18804 8.20 8.13 5.11
27746  2.00 Very Good     H     SI1  62.8    57 18803 7.95 8.00 5.01
27745  2.29   Premium     I     SI1  61.8    59 18797 8.52 8.45 5.24

# sort with two variables, one ascending, one descending
> head(diamonds[order(diamonds$color, -diamonds$depth),])
      carat  cut color clarity depth table price    x    y    z
45689  0.70 Fair     D     SI2  71.6    55  1696 5.47 5.28 3.85
8357   1.02 Fair     D     SI1  70.6    57  4398 6.08 6.01 4.27
1439   1.00 Fair     D     SI2  69.3    58  2974 5.96 5.87 4.10
18359  1.50 Fair     D     SI2  68.8    57  7469 6.90 6.86 4.73
46431  0.70 Fair     D     SI2  67.8    58  1770 5.51 5.44 3.71
48489  0.50 Fair     D    VVS2  67.6    57  1980 4.95 4.84 3.31
```

Aggregating rows in data frames

[Like SQL GROUP BY]

```
> aggregate(diamonds$price, by=list(diamonds$color, diamonds$clarity), FUN=mean, na.rm=TRUE)
   Group.1 Group.2        x
1        D      I1 3863.024
2        E      I1 3488.422
3        F      I1 3342.182
4        G      I1 3545.693
5        H      I1 4453.414
6        I      I1 4302.185
7        J      I1 5254.060
8        D     SI2 3931.101
9        E     SI2 4173.826
10       F     SI2 4472.625
11       G     SI2 5021.684
12       H     SI2 6099.895
13       I     SI2 7002.649
14       J     SI2 6520.958
15       D     SI1 2976.146
16       E     SI1 3161.838
17       F     SI1 3714.226
18       G     SI1 3774.787
19       H     SI1 5032.415
[and 37 more rows]
```

Aggregating using max output instead of mean

```
> aggregate(diamonds$price, by=list(diamonds$color, diamonds$clarity), FUN=max, na.rm=TRUE)
   Group.1 Group.2     x
1        D      I1 15964
2        E      I1 11548
3        F      I1 10685
4        G      I1 13203
5        H      I1 17329
6        I      I1 16193
7        J      I1 18531
8        D     SI2 18693
9        E     SI2 18477
10       F     SI2 18784
11       G     SI2 18804
12       H     SI2 18745
13       I     SI2 18756
14       J     SI2 18710
15       D     SI1 18468
16       E     SI1 18731
17       F     SI1 18759
18       G     SI1 18818
19       H     SI1 18803
[and 37 more rows]
```

Merging data frames

[Like SQL JOIN]

```
> books
      name                              title     other.author
1    Tukey     Exploratory Data Analysis                 <NA>
2 Venables Modern Applied Statistics ...              Ripley
3  Tierney                      LISP-STAT                 <NA>
4   Ripley             Spatial Statistics                 <NA>
5   Ripley          Stochastic Simulation                 <NA>
6   McNeil     Interactive Data Analysis                 <NA>
7   R Core           An Introduction to R Venables & Smith
> authors
   surname nationality deceased
1    Tukey          US      yes
2 Venables   Australia       no
3  Tierney          US       no
4   Ripley          UK       no
5   McNeil   Australia       no
```

Source: http://stat.ethz.ch/R-manual/R-patched/library/base/html/merge.html

Merging data frames

```
> m1 <- merge(authors, books, by.x = "surname", by.y = "name")
> head(m1)
   surname nationality deceased                          title other.author
1   McNeil   Australia       no      Interactive Data Analysis         <NA>
2   Ripley          UK       no             Spatial Statistics         <NA>
3   Ripley          UK       no          Stochastic Simulation         <NA>
4  Tierney          US       no                     LISP-STAT         <NA>
5    Tukey          US      yes      Exploratory Data Analysis         <NA>
6 Venables   Australia       no Modern Applied Statistics ...       Ripley

## "R core" is missing from authors and appears only with all = TRUE:
> merge(authors, books, by.x = "surname", by.y = "name", all = TRUE)
 surname nationality deceased                          title     other.author
1   McNeil   Australia       no      Interactive Data Analysis             <NA>
2   R Core        <NA>     <NA>           An Introduction to R Venables & Smith
```

Source: http://stat.ethz.ch/R-manual/R-patched/library/base/html/merge.html

Merging with NA values present

```
x <- data.frame(k1 = c(NA,NA,3,4,5),
                      k2 = c(1,NA,NA,4,5), data = 1:5)
y <- data.frame(k1 = c(NA,2,NA,4,5),
                      k2 = c(NA,NA,3,4,5), data = 1:5)
> merge(x,y, by = c("k1","k2")) # NA's match
  k1 k2 data.x data.y
1  4  4      4      4
2  5  5      5      5
3 NA NA      2      1
> merge(x, y, by = "k1") # NA's match, so 6 rows
  k1 k2.x data.x k2.y data.y
1  4    4      4    4      4
2  5    5      5    5      5
3 NA    1      1   NA      1
4 NA    1      1    3      3
5 NA   NA      2   NA      1
6 NA   NA      2    3      3
> merge(x, y, by = "k2", incomparables = NA) # 2 rows
  k2 k1.x data.x k1.y data.y
1  4    4      4    4      4
2  5    5      5    5      5
```

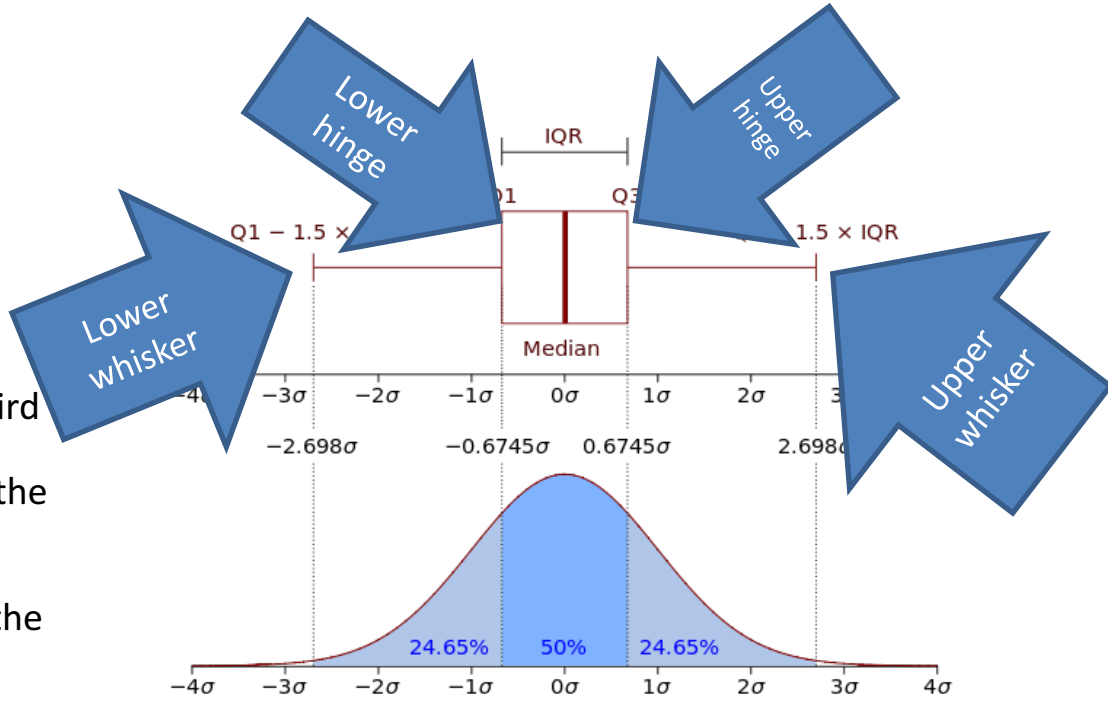
Details on what ddply does

```
# ddply basically works as follows:

pieces <- split(bnames, list(bnames$sex, bnames$year))
results <- vector("list", length(pieces))
for(i in seq_along(pieces)) {
  piece <- pieces[[i]]
  piece <- transform(piece, rank = rank(-percent, ties.method = "first"))
  results[[i]] <- piece
}
result <- do.call("rbind", results)
```

Source: http://plyr.had.co.nz/09-user/

Boxplots help summarize and compare data distributions, and find outliers

- Some differences between stats packages. In ggplot2:
  - Median corresponds to 50%-ile
  - The upper and lower "hinges" correspond to the first and third quartiles (the 25th and 75th percentiles).
  - IQR is the inter-quartile range, or distance between the first and third quartiles.
  - The upper whisker extends from the hinge to the highest value that is within 1.5 * IQR of the hinge.
  - The lower whisker extends from the hinge to the lowest value within 1.5 * IQR of the hinge.
  - Data beyond the end of the whiskers are outliers and plotted as points (as specified by Tukey).



Source: http://docs.ggplot2.org/0.9.3.1/geom_boxplot.html

The reshape2 package

http://seananderson.ca/2013/10/19/reshape.html

A way to easily "mold" dataframes into a desired configuration.