Quiz Review

Peter Kabai

Things We Will Review

- Basic data science
- Classification, regression and clustering
- KNN
- R coding

The Basics

- What do we want to make good predictions on?
- Training or test data? Why?
- What happens if our model is too specific to the training data?
- What about if it doesn't fit the training data enough?
- What is a feature? What about a feature space?

Classification

- We're trying to predict a label, or category.
- This is supervised learning, we know the label of each training example, that is, what category it belongs to.
- Example: given information about credit card debt, we try to predict whether or not the person defaulted on their debt.
- What could some possible target features be categories be?
- If the data shows the output to be 1 or 0, is that correct? Aren't numerical features for regression?
- What is an example of something that cannot be the output of a model that does classification?

Regression

- Here we're trying to predict a numerical output.
- This is supervised learning, we know the value of the target feature for each training example.
- Example: given the engine size, transmission type, and vehicle weight, predict the mpg.
- What are other possible target features of a model that uses car data?
- What are features that would not be target features in a regression problem?

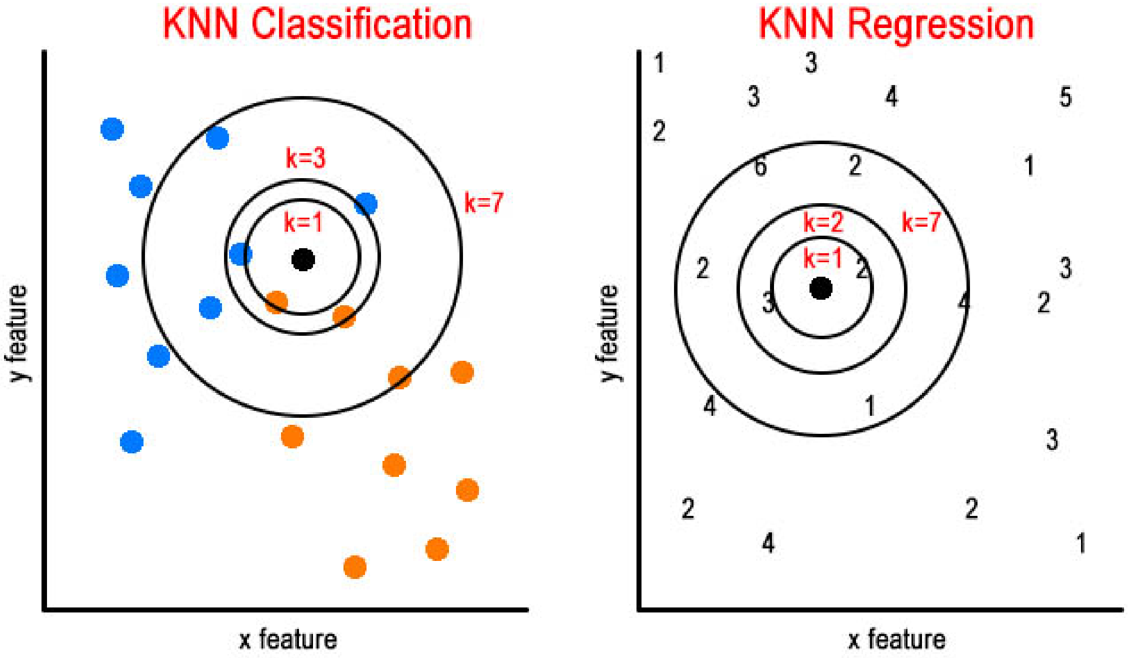
Clustering

- Here we're grouping data together into any number of categories.
- Unsupervised learning, we give the model some data, and it finds similarities on its own.

KNN

- KNN can be used for both regression, and classification.
- In KNN classification the output is the most frequent label of the K closest points.
- In KNN regression the output is the mean of the K closest points.

KNN Visual


KNN Classification
k=3
k=7
k=1
y feature
x feature
KNN Regression
1
3
3
4
5
2
6
2
1
k=2
k=7
k=1
2
2
3
3
4
2
4
1
3
2
2
4
1
y feature
x feature

R Coding!

Vectors

```
# There are no scalars! This expression is equal to 1
length(5)
```

```
## [1] 1
```

```
# The function length() returns the number of items in a vector
# All item in a vector must be the same type

# If they aren't the lower types get converted to the higher type
c(1, 2, "three")
```

```
## [1] "1"     "2"     "three"
```

Sampling

```r
# Sampling is a way to create a vector with random values
sample(1:10, 100)
```

```
## Error in sample.int(length(x), size, replace, prob): cannot take a sample larger than the po
```

```r
# This fails, because there are only 10 options,
# but we're trying to pick 100.
# To fix it, we can set replace to TRUE
```

Sampling

```
sample(1:10, 100, replace=TRUE)
```

```
##   [1]  1  8  9  1  1  4  7  2  2  9  1  4  8  8  5  2  7 10  3  7  2  4  5  2  9  4  2  7  2
##  [32]  3  7  4  5  5  9  7  3  3  5  4  3 10  6  6  1  8  3  9 10  4  6  5  1  9  1  8  1  3
##  [63]  1  3  2  4  1  9  7  8  5  6  8  8  6  2  5  6  4  8  5  4 10  3  9 10 10  6  5  1  2
##  [94]  2  5  3  7  5  6  4
```

```
# We can also simply add the TRUE parameter.
sample(1:10, 100, TRUE)
```

```
##   [1]  9  4  8  7 10  6  4 10  6  5  9  9 10  1  6 10  5  8  4  6 10  8  6  2  8  3  6  7  7
##  [32]  8  2  7  1  1  6  5  1 10  9  6  6  4  9  5  4  3  1  8  7  4  3  1  8  5  4  6  9  3
##  [63]  1  3  8  9  3  2  1  4  8 10  6 10  1  8  5  2  3  1  1  6  9  8  5  1  8  8  9  5  4
##  [94]  9 10  1 10  3  7  3
```

Sequences

```
# There are multiple ways to do write parameters for sequences.
seq(from=1, to=10, by=2)
```

```
## [1] 1 3 5 7 9
```

```
seq(1, 10, 2)
```

```
## [1] 1 3 5 7 9
```

Boolean Vectors

```
# Just a vector of true or false values
# These can be used to index into vectors
n = c(TRUE, FALSE, TRUE, FALSE)
m = c(TRUE, FALSE)
v = c(1:4)
```

Boolean Vectors

```
# What is the output of both?
# Which of these uses recycling?
v[m]
```

```
## [1] 1 3
```

```
v[n]
```

```
## [1] 1 3
```

```
# The first one uses recycling, because length(v) is 4
# but length(m) is 2
```

Boolean Vectors

```
# Use sequence to create an identical variable
c(1:100)[c(c(TRUE, FALSE, FALSE), c(FALSE, FALSE))]
```

```
##  [1]  1  6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96
```

```
# Answer:
seq(1,100,5)
```

```
##  [1]  1  6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96
```

```
# A boolean vector is created that then is used to index
# the vector c(1:100)
```

Recycling

```r
# What is the result of this expression?
c(1:2) * c(1:4)
```

```
## [1] 1 4 3 8
```

```r
# Answer:
# The first vector is recycled, so the result is c(1,4,3,8)

# What is the result of this expression?
c(1:2) * c(1:5)
```

```
## Warning in c(1:2) * c(1:5): longer object length is not a multiple of shorter object length
```

```
## [1] 1 4 3 8 5
```

```r
# This runs, but also creates a warning
```

Vector Operations

```
c(1,2,3,4,5,6,7,8,9,10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
# Re-write the above vector in a cleaner way
c(1:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
# What will be the content of y?
x = c(1:10)
y = x + 5

# This is a vectorized operation, so 5 gets added
# to ALL the values in x, and x becomes c(6,7,8,9,10,11,12,13,14,15)
```

Vectorized vs. Aggregate Functions

```
x = c(1:10)
sqrt(x)
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427 3.000000 3.1622
```

```
mean(x)
```

```
## [1] 5.5
```

```
min(x)
```

```
## [1] 1
```

```
max(x)
```

```
## [1] 10
```

Vectorized vs. Aggregate Functions

```
# What will be the result of this expression?
mean(x) < 5
```

```
## [1] FALSE
```

```
# How about this one?
mean(x < 5)
```

```
## [1] 0.4
```

```
# What's the result of the intermediate step above?
x < 5
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

Vectorized vs. Aggregate Functions

```
# Remember, when aggregator functions are called on boolean
# vectors, the values are converted to 1 or 0
v = c(TRUE,TRUE,TRUE,TRUE,FALSE,FALSE,FALSE,FALSE,FALSE,FALSE)
# What is the output of this expression?
sum(v)
```

```
## [1] 4
```

```
# And this one?
sqrt(v)
```

```
##  [1] 1 1 1 1 0 0 0 0 0 0
```

Data Frames (Using Indices)

```
# Using the built in mtcars data frame
# How do you get the all rows but just columns 3 to 5?
dat = mtcars[,3:5]
dat
```

```
##                       disp  hp drat
## Mazda RX4           160.0 110 3.90
## Mazda RX4 Wag       160.0 110 3.90
## Datsun 710          108.0  93 3.85
## Hornet 4 Drive      258.0 110 3.08
## Hornet Sportabout   360.0 175 3.15
## Valiant             225.0 105 2.76
## Duster 360          360.0 245 3.21
## Merc 240D           146.7  62 3.69
## Merc 230            140.8  95 3.92
## Merc 280            167.6 123 3.92
## Merc 280C           167.6 123 3.92
## Merc 450SE          275.8 180 3.07
## Merc 450SL          275.8 180 3.07
## Merc 450SLC         275.8 180 3.07
## Cadillac Fleetwood  472.0 205 2.93
## Lincoln Continental 460.0 215 3.00
## Chrysler Imperial   440.0 230 3.23
```

Data Frames (Using Column Names)

```
# How about if we want column names?
dat = mtcars[,c("cyl","hp")]
dat
```

```
##                     cyl  hp
## Mazda RX4             6 110
## Mazda RX4 Wag         6 110
## Datsun 710            4  93
## Hornet 4 Drive        6 110
## Hornet Sportabout     8 175
## Valiant               6 105
## Duster 360            8 245
## Merc 240D             4  62
## Merc 230              4  95
## Merc 280              6 123
## Merc 280C             6 123
## Merc 450SE            8 180
## Merc 450SL            8 180
## Merc 450SLC           8 180
## Cadillac Fleetwood    8 205
## Lincoln Continental   8 215
## Chrysler Imperial     8 230
## Fiat 128              4  66
```

Data Frames

```
# How about the same as before, but only rows 1 to 5?
dat = mtcars[1:5,c("cyl","hp")]
dat
```

```
##                    cyl  hp
## Mazda RX4            6 110
## Mazda RX4 Wag        6 110
## Datsun 710           4  93
## Hornet 4 Drive       6 110
## Hornet Sportabout    8 175
```

Data Frames

```
# Add the mpg column but only show rows where mpg > 20?
dat = mtcars[mtcars$mpg > 20,c("cyl","hp","mpg")]
dat
```

```
##                cyl  hp  mpg
## Mazda RX4        6 110 21.0
## Mazda RX4 Wag    6 110 21.0
## Datsun 710       4  93 22.8
## Hornet 4 Drive   6 110 21.4
## Merc 240D        4  62 24.4
## Merc 230         4  95 22.8
## Fiat 128         4  66 32.4
## Honda Civic      4  52 30.4
## Toyota Corolla   4  65 33.9
## Toyota Corona    4  97 21.5
## Fiat X1-9        4  66 27.3
## Porsche 914-2    4  91 26.0
## Lotus Europa     4 113 30.4
## Volvo 142E       4 109 21.4
```

Data Frames

```
# We're using a boolean vector to index the rows
mtcars$mpg > 20
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FAI
## [17] FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FAI
```

```
# This shows which rows to include, and which rows to ignore
```

CSV to Plot

```
# Read the CSV
dat_raw = read.csv("data/quizReview.csv")
dat_raw
```

```
##                      Timestamp Wednesday.10.to.11 Wednesday.11.to.12 Wednesday.2.to.3
## 1  2019/02/05 12:08:25 PM PST     I can make it!     I can make it!                 
## 2  2019/02/05 12:08:27 PM PST                        I can make it!   I can make it!
## 3  2019/02/05 12:17:12 PM PST                                         I can make it!
## 4  2019/02/05 12:22:43 PM PST                                                       
## 5   2019/02/05 2:00:43 PM PST                                                       
## 6   2019/02/05 2:01:00 PM PST     I can make it!     I can make it!   I can make it!
## 7   2019/02/05 2:01:10 PM PST                                                       
## 8   2019/02/05 2:01:19 PM PST                                         I can make it!
## 9   2019/02/05 2:01:29 PM PST                        I can make it!                 
## 10  2019/02/05 2:01:32 PM PST                                                       
## 11  2019/02/05 2:01:35 PM PST     I can make it!                                    
## 12  2019/02/05 2:01:47 PM PST     I can make it!     I can make it!                 
## 13  2019/02/05 2:01:57 PM PST                                                       
## 14  2019/02/05 2:01:58 PM PST                        I can make it!   I can make it!
## 15  2019/02/05 2:02:02 PM PST     I can make it!     I can make it!                 
## 16  2019/02/05 2:02:08 PM PST                                                       
## 17  2019/02/05 2:02:24 PM PST                                         I can make it!
## 18  2019/02/05 2:02:46 PM PST
```

CSV to Plot

```
# Remove the first row, which is just a timestamp
dat = dat_raw[,-1]

# Convert all values to either a 1 or a 0
dat[,] = ifelse (is.na(dat) | dat != "I can make it!", 0,  1)

dat
```

```
##    Wednesday.10.to.11 Wednesday.11.to.12 Wednesday.2.to.3 Wednesday.3.to.4 Thursday.10.to.1
## 1                   1                  1                0                0                 (
## 2                   0                  1                1                0                 (
## 3                   0                  0                1                1                 (
## 4                   0                  0                0                0                 1
## 5                   0                  0                0                0                 (
## 6                   1                  1                1                1                 (
## 7                   0                  0                0                0                 (
## 8                   0                  0                1                1                 (
## 9                   0                  1                0                0                 1
## 10                  0                  0                0                0                 (
## 11                  1                  0                0                0                 1
## 12                  1                  1                0                0                 (
## 13                  0                  0                0                0                 (
## 14                  0                  1                1                1                 (
```

CSV to Plot

```
# Check to see if all columns are now numeric
sapply(dat, class)
```

```
## Wednesday.10.to.11 Wednesday.11.to.12   Wednesday.2.to.3   Wednesday.3.to.4  Thursday.10.to.
##          "numeric"          "numeric"          "numeric"          "numeric"          "numeri
##  Thursday.11.to.12   Thursday.12.to.1    Thursday.1.to.2
##          "numeric"          "numeric"          "numeric"
```

```
# Change the names of the columns (gsub wasn't covered in class)
names(dat) = gsub("[.]", " ", names(dat))

# Show the number of "I can make it!" votes for each day
sapply(dat, sum)
```

```
## Wednesday 10 to 11 Wednesday 11 to 12   Wednesday 2 to 3   Wednesday 3 to 4  Thursday 10 to
##                  5                  8                  7                  7
##  Thursday 11 to 12   Thursday 12 to 1    Thursday 1 to 2
##                  6                 10                 13
```

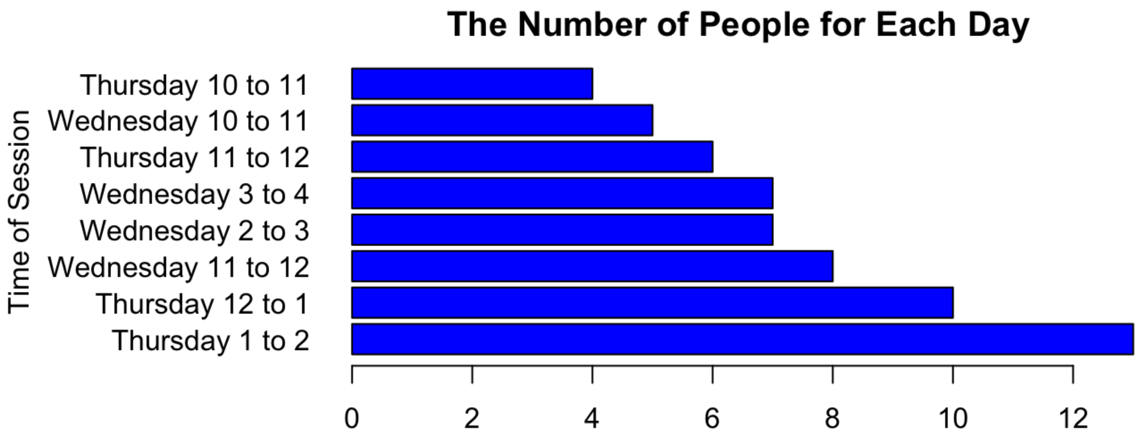
CSV to Plot

```
# Sorting that vector
vect_of_times = sort(sapply(dat, sum), decreasing = TRUE)
vect_of_times
```

```
##    Thursday 1 to 2   Thursday 12 to 1 Wednesday 11 to 12   Wednesday 2 to 3   Wednesday 3 to
##                 13                 10                  8                  7
##  Thursday 11 to 12 Wednesday 10 to 11  Thursday 10 to 11
##                  6                  5                  4
```

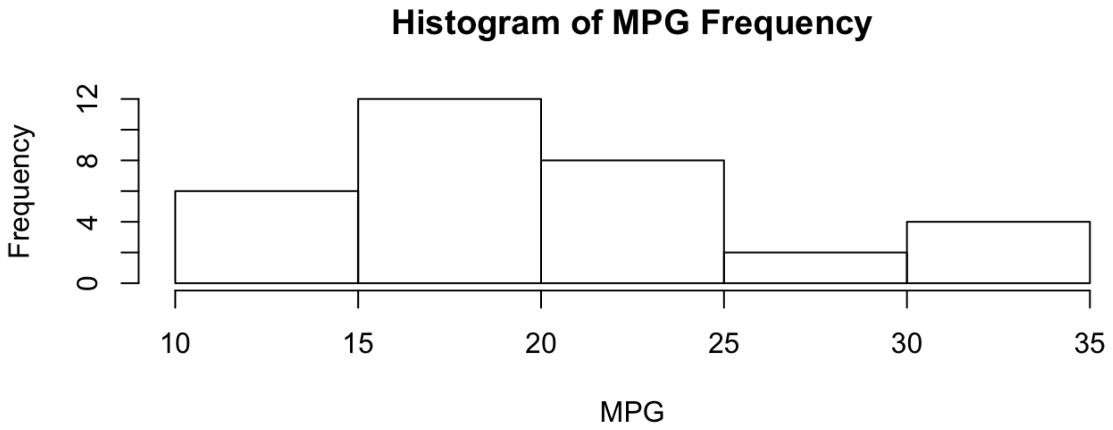
CSV to Plot

```
# plot the named vector of times
par(mar = c(4, 11, 2, 1), mgp = c(9,1,0))
barplot(
  vect_of_times, las = 1,horiz = TRUE, col = "blue",
  ylab = "Time of Session", main = "The Number of People for Each Day"
)
```



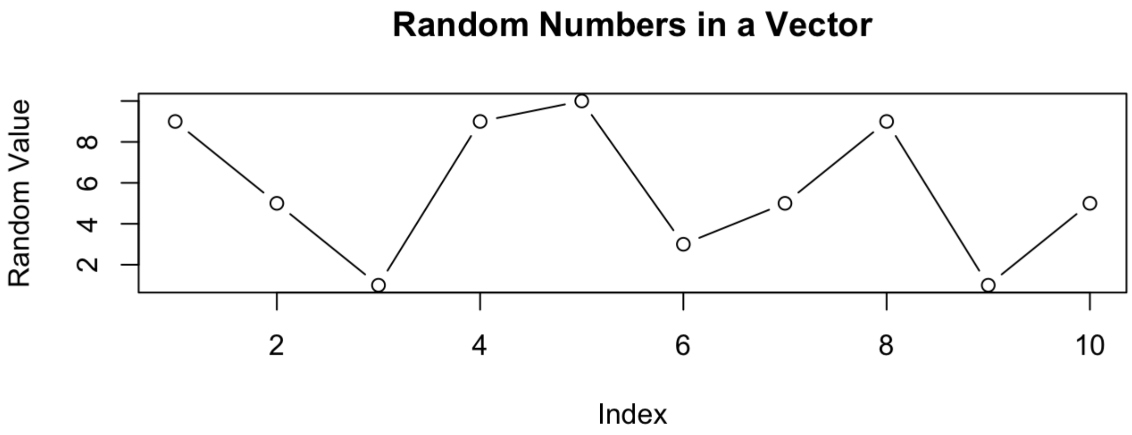
Other Types of Plots (Histogram)

```
hist(
  mtcars$mpg,
  main = "Histogram of MPG Frequency", xlab = "MPG"
)
```



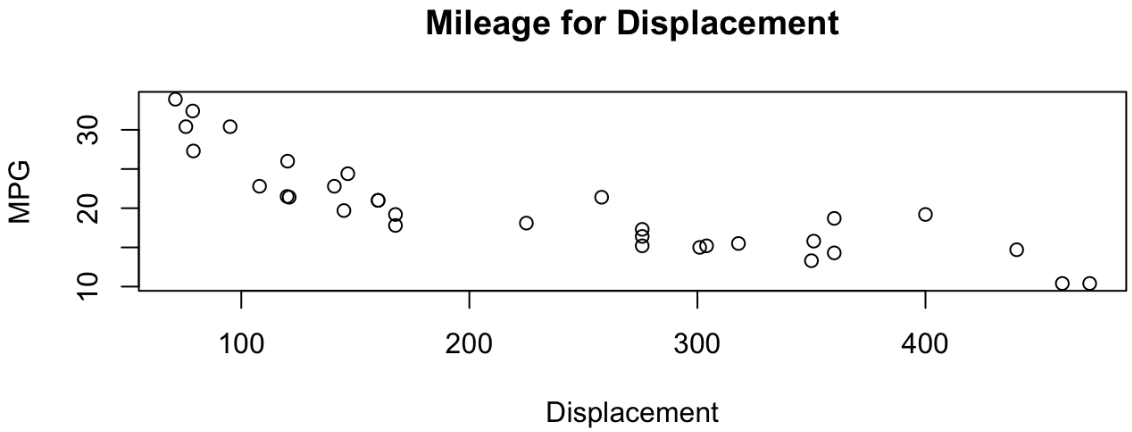
Other Types of Plots (Line)

```
x = sample(1:10, 10, TRUE) # vector of random values
plot(
  x, type="b",
  main = "Random Numbers in a Vector", xlab = "Index", ylab = "Random Value"
)
```



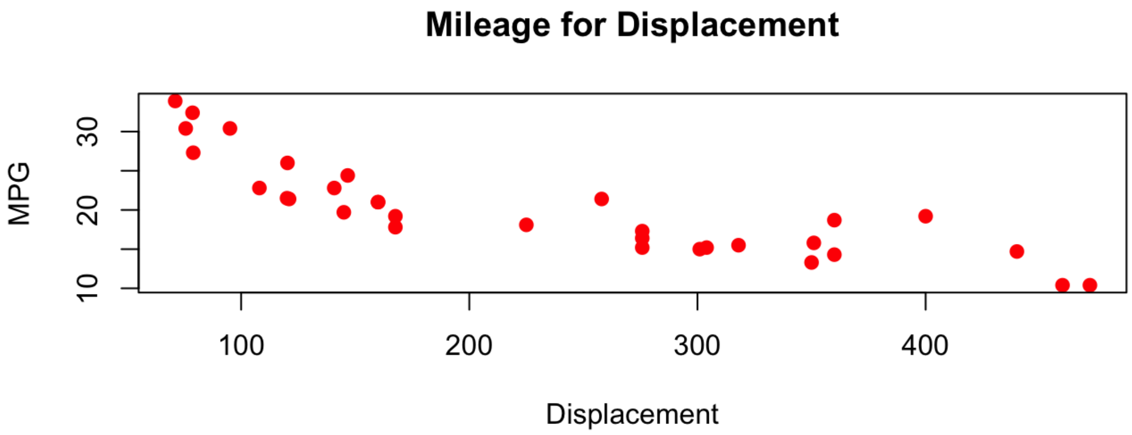
Other Types of Plots (Scatter)

```
plot(
  mtcars$disp, mtcars$mpg,
  main = "Mileage for Displacement", xlab = "Displacement", ylab = "MPG"
)
```



Other Types of Plots (Scatter)

```
# a different way to do the same thing
plot(
  mpg ~ disp, data=mtcars, col="red", pch=19,
  main = "Mileage for Displacement", xlab = "Displacement", ylab = "MPG"
)
```



Anything else?