Katherine Olson

STA 141 Assignment Four

Report

Task 1: Extracting prices

The regular expression I used was "\\$([1-9]{0,1}[ ]{0,1}[0-9,]{1,2}[ ]{0,1}[0-9][ ]{0,1}[0-9][ ]{0,1}[0-9])". I started with a simplified down version and made it more specific until I felt it captured most of the prices. I decided to look in the body and the title of each post and found 13,010 posts with a match in the body and 29,303 posts with a match in the title. For dealing with posts where multiple prices were found, I decided to take the maximum price. I made it so my regular expression would not take prices above $999,999 or below $999 so less huge or small prices would be picked up. I could not think of a way, other than looking at each post individually, which would be too time consuming, to pick a price other than picking the maximum, so I went with that. The percentage of posts with a price in the body and in the price column that had the same price in each was 85.5%. The percentage of posts with a price in the title and in the price column that had the same price in each was 99.3%. I found that the title was much more accurate than the body, probably due to my failure to think of any other way than taking the maximum price when many prices were found. But of course, the price in the price column could be wrong.

Task 2: Extracting VINs

The regular expression I used for this task was "[0-9A-HJ-NPR-Z]{14}[0-9]{3}". I started with "[0-9A-Z]" and made it more specific after researching more about VINs. I made it more specific one step at a time, checking the number of matches after each step and looking at some of the matches. I ended up with 8,570 results and then added them to the data frame, putting NA for the rest.

Task 3: Extracting phone numbers

The regular expression I used was "([0-9:numsOr:]{3}|\\([0-9:numsOr:]{3}\\))[-.]{0,10}[0-9:numsOr:]{3}[-.]{0,10}[0-9:numsOr:]{4}" with numsOr = paste(c("zero", "ZERO, ..., "nine", "NINE"), collapse = "|"). I started with "[0-9]{3}-[0-9]{3}-[0-9]{4}" and made the expression more specific until I felt I had found most of the phone numbers. The hardest part of this task for me was figuring out that I could insert :numsOr: like the other ones such as :punct: that R has in its library. I ended up finding 16,734 phone numbers.

Task 4: Extracting emails

The regular expression I used was "[0-9A-Za-z.#_~!$'()*+,;=%-]{1,20}@[0-9a-zA-Z-]{1,20}\\.[a-zA-Z]{3}". I started with the idea of "(something)@(something)\\.(something)" and then filled in all of the somethings. I had to mess around with the expression to get it just right and I found 107 emails.

Task 5: Extracting year

The regular expression I used was "[1][9][4-9][0-9]|[2][0][0|1][0-9]". I did not struggle to much with this expression. The hardest part was deciding the earliest year to include. I ended up deciding on 1940. I got 25,244 matches in the body and 29,376 in the description. There were 23,093 posts with a value in the year column and a year found in both the body and description. 92.6% of these had the same value for year found in all three. I then decided to trust the year entered in the year column the least, mostly because of what I learned from assignment one. I decided to trust the description the most because there are less other numbers such as price and VIN that could be mistaken for a year. So I went through each row of vposts and if there was a year found in the description, I moved that into the year column. If there was no year found, I checked if anything was found in the body, and used that, and if not, I left the year as is. I started with a mean year of 2004.06 and ended with a mean year of 2004.03. So this process did not have any huge changes, but did help make the data more clean and accurate.

Task 6: Extracting model

I tried to find a way to insert the makers as a variable to make the expression shorter like I did for the phone number task, but I could not find a way. I started this problem by finding all the matches with proper spellings, I found 22,791 matches. I used paste and the maker column of vposts to find matches using the regular expression:

"(chevrolet|model|chevy|nissan|infiniti|acura|toyota|lexus|honda|bmw|dodge|ford|NA|chrysler|mazda|jeep|subaru|mercedes|mercedesbenz|hyundai|volkswagen|vw|cadillac|gmc|mini|saturn|mitsubishi|mercury|volvo|kia|land rover|audi|hummer|pontiac|harleydavidson|smart|peterbilt|jaguar|buick|lincoln|scion|saab|tesla|fiat|international|astonmartin|porsche|isuzu|suzuki|bentley|plymouth|oldsmobile|mack|shelby|studebaker|eagle|hudson|alfa romeo|freightliner|geo|mg|rolls royce|maserati|daewoo|leaf|datsun|willys|amc|lamborghini|triumph|ferrari|bugatti|yerfdog|desoto|peugeot|bricklin|zap)[:]{1,6}[a-zA-Z0-9-]{2,20}"

I entered these matches into the model column and then used the top 150 to make another regular expression that I used to look through each body for the 150 most popular models found from the first regular expression. I edited this expression to add misspellings and remove words that were not models. I reached a point where looking for misspellings was taking too much time for the few models I was finding. Since the result was not worth my time, I stopped, and ended up with 33,854 matches using the regular expression:

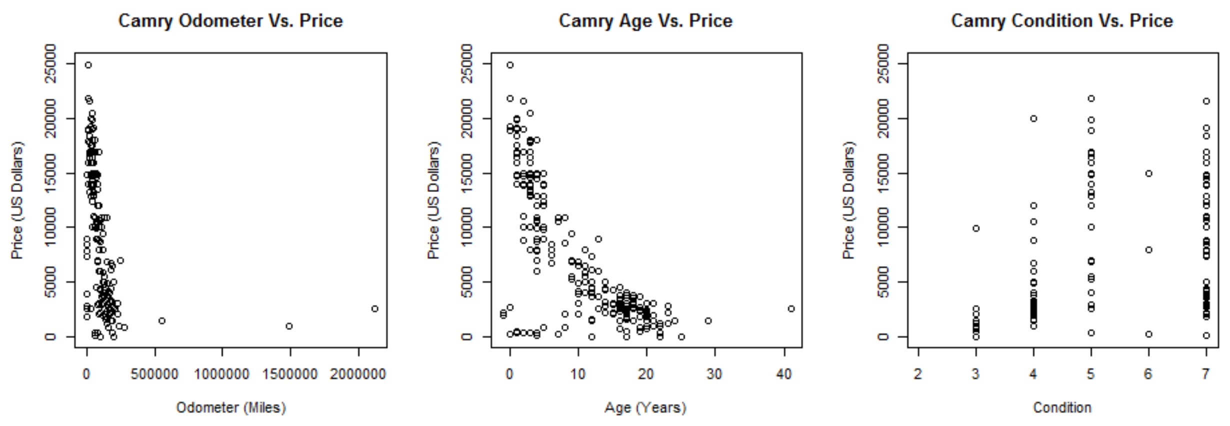
"avalanche|avalon|arapahoeask|a6|accord|altima|a4|buick|blazer|beetle|civic|challenger|cruze|

cooper|corolla|cts|c-class|cr-v|camaro|cheyenna|chevelle|c10|c-60|caprice|cabrio|camry|corvette|carolla|cornett|continental|dakota|deville|dodge|dts|durango|escape|express|edge|exlipse|equinox|escalade|es|elantra|el camino|echo|expedition|explorer|eldorado|fury|f-350|f350|f250|f-250|forester|fusion|f150|f-150|focus|gti|g6|gmc|grand|gs|g37|g35|highlander|impala|impreza|jeep|jetta|legacy|letmoore|ls|lumina|liberty|maxima|mustang|murano|mdx|malibu|matrix|montero|mini cooper|nova|new yorker|navigator|odyssey|optima|outback|pathfinder|pilot|pioneer|pt|prius|previa|passat|protege|rover|rogue|ram|ranger|rav4|rx|rl|s60|silverado|sebring|suburban|sierra|santa|sentra|sienna|sonata|satellite|sr5|s80|scion|serville|tribute|taurus|town|tacoma|tundra|ts80|t800|tahoe|tl|tsx|versa|veloster|wrangler|windsor|x5|xterra|x3|yukon|yaris|zdx|9-3|200|328i|4runner|325i|300|525i|928|1500|6s"

After this, I entered the results into the model column of vposts, replacing anything found by the first expression. I used unique() to correct some spelling mistakes and remove any words that got picked up that were clearly not models like "is" and "was." I ended up with a total of 33,947 entries of vposts with a model.

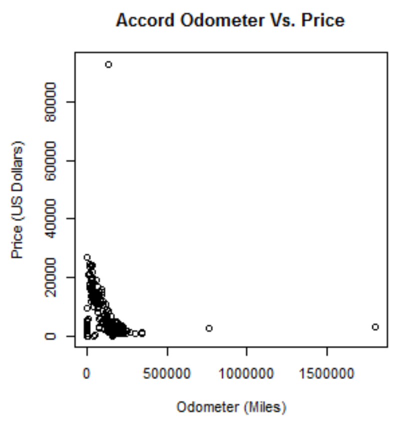
Part 2: Modeling

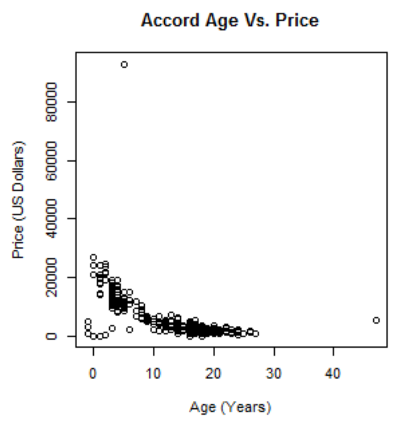
I decided to pick the two models Camry and Accord. They both had around 300 posts and seemed fairly clean. Starting with Camry, I plotted price with the three variables. The plot is below.

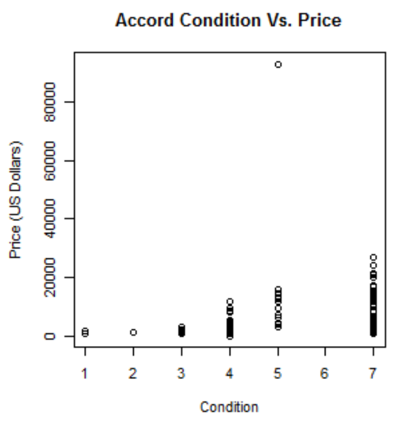


Looking at the plots, there seems to be a relationship present for all three, so a linear model seems appropriate.

Next for Accord, I plotted price with the three variables:







There appears to be a relationship between price and the three variables, so a linear model also seems appropriate. For both models, we are assuming that a linear relationship is present between price and the three variables. We are also assuming that the three predictor variables can be treated as fixed values, independence of errors, lack of multicollinearity in the predictors, and constant variance.

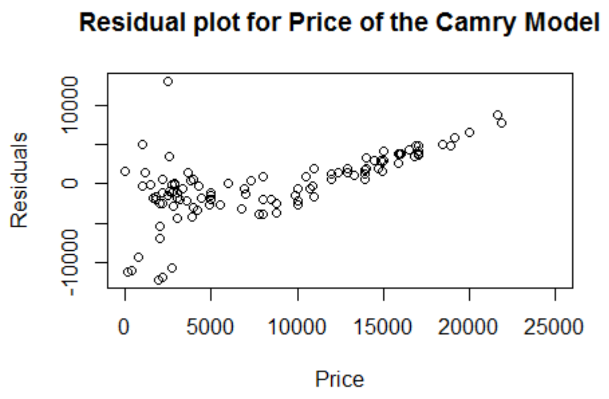
Using the lm() function, I created the liner models for Camry and Accord:

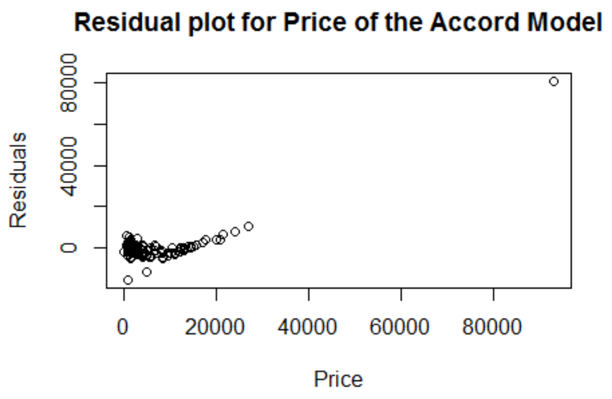
Camry: $\text{yhat} = 14140.0\beta_0 - 0.003\beta_1 - 598.2\beta_2 - 16.94\beta_3$

Accord: $\text{yhat} = 14660.0\beta_0 - 0.0005\beta_1 - 764.2\beta_2 + 284.7\beta_3$

Where β_0 = intercept, β_1 = odometer, β_2 = age, β_3 = condition

I used the predict() function to get a vector of predictions for both and then plotted the actual values for price against the residuals:





Residual plots ideally look random and have no noticeable pattern or relationship. Looking at the plots above, there are noticeable patterns. So it seems like a linear model may not be a very good model. I would not use either of these models if I were buying a car.

Resources

- https://stat.ethz.ch/R-manual/R-devel/library/base/html/regmatches.html
- https://en.wikipedia.org/wiki/Vehicle_identification_number
- https://stat.ethz.ch/R-manual/R-devel/library/base/html/grep.html
- http://stackoverflow.com/questions/7597559/grep-in-r-with-a-list-of-patterns
- https://en.wikipedia.org/wiki/Email_address#Valid_email_addresses
- http://stackoverflow.com/questions/6286313/remove-an-entire-column-from-a-data-frame-in-r
- http://stackoverflow.com/questions/13640188/converting-text-to-lowercase
- https://en.wikipedia.org/wiki/Linear_regression#Assumptions
- Piazza
- Lecture

Code Appendix

```
# Loading the data:
load("~/UC Davis/STA 141/HW 1/vehicles.rda")
```

# Task 1: Price
# ==========

```
rx1 = "\\$([1-9]{0,1}[ ]{0,1}[0-9,]{1,2}[ ]{0,1}[0-9][ ]{0,1}[0-9][ ]{0,1}[0-9])"
table(grepl(rx1, vposts$body)) # 13010

matches1 = gregexpr(rx1, vposts$body)
whatMatched1 = regmatches(vposts$body, matches1) # Some will have more than one

bodyPrices = sapply(whatMatched1, function(body){
        numPrices = length(body)
        # Nothing was found in the body:
        if(numPrices == 0){
                value = NA
        }
        # Only one price found: ( need to remove $ and , and change to numeric)
        else if(numPrices == 1) {
                value = gsub("\\$|,| ", "", body[1])
                value = as.numeric(value)
        }
        # More than one price found:
        else{
                # Rm $ and , and change to numeric for each price:
```

```
                prices = sapply(body, function(p){
                        result = gsub("\\$|,| ", "", body[p])
                         as.numeric(result)
                })
                # Then compare and decide what price to use:
                value = max(prices) # Returning max for now
        }
})

# Same for description or title?:
table(grepl(rx1, vposts$description)) # 1031
table(grepl(rx1, vposts$title)) # 29303

# Much more for title:
matchesT = gregexpr(rx1, vposts$title)
whatMatchedT = regmatches(vposts$title, matchesT)
titlePrices = sapply(whatMatchedT, function(body){
        numPrices = length(body)
        # Nothing was found in the body:
        if(numPrices == 0){
                value = NA
        }
        # Only one price found: ( need to remove $ and , and change to numeric)
        else if(numPrices == 1) {
                value = gsub("\\$|,| ", "", body[1])
                value = as.numeric(value)
        }
        # More than one price found:
        else{
                # Rm $ and , and change to numeric for each price:
                prices = sapply(body, function(p){
                        result = gsub("\\$|,| ", "", body[p])
                        as.numeric(result)
                })
                # Then compare and decide what price to use:
                value = max(prices) # Returning max for now
        }
})

# Comparing both to the price column:
```

```
bodyCompare = sapply(1:34677, function(row){
        if((!is.na(vposts[row, ]$price)) & (!is.na(bodyPrices[row]))){
                if(vposts[row, ]$price == bodyPrices[row])
                        result = TRUE
                else{
                        result = FALSE
                }
        }
        else{
                result = NA
        }
})

titleCompare = sapply(1:34677, function(row){
        if((!is.na(vposts[row, ]$price)) & (!is.na(titlePrices[row]))){
                if(vposts[row, ]$price == titlePrices[row])
                        result = TRUE
                else{
                        result = FALSE
                }
        }
        else{
                result = NA
        }
})
percSameB = mean(bodyCompare == TRUE, na.rm = TRUE) # .85
percSameT = mean(titleCompare == TRUE, na.rm = TRUE) # .99
```

# Task 2: VIN
# ==========

```
# Finding VINs:
rx = "[0-9A-HJ-NPR-Z]{14}[0-9]{3}"
matches = regexpr(rx, vposts$body)
whatMatched = regmatches(vposts$body, matches)
table(grepl(rx, vposts$body)) # 8570 true

# Adding to vposts:
rowsMatch = grep(rx, vposts$body)
vposts$VIN = NA
vposts[rowsMatch,]$VIN = whatMatched
```

# Task 3: Phone numbers

# ==================

```
# Finding best regular expression:
# start: rx3 = "[0-9]{3}-[0-9]{3}-[0-9]{4}" True: 8640
nums = c("zero", "ZERO", "one", "ONE", "two", "TWO", "three", "THREE", "four", "five",
        "FIVE", "six", "SIX", "seven", "SEVEN", "eight", "EIGHT", "nine", "NINE")
numsOr = paste(nums, collapse="|")
rx3 = "([0-9:numsOr:]{3}|\\([0-9:numsOr:]{3}\\))[ -.]{0,10}[0-9:numsOr:]{3}[ -.]{0,10}[0-
        9:numsOr:]{4}"
table(grepl(rx3, vposts$body)) # 16734

# Adding to vposts:
matches3 = regexpr(rx3, vposts$body)
whatMatched3 = regmatches(vposts$body, matches3) # Might be more than one
rowsMatch3 = grep(rx3, vposts$body)
vposts$phoneNumber = NA
vposts[rowsMatch3, ]$phoneNumber = whatMatched3
```

# Task 4: Emails

# ============

```
# Finding best regular expression:
rx4 = "[0-9A-Za-z.#_~!$'()*+,;=%-]{1,20}@[0-9a-zA-Z-]{1,20}\\.[a-zA-Z]{3}"
table(grepl(rx4, vposts$body)) # 107

# Adding to vposts:
matches4 = regexpr(rx4, vposts$body)
whatMatched4 = regmatches(vposts$body, matches4)
rowsMatch4 = grep(rx4, vposts$body)
vposts$email = NA
vposts[rowsMatch4, ]$email = whatMatched4
```

# Task 5: Year

# ==========

```
mean(vposts$year) # 2004.064 Notice- no NAs

# Finding best regular expression:
rx5 = "[1][9][4-9][0-9]|[2][0][0|1][0-9]"
table(grepl(rx5, vposts$body)) # True for 25244
table(grepl(rx5, vposts$description)) # True for 29376
```

```
# Getting values from body:
matchesB = regexpr(rx5, vposts$body)
whatMatchedB = as.numeric(regmatches(vposts$body, matchesB))

# Getting values from description:
matchesD = regexpr(rx5, vposts$description)
whatMatchedD = as.numeric(regmatches(vposts$description, matchesD))

# Creating temporary column to compare with:
vposts$yearB = NA
vposts$yearD = NA
rowsMatchB = grep(rx5, vposts$body)
rowsMatchD = grep(rx5, vposts$description)
vposts[rowsMatchB, ]$yearB = whatMatchedB
vposts[rowsMatchD, ]$yearD = whatMatchedD

# Now have year (in data), yearB (year found in body), yearD (year found in description)
# Comparing:
haveAll3 = (!is.na(vposts$year) & !is.na(vposts$yearB) & !is.na(vposts$yearD))
numAll3 = sum(haveAll3) # 23093
all3Same = ((vposts$year == vposts$yearB) & (vposts$year == vposts$yearD) & (vposts$yearB
        == vposts$yearD))
# The percent of the posts that have a year in all three that have the same year in all three:
perc3Same = sum(all3Same, na.rm = TRUE) / numAll3 # 0.9263

# Go through each row of vpost and if there is a year found in the description,
# move that into the year column. If there is no year found, check if anything is
# found in the body, and use that, and if not, I leave the year as is:
yearChange = sapply(1:34677, function(row){
        if (!is.na(vposts[row, ]$yearD) == TRUE){
                year = vposts[row, ]$yearD
        }
        else if (!is.na(vposts[row, ]$yearB) == TRUE){
                year = vposts[row, ]$yearB
        }
        else{
                year = vposts[row, ]$year
        }
 })
```

```
vposts$year = yearChange
# How did that change the col?
mean(vposts$year) # 2004.026

# Deleting cols I don't need any more:
vposts$yearB = NULL
vposts$yearD = NULL
```

# Task 6: Model
# ===========

```
# Finding regular expression:
makers = paste(unique(vposts$maker), collapse = "|")
rx6 = "(chevrolet|model|chevy|nissan|infiniti|acura|toyota|lexus|honda|bmw|dodge|ford|NA|
        chrysler|mazda|jeep|subaru|mercedes|mercedes-benz|hyundai|volkswagen|vw|cadillac|
        gmc|mini|saturn|mitsubishi|mercury|volvo|kia|land rover|audi|hummer|pontiac|harley
        davidson|smart|peterbilt|jaguar|buick|lincoln|scion|saab|tesla|fiat|international|aston
        martin|porsche|isuzu|suzuki|bentley|plymouth|oldsmobile|mack|shelby|studebaker|eagle|h
        udson|alfa romeo|freightliner|geo|mg|rolls royce|maserati|daewoo|leaf|datsun|willys|amc|
        lamborghini|triumph|ferrari|bugatti|yerfdog|desoto|peugeot|bricklin|zap)[ :]{1,6}[a-zA-
        Z0-9-]{2,20}"
table(grepl(rx6, vposts$body, ignore.case = TRUE)) # 26083

matches6 = regexpr(rx6, vposts$body, ignore.case = TRUE)
whatMatched6 = regmatches(vposts$body, matches6)
rowsMatch6 = grep(rx6, vposts$body, ignore.case = TRUE)

# Used to look for misspellings or other ways to get model:
noMatches = vposts[-(rowsMatch6), ]
head(noMatches$body, 100)

# Now want to take the unique() of the models found and search for those instead of the maker
# and then the model:
head(whatMatched6, 100)
whatMSplit = strsplit(whatMatched6, " ") # Now a list
head(whatMSplit, 100)
models = sapply(1:length(whatMSplit), function(list){
        whatMSplit[[list]][2]
})
```

```
# Enter these into vposts:
vposts$model = NA
vposts[rowsMatch6, ]$model = models

# Use the more popular of the models and try to find those in other bodies:
# Like before but with only the more popular ones:
top150 = tail(sort(table(vposts$model)), 150)
mods = paste(names(top150), collapse = "|")

rx7 =
"avalanche|avalon|arapahoeask|a6|accord|altima|a4|buick|blazer|beetle|civic|challenger|cruze|
        cooper|corolla|cts|c-class|cr-v|camaro|cheyenna|chevelle|c10|c-60|caprice|cabrio|camry|
        corvette|carolla|cornett|continental|dakota|deville|dodge|dts|durango|escape|express|
        edge|exlipse|equinox|escalade|es|elantra|el camino|echo|expedition|explorer|eldorado|fury
        |f-350|f350|f250|f-250|forester|fusion|f150|f-150|focus|gti|g6|gmc|grand|gs|g37|g35
        |highlander|impala|impreza|jeep|jetta|legacy|letmoore|ls|lumina|liberty|maxima|mustang|
        murano|mdx|malibu|matrix|montero|mini cooper|nova|new yorker|navigator|odyssey|
        optima|outback|pathfinder|pilot|pioneer|pt|prius|previa|passat|protege|rover|rogue|ram|rang
        er|rav4|rx|rl|s60|silverado|sebring|suburban|sierra|santa|sentra|sienna|sonata|satellite|sr5|s8
        0|scion|serville|tribute|taurus|town|tacoma|tundra|ts80|t800|tahoe|tl|tsx|versa|veloster|wran
        gler|windsor|x5|xterra|x3|yukon|yaris|zdx|9-3|200|328i|4runner|325i|300|525i|928|1500|6s"

table(grepl(rx7, vposts$body, ignore.case = TRUE)) # 33854

matches7 = regexpr(rx7, vposts$body, ignore.case = TRUE)
whatMatched7 = regmatches(vposts$body, matches7)
rowsMatch7 = grep(rx7, vposts$body, ignore.case = TRUE)

# Used to look for misspellings or other ways to get model:
noMatches7 = vposts[-(rowsMatch7), ]
head(noMatches7$body, 100)
noMatches7[401:450,]$body

# Now entering those results into vposts$model
vposts[rowsMatch7, ]$model = whatMatched7

# Correcting misspellings and incorrect models (some):
vposts$model = tolower(vposts$model)
sort(unique(vposts$model))
```

```
vposts[which(vposts$model == "windstar"),]$model = "winstar"
vposts[which(vposts$model == "was"),]$model = NA
vposts[which(vposts$model == "toyota"),]$model = NA
vposts[which(vposts$model == "to"),]$model = NA
vposts[which(vposts$model == "runs"),]$model = NA
vposts[which(vposts$model == "ready"),]$model = NA
vposts[which(vposts$model == "insurance"),]$model = NA
vposts[which(vposts$model == "in"),]$model = NA
vposts[which(vposts$model == "it"),]$model = NA
vposts[which(vposts$model == "is"),]$model = NA
vposts[which(vposts$model == "has"),]$model = NA
vposts[which(vposts$model == "for"),]$model = NA
vposts[which(vposts$model == "fuel"),]$model = NA
vposts[which(vposts$model == "gas"),]$model = NA
vposts[which(vposts$model == "carolla"),]$model = "corolla"
vposts[which(vposts$model == "auction"),]$model = NA
vposts[which(vposts$model == "\npathfinder"),]$model = "pathfinder"

# Total number of models:
sum(!is.na(vposts$model) == TRUE) # 33947
```

# Modeling
# =======

```
# Picking the two models:
sort(table(vposts$model))
# camry and accord? - Diff makers, around 300, seems good
camry = subset(vposts, vposts$model == "camry") # 289
accord = subset(vposts, vposts$model == "accord") # 319

# Making condition not categorical:
condC = sapply(1:289, function(row){
  if (!is.na(camry[row, ]$condition) == TRUE){
    if(camry[row, ]$condition == "excellent"){
      val = 7
    }
    else if(camry[row, ]$condition == "new"){
      val = 6
    }
    else if(camry[row, ]$condition == "like new"){
      val = 5
    }
    else if(camry[row, ]$condition == "good"){
      val = 4
    }
    else if(camry[row, ]$condition == "fair"){
      val = 3
    }
    else if(camry[row, ]$condition == "used"){
      val = 2
    }
    else{
      val = NA
    }
  }
  else{
    val = NA
  }
})

condA = sapply(1:319, function(row){
  if (!is.na(accord[row, ]$condition) == TRUE){
    if(accord[row, ]$condition == "excellent"){
      val = 7
    }
    else if(accord[row, ]$condition == "new"){
      val = 6
    }
    else if(accord[row, ]$condition == "like new"){
      val = 5
    }
    else if(accord[row, ]$condition == "good"){
      val = 4
    }
    else if(accord[row, ]$condition == "fair"){
      val = 3
    }
    else if(accord[row, ]$condition == "used"){
      val = 2
    }
    else if(accord[row, ]$condition == "salvage"){
      val = 1
    }
    else{
      val = NA
    }
  }
  else{
    val = NA
  }
})

camry$condition2 = condC
accord$condition2 = condA

vposts$age = 2015 - vposts$year

par(mfrow = c(1,3))
plot(camry$odometer, camry$price, xlab = "Odometer (Miles)", ylab = "Price (US Dollars)",
main = "Camry Odometer Vs. Price")
plot(camry$age, camry$price, xlab = "Age (Years)", ylab = "Price (US Dollars)", main =
"Camry Age Vs. Price")
plot(camry$condition2, camry$price, xlab = "Condition", ylab = "Price (US Dollars)", main =
"Camry Condition Vs. Price")

plot(accord$odometer, accord$price, xlab = "Odometer (Miles)", ylab = "Price (US Dollars)",
main = "Accord Odometer Vs. Price")
plot(accord$age, accord$price, xlab = "Age (Years)", ylab = "Price (US Dollars)", main =
"Accord Age Vs. Price")
plot(accord$condition2, accord$price, xlab = "Condition", ylab = "Price (US Dollars)", main =
"Accord Condition Vs. Price")

camryModel = lm(price~odometer+age+condition2, data = camry)
summary(camryModel)
predC = predict(camryModel, camry)

accordModel = lm(price~odometer+age+condition2, data = accord)
summary(accordModel)
predA = predict(accordModel, accord)

par(mfrow = c(1,2))
```

```
resC = camry$price - predC
plot(camry$price, resC, xlab = "Price", ylab = "Residuals", main = "Residual plot for Price of the
Camry Model")

resA = accord$price - predA
plot(accord$price, resA, xlab = "Price", ylab = "Residuals", main = "Residual plot for Price of
the Accord Model")
```