Regular Expressions

Hasan Abuelreich

November 15, 2015

# Step 1

First I figured out the regular expression for one price. I approached it by telling the regular expression to look for "$" followed by numbers and later discovered that I had to also include an optional comma so I can capture prices with commas as well. I later created a function that will grab all the prices out for me and realized that they were in order but the length did not match the price column anmyore. Hence, I created another function that had a vector with the correct length with NA observations inside it. I used grepl to find out which ones had prices in the body and filled all the true with the prices I found in the get\_price function leaving the unknown prices NAs. I tabled my matched priced with vposts$prices and got 3,128 False and 10,090 True. Hence, 10,090 prices from the body matched giving us an error rate of 31%.

load(url("http://eeyore.ucdavis.edu/stat141/Data/vehicles.rda"))  
#regular expression used to get price accounting for commas and finding price by "$"  
myprice = regexpr("\\$[0-9]+,?[0-9]+", vposts$body[2])  
regmatches(vposts$body[2], myprice)

## [1] "$18,797"

get\_price = function(s){  
 #Pick out the price from a string s  
 #This regular expression searches for a $ and a comma is optional to find  
 #I Used \\ because $ is a metacharacter  
 reg\_1=regexpr("\\$[0-9]+,?[0-9]+",s)  
 regmatch\_1=regmatches(s,reg\_1)  
 #I use gsub here to subsitute the $ with nothing so I can match it later to the price column  
 gsub\_1=gsub("\\$|,", "", regmatch\_1)  
 #I make it numeric for the comparison later  
 as.numeric(gsub\_1)  
}  
#In order to compare them to the price column they must be in the same positions but are not  
#I approach this by creating a fuunction that has a vector of the correct length NA's and will use grepl to find the corresponding positions true and fill them up with the observations I found  
  
match = function(s){  
   
 # want to see which ones have price into s if not NA  
 # Create a vector of the correct length with NA; I will plug in the matches found  
 # When a match is not found it will return NA  
 out = rep(NA, length(s))  
   
 # Find the ones that do have matchs  
 grepl\_1=grepl("\\$[0-9]+,?[0-9]+",s)  
   
 # Replace NA's with price, since we know we can find it  
 out[grepl\_1] = get\_price(s[grepl\_1])  
   
 return(out)  
 # Same as  
 # out  
}   
table(match(vposts$body) == vposts$price)

##   
## FALSE TRUE   
## 3128 10090

# Step 2

I first figured out the regular expression needed by testing it on one body. I realized that "VIN" & "Stock" appear in many cases instead of the VIN number, hence I used gsub to take care of that issue. I created a function that grabbed all the VIN numbers back and found that there were 5,232 VIN numbers present in the body for different vehicles. In order to add these to the the correct vehicles, like previous in prices I made a function that had a vector of the correct length NA's and then used grepl to find out the TRUE cases that had VIN and for the TRUE cases I matched the VIN numbers I found. Now that they are in the correct order, I added them to my data by creating a column "VIN" and making it equal to my matches.

# I first looked at an individual string to find a pattern for VIN number in order to know what to look for  
# I noticed that VIN number starts with VIN: and there are letters and numbers after that.  
# after i read about the VIN number I noticed that the VIN number finishes with numbers  
# I also read that VIN number must be upper case but since there are so many observations in the data that is not written appropriatley included lower case to get the best result.  
# it can have as many as spaces after that or even no spaces [ \*]  
regexpr\_VIN=regexpr("VIN:?[ \*][0-9a-zA-Z]+[0-9]+",vposts$body[2])  
substr(vposts$body[1], 106,106+22-1)

## [1] "G1FT1EW1C9106920Stock "

regmatches(vposts$body[1], regexpr\_VIN)

## [1] "IN: 2G1FT1EW1C9106920S"

#I can now function this for all the vehichles   
get\_VIN = function(v){  
 #pick out vin from a string  
 regexpr\_VIN = regexpr("VIN:?[ \*][0-9a-zA-Z]+[0-9]+",v)  
 regmatches\_VIN = regmatches(v,regexpr\_VIN)  
 # so I got the vin number followed with VIN:  
# so now I have to remove the VIN: and any spaces so I use gsub  
 gsub("VIN:?( \*)","",regmatches\_VIN)  
}  
#To get the number of NA's  
length(get\_VIN(vposts$body))

## [1] 5232

#In order to combine the VIN numbers to the our observations we need equal length vectors  
#hence like previously in price I created a vector of NA's with correct length & filled up my observations  
match\_VIN=function(v){  
 # want to see which ones have VIN into v if not NA  
   
 # Create a vector of the correct length  
 out = rep(NA, length(v))  
   
 # Find the ones that do have matchs  
 grepl\_VIN=grepl("VIN:?[ \*][0-9a-zA-Z]+[0-9]+",v)  
   
 # Replace NA's with price, since we know we can find it  
 out[grepl\_VIN] = get\_VIN(v[grepl\_VIN])  
   
 return(out)  
 # Same as  
 # out  
}   
  
# making a column and add to data.frame  
vposts$VIN=match\_VIN(vposts$body)

# Step 3

To match the phones I noticed that many have parenthesis and a dash, hence in my regular expression I started it with an otional parenthesis and had or a "-" as well to capture as many phone numbers as possible. I also realized that many phone numbers also had a dash after the first three numbers, hence I also added an optional "-" over there using ? in my regular expression. Now that I have my regular expression I used a function similar to what I have been using for the previous steps to make it general and used it on the each body of each vehicle. I used length() for these matches and found that there are 16,818 phone numbers. In order to add phone numbers to the correct observations I created a function that had a vector of NA's and used grepl to determine which observtaions had TRUE for phone numbers and I fit the TRUE positions with my observations. Therefore, now I can add the column because it is the same lenghth and the phone numbers are in the right place for each observation.

#I tried my regular expression on one before generalizing a function  
#I included optional parenthesis and discovered an optional dash was needed as well to capture as many phone numbers as possible  
reg\_phone = regexpr("\\(?[0-9]+[\\)|-] ?[0-9]+-?[0-9]+", vposts$body[1])  
regmatches(vposts$body[1],reg\_phone)

## [1] "(508) 205-1046"

get\_phone=function(p){  
   
 regexpr\_phone=regexpr("\\(?[0-9]+[\\)|-] ?[0-9]+-?[0-9]+", p)  
 regmatches(p,regexpr\_phone)  
   
}  
  
length(get\_phone(vposts$body))

## [1] 16818

# In order to add the phone numbers to my data same length is needed  
match\_phone=function(p){  
 # Create a vector of the correct length  
 out = rep(NA, length(p))  
   
 # Find the ones that do have matchs  
 grepl\_3=grepl("\\(?[0-9]+[\\)|-] ?[0-9]+-?[0-9]+",p)  
   
 # Replace NA's with price, since we know we can find it  
 out[grepl\_3] = get\_phone(p[grepl\_3])  
   
 return(out)  
 # Same as  
 # out  
}   
  
vposts$phone = match\_phone(vposts$body)

# Step 4

I first randomly sampled 30 observations in order to find an e-mail to work off of. However, no e-mails were found which does make sense because there were very few. I used intuition that e-mail has an "@" sign and used a combination of different endings in my regular expression. I plugged it into a generalized function right away since I couldn't find any e-mails in my random smapling. The results were 101 e-mails found, which makes sense why there were no e-mails found when I was random sampling observations. In order to add these to my data I created a function that had a length of NA's corresponding to vposts and used grepl to find the postition of he e-mails. Then I subsetted my matched e-mails for all the true corresponding positions. That gave me a vector with the same length as my data and the e-mails were in the correct positions, hence I could just add that column to my data now.

#I randomly smapled about 30 vehicle bodies to try and find an e-mail to work off of for my regular expression and did not find any  
#Hence I began working on my function straight away using intuition that e-mail requires "@" and possible endings of an e-mail  
  
get\_mail = function(m){  
 #this will pick out e-mail from a string  
 reg\_mail = regexpr("[a-zA-Z0-9]+@[a-zA-Z0-9]+\\.(com|net|edu|org)+",m)  
 #this matches that e-mail and returns its output  
 regmatches(m,reg\_mail)  
}  
length(get\_mail(vposts$body))

## [1] 101

#In order to be able to add the e-mails to the corresponding vehicles in my data i create the following function leaving NA's for no e-mails  
match\_mail=function(m){  
 # want to see which ones have Mail into m if not NA  
   
 # Create a vector of the correct length  
 out = rep(NA, length(m))  
   
 # Find the ones that do have matchs  
 grepl\_4=grepl("[a-zA-Z0-9]+@[a-zA-Z0-9]+\\.(com|net|edu|org)+",m)  
   
 # Replace NA's with emails, since we know we can find it  
 out[grepl\_4] = get\_mail(m[grepl\_4])  
   
 return(out)  
 # Same as  
 # out  
}   
mail = match\_mail(vposts$body)  
#This will add mail as a coulmn to vposts  
vposts$mail = mail  
colnames(vposts)

## [1] "id" "title" "body" "lat"   
## [5] "long" "posted" "updated" "drive"   
## [9] "odometer" "type" "header" "condition"   
## [13] "cylinders" "fuel" "size" "transmission"  
## [17] "byOwner" "city" "time" "description"   
## [21] "location" "url" "price" "year"   
## [25] "maker" "makerMethod" "VIN" "phone"   
## [29] "mail"

# Step 5

To find the years in the body I created a function to look for the regex contatining the numbers 19 foloowed by two numbers [0-9] or 2000-2009 or 2010-2015. I function this to look for these numebrs and then looked in the body for them. I found that there are 22,169 years for different observations in the body. In order to add them to my data and check the corresponding matches I created a function that had a vector of NA's with the correct length of my data. Then I used grepl to find the TRUE positions and subsetted the true positions in my vector and used the match function I previously created to match the years found to the corresponding positions. Following that I set my matched years to numeric and compared them with vposts$year column and got 20,549 matches out of the 22,169 I found. An impressive error rate of 7.8% of this messy data.

get\_year = function(y){  
 #Using intution I created a regex function for what seemed to me possible years so anything starting with 19 contating two numbers or 200[0-9] or 201[0-5] for the last five years  
 reg\_year = regexpr("(19[0-9]{2})|(200[0-9])|(201[0-5])",y)  
 regmatches(y,reg\_year)  
   
}  
  
  
length(get\_year(vposts$body))

## [1] 25136

match\_year=function(y){  
 # want to see which ones have year into y if not NA  
   
 # Create a vector of the correct length  
 out = rep(NA, length(y))  
   
 # Find the ones that do have matchs  
 grepl\_5=grepl("(19[0-9]{2})|(200[0-9])|(201[0-5])",y)  
   
 # Replace NA's with year, since we know we can find it  
 out[grepl\_5] = get\_year(y[grepl\_5])  
   
 return(out)  
 # Same as  
 # out  
}   
  
my\_year = as.numeric(match\_year(vposts$body))  
  
table(my\_year==vposts$year)

##   
## FALSE TRUE   
## 2121 23015

# Step 6

After looking at different columns in the data set I decided that the description almost always had a maker followed by the model so I chose the description to look for the model names. I noticed that the model is also usually followed by the maker so I wanted to make a regular expression conditional on the unique maker so I can capture all kinds of different models for the unique makers. Hence, I used paste to collapse all the unique makers by the "|" condition/or sign so I can capture all the different unqique makers. That was the middle part of my regular expression and the last part is to look for letters and numbers because many models contain a number as well. I functioned this and used gsub to get rid of the maker name so my code would just return the model. Hence, I have a vector now with all the model names and I can subset the models I want from that vector and build a regression model for my next step.

#I assigned all unique makers to makers for the steps to follow   
makers = unique(vposts$maker)  
makers

## [1] "chevrolet" "nissan" "infiniti"   
## [4] "acura" "toyota" "lexus"   
## [7] "honda" "bmw" "dodge"   
## [10] "ford" NA "chrysler"   
## [13] "mazda" "jeep" "subaru"   
## [16] "mercedes" "hyundai" "volkswagen"   
## [19] "cadillac" "gmc" "mini"   
## [22] "saturn" "mitsubishi" "mercury"   
## [25] "volvo" "kia" "land rover"   
## [28] "audi" "hummer" "pontiac"   
## [31] "harley davidson" "smart" "peterbilt"   
## [34] "jaguar" "buick" "lincoln"   
## [37] "scion" "saab" "tesla"   
## [40] "fiat" "international" "aston martin"   
## [43] "porsche" "isuzu" "suzuki"   
## [46] "bentley" "plymouth" "oldsmobile"   
## [49] "mack" "shelby" "studebaker"   
## [52] "eagle" "hudson" "alfa romeo"   
## [55] "freightliner" "geo" "mg"   
## [58] "rolls royce" "maserati" "daewoo"   
## [61] "leaf" "datsun" "willys"   
## [64] "amc" "lamborghini" "triumph"   
## [67] "ferrari" "bugatti" "yerfdog"   
## [70] "desoto" "peugeot" "bricklin"   
## [73] "zap"

# Idea - build up a regular expression  
# build a regular expression in to parts because I realized many have the maker followed by the model name in description  
firstpart = ".\*("  
# Using paste here will collapse all my makers and have then next to each other and it will seperate them by the or "|" sign so my regexpression will work for all makers  
models = paste(makers, collapse='|')  
#last part of the regular expression will return my model with letters and numbers since many models are followed by the model number  
lastpart = ") ([0-9a-zA-Z]+).\*"  
  
# Final regular expr to use:  
#paste0 will glue all my regular expressions together and the middle part will be all the unique makers  
pattern = paste0(firstpart, models, lastpart)  
  
findmodel = function(s){  
 # return model   
 # gsub subsitutes the maker name and will only return the model  
 models = gsub(pattern, "\\2", s, ignore.case=TRUE)  
 tolower(models)  
}  
my\_models = findmodel(vposts$description)  
class(my\_models)

## [1] "character"

match\_model=function(z){  
 # want to see which ones have price into s if not NA  
 # Create a vector of the correct length with NA; I will plug in the matches found  
 # When a match is not found it will return NA  
 out = rep(NA, length(z))  
   
 # Find the ones that do have matchs  
 grepl\_6=grepl(pattern,z)  
   
 # Replace NA's with price, since we know we can find it  
 out[grepl\_6] = findmodel(z[grepl\_6])  
   
 return(out)  
 # Same as  
 # out  
}   
  
  
allmodels = match\_model(vposts$description)  
vposts$Model = allmodels

# Part 2: Modeling

I chose the honda civic and totyota camry for my regression. Hence, I subsetted camry and and civic in there own data sets each so I can run two seperate regression models and analyzie the two. For civic it seems that odometer is postive and would increase the price of the car, which most might find a little weird. However, it appears that the variable is not statistically significant over here and that may be because generally civics last a very long time relative to other vehicles (american manufcaturers) and hence the odomoter might effect price but not really. Condition on the other hand effect price in a postive way when the car is new/like new and is statistically significant. Hence, it is safe to say that when a honda civic is like new it would sell for a higher value than one in worse condition. Fair and good conidtion appear to have a negative effect on price, however they are not statistically significant in this model. Year is statistically significant even at the 1% significance level testing with a p-value close to zero. The coefficient is postive and since the higher the year the newer the car, this implies that as year gets higher the cars price will increase, which is what we would expect from a newer car. Hence, for the honda civic the main factors that contribute to the price are condition and year, while odomoter really does not because it is not statistically significant at the 5% level of significance. The regression for Toyota camry returned different results that for the honda civic. It appears to be that for the camry odometer has a negtive impact, however it is not statisticaly significant at the 5% level of significance with a p-value closeer to 1. Condition also appears to not be statistically siginficant at all with really high p-values closer to 1. Year on the other hand is a statistically significant variable and it has a postive coefficient, which implies as year gets larger or the car gets newer the price should go up as well. Hence, it is safe to say that for a camry the main influence for the price is year. This sounds reasonable because camrys are widely used as rental cars because of the low maintenence and a one year old camry standard option could usually be obtained for around 17,000 at an auction. Hence, odometer does not have much influence in that case and condition is usually does not vary that much for a rental car as they sell them after a year. This may be one of the reasons that we got these results. Following that to find our if location effects these vehicles prices in general I decided to model camry and civic together to get a greater number observations and more accuracy for the loaction. It turns out that very few locations are statistically significant and most have really high p-values and are not significant. The locations that do seem to appear significant are the ones that are "nicer" or richer zip codes. For instance bay area locations such as palo alto is statistically significant, but that is one of the richest zip codes in theh United States and that may be why. Hence, it is safe to say that in general location is not statistically significant for the most part, except the really rich areas were some people may be willing to pay over the price of the actual used car.In order to predict the price for a certain camry or civic from the process will be the same. First we will look at the intercept and that will be our beta0 which will be the same from all the civics and a different b0 that is the same for all the camrys, which is obtained as the intercept from the model summary . Then for condition, depeding on the car's condition we will either add or subtract in the equation according to the coefficient of condition. Respectively odomoeter number will be multiplied by the odometer coefficient and the year will be multiplied by the year coefficient and that should give us an estimate for the price of the vehicle. I would not use this tool if I was buying or selling a car because some of the variables in my model were not statistically significant. This is a good tool and gives an approximate estimate, but I would rather use a kellys blue book because they would be much more accurate and probably sampled many cars to get those price estimates.

# I used the "fixes" from assignment 1 for age, odometer, and price before starting to regress  
#odometer greater than one million is not reasonable and i eleminated results above that assigning them to NA  
vposts$odometer[vposts$odometer > 1e6] <- NA  
  
#fixed price for unreasonable price under $200 the maker median was assigned  
idx = !is.na(vposts$price) & vposts$price < 200  
fixprice = c(acura = 7500, audi = 10499.5, bmw = 10500, buick = 3250, cadillac = 6995, chevrolet = 6000, chrysler = 4996.5, dodge = 6991.5, ford = 5900, gmc = 7750, harleydavidson = 7500, honda = 5500, hyundai = 6319, infiniti = 9697.5, international = 11000, jeep = 7050, kia = 6988, landrover = 11995, lexus = 11999.5, lincoln = 4500, mack = 19000, mazda = 5998, mercedes = 7995, mercury = 3100, mini = 9495, mitsubishi = 3800, nissan = 6995, oldsmobile = 2500, plymouth = 2500, pontiac = 2995, porsche = 20000, scion = 7992.5, shelby = 39500, subaru = 7797.5, suzuki = 3600, toyota = 7857.5, volkswagen = 5995, volov = 5800)   
vposts$price[idx] = fixprice[vposts$maker[idx]] # set the prices I want for each maker  
  
  
#selecting my models I chose civic and accord   
#subsetting only for civic so I can fit a regression model  
civic = vposts$Model == 'civic'  
civic[is.na(civic)] = FALSE  
vposts\_civic = vposts[civic, ]  
model\_civic = lm(vposts\_civic$price ~ vposts\_civic$odometer + vposts\_civic$condition + vposts\_civic$year)  
summary(model\_civic)

##   
## Call:  
## lm(formula = vposts\_civic$price ~ vposts\_civic$odometer + vposts\_civic$condition +   
## vposts\_civic$year)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3472.9 -630.6 144.4 495.9 2504.5   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.125e+06 9.738e+04 -11.554 2.55e-13 \*\*\*  
## vposts\_civic$odometer 7.129e-03 3.931e-03 1.814 0.07857 .   
## vposts\_civic$conditionfair -8.880e+02 1.461e+03 -0.608 0.54744   
## vposts\_civic$conditiongood -7.865e+02 5.664e+02 -1.389 0.17395   
## vposts\_civic$conditionlike new 2.058e+03 6.988e+02 2.946 0.00578 \*\*   
## vposts\_civic$conditionnew 3.754e+03 7.929e+02 4.734 3.79e-05 \*\*\*  
## vposts\_civic$year 5.639e+02 4.857e+01 11.610 2.24e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1373 on 34 degrees of freedom  
## (63 observations deleted due to missingness)  
## Multiple R-squared: 0.9346, Adjusted R-squared: 0.9231   
## F-statistic: 80.99 on 6 and 34 DF, p-value: < 2.2e-16

#subsetting only for camry so I can fit a regression model  
camry = vposts$Model == 'camry'  
camry[is.na(camry)] = FALSE  
vposts\_camry = vposts[camry, ]  
model\_camry = lm(vposts\_camry$price ~ vposts\_camry$odometer + vposts\_camry$condition + vposts\_camry$year)  
summary(model\_camry)

##   
## Call:  
## lm(formula = vposts\_camry$price ~ vposts\_camry$odometer + vposts\_camry$condition +   
## vposts\_camry$year)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2094.6 -958.4 0.0 958.4 2141.6   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.162e+06 2.285e+05 -5.088 0.00142 \*\*  
## vposts\_camry$odometer -1.438e-03 1.236e-02 -0.116 0.91061   
## vposts\_camry$conditionfair 8.017e+02 1.412e+03 0.568 0.58785   
## vposts\_camry$conditiongood 1.543e+03 1.656e+03 0.932 0.38252   
## vposts\_camry$conditionlike new 2.061e+02 2.025e+03 0.102 0.92180   
## vposts\_camry$year 5.828e+02 1.139e+02 5.118 0.00137 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1768 on 7 degrees of freedom  
## (26 observations deleted due to missingness)  
## Multiple R-squared: 0.8123, Adjusted R-squared: 0.6782   
## F-statistic: 6.059 on 5 and 7 DF, p-value: 0.01756

#camry and civic by location to see if there are any effects on price by location  
#subsetting the rows for civic and camry  
regress\_rows = vposts$Model == 'civic' | vposts$Model == 'camry'  
regress\_rows[is.na(regress\_rows)] = FALSE  
vposts\_regress = vposts[regress\_rows, ]  
  
model\_location = lm(vposts\_regress$price ~ vposts\_regress$location)  
# I did not want to print this summary because it is more than two pages  
location\_summary = summary(model\_location)