Jiten Punjabi
U31893003

Web Scraping in Python

The aim of the project is build an effective and efficient web scrapper in Python, which will navigate through terse HTML files, using various packages in Python, to collect data. Web Scrapping is to be done on a website that specializes in selling and buying used boats (www.boattrader.com). After scrapping the website, the collected data is to be analyzed and explain the variation in prices of boats according to their various features.

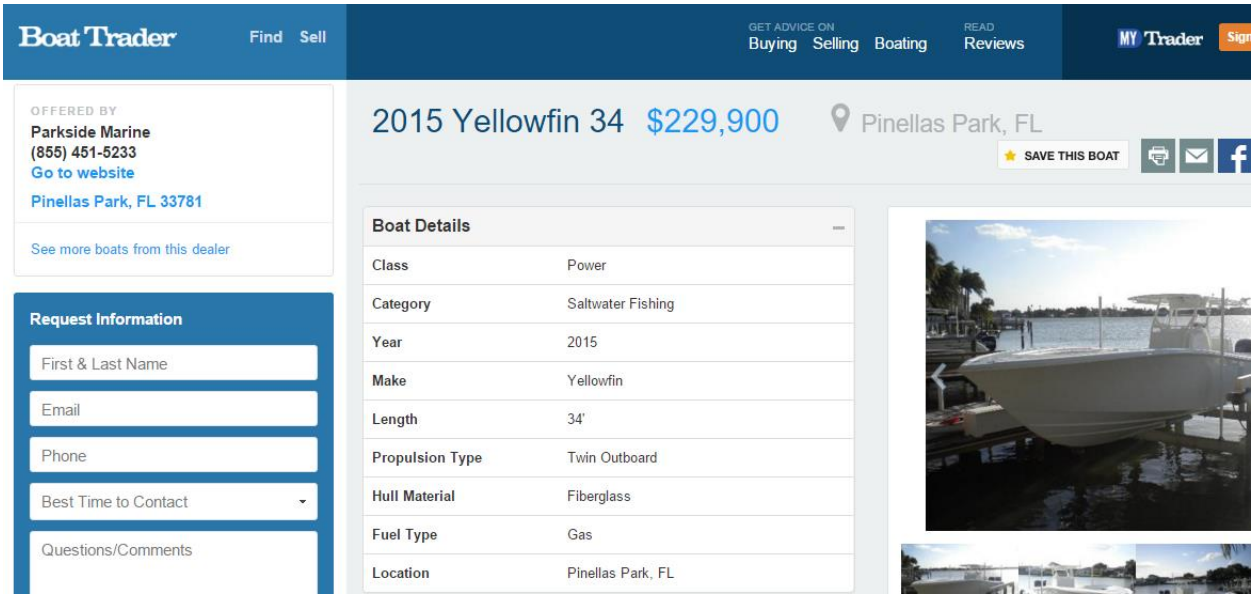
Pre-requisites

In order to start scraping the web and build an automatic web scrapper, I had to understand and learn HTML parsing techniques and Python programming. Team Leada modules set me on track to understanding the basics of both these concepts and Google helped me in learning more advanced concepts as I started coding.

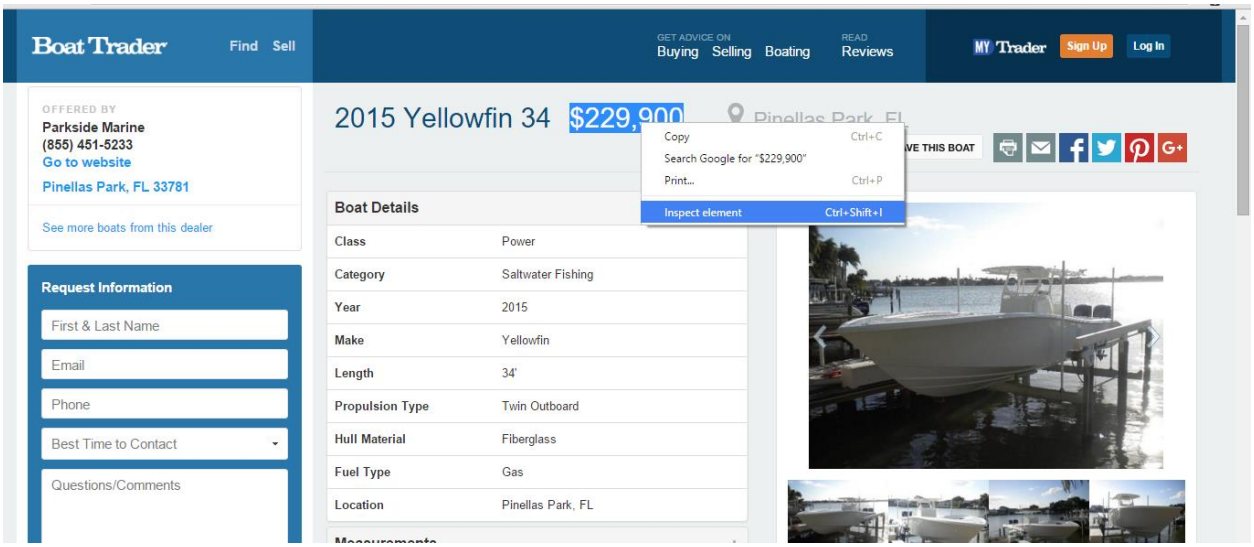
HTML parsing techniques made me familiar with selecting HTML elements. It gave me the basic understanding of HTML and element selection. I further used these techniques to inspect the Boat Trader website and also the individual ad pages. Parsing was to be done at the element level in order to fetch the relevant boat details of the advertisement. Team Leada's tutorial on Python programming laid the foundation on the basics of Python and object oriented programming. It also gave an introduction to web scraping basics such as requesting a web page HTML source, extracting links, crawling search results, and more.

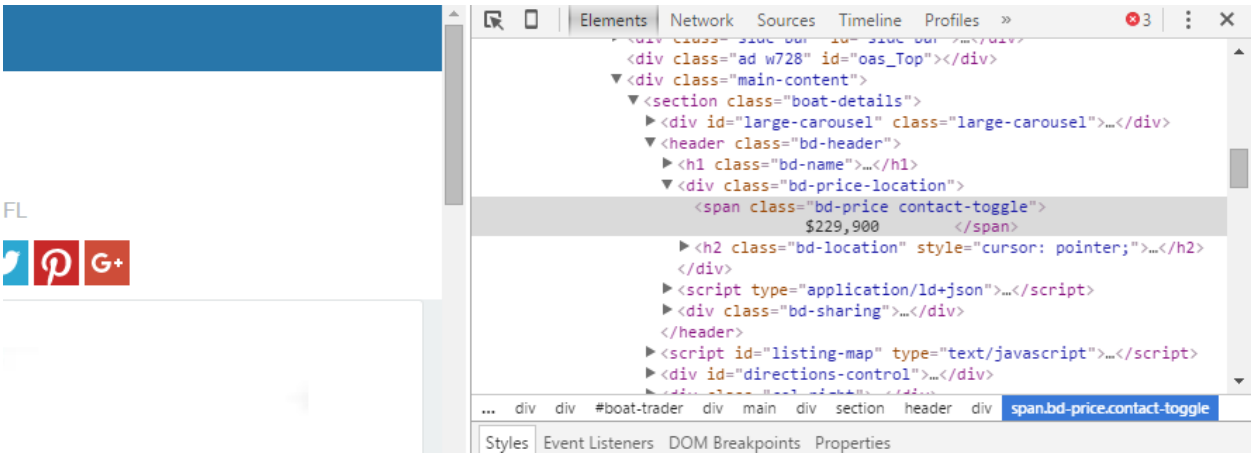
Inspecting the Webpage

First step towards web scraping was to inspect the Boat Trader website and fetch the links of all the individual ads of the boats listed on Boat Trader. Then, on each individual ad, inspect all the elements of interest and try to fetch those details about the boat. For example, the below screenshot shows the webpage of an individual ad along with the features or details we desire to fetch.



In order to fetch these details, we have to inspect each element by looking at the HTML source code. I inspected the code for each element and gathered the “class” or “tag” they belonged to, which would then be used in Python to automate the web scrapper. Below screenshot shows the element inspection of Price of a boat. Similarly, I have parsed for all the other details as well.





After clicking on inspect element, the source code behind that element is shown. I can use the "class" attribute of "span" tag to fetch the price of this particular code. Similarly, I followed this procedure to get data for all other boats as well.

Python Code

After I found out the "tags", "class" and "id" of various fields I was interested in, I could start coding the web scraper. Various packages were needed to scrap the web. Requests packages was used to make connection to the web, it is used for sending HTTP requests. It can fetch the raw HTML file from the web and using Python you can parse that HTML file. Another package that was used is BeautifulSoup. It is a library for pulling out HTML and XML files. It provides ways of navigating, searching and modifying the parse tree. And csv package was used to write the parsed data from the web on a csv spreadsheet, which can then be used to analyze the boat details. Below is the code I wrote to fetch the data from www.boattrader.com.

```
import requests
from bs4 import BeautifulSoup , SoupStrainer
import csv

WriteResultsFile = csv.writer(open("BoatDetails1001-1150.csv","w"))
WriteResultsFile.writerow(["Make","Model","Year","Price","Contact
Number","Class","Category","Length","Hull Material"])
x=1
y=0
get_links=[]
boat_details=[]
j=0
i=0
page=1001
#below while loop collects the individual ad links from all the pages of the website
in get_links list
while page<=1150:
    next = "http://www.boattrader.com/search-results/NewOrUsed-any/Type-any/Category-
all/Radius-4000/Sort-Length:DESC/Page-"+str(page)+",28?"
    r = requests.get(next)
```

```
    raw_html = r.text
    soup = BeautifulSoup(raw_html,'html.parser')
    search_link = soup.find_all('a',{'data-reporting-click-listing-type':'standard
listing'})
    #print next
    get_links.append(search_link)



    y=0
    while y<len(get_links[j]):

#the while loop takes links of all individual ads from the page link and scraps all
individual ads to fetch various boat details
#after scraping data from individual ads, it writes them into a CSV file
        link_next = "http:"+get_links[j][y]['href']
        r1 = requests.get(link_next)
        raw_html1 = r1.text
        soup = BeautifulSoup(raw_html1,'html.parser')
        search_make = soup.find_all('span',{'class':'bd-make'})
        search_model = soup.find_all('span',{'class':'bd-model'})
        search_year = soup.find_all('span',{'class':'bd-year'})
        search_price = soup.find_all('span',{'class':'bd-price contact-toggle'})
        search_contact = soup.find_all('div',{'class':'contact'})
        search_class = soup.find('div',{'id':'boat-details'})
        #print search_make[0].text
        #print search_model[0].text
        #print search_year[0].text
        #print search_price[0].text
        #print search_contact[0].text


        try:
            boat_details=[]
            for row in search_class.find_all('tr'):

                for cell in row.find_all('td'):
                    boat_details.append(cell)


WriteResultsFile.writerow([search_make[0].text,search_model[0].text,search_year[0].tex
t,(search_price[0].text).strip(),search_contact[0].text
                                ,boat_details[0].text,
boat_details[1].text,boat_details[4].text,boat_details[6].text])

        except AttributeError:
            print 'Unable to parse';

        print page,y;

        y=y+1
    page=page+1
    j=j+1
```

Jiten Punjabi
U31893003

The above code first loads all the libraries needed, then initializes the variables and creates a csv spreadsheet and writes the headers with all the variables or details of the boat. A while loop is set for iteration on various webpages, and each webpage has around 30 individual ads. So, first from each page the links to all the individual ads is fetched. I have created another while loop to parse through those individual ads. An object of BeautifulSoup is created so that it parses through all the “tags” and “class” that I found through inspection of the source code. Once all the data is collected, it is written row by row using the csv functions to build the csv spreadsheet. The above code parses around 150 webpages. Boat Trader has totally 4916 webpages listing all the boat ads. However, all these webpages couldn’t be parsed at once, since the programs always returned a Memory Error. The largest number of pages it could scrap without crashing because of Memory Error was 150-170, hence I collected the data in chunks of 150 webpages. It took around 2 hours for a single program to run, for scrapping just 150 pages.

Below is the screenshot of how the data looked like after the program ran successfully.

A	B	C	D	E	F	G	H	I
Make	Model	Year	Price	Contact Number	Class	Category	Length	Hull Material
Custom	Double Hull Crude	2008	Request a Price	(866) 461-6540	Power	Commercial Boat	820'	Steel
Custom	Tanker	1995	Request a Price	(866) 461-6540	Power	Commercial Boat	748'	Steel
Tanker	LPG Tanker	2003	Request a Price	(866) 461-6540	Power	Commercial Boat	739'	Steel
CARGO VESSEL	Cellular Containe	1999	Request a Price	(866) 461-6540	Power	Commercial Boat	683'	Steel
Bulkcarrier	Geared Bulkcarri	1997	Request a Price	(866) 461-6540	Power	Commercial Boat	622'	Steel
Bulkcarrier	General Cargo Ve	1997	Request a Price	(866) 461-6540	Power	Commercial Boat	622'	Steel
Tanker	Product Tanker	2002	Request a Price	(866) 461-6540	Power	Commercial Boat	620'	Steel

Jiten Punjabi
U31893003

Data Consolidation, Data Preparation, Data Cleaning for Analysis

Since I ran the code in chunks of 150 pages at a time, there was a lot of data to be consolidated from many spreadsheets. After consolidation of data, I checked for various discrepancies in the collected data. Many duplicates were found in the data, one of the reasons for that could be, people post multiple ads of their boat, hoping for a quick sale. I then got rid of all the blanks involving in any row of the data set. All of this was done using Microsoft Excel.

Data formatting changes were also involved in this consolidated data set file. For instance, in the Length of the boat field, the length that was fetched from the webpages was in the format eg. 51' , 252' and so on. All such values were converted to numeric values. Another case in point is Price of the boat. Price fetched for eg. $15,250. I converted such currency values to numbers for eg. 15250.00 using Microsoft Excel. After all the data consolidation, preparation and cleaning, 18,159 rows of data was collected.

Below screenshot shows how the final consolidated csv spreadsheet looks like.

	A	B	C	D	E	F	G	H	I	J
1	Make	Model	Year	Price	Contact N	Class	Category	Length	Hull Material	
2	ISHIKAWA	Cruise Shi	1990	35000000	(866) 469-	Power	Cruise Sh	579	Steel	
3	Cruise Shi	Stock No.	1992	33333	1.52E+10	Power	Commerc	538	Steel	
4	RIO	Test Ad	2015	27999	(757) 351-	Power	Bass Boat	456	Aluminum	
5	Ferry Car I	344	2010	2700000	(877) 564-	Power	Cargo Shi	344	Steel	
6	Chaparral	28 ssx	2014	615000	(866) 347-	Power	Cruisers,	287	Fiberglass	
7	Custom	Tanker	1981	750000	(866) 461-	Power	Commerc	271	Steel	
8	Pavati	AL-24	2015	5950000	(855) 405-	Power	Ski and W	248	Aluminum	
9	Cooks & V	Caledonia	1947	649000	(216) 391-	Sails	Schooner	245	Fiberglass	
10	Sea Hunt	BX	2014	6184540		Power	Center Co	244	FIBERGLASS/COMPO	
11	Custom Cr	239 Nil Sh	2007	4590000	(866) 469-	Power	Cruise Sh	239	Steel	
12	Shadow M	Custom Ex	2007	319000	(888) 573-	Power	Mega Yac	220	Steel	
13	Astro	18 foot Fis	1996	3899000		Small Boa	Tenders/S	216	FIBERGLASS/COMPO	
14	Blount	Luxury Dir	1987	525000	(877) 430-	Power	Commerc	197	Steel	
15	Amels Shi	Expeditior	1972	8990000	(866) 694-	Power	Motor Yac	196	Steel	
16	Supply	General Su	1973	650000	(866) 461-	Power	Commerc	195	Steel	
17	Release	Center Co	2015	379000	(888) 430-	Power	Center Co	194	Composite	
18	Sportsmar	19 Island F	2016	1389000	(855) 751-	Power	Bay Boats	191	Fiberglass	
19	Sportsmar	19 Island F	2016	12900000	(888) 597-	Power	Bay Boats	191	Fiberglass	
20	Mastercra	SPORTSTA	2000	600000	(888) 627-	Power	Ski and W	190	Composite	
21	Pullman	MY	1943	275000	(888) 319-	Power	Commerc	185	Steel	
22	Wesmac	Warsan 18	2008	499999	(866) 469-	Power	Mega Yac	184	Fiberglass	
23	Yamaha M	SX190	2016	950000	(866) 836-	Power	Bowrider	180	Fiberglass	
24	Halter Ma	Geophysic	1979	5450000	(866) 469-	Power	Commerc	180	Steel	
25	Supply	Supply Ve	1977	310000	(866) 461-	Power	Commerc	180	Steel	
26	Amels	Tri-Deck	1998	2000000	(866) 836-	Power	Mega Yac	178	Steel	

Jiten Punjabi
U31893003

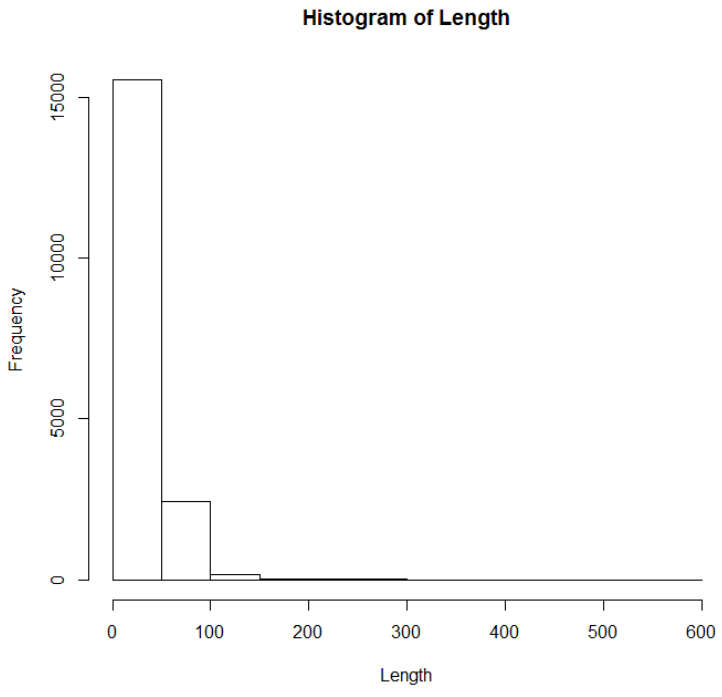
Data Visualization

First I loaded the data from the clean csv spreadsheet that I prepared for analysis. I did basic checks on the data.

```
> data <-read.csv("Boat_new.csv", header = TRUE,stringsAsFactors = FALSE )
> attach(data)
> dim(data)
[1] 18159     9
> head(data)
                        Make           Model Year    Price Contact.Number Class
1                  ISHIKAWAJIMA     Cruise Ship 1990 35000000 (866) 469-9640 Power
2 Cruise Ship, 800 Passenger - Stock No. S2115 1992    33333    15166475129 Power
3                          RIO         Test Ad 2015    27999 (757) 351-7111 Power
4          Ferry Car Passenger             344 2010  2700000 (877) 564-9702 Power
5                    Chaparral          28 ssx 2014   615000 (866) 347-9401 Power
6                       Custom          Tanker 1981   750000 (866) 461-6540 Power
                     Category Length Hull.Material
1  Cruise Ship, Commercial Boats    579         Steel
2  Commercial Boats, Cruise Ship    538         Steel
3                    Bass Boats    456      Aluminum
4             Cargo Ship, Other    344         Steel
5               Cruisers, Other    287    Fiberglass
6              Commercial Boats    271         Steel
> |
```

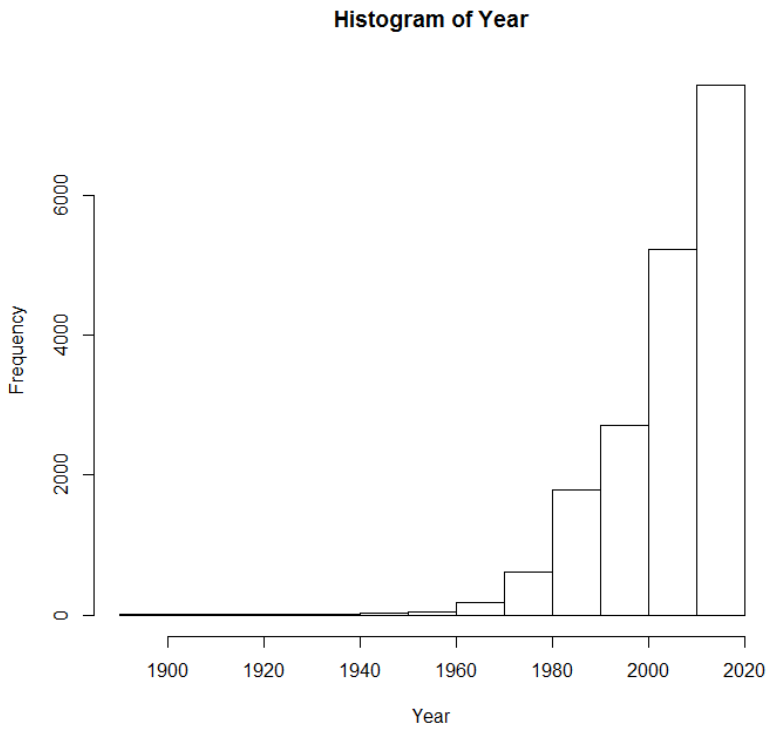
Now, performing some basic visualizations to understand the behavior and trend of data.

First Plotting the histogram of Length of the Boat

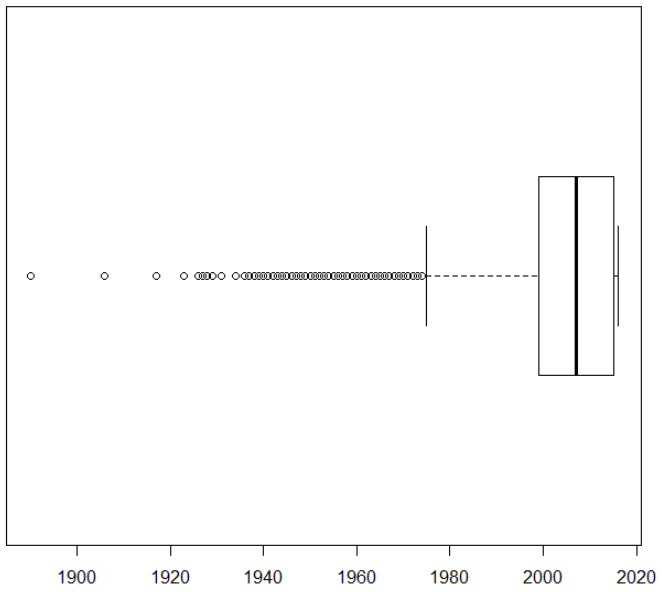


Jiten Punjabi
U31893003

Histogram of Year of Make of the Boat

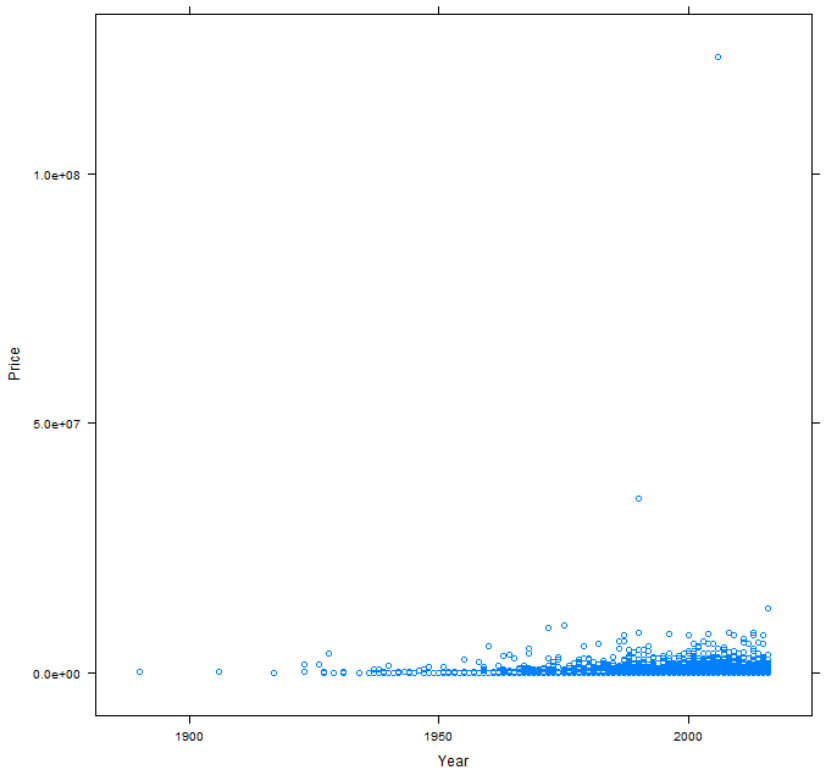


Boxplot of Year of Make of the Boat

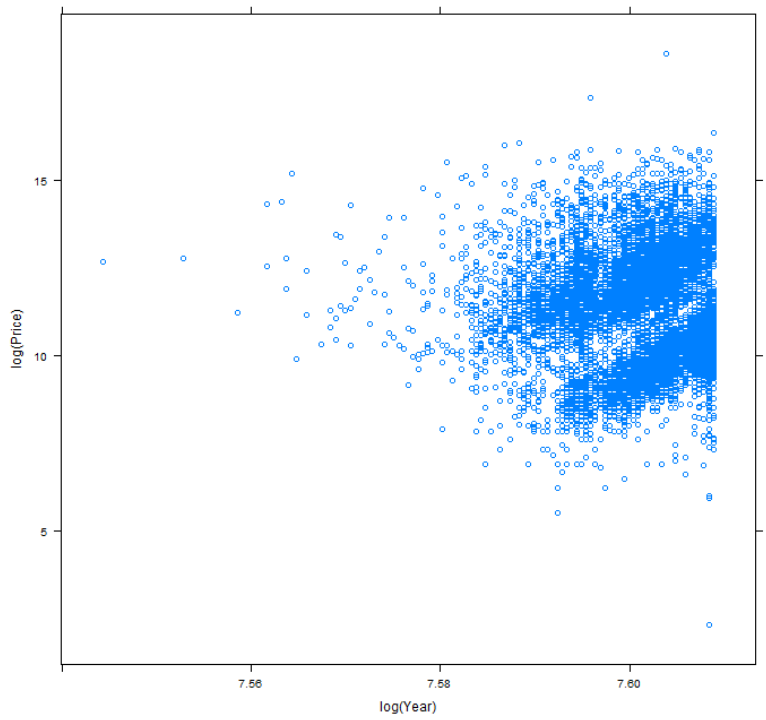


Jiten Punjabi
U31893003

Scatterplot – Price v/s Year

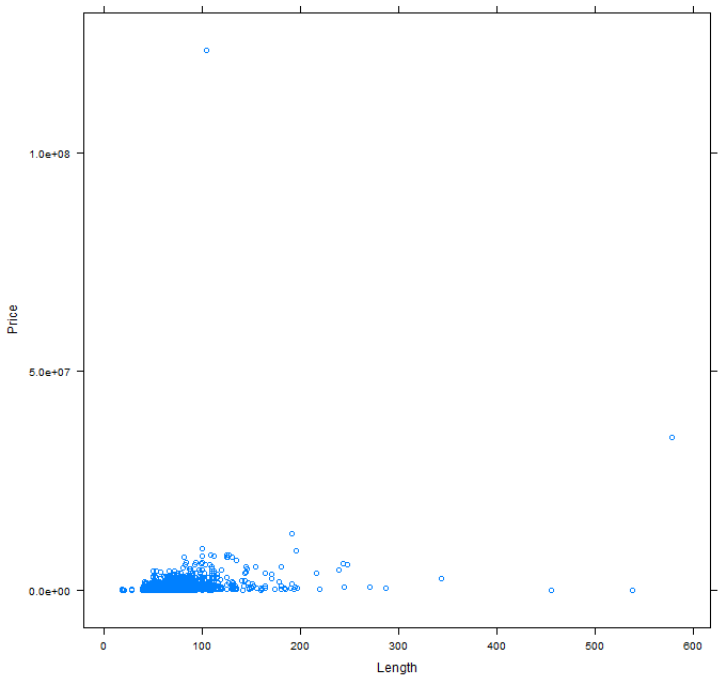


Now, taking log on both sides, to see a better trend. Log(Price) v/s Log(Year)

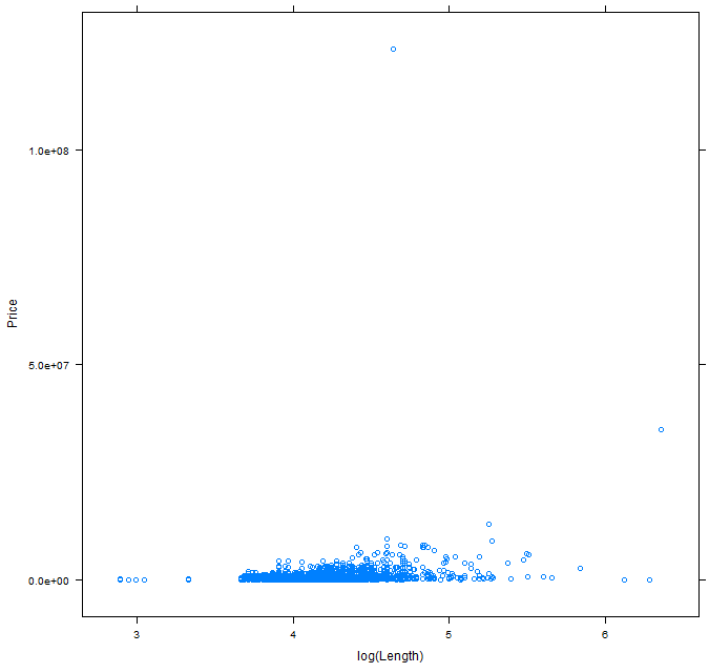


Jiten Punjabi
U31893003

Scatterplot – Price v/s Length



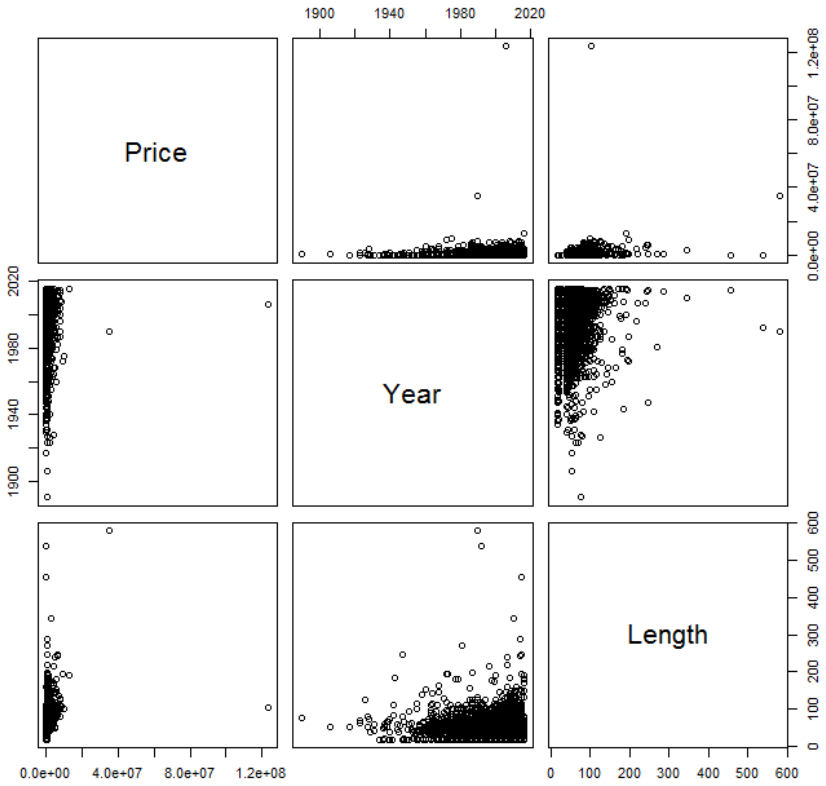
Now taking log on Length, we see the trend. Price v/s Log(Length)



Jiten Punjabi
U31893003

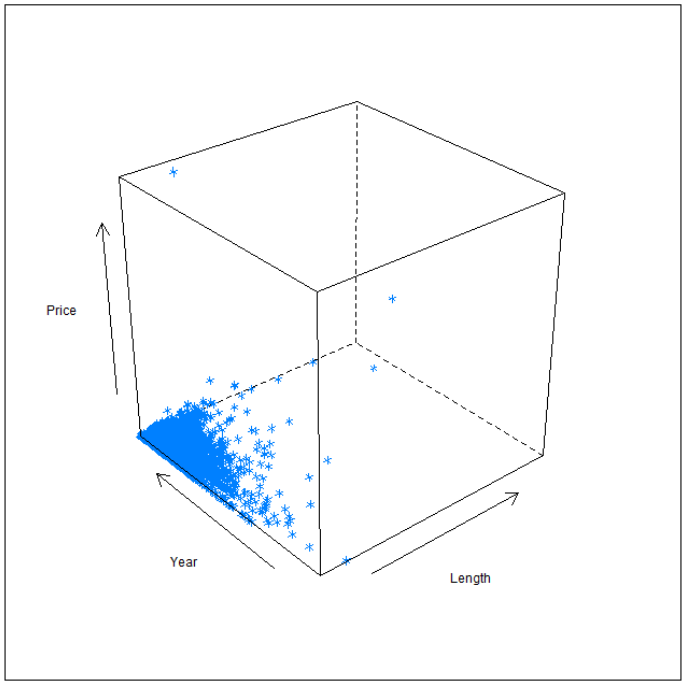
Scatterplot Matrices

Pairs(Price~Year+Length)



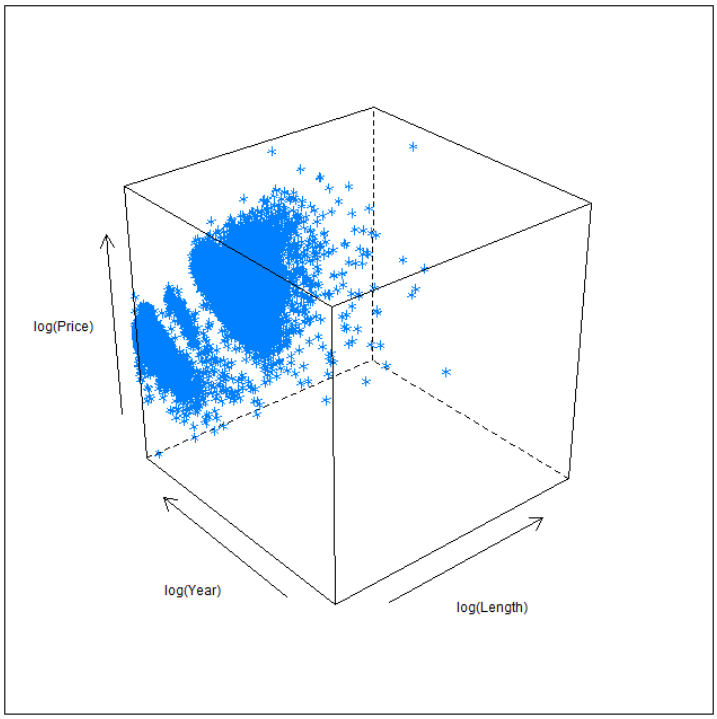
3-D Scatterplot

cloud(Price~Length+Year)



We get a better 3-D visualization, if we take log on both the sides. We get,

cloud(log(Price)~log(Length)+log(Year))



Data Analysis

Running regression models first,

Building the first regression model, I used the response variable as Price, and two predictor variables Length and Year. Since these two variables are highly correlated to the Price of the boat.

Model 1

reg1<-lm(Price~Length+Year)

summary(reg1)

Result:

```
Call:
lm(formula = Price ~ Length + Year)

Residuals:
      Min        1Q    Median        3Q       Max
 -8544988    -90874     10207     48870 122075447

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.341e+07  1.332e+06 -10.066   <2e-16 ***
Length       1.679e+04  3.746e+02  44.826   <2e-16 ***
Year         6.503e+03  6.624e+02   9.818   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1004000 on 18156 degrees of freedom
Multiple R-squared:  0.1019,    Adjusted R-squared:  0.1018
F-statistic:  1030 on 2 and 18156 DF,  p-value: < 2.2e-16
```

We can see that from this model around 10% of the variability in Price can be explained. We clearly need a better model that can capture more variation in Price.

Now, taking log on the Predictor variable Year, and see if it captures more variability.

Model 2

reg2<-lm(Price~Length+log(Year))

summary(reg2)

Result:

```
Call:
lm(formula = Price ~ Length + log(Year))

Residuals:
      Min        1Q    Median        3Q       Max
 -8542750    -90808     10488     48785 122075667

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -9.883e+07  1.005e+07  -9.834   <2e-16 ***
Length       1.679e+04  3.745e+02  44.825   <2e-16 ***
log(Year)    1.295e+07  1.321e+06   9.801   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1004000 on 18156 degrees of freedom
Multiple R-squared:  0.1019,    Adjusted R-squared:  0.1018
F-statistic:  1030 on 2 and 18156 DF,  p-value: < 2.2e-16

> |
```

The model however does not improve in the above step. Now, transforming the predictor variable Length.

Model 3 :

reg3<-lm(Price~log(Length)+log(Year))

summary(reg3)

Result:

```
Call:
lm(formula = Price ~ log(Length) + log(Year))

Residuals:
      Min        1Q    Median        3Q       Max
 -1940320   -150787      8750     53103 122467483

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -112392641   10716660  -10.49   <2e-16 ***
log(Length)     618328      17051   36.26   <2e-16 ***
log(Year)     14533361    1406068   10.34   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1022000 on 18156 degrees of freedom
Multiple R-squared:  0.06986,   Adjusted R-squared:  0.06976
F-statistic: 681.8 on 2 and 18156 DF,  p-value: < 2.2e-16
```

The above model returns the R-squared values as around 70%. Transforming the Length variable improves the model greatly. It can now explain the price variability of around 70%. Now, lets try to improve the model further and transform the Price to log as well.

Model 4:

reg4<-lm(log(Price)~log(Length)+log(Year))

summary(reg4)

Result :

```
Call:
lm(formula = log(Price) ~ log(Length) + log(Year))

Residuals:
    Min      1Q  Median      3Q     Max
-8.2174 -0.3176  0.0484  0.3682  4.3397

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -455.09391    7.51837  -60.53   <2e-16 ***
log(Length)    2.63614    0.01196  220.38   <2e-16 ***
log(Year)     60.11947    0.98644   60.95   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7169 on 18156 degrees of freedom
Multiple R-squared:  0.7357,    Adjusted R-squared:  0.7357
F-statistic: 2.527e+04 on 2 and 18156 DF,  p-value: < 2.2e-16
```

There is a great improvement in the model by taking log on Price variable. We have achieved 73.57% R squared. This model captures a good amount of uncertainty of Price.

Now that we have a relationship between Price and two predictor variables, we can use other categorical variables collected to try to improve the model.

Now taking Hull Material of the boat as one of the predictor variable and check the results. Since Hull material is a categorical variable, it will create Dummy variables for the number of types of Hull material.

Model 5:

reg5 <- lm(log(Price)~log(Length)+log(Year)+Hull.Material)

summary(reg5)

Result :

```
Call:
lm(formula = log(Price) ~ log(Length) + log(Year) + Hull.Material)

Residuals:
    Min      1Q  Median      3Q     Max
-8.1713 -0.3003  0.0283  0.3369  4.1317

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                   -519.48559    7.91926 -65.598  < 2e-16 ***
log(Length)                      2.62697    0.01197 219.382  < 2e-16 ***
log(Year)                       68.55972    1.03910  65.980  < 2e-16 ***
Hull.MaterialALUMINUM           -0.17654    0.08499  -2.077 0.037793 *
Hull.MaterialComposite           0.46341    0.03157  14.680  < 2e-16 ***
Hull.MaterialFERRO-CEMENT       -2.37738    0.49405  -4.812 1.51e-06 ***
Hull.MaterialFerro cement        0.50147    0.31287   1.603 0.108997
Hull.MaterialFiberglass          0.33356    0.01318  25.317  < 2e-16 ***
Hull.MaterialFiberglass Reinforced  0.56483  0.31265   1.807 0.070842 .
Hull.MaterialFIBERGLASS REINFORCED  0.16907  0.03726   4.537 5.74e-06 ***
Hull.MaterialFIBERGLASS/COMPOSITE   0.21273  0.04136   5.144 2.72e-07 ***
Hull.MaterialHypalon             0.55658    0.18068   3.080 0.002070 **
Hull.MaterialINFLATABLE          0.19898    0.69868   0.285 0.775803
Hull.MaterialOther               0.10707    0.03034   3.529 0.000419 ***
Hull.MaterialOTHER/NA            0.22760    0.28538   0.798 0.425153
Hull.MaterialPvc                -0.54859    0.69860  -0.785 0.432305
Hull.MaterialRIGID INFLATABLE    0.27831    0.69861   0.398 0.690354
Hull.MaterialRoplene             0.17569    0.28538   0.616 0.538156
Hull.MaterialSteel               0.11274    0.04709   2.394 0.016661 *
Hull.MaterialSTEEL               0.71274    0.24746   2.880 0.003979 **
Hull.MaterialWood                1.07416    0.05834  18.411  < 2e-16 ***
Hull.MaterialWOOD                0.93628    0.26514   3.531 0.000415 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6985 on 18137 degrees of freedom
Multiple R-squared:  0.7494,    Adjusted R-squared:  0.7491
F-statistic:  2583 on 21 and 18137 DF,  p-value: < 2.2e-16
```

.

With this model, we achieve R-squared value as 75%. It is better than the previous model, however it doesn’t change the model drastically. It improves the variability by 2%, and adding a large number of dummy variables.

We can also use the make of the boat as one of the predictors for the model, although, even Make variable would have a large number of categorical factors, so would add many dummy variables to the regression. I tired the regression model with Make as one of the factors and see how it changes the results.

Model 6:

model <- lm(log(Price)~log(Length)+log(Year)+Make)

summary(model)

Result :

```
MakeYoung Brothers                          1.782e-01  8.254e-01   0.216 0.829040
MakeYoung Sun                               1.039e-01  7.149e-01   0.145 0.884471
MakeYOUNGBLOOD                              4.814e-01  7.153e-01   0.673 0.500958
MakeYoungquist                             -6.106e-01  8.257e-01  -0.739 0.459618
MakeYPL                                    -3.284e-01  8.254e-01  -0.398 0.690714
MakeYuka                                    6.991e-01  8.257e-01   0.847 0.397181
MakeZar Formenti                           -1.659e-01  7.152e-01  -0.232 0.816541
MakeZiegler Shipyards                       1.630e+00  8.262e-01   1.973 0.048535 *
MakeZodiac                                 -2.873e-01  6.308e-01  -0.455 0.648777
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5836 on 16343 degrees of freedom
Multiple R-squared:  0.8424,    Adjusted R-squared:  0.8249
F-statistic: 48.12 on 1815 and 16343 DF,  p-value: < 2.2e-16
```

This model created a long list of dummy variables and returned a R squared values as 85%.

This model captures the uncertainty of to a great level, however it is a very complex model to implement, with so many dummy variables it becomes a tedious task to interpret and predict the results.

Non-parametric Model

Model :

library(mgcv)

non<-gam(Price~s(Length)+s(Year))

summary(non)

Result :

Jiten Punjabi
U31893003

```
> summary(non)

Family: gaussian
Link function: identity

Formula:
Price ~ s(Length) + s(Year)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   174499       7278   23.98   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
            edf Ref.df      F  p-value
s(Length) 8.978  9.000 331.12  < 2e-16 ***
s(Year)   1.611  2.029  16.22 8.21e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.143   Deviance explained = 14.4%
GCV = 9.6237e+11  Scale est. = 9.6175e+11  n = 18159
>
```

The non parametric model captures only 14.3% uncertainty in the price variation. Using a non parametric model for this data is not suggestable. Using the linear regression model gives us the best results as seen in above cases.