Danielle Heymann

Boston Marathon Linear Regression Project

CSCI 688

Executive Summary

Each year, on Patriot's Day, thousands of runners assemble at the start of the Boston Marathon and anticipate the sound of the gunfire, indicating the beginning of a trying 26.2 miles. The oldest annually held marathon, the Boston Marathon is a unique race in that competitors must qualify to earn their spot in the race (with the exception of a fraction of runners who did not qualify but bought entry through charity donations or other organizations). This makes the field ultra-competitive. However, the exclusiveness of the Boston Marathon lends itself to interesting results amongst participants: some come to set a PR (personal record), while others are content with merely qualifying and then "underperform." It will be interesting to see where the breakeven point is between these categories. I will address the following questions in this study: 1. How can finish time be modeled given a set of predictor variables? Are there trends between groupings, called waves, of runners that side models can reveal?

In this study, I used historic race results from the 2017 Boston Marathon and created a model to predict an individual's finish time based on a number of predictor variables including: age, first 5K pace (minutes per kilometer) of the race, first half marathon pace (minutes per kilometer) of the race, gender, and wave group (Elite, one, two, three, four). The response variable is the official time, which is the time that each individual completes the marathon in (net time, not gun time). I also created additional small models for each wave group to analyze trends and topics of interest. I used forward selection stepwise regression after comparing the results of similar methods to accomplish this.

The regression equation for my general (all waves) model is:

$$Y = 24.1 + -0.0609X_1 – 22.314X_2 + 66.460X_3 – 6.829X_4 – 7.996X_6 – 5.332X_7 – 2.543X_8$$

I also made 5 smaller models to examine performance factors within each wave group and came up with the theory that most runners who are not elite run their first 5K of the race too quickly, which could give insight to improve their pacing strategy. I also found that as pace increases among runners, women are tending to run too quickly in the initial phase of the race. Moreover, as we analyze the wave groups, the slower groups are running the second half of their marathon slower relative to the first half of their marathon, which would correspond to a "positive split." As we analyze faster wave groups like the Elite group and Wave one, there is an increasingly greater amount of "negative splits" among runners, which would correspond to a faster second half of the race. This strategy is more favorable as it allows a runner to economically expend energy. This study will highlight the process and methods taken along with visualizations that reveal these findings.

Abstract & Introduction

Predictor Variables:

X1: Age (integer)

X2: First 5K Pace (minutes per kilometer)

X3: First Half Marathon Pace (minutes per kilometer)

X4: Gender (Indicator Variable: 1 for Females; 0 for males)

X5: Wave: Elite (Indicator Variable: 1 for Elite Wave member; 0 otherwise)

X6: Wave: One (Indicator Variable: 1 for Wave One member; 0 otherwise)

X7: Wave: Two (Indicator Variable: 1 for Wave Two member; 0 otherwise)

X8: Wave: Three (Indicator Variable: 1 for Wave Three member; 0 otherwise)

Response Variable:

Y: Official Time (minutes)

*Wave Four Members have values of 0 for indicator variables X5-X8 and they are considered the "base". An unspecified amount of wave four runners are charity runners, which indicates that they did not qualify for the race but raised money which granted them entry. There is not enough data to determine which runners are charity runners.

X1, Age

With different ages comes different levels of experience. The more mature runners have most likely had experience on the course or have run multiple marathons. This would allow them to appropriately pace and not fall under the pressures of starting off too quickly.

X2, First 5k Pace

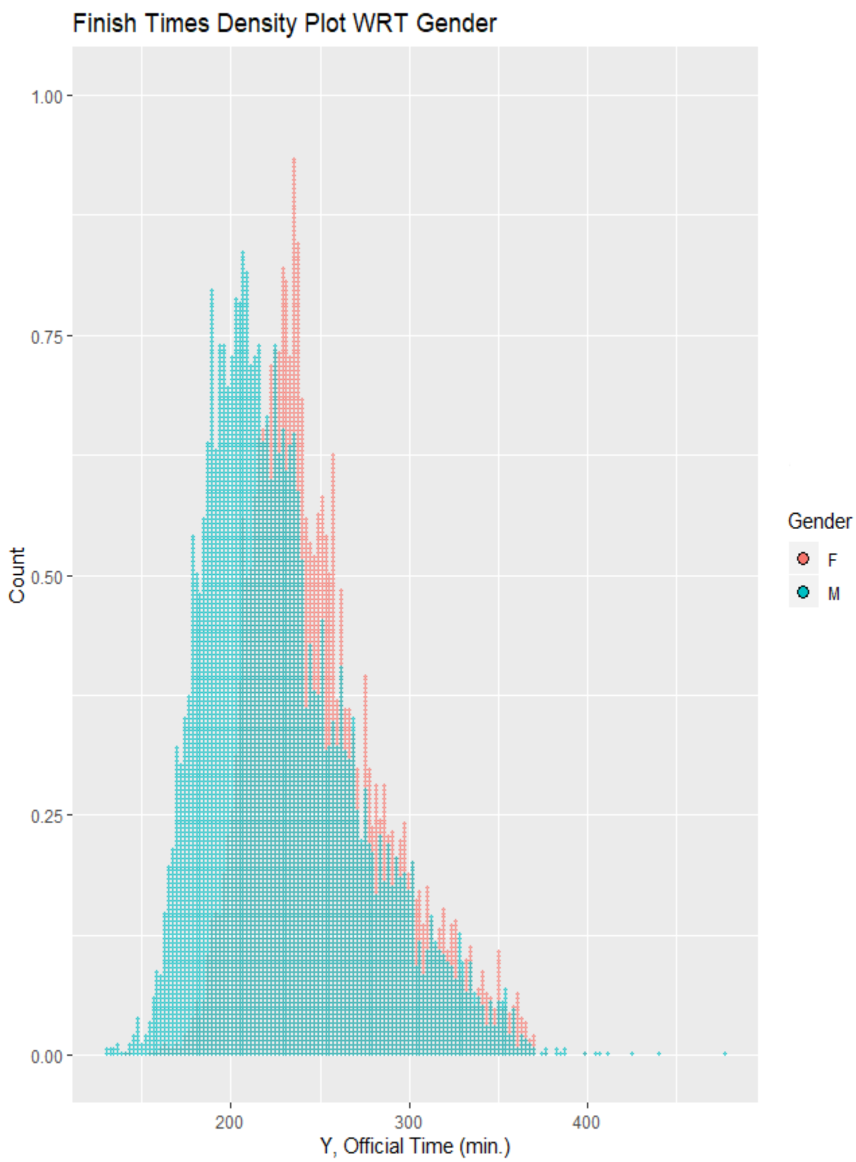
This predictor variable along with qualifying time will tell a lot about a runner's performance. It is highly unlikely that the runner's first 5k pace would be faster than the average 5k pace of their overall qualifying time.

X3, First Half Marathon Pace

One of the most interesting predictor variables in the model, the half marathon pace will give insight to the “negative split” factor of the runner’s time. It is generally accepted that the runners who run the second half of the marathon faster than the first half of the marathon do better with respect to their peers. This is called a “negative split.” A negative split is favorable and indicates an economic racing strategy, where the runner is less likely to “hit the wall” or burnout due to wasting too much energy initially.

X4, Gender

Gender is an important qualitative variable in the model because male and female athletes take different training approaches, and males are overall faster than females. This study reveals differences in pacing consistency between the genders, so it is an important valuable predictor variable.



It is also interesting to look at the distribution of females and males in the race in the density graph on the right. This graph is standardized, so it is easy to see that the mean finish time for women is slightly slower than the mean finish time for men. It is also important to note that there are more males included in the study than there are females. The dataset that I used includes 11972 females and 14438 males, which would be a male/female ratio of about 5:6.

A note on Qualifying Time-

The Boston Athletic Association determines qualifying time for each gender-age group division in order to limit the field and ensure that the race will be competitive. This determines the bib assignment, which then determines the wave assignment. The table below displays the assignment criteria. Waves are determined by bib number which is determined by qualifying time cutoff zones. This means that the runners first achieve entry to the race if they meet qualifying criteria for their age group (Qualifying Time Table), and then, they are placed into waves based on the best marathon time they achieved that granted them entry to the Boston Marathon (which is referred to as a "BQ" for Boston-Qualify). The "Qualifying Time Interval" was determined by the mathematicians at the Boston Athletics Association in order to create optimal wave size to allow for maximum space on the road while promoting similar paces amongst runners in the wave. Furthermore, while this study does not incorporate it, each wave is broken up into separate corrals for further groupings of runners according to their BQ times. This strategy promotes safety on the roads and a collaborative initiative amongst runners in the corrals.

Wave Assignment

Bib Range	**"BQ" Time Interval**	**Wave**
0-100	≤ 2:25:21	Elite
101-7,700	2:25:21 - 3:10:43	1
8,000-15,600	3:10:44 - 3:29:27	2
16,000-23,600	3:29:28 - 3:57:18	3
24,000-32,500	≥ 3:57:19	4

X5, X6, X7, and X8 correspond to indicator variables based on placement in the specified wave group (elite, one, two three). A value of "1" denotes that the runner is part of the specified wave group. Wave four is the base case and does not require an indicator variable as all other wave indicator variables will take on value "0".

Qualifying Times

Age Group	**MEN**	**WOMEN**
18-34	3:00	3:30
35-39	3:05	3:35
40-44	3:10	3:40
45-49	3:20	3:50
50-54	3:25	3:55
55-59	3:35	4:05
60-64	3:50	4:20
65-69	4:05	4:35
70-74	4:20	4:50
75-79	4:35	5:05
>= 80	4:50	5:20

INITIAL DATA CLEANING PROCESS

I used Python for most of my project, in addition to R and Minitab and have included the files, but I will describe here the process that I used to clean the data for further clarity:

- Elite Bib numbers below 101 were assigned to "elite" wave. There were repeat Bib numbers as Male competitors in the elite category had assignments XX while Female competitors in the elite category had assignments XXF. Therefore, I immediately grouped them all in the elite category to reduce any misunderstandings

-Assigned wave/bib color based on research and memo from Boston Athletic Association

-After Bib number to wave assignment was complete, I no longer required the Bib number variable and dropped it from any further analysis

-Dropped any row with an NA value for any category as it would be an incomplete data point

-Split the data into training and test data. Method for accomplishing this: I first divided the data on category of gender. This left me with a dataframe for female runners and male runners. I shuffled the entries and then divided each subset into two additional subsets. I stay consistent with the seed used by the random algorithm to shuffle my dataframes so that each time I run my program, I will be using the same two final sets for training and validation. At that point, I had 4 subsets of shuffled entries split between genders (Females A, Females B, Males A, Males B). I then concatenated Females A with Males A and concatenated Females B with Males B. This left me with 2 dataframes. Finally, I sorted based on Official Time so that I have 2 complete datasets for training and validation: A and B.

-The last step in my data cleaning was to reformat numeric information held in string data type to a numeric data type for analytical purposes.

-I repeated this process after dividing the dataset into 5 subsets based on wave group so that I could have training and validation datasets, A and B, for my side study (wave performance analysis).

-Finally, I exported these twelve final sets of data to CSVs in order to use Minitab in the later part of my model development.

DATA STATISTICAL SUMMARY

Descriptive Statistics:

	Y, Official Time	X1, Age	X2, 5K Pace	X3, Half Pace	X4, Gender	X5, Wave: Elite	X6, Wave: One	X7, Wave: Two	X8, Wave: Three
count	13183.000000	13183.000000	13183.000000	13183.000000	13183.000000	13183.000000	13183.000000	13183.000000	13183.000000
mean	237.890005	42.515133	5.109776	5.235105	0.453159	0.003110	0.249336	0.250626	0.252370
std	42.427865	11.364216	0.801240	0.870168	0.497820	0.055683	0.432645	0.433390	0.434389
min	129.620000	18.000000	3.080000	3.061030	0.000000	0.000000	0.000000	0.000000	0.000000
25%	207.800000	34.000000	4.530000	4.622350	0.000000	0.000000	0.000000	0.000000	0.000000
50%	231.430000	43.000000	5.010000	5.105820	0.000000	0.000000	0.000000	0.000000	0.000000
75%	261.780000	51.000000	5.556000	5.684090	1.000000	0.000000	0.000000	1.000000	1.000000
max	478.230000	80.000000	10.846000	9.598300	1.000000	1.000000	1.000000	1.000000	1.000000

It is important to note that the mean time in the 2017 Boston Marathon was 3 hours and 58 minutes, which is above the median qualifying time (refer to qualifying time table in Abstract). This can be attributed to the challenging elevation profile of the course. It is common for runners to choose a "fast and flat" race in order to accomplish a "BQ" time, which has a more favorable elevation profile than the Boston Marathon course.

Normality of Residuals:

I fit an Ordinary Least Squares (OLS) Regression Model as a starting point and to accomplish residual analysis, identification of outliers, creation of correlation matrix and plots, insight of potential transforms, and creation of additional plots for exploratory purposes.

On the right is the initial OLS Model output, which is a benchmark and starting point to the model which I develop. Below is residual plot.

OLS Regression Results

Dep. Variable:	Y, Official Time	**R-squared:**	0.907
Model:	OLS	**Adj. R-squared:**	0.907
Method:	Least Squares	**F-statistic:**	1.615e+04
Date:	Sat, 07 Dec 2019	**Prob (F-statistic):**	0.00
Time:	18:06:34	**Log-Likelihood:**	-52424.
No. Observations:	13183	**AIC:**	1.049e+05
Df Residuals:	13174	**BIC:**	1.049e+05
Df Model:	8		
Covariance Type:	nonrobust		

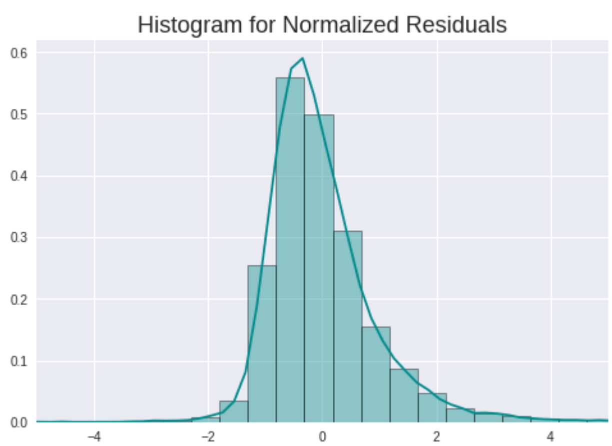
	coef	std err	t	P>|t|	[0.025	0.975]
const	23.7932	1.609	14.787	0.000	20.639	26.947
X1, Age	-0.0511	0.012	-4.257	0.000	-0.075	-0.028
X2, 5K Pace	-17.8734	0.510	-35.029	0.000	-18.874	-16.873
X3, Half Pace	60.1848	0.434	138.758	0.000	59.335	61.035
X4, Gender	-7.0166	0.297	-23.640	0.000	-7.598	-6.435
X5, Wave: Elite	-13.7280	2.147	-6.395	0.000	-17.936	-9.520
X6, Wave: One	-9.0043	0.574	-15.696	0.000	-10.129	-7.880
X7, Wave: Two	-5.4202	0.454	-11.952	0.000	-6.309	-4.531
X8, Wave: Three	-2.5755	0.409	-6.298	0.000	-3.377	-1.774

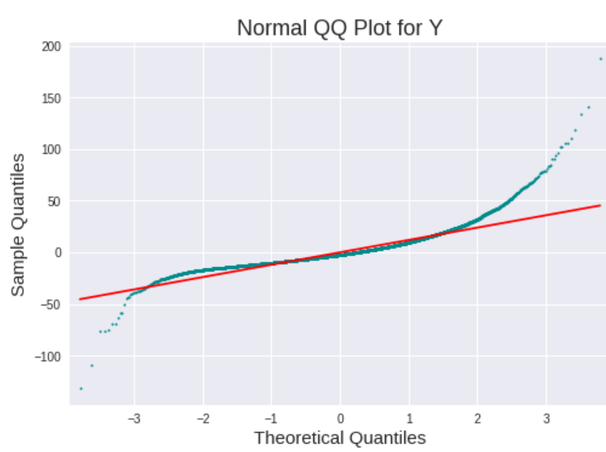
Omnibus:	6383.341	**Durbin-Watson:**	1.758
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	109854.082
Skew:	1.904	**Prob(JB):**	0.00
Kurtosis:	16.619	**Cond. No.**	915.

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Regression	8	21533226	2691653	16147.44	0.000
X1, Age	1	3020	3020	18.12	0.000
X2, 5K Pace	1	204531	204531	1227.00	0.000
X3, Half Pace	1	3209462	3209462	19253.81	0.000
X4, Gender	1	93159	93159	558.87	0.000
X5, Wave: Elite	1	6818	6818	40.90	0.000
X6, Wave: One	1	41068	41068	246.37	0.000
X7, Wave: Two	1	23812	23812	142.85	0.000
X8, Wave: Three	1	6612	6612	39.67	0.000
Error	13174	2196004	167		
Lack-of-Fit	13158	2194858	167	2.33	0.024
Pure Error	16	1147	72		
Total	13182	23729231			

I First Plotted residual plot for Y, histogram, and Normal QQ Plot for Y. The results showed heavier variance along the tails, which is expected and understandable as the early and late waves are more extreme (elite and wave four) in their performances. Kurtosis is 16.619, which indicates leptokurtic distributed data. This gives insight that a large amount of variance in the model is due to the infrequent extremes. After an ln(Y) transformation, the normality of the residuals seemed to decrease, R^2 decreased as well





I then Transformed Y to ln(Y). Below is the OLS Model output:

OLS Regression Results

Dep. Variable:	lnY, Official Time	**R-squared:**	0.896
Model:	OLS	**Adj. R-squared:**	0.896
Method:	Least Squares	**F-statistic:**	1.420e+04
Date:	Sat, 07 Dec 2019	**Prob (F-statistic):**	0.00
Time:	18:22:39	**Log-Likelihood:**	19301.
No. Observations:	13183	**AIC:**	-3.858e+04
Df Residuals:	13174	**BIC:**	-3.852e+04
Df Model:	8		
Covariance Type:	nonrobust		

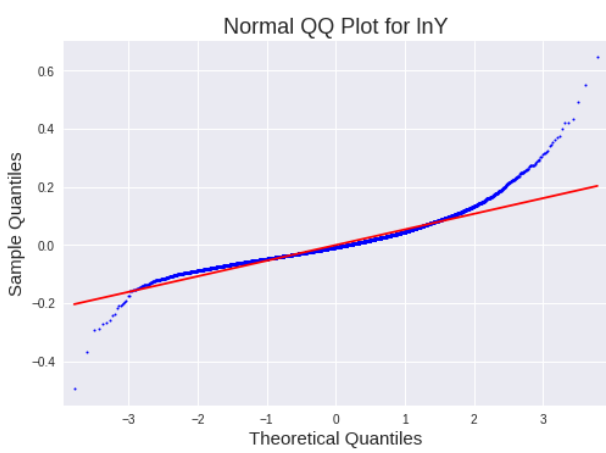
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	4.6134	0.007	661.197	0.000	4.600	4.627
X1, Age	-3.796e-05	5.2e-05	-0.730	0.466	-0.000	6.4e-05
X2, 5K Pace	-0.0616	0.002	-27.838	0.000	-0.066	-0.057
X3, Half Pace	0.2275	0.002	120.962	0.000	0.224	0.231
X4, Gender	-0.0257	0.001	-19.995	0.000	-0.028	-0.023
X5, Wave: Elite	-0.1578	0.009	-16.950	0.000	-0.176	-0.140
X6, Wave: One	-0.0556	0.002	-22.365	0.000	-0.061	-0.051
X7, Wave: Two	-0.0198	0.002	-10.074	0.000	-0.024	-0.016
X8, Wave: Three	-0.0022	0.002	-1.254	0.210	-0.006	0.001

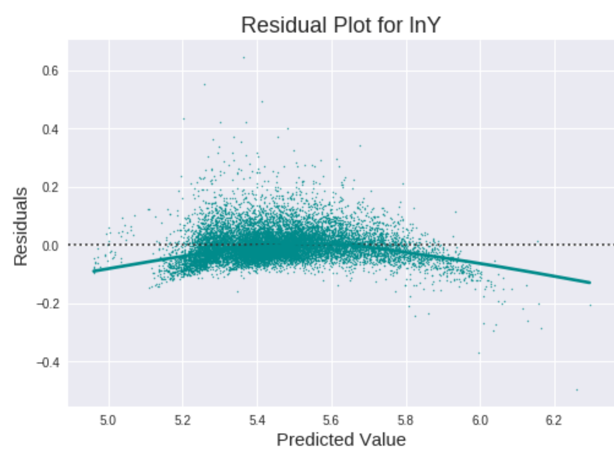
Omnibus:	4164.583	**Durbin-Watson:**	1.614
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	32339.126
Skew:	1.303	**Prob(JB):**	0.00
Kurtosis:	10.217	**Cond. No.**	915.

-By looking at the P values associated with the predictor variables, this ln(Y) transformation would hint towards dropping X1, Age, and X8, Wave Three, from the model due to the p value exceeding an alpha level of .05. Also, in this model, R^2 decreased from .907 to .896.

-Moreover, the residual plot for ln(Y) transformed OLS looks less linear and more curved in form, which is not desirable.

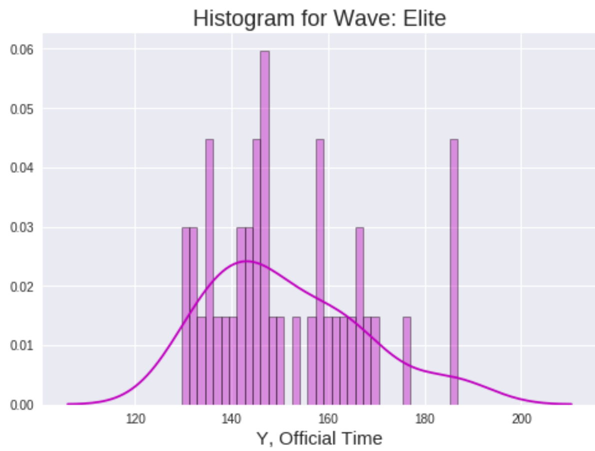
-The histogram reveals semi constant variance, but greater variance along the tails. This could be due to the following: 1) wave four's charity runners who did not qualify to run; 2) the elite wave's small group of elite runners who have an unnatural talent and facility compared to the general population.

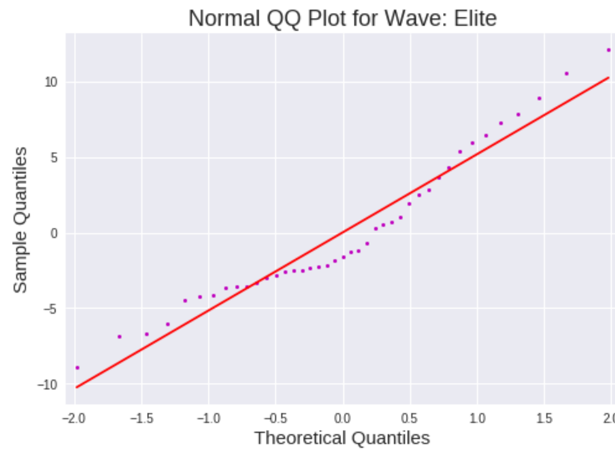


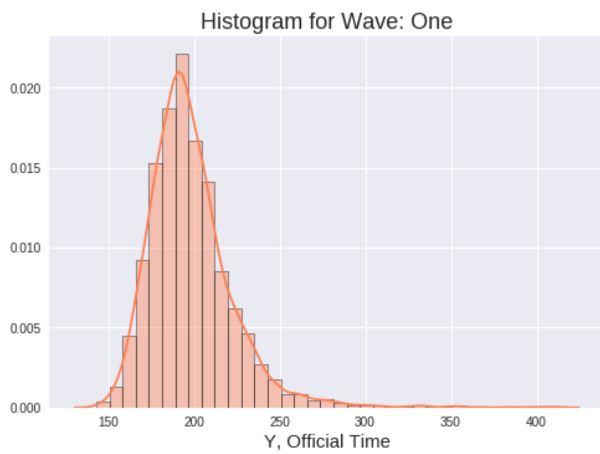


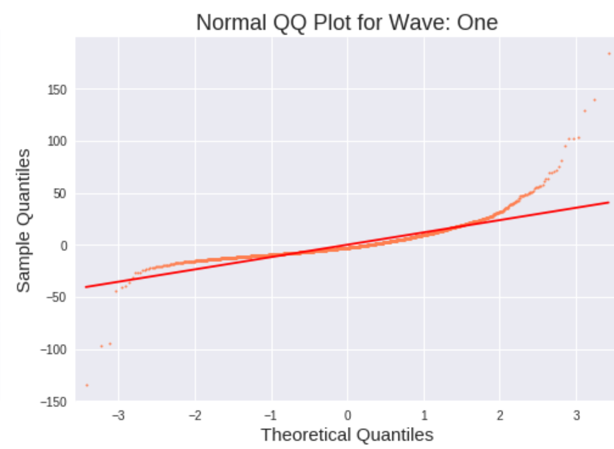
Therefore, I decided to build my model without the Y transform.

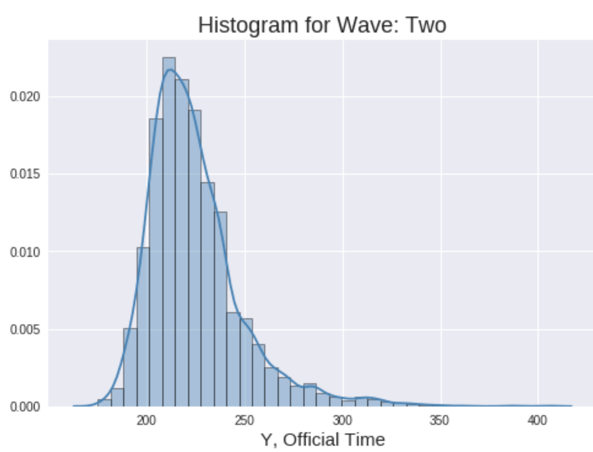
In the second part of my study, I broke the dataset into 5 groups based on waves and built 5 miniature models accordingly. The plots below show visualizations of the distribution of the data, and the appendix contains more information on the OLS details from the initial regression.

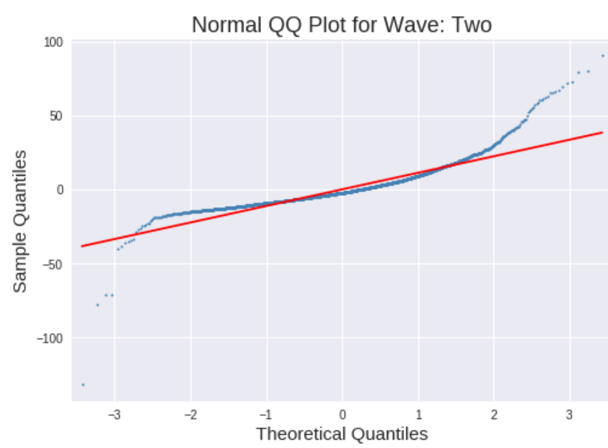


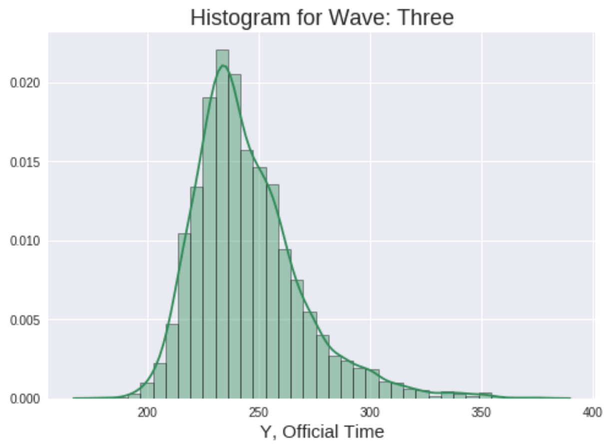


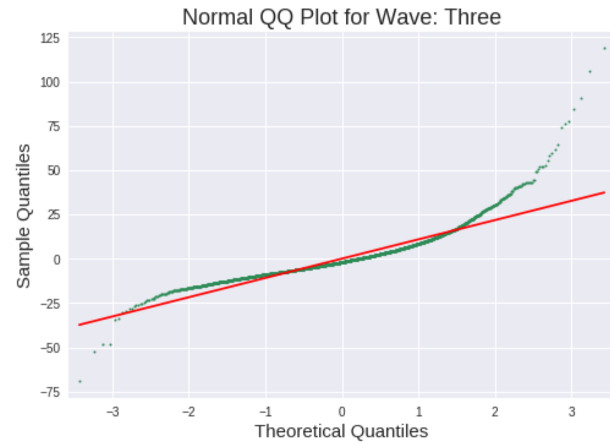


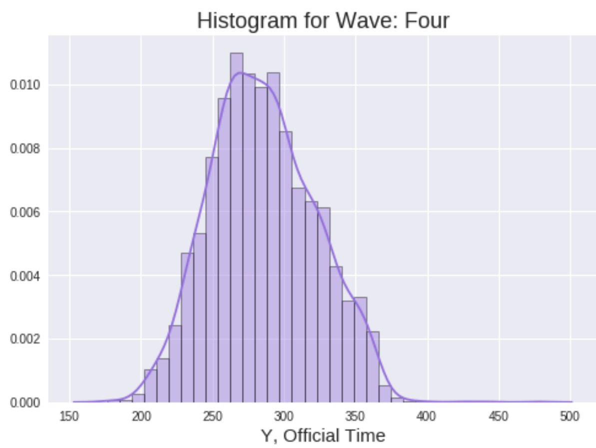


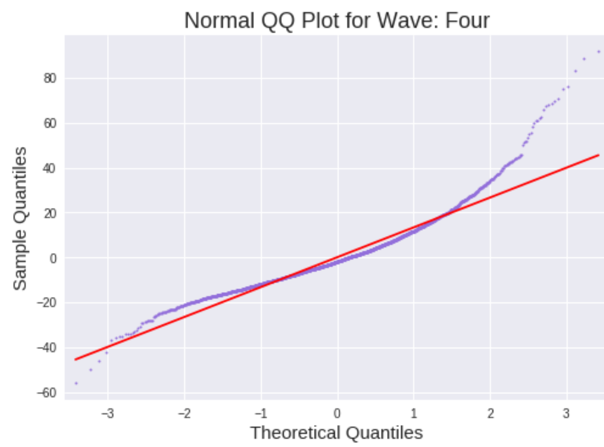












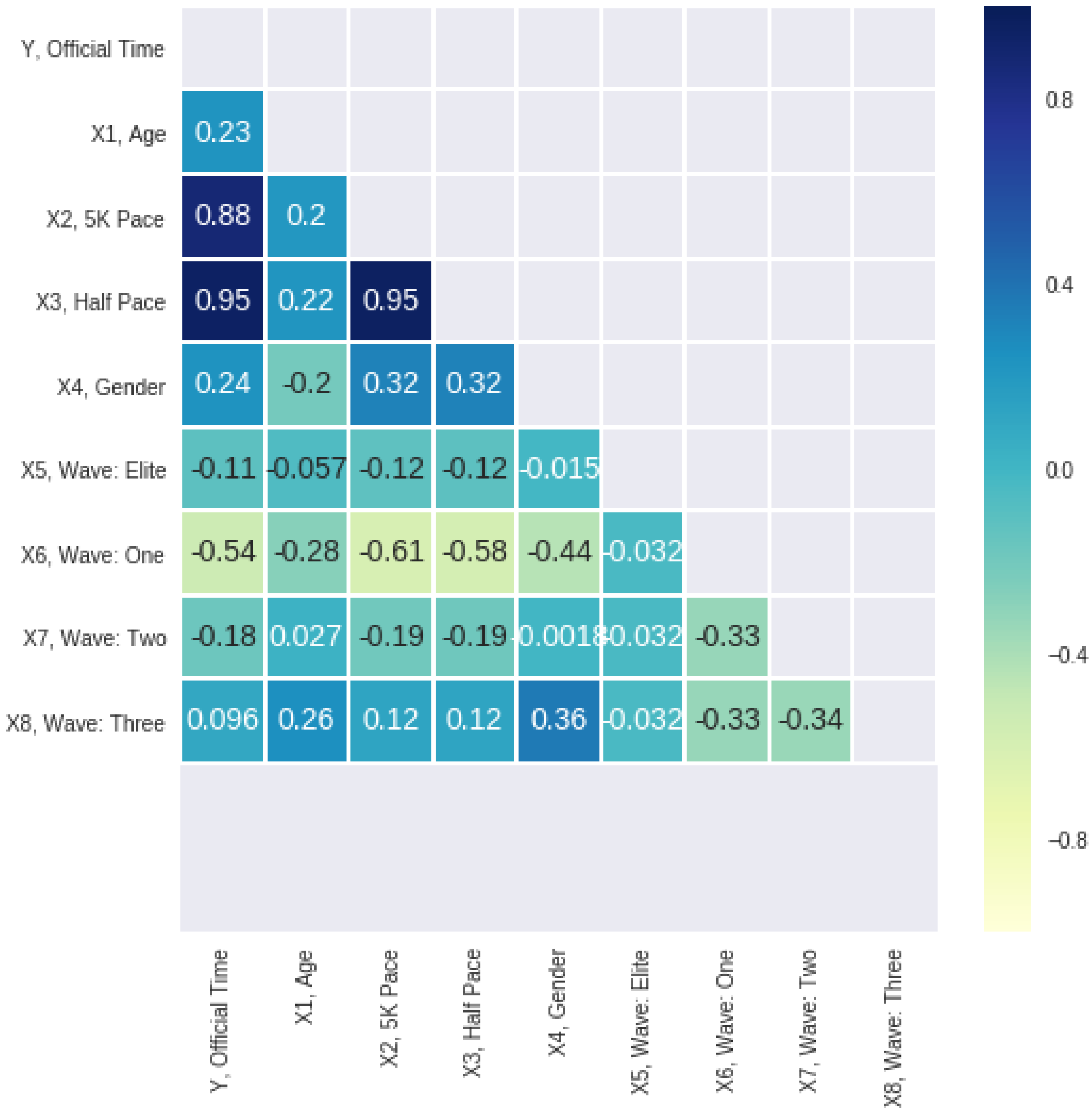
We can note from the wave histograms and normal plots that they all have some slight skewness, with the exception of wave 4. We can also note that the elite wave is not particularly normally distributed as the field is so small. The goal of these smaller models is to find relationships between the coefficients, so there is not too much concern, as the main model is more robust and will be used as the main tool to

Correlation Matrix

	Y, Official Time	X1, Age	X2, 5K Pace	X3, Half Pace	X4, Gender	X5, Wave: Elite	X6, Wave: One	X7, Wave: Two	X8, Wave: Three
Y, Official Time	1.000000	0.227210	0.875601	0.945579	0.239609	-0.113838	-0.535999	-0.179481	0.096410
X1, Age	0.227210	1.000000	0.204271	0.222481	-0.204270	-0.057078	-0.279213	0.027032	0.256546
X2, 5K Pace	0.875601	0.204271	1.000000	0.953547	0.321853	-0.122000	-0.608814	-0.190943	0.123380
X3, Half Pace	0.945579	0.222481	0.953547	1.000000	0.321262	-0.117178	-0.581353	-0.185030	0.119776
X4, Gender	0.239609	-0.204270	0.321853	0.321262	1.000000	-0.015269	-0.440112	-0.001842	0.362502
X5, Wave: Elite	-0.113838	-0.057078	-0.122000	-0.117178	-0.015269	1.000000	-0.032191	-0.032302	-0.032452
X6, Wave: One	-0.535999	-0.279213	-0.608814	-0.581353	-0.440112	-0.032191	1.000000	-0.333299	-0.334847
X7, Wave: Two	-0.179481	0.027032	-0.190943	-0.185030	-0.001842	-0.032302	-0.333299	1.000000	-0.336000
X8, Wave: Three	0.096410	0.256546	0.123380	0.119776	0.362502	-0.032452	-0.334847	-0.336000	1.000000

Heatmap of Correlation Matrix

The heatmap shows the correlations between the variables. The more extreme towards blue and yellow hues, the higher the correlation (positive or negative).



MODEL DEVELOPMENT AND ANALYSIS

I conducted an initial OLS for my general model as well as my five subset wave models, followed by a thorough outlier detection and influencing point procedure. The python code in the appendix displays the procedure, which eliminates data points with t_i values greater than t value for test for Bonferroni simultaneous test procedure with a family significance level of alpha = .10, h_{ii} value greater than 3*(p/n), $DFFITS_i$ greater than 4*sqrt(p/n). I had to change the cutoff points for the leverages h_{ii} from the traditional 2*(p/n) to 3*(p/n) and DFFITS from the traditional 2*sqrt(p/n) due to large number of points that violated the criteria. The output below shows how each of the 6 models' datasets were altered as a result of my outlier and leverage detection and reduction process:

GENERAL MODEL

```
t value for test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10:  4.478348339712635
Hii cutoff point for declaring outliers in X:  0.0020480922400060685
DFFITS cutoff point for declaring highly influential points:  0.10451391588379845
F statistic from F(p,n-p) distribution at 50th percentile:  0.9270286556620607
The number of outliers and influential points deleted from the data is:  241
```

WAVE ELITE MODEL

```
t value for test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10:  3.299677798044307
Hii cutoff point for declaring outliers in X:  0.6585365853658537
DFFITS cutoff point for declaring highly influential points:  1.8740851426632728
F statistic from F(p,n-p) distribution at 50th percentile:  0.9467110802014125
The number of outliers and influential points deleted from the data is:  0
```

WAVE ONE MODEL

```
t value for test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10:  4.17613601486594
Hii cutoff point for declaring outliers in X:  0.008214177061149984
DFFITS cutoff point for declaring highly influential points:  0.20930586309545476
F statistic from F(p,n-p) distribution at 50th percentile:  0.9271713044432743
The number of outliers and influential points deleted from the data is:  53
```

WAVE TWO MODEL

```
t value for test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10:  4.177285902697084
Hii cutoff point for declaring outliers in X:  0.008171912832929782
DFFITS cutoff point for declaring highly influential points:  0.2087667001917663
F statistic from F(p,n-p) distribution at 50th percentile:  0.9271703245863058
The number of outliers and influential points deleted from the data is:  66
```

WAVE THREE MODEL

```
t value for test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10:  4.178831949367677
Hii cutoff point for declaring outliers in X:  0.008115419296663661
DFFITS cutoff point for declaring highly influential points:  0.20804383251822886
F statistic from F(p,n-p) distribution at 50th percentile:  0.9271690148809925
The number of outliers and influential points deleted from the data is:  49
```

WAVE FOUR MODEL

```
t value for test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10:  4.171820478518554
Hii cutoff point for declaring outliers in X:  0.008374689826302729
DFFITS cutoff point for declaring highly influential points:  0.21134098610290405
F statistic from F(p,n-p) distribution at 50th percentile:  0.9271750260467924
The number of outliers and influential points deleted from the data is:  24
```

It is fascinating that wave four has the least amount of outliers for the normally distributed datasubsets (the elite model does not have enough datapoints to have significant outliers). Initially, wave four seemed to be a potentially problematic wave group as it is mixed with both qualifiers and charity runners (non-qualifiers). However, it is in fact the most normally distributed group.

Initially, I conducted best subsets with interaction terms and second order terms. However, for the extra 10+ variables added to the model, there was no increase in R^2, and almost a twofold increase in selection criteria such as PRESS, AIC, and BIC. Therefore, I decided to stick with a simpler, strictly first order model (and achieve an R^2 of the same quality). For such a large dataset, this is sensible as it is less expensive to process in terms of time and resources.

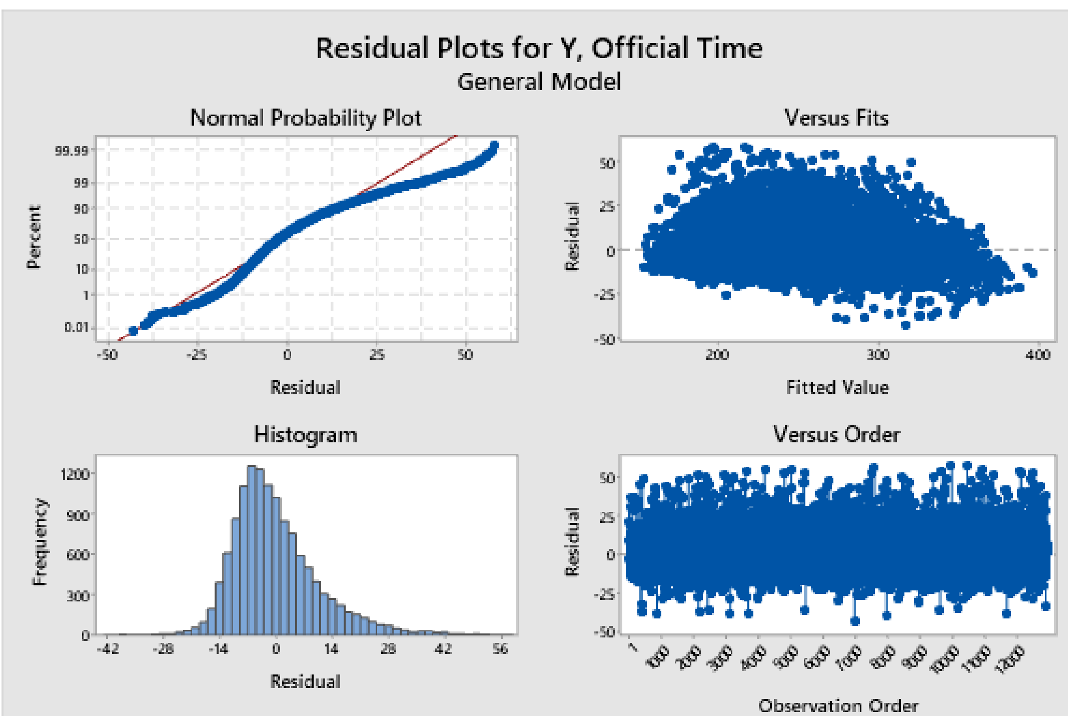
For each model (the general model and the 5 wave subset models), I started my analysis by using the exported clean datasets from python and imported them to Minitab for a detailed regression model building environment. I started with a standard regression model best fit, followed by best subsets, stepwise, forward selection (alpha to enter = .25), and backward elimination (alpha to remove = .1). I then put the “B” data file (the validation data set that was the B partition of the dataset) through the regression model selected from the cleaned “A” data file and created a composite results table (in the results section) based on my findings.

The regression model for my general model, attained from using forward selection with alpha = .25, led me to selecting the model with equation:

$$Y = 24.1 + -0.0609X_1 – 22.314X_2 + 66.460X_3 – 6.829X_4 – 7.996X_6 – 5.332X_7 – 2.543X_8$$

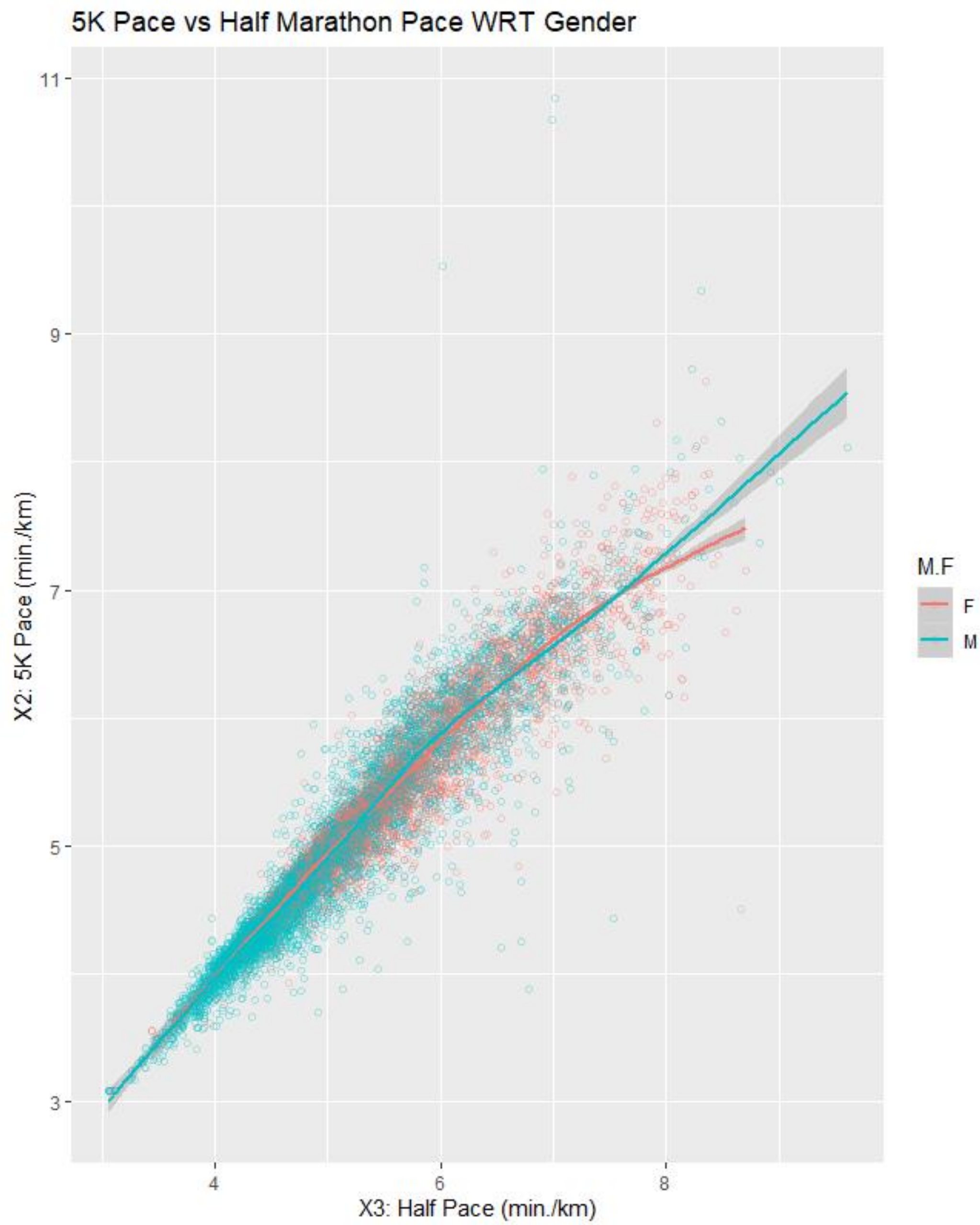
As the outlier and influence detection and reduction method deleted all of the elites from the general model, there is no need for X_5, the indicator of the elite wave.

The Residual Plots below show the relatively normally distributed residuals. Refer to results section for findings and analysis of regression results. The appendix contains each smaller model’s final regression analysis of the chosen model. The minitab file details the process of regressions to attain results in the results table.



A side observation:

An interesting observation is the link between the first 5K pace (X_2) and the first half marathon pace (X_3). Ideally, depending on racing strategy, one would think that the first 5K pace should not be lower (meaning faster) than the first half marathon pace. In smaller races that are not well organized, this strategy may be beneficial as a faster runner may need to stride ahead of slower runners to avoid being confined in an unfavorable pace pack. However, in an organized race like the Boston Marathon, where runners are organized according into corrals and waves, there is no need to speed up in the beginning to achieve a good positioning. Nonetheless, the data from the 2017 Boston Marathon shows that in fact, runners will tend to run the first 5K at a faster speed than their first half marathon of the race.



From the plot below, and our knowledge that slope of one between X2, 5K pace, and X3, half marathon pace, would indicate constant speed, we can understand more about gender impact on consistency in pacing. It is clear that as the trendlines overlap each other until the half pace of 7:30, the females and males are at the same pace ratio. However, after average pace of 7:30 per km, it is evident that females are slower relative to their first 5K pace, which indicated lack of economic efficiency in race strategy.

RESULTS

Results table incorporating the findings of regression models: (models were selected based on forward selection regression process)

	Model Number	1	1		2	2	3	3	4	4	5	5	6	6
		General	**General**	**Wave Category**	**Elite**	**Elite**	**One**	**One**	**Two**	**Two**	**Three**	**Three**	**Four**	**Four**
	Data Set type	Training	Validation		Training	Validation	Training	Validation	Training	Validation	Training	Validation	Training	Validation
	Statistic													
	p	8	8		3	3	5	5	4	4	4	4	4	4
Constant	b0	24.1	22.11		9.8	28.2	-2.26	3.3	7.87	16.06	11.7	11.15	32.74	32.2
	s{b0}	1.5	1.52		10.9	12	2.47	2.5	2.81	3.38	2.85	3.44	2.26	2.27
X1	b1	-0.0609	-0.0681				-0.1905	-0.1384						
Age	s{b1}	0.0105	0.0117				0.0234	0.027						
X2	b2	-22.314	-18.802		-48.2	-48.13	-34.24	-30.29	-27.6	-27.48	-27.5	-17.77	-15.822	-15.094
5K Pace	s{b2}	0.515	0.536		10.1	8.49	1.44	1.43	1.2	1.36	1.07	1.21	0.764	0.788
X3	b3	64.46	61.368		89.03	83.62	81.2	75.67	71.11	69.53	70.632	61.234	56.339	55.717
Half Pace	s{b3}	0.435	0.457		8.82	7.54	1.31	1.3	1.01	1.11	0.916	0.989	0.628	0.646
X4	b4	-6.788	-6.829				-6.491	-5.784	-4.787	-5.208	-5.61	-4.939	-6.855	-7.001
Gender	s{b4}	0.26	0.289				0.72	0.828	0.353	0.445	0.419	0.489	0.464	0.476
X5	b5													
Wave Elite	s{b5}													
X6	b6	-9.154	-7.996											
Wave One	s{b6}	0.525	0.544											
X7	b7	-5.332	-3.906											
Wave Two	s{b7}	0.411	0.435											
X8	b8	-2.543	-1.624											
Wave Three	s{b8}	0.364	0.393											
	SSEp	1595061	2152750		992.5	1605.6	354689	489377	320794	532226	333923	512636	513839	558343
	PRESSp	1597227.3	2158364		1149.37	1885.55	355800	492499.8	321747	535111	335018.1	527281	515160.8	559889.1
	Cp	8	8		1.3	5.7	5	5	4.1	11.9	3.8	3.4	3.5	8
	MSEp	123.332	163.2974		26.12	36.49	109.879	152.406	99.22	161.427	102.024	152.072	160.826	172.06
	R Sqr. a,p	**92.6**	90.66		**89.26**	81.82	**75.97**	74.4	**77.72**	71.42	**80.05**	72.16	**87.52**	87.1

Findings:

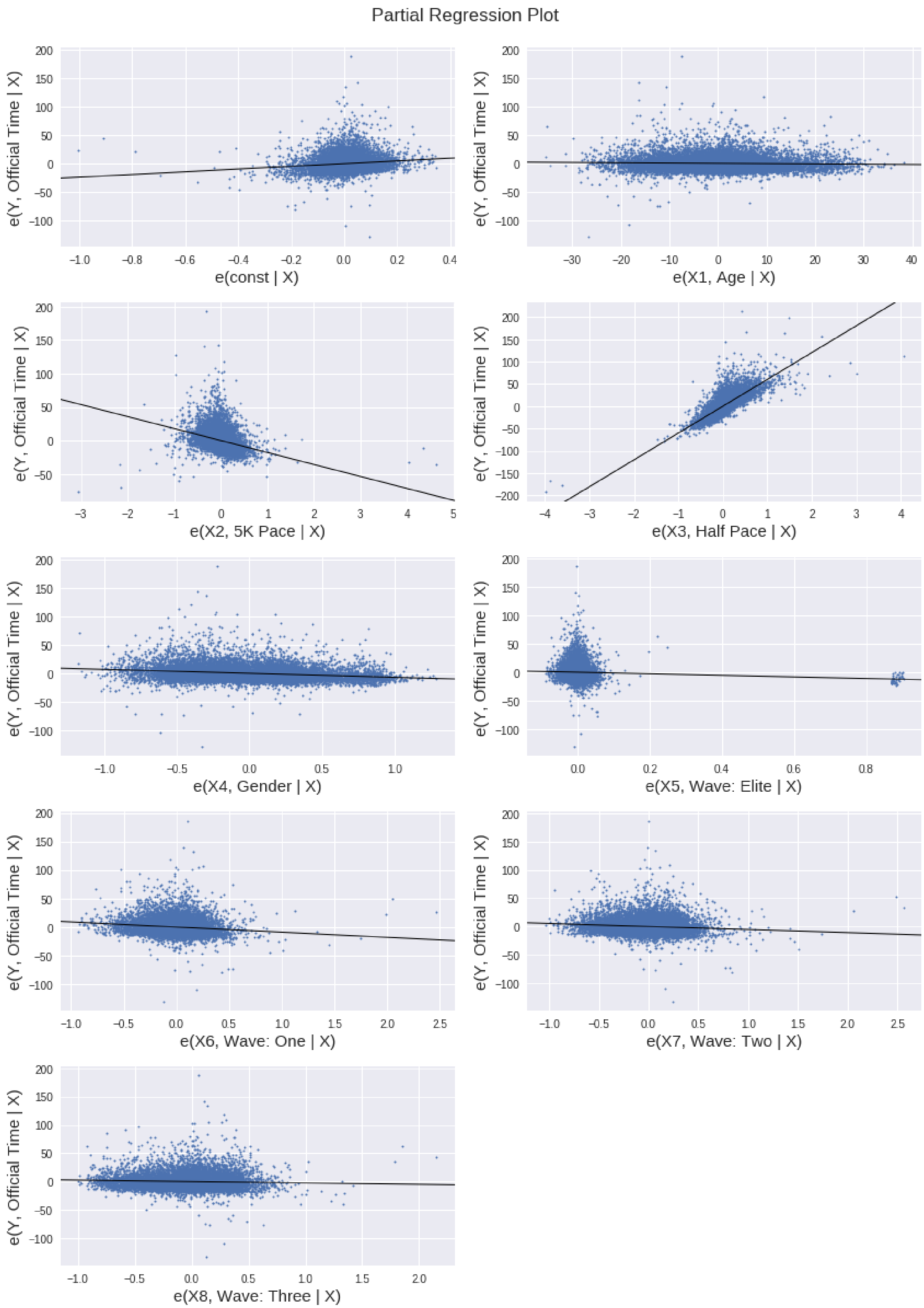
1. As we move from fast to slow wave group, the ratio of coefficient of 5K pace to half marathon pace is smaller. This can be interpreted that the faster the wave, the better they are at keeping consistent 5K and half pace.
2. The elite group and group 1 have the smallest B_0 coefficients. This can be interpreted as a negative split if the ratio of 5K to half marathon pace is greater than ½.
3. The general model has a good R^2 value (92.6), which held up at 90.6 after running the validation data through the model. Moreover, the other models have consistent selection criteria between the training and validation sets. This supports strong models.
4. The coefficient of gender is negative. Gender take value 1 if female and 0 if male. Given that overall times for female are slower than males (so greater response value of Y, official time in minutes), this coefficient is difficult to interpret as we only see crossover point in performance of women's pacing after the 7:30 minute/km pace range. This was shown earlier in a scatter plot of the interesting finding.

5. The training and validation sets have parameter coefficients and errors mostly within confidence intervals produced by the regression model (refer to Minitab regression model outputs and appendix for confidence intervals). Although in the wave one validation model the constant b_0 changes sign, the values are contained in the model's intervals, so this is acceptable.

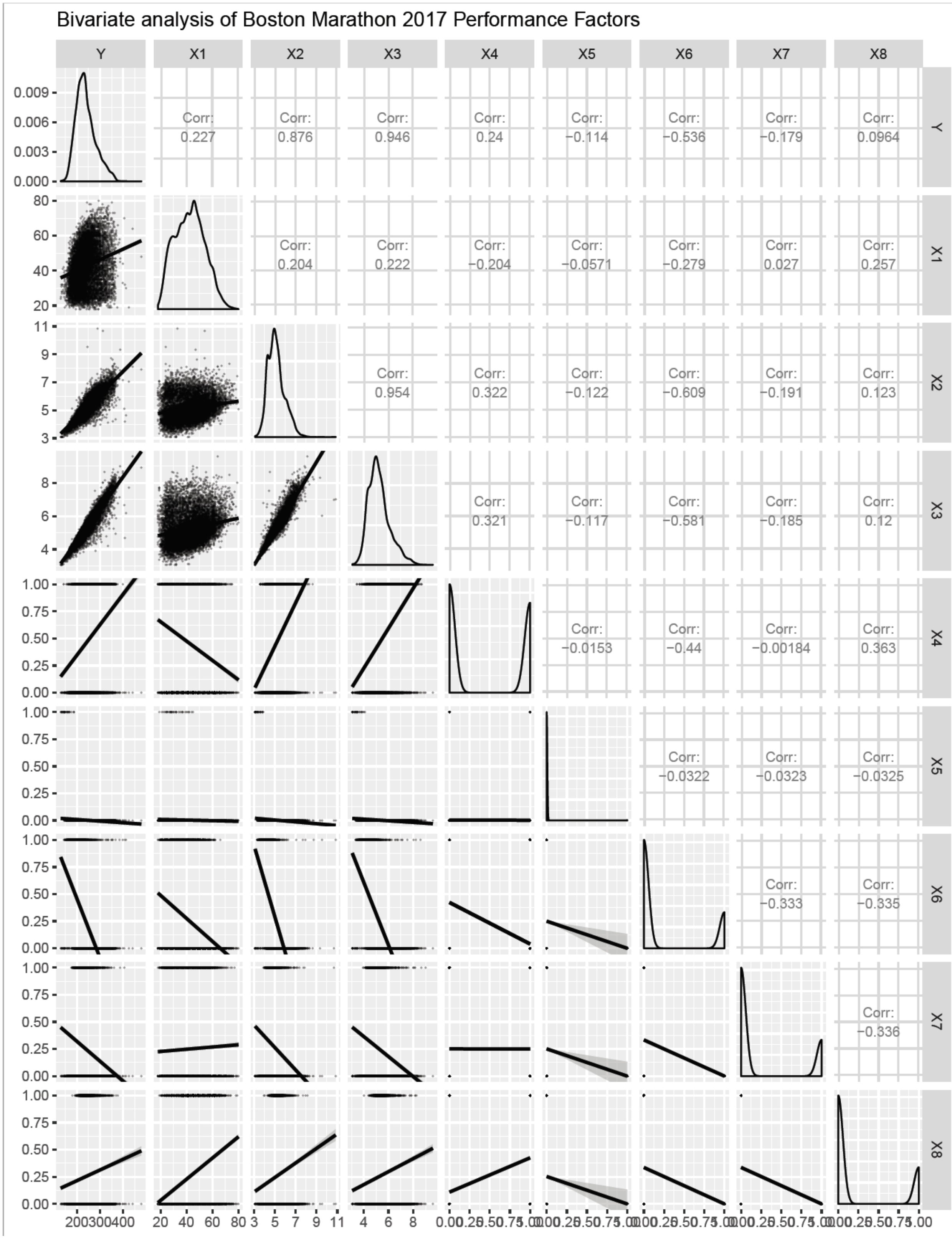
APPENDIX

Sources from Abstract:

(1) https://myemail.constantcontact.com/2017-Boston-Marathon---Participant-News.html?soid=1102184342553&aid=Rigf1jDJ2IE
(2) http://archive.boston.com/sports/marathon/articles/2011/02/17/marathon_qualifying_is_revised/
(3) https://www.baa.org/races/boston-marathon/enter/qualify

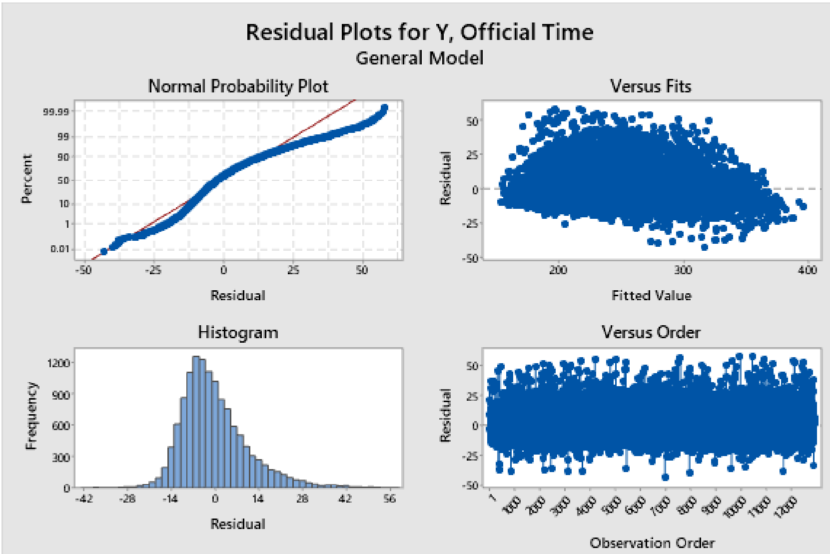


Partial Regression Plot
e(Y, Official Time | X)
e(const | X)
e(X1, Age | X)
e(X2, 5K Pace | X)
e(X3, Half Pace | X)
e(X4, Gender | X)
e(X5, Wave: Elite | X)
e(X6, Wave: One | X)
e(X7, Wave: Two | X)
e(X8, Wave: Three | X)


Bivariate analysis of Boston Marathon 2017 Performance Factors
Y
X1
X2
X3
X4
X5
X6
X7
X8
Corr: 0.227
Corr: 0.876
Corr: 0.946
Corr: 0.24
Corr: −0.114
Corr: −0.536
Corr: −0.179
Corr: 0.0964
Corr: 0.204
Corr: 0.222
Corr: −0.204
Corr: −0.0571
Corr: −0.279
Corr: 0.027
Corr: 0.257
Corr: 0.954
Corr: 0.322
Corr: −0.122
Corr: −0.609
Corr: −0.191
Corr: 0.123
Corr: 0.321
Corr: −0.117
Corr: −0.581
Corr: −0.185
Corr: 0.12
Corr: −0.0153
Corr: −0.44
Corr: −0.00184
Corr: 0.363
Corr: −0.0322
Corr: −0.0323
Corr: −0.0325
Corr: −0.333
Corr: −0.335
Corr: −0.336

Regression Models:

GENERAL MODEL



OLS Regression Results

Dep. Variable:	Y, Official Time	**R-squared:**	0.926
Model:	OLS	**Adj. R-squared:**	0.926
Method:	Least Squares	**F-statistic:**	2.322e+04
Date:	Thu, 12 Dec 2019	**Prob (F-statistic):**	0.00
Time:	20:38:21	**Log-Likelihood:**	-49517.
No. Observations:	12942	**AIC:**	9.905e+04
Df Residuals:	12934	**BIC:**	9.911e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	24.1028	1.502	16.049	0.000	21.159	27.047
X1, Age	-0.0609	0.011	-5.784	0.000	-0.082	-0.040
X2, 5K Pace	-22.3140	0.515	-43.363	0.000	-23.323	-21.305
X3, Half Pace	64.4603	0.435	148.199	0.000	63.608	65.313
X4, Gender	-6.7877	0.260	-26.141	0.000	-7.297	-6.279
X5, Wave: Elite	-8.321e-14	8.91e-16	-93.380	0.000	-8.5e-14	-8.15e-14
X6, Wave: One	-9.1536	0.525	-17.440	0.000	-10.182	-8.125
X7, Wave: Two	-5.3315	0.411	-12.981	0.000	-6.137	-4.526
X8, Wave: Three	-2.5431	0.364	-6.996	0.000	-3.256	-1.831

Omnibus:	2618.927	**Durbin-Watson:**	1.769
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6081.544
Skew:	1.144	**Prob(JB):**	0.00
Kurtosis:	5.458	**Cond. No.**	7.98e+18

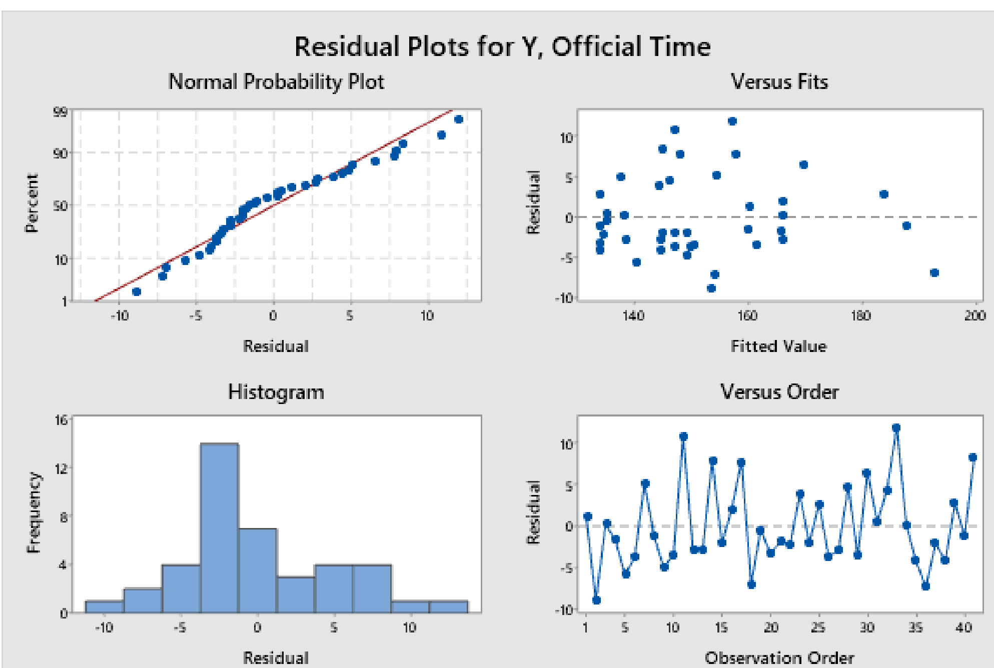
WAVE ELITE

OLS Regression Results

Dep. Variable:	Y, Official Time	**R-squared:**	0.898
Model:	OLS	**Adj. R-squared:**	0.893
Method:	Least Squares	**F-statistic:**	167.2
Date:	Thu, 12 Dec 2019	**Prob (F-statistic):**	1.47e-19
Time:	21:01:37	**Log-Likelihood:**	-123.50
No. Observations:	41	**AIC:**	253.0
Df Residuals:	38	**BIC:**	258.1
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	9.7853	10.899	0.898	0.375	-12.278	31.848
X2, 5K Pace	-48.1964	10.079	-4.782	0.000	-68.600	-27.793
X3, Half Pace	89.0306	8.815	10.099	0.000	71.185	106.876

Omnibus:	3.300	**Durbin-Watson:**	1.205
Prob(Omnibus):	0.192	**Jarque-Bera (JB):**	2.959
Skew:	0.648	**Prob(JB):**	0.228
Kurtosis:	2.773	**Cond. No.**	88.4



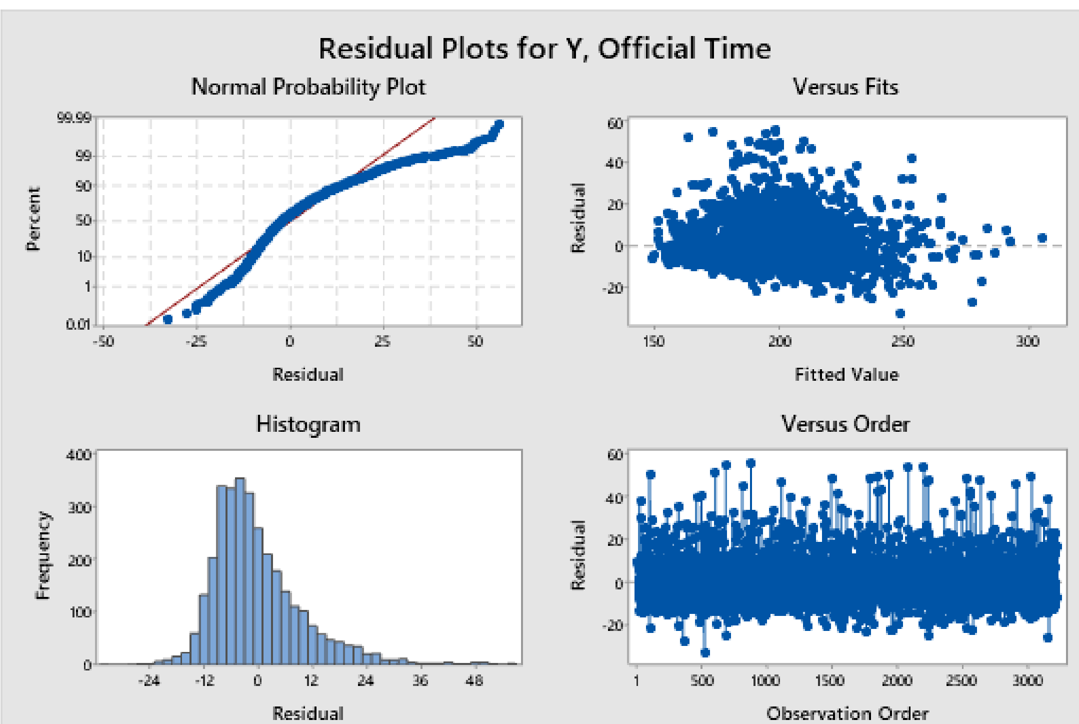
WAVE 1

OLS Regression Results

Dep. Variable:	Y, Official Time	**R-squared:**	0.760
Model:	OLS	**Adj. R-squared:**	0.760
Method:	Least Squares	**F-statistic:**	2563.
Date:	Thu, 12 Dec 2019	**Prob (F-statistic):**	0.00
Time:	21:03:38	**Log-Likelihood:**	-12185.
No. Observations:	3234	**AIC:**	2.438e+04
Df Residuals:	3229	**BIC:**	2.441e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-2.2631	2.471	-0.916	0.360	-7.109	2.583
X1, Age	-0.1905	0.023	-8.137	0.000	-0.236	-0.145
X2, 5K Pace	-34.2372	1.437	-23.826	0.000	-37.055	-31.420
X3, Half Pace	81.1999	1.305	62.209	0.000	78.641	83.759
X4, Gender	-6.4915	0.720	-9.017	0.000	-7.903	-5.080

Omnibus:	928.667	**Durbin-Watson:**	1.966
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2902.171
Skew:	1.457	**Prob(JB):**	0.00
Kurtosis:	6.613	**Cond. No.**	533.



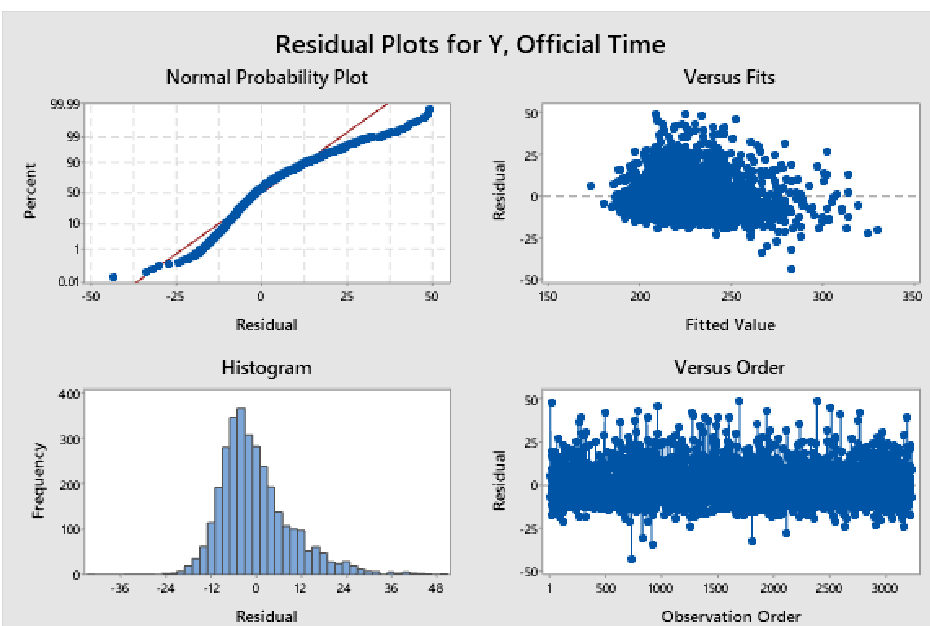
WAVE 2

OLS Regression Results

Dep. Variable:	Y, Official Time	**R-squared:**	0.778
Model:	OLS	**Adj. R-squared:**	0.778
Method:	Least Squares	**F-statistic:**	3776.
Date:	Thu, 12 Dec 2019	**Prob (F-statistic):**	0.00
Time:	21:05:43	**Log-Likelihood:**	-12035.
No. Observations:	3238	**AIC:**	2.408e+04
Df Residuals:	3234	**BIC:**	2.410e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	7.8726	2.813	2.798	0.005	2.357	13.389
X2, 5K Pace	-27.6033	1.201	-22.990	0.000	-29.957	-25.249
X3, Half Pace	71.1111	1.012	70.237	0.000	69.126	73.096
X4, Gender	-4.7874	0.353	-13.559	0.000	-5.480	-4.095

Omnibus:	650.930	**Durbin-Watson:**	2.017
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1489.036
Skew:	1.131	**Prob(JB):**	0.00
Kurtosis:	5.433	**Cond. No.**	116.



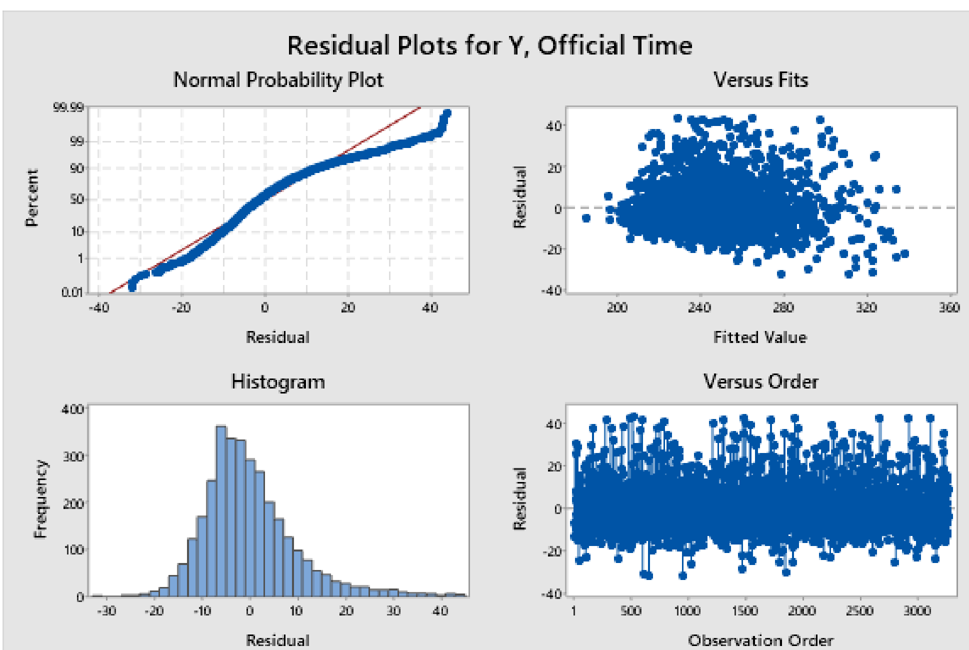
WAVE 3

OLS Regression Results

Dep. Variable:	Y, Official Time	**R-squared:**	0.801
Model:	OLS	**Adj. R-squared:**	0.801
Method:	Least Squares	**F-statistic:**	4398.
Date:	Thu, 12 Dec 2019	**Prob (F-statistic):**	0.00
Time:	21:08:08	**Log-Likelihood:**	-12229.
No. Observations:	3278	**AIC:**	2.447e+04
Df Residuals:	3274	**BIC:**	2.449e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	11.7013	2.855	4.099	0.000	6.104	17.299
X2, 5K Pace	-27.4980	1.074	-25.596	0.000	-29.604	-25.392
X3, Half Pace	70.6316	0.916	77.132	0.000	68.836	72.427
X4, Gender	-5.6097	0.419	-13.386	0.000	-6.431	-4.788

Omnibus:	656.851	**Durbin-Watson:**	1.987
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1502.419
Skew:	1.128	**Prob(JB):**	0.00
Kurtosis:	5.432	**Cond. No.**	126.



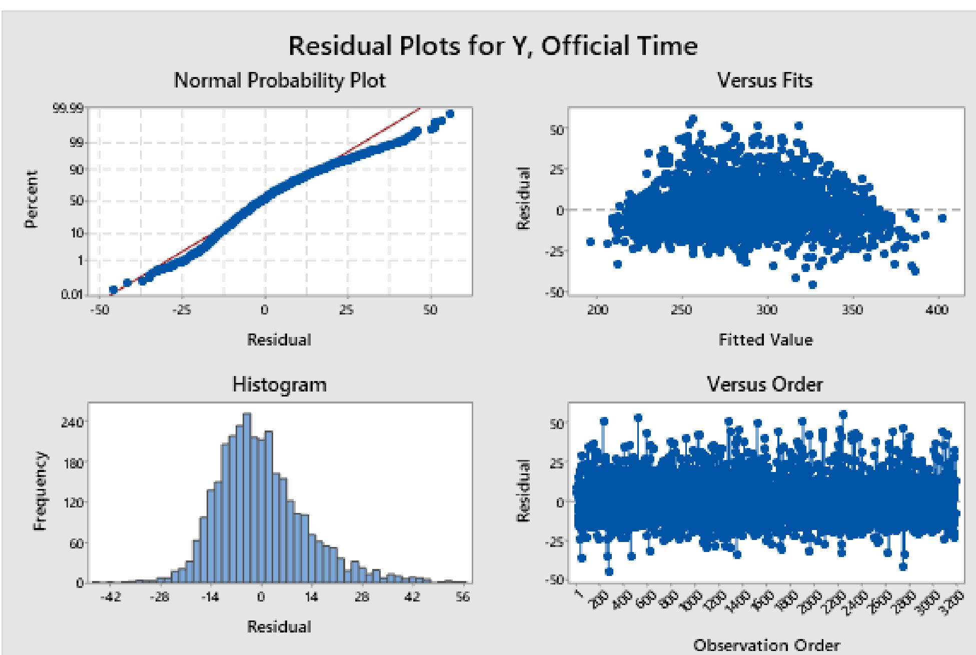
WAVE 4

OLS Regression Results

Dep. Variable:	Y, Official Time	**R-squared:**	0.876
Model:	OLS	**Adj. R-squared:**	0.875
Method:	Least Squares	**F-statistic:**	7495.
Date:	Thu, 12 Dec 2019	**Prob (F-statistic):**	0.00
Time:	21:08:55	**Log-Likelihood:**	-12667.
No. Observations:	3200	**AIC:**	2.534e+04
Df Residuals:	3196	**BIC:**	2.537e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	32.7370	2.258	14.500	0.000	28.310	37.164
X2, 5K Pace	-15.8219	0.764	-20.699	0.000	-17.321	-14.323
X3, Half Pace	56.3387	0.628	89.742	0.000	55.108	57.570
X4, Gender	-6.8547	0.464	-14.762	0.000	-7.765	-5.944

Omnibus:	352.435	**Durbin-Watson:**	2.053
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	553.545
Skew:	0.790	**Prob(JB):**	6.30e-121
Kurtosis:	4.286	**Cond. No.**	90.9



Python Code

Python Packages Import

```
"""

#initial statements for numpy and pandas packages
import math
import numpy as np
import pandas as pd
import csv
import seaborn as sns
import scipy as sp
import statsmodels.api as sm
from statsmodels.formula.api import ols
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.graphics.api as smg
import random
from sklearn.utils import shuffle
```

Reading in Data File

```
from google.colab import files
uploaded = files.upload()

#boston_data = pd.read_csv(r'C:\\Users\\dheym\\OneDrive\\Documents\\WilliamMary19-20\\LinearRegression\\marathon_results_2017.csv\\marathon_results_2017.csv')
boston_data= pd.read_csv('marathon_results_2017_best.csv')
#boston_data.head(100)

#Creation of dataframe only incorporating the variables that I am using in my model
boston_data_s = boston_data.loc[:,['Bib','M/F', 'Age', 'X2, 5K Pace', 'X3, Half Pace', 'Y, Official Time']]
#1 for Female and 0 for Male
boston_data_s['X4, Gender'] = np.where(boston_data_s['M/F']=='F', 1, 0)
# don't need this- if female X4 = 0, then it is a male
#boston_data_s['X5, Male'] = np.where(boston_data_s['M/F']=='M', 1, 0)
boston_data_s['X5, Wave: Elite'] = np.where(boston_data_s['Bib']<=100, 1, 0)
boston_data_s['X6, Wave: One'] = np.where(((boston_data_s['Bib']<=7700) &
(boston_data_s['Bib']>=100)) == True, 1, 0)
boston_data_s['X7, Wave: Two'] = np.where(((boston_data_s['Bib']<=15600) & (boston_data_s['Bib'] >=
8000)) == True, 1, 0)
boston_data_s['X8, Wave: Three'] = np.where(((boston_data_s['Bib']<=23600) & (boston_data_s['Bib']
>= 16000)) == True, 1, 0)
# don't need this - case when all other indicator variables = 0
#boston_data_s['X10, Wave: Four'] = np.where(((boston_data_s['Bib']<=32500) & (boston_data_s['Bib']
>= 24000)) == True, 1, 0)
```

#cleaning data types

#Note: Now, 'Bib' and 'M/F' are not needed as waves take on indicator variables and gender also takes on indicator variable
boston_data_s.rename(columns={'Age': 'X1, Age'}, inplace=True)

#boston_data_s.sample(10)

Data Cleaning and Sample Selection

#Splitting Data into training and test data before creation of model
#set seed in random_state argument so that we consistently choose to start at this seed every time
#this code is run so that that training data for model building will be consistent

#get subsets of females, males; shuffle; split into two; bind set F1 and M1 as training and F2 and M2 as validation
boston_F = boston_data_s[boston_data_s['X4, Gender']== 1]
boston_Fs = shuffle(boston_F, random_state = 3)

#There are 11972 females, so split into 2 groups of 5986
boston_Fa = boston_Fs.iloc[0:5986,:]
boston_Fb = boston_Fs.iloc[5986:11972,:]

boston_M = boston_data_s[boston_data_s['X4, Gender']== 0]
boston_Ms = shuffle(boston_M, random_state = 4)

#There are 14438 males, so split into 2 groups of 7219
boston_Ma = boston_Ms.iloc[0:7219,:]
boston_Mb = boston_Ms.iloc[7219:14438,:]

#Now we merge females "a" and males "a" for dataset "a" (for training) and merge
#females "b" and males "b" for dataset "b" (validation)
boston_data_A = pd.concat([boston_Fa, boston_Ma])
boston_data_B = pd.concat([boston_Fb, boston_Mb])
#Now sort the two datasets by Y, official time
boston_data_A = boston_data_A.sort_values(by=['Y, Official Time'])
boston_data_A.dropna
boston_data_B = boston_data_B.sort_values(by=['Y, Official Time'])
boston_data_B.dropna
#boston_data_B.head(20)

#Splitting Data into training and test data before creation of model
#set seed to 30 in random_state argument so that we consistently choose to start at this seed every time
#this code is run so that that training data for model building will be consistent

#boston_data_A.drop('Bib', axis = 1)

```
col_set = ['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender', 'X5, Wave: Elite', 'X6, Wave: One', 'X7,
Wave: Two', 'X8, Wave: Three']
for i in range(len(col_set)):
  indexNames = 0
  indexNames = boston_data_A[boston_data_A[col_set[i]] == '#VALUE!' ].index
  if len(indexNames>0):
    # Delete these row indexes from dataFrame
    boston_data_A.drop(indexNames , inplace=True)

#Do for B as well
for i in range(len(col_set)):
  indexNames = 0
  indexNames = boston_data_B[boston_data_B[col_set[i]] == '#VALUE!' ].index
  if len(indexNames>0):
    # Delete these row indexes from dataFrame
    boston_data_B.drop(indexNames , inplace=True)

#more data cleaning
boston_data_A['X2, 5K Pace'] = pd.to_numeric(boston_data_A['X2, 5K Pace'])
boston_data_A['X3, Half Pace'] = pd.to_numeric(boston_data_A['X3, Half Pace'])
boston_data_A['Index'] = list(range(len(boston_data_A)))

boston_data_B['X2, 5K Pace'] = pd.to_numeric(boston_data_B['X2, 5K Pace'])
boston_data_B['X3, Half Pace'] = pd.to_numeric(boston_data_B['X3, Half Pace'])
boston_data_B['Index'] = list(range(len(boston_data_B)))

#Export to CSV
boston_data_A.to_csv('boston_data_A.csv')
files.download('boston_data_A.csv')

boston_data_B.to_csv('boston_data_B.csv')
files.download('boston_data_B.csv')

"""# Data Statistical Summary: Normality of Residuals"""

#@title OLS Regression Model starting point{ form-width: "120px" }
#OLS regression
X = boston_data_A[['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender', 'X5, Wave: Elite', 'X6, Wave:
One', 'X7, Wave: Two', 'X8, Wave: Three']]
y = boston_data_A['Y, Official Time']
## fit a OLS
X = sm.add_constant(X)
boston_fit_A = sm.OLS(y, X.astype(float)).fit()

# fitted values (need a constant term for intercept)
boston_fit_A_y = boston_fit_A.fittedvalues
boston_data_A['Yfit'] = boston_fit_A_y
```

```
# residuals
boston_fit_A_residuals = boston_fit_A.resid
boston_data_A['Res'] = boston_fit_A_residuals

boston_fit_A.summary()

#@title Residual Plots{ form-width: "120px" }
#residual plot for Y
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
resplot = sns.residplot(boston_data_A['Yfit'], boston_data_A['Res'], lowess=True, color="darkcyan",
scatter_kws={'s':2})
resplot.axes.set_title('Residual Plot for Y')
resplot.axes.set_xlabel('Predicted Value')
resplot.axes.set_ylabel('Residuals')

#Normal QQ plot for Y
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
plt.rc('lines', markersize = 2)
qqplot = sm.qqplot(boston_data_A['Res'], line = 'r',color="darkcyan" )
plt.title('Normal QQ Plot for Y')
plt.show()
#Plot is symmetric with heavy tails

#OLS regression for Y' = lnY
X = boston_data_A[['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender', 'X5, Wave: Elite', 'X6, Wave:
One', 'X7, Wave: Two', 'X8, Wave: Three']]
#new colun for Y' = ln
boston_data_A['lnY, Official Time']= np.log(boston_data_A['Y, Official Time'])
y = boston_data_A['lnY, Official Time']
## fit a OLS
X = sm.add_constant(X)
boston_fit_B_lnY = sm.OLS(y, X.astype(float)).fit()

# fitted values (need a constant term for intercept)
boston_fit_B_lnY_fit = boston_fit_B_lnY.fittedvalues
boston_data_A['lnYfit'] = boston_fit_B_lnY_fit

# residuals
boston_fit_B_lnY_residuals = boston_fit_B_lnY.resid
boston_data_A['Res, lnY'] = boston_fit_B_lnY_residuals
```

```
boston_fit_B_lnY.summary()

#residual plot for lnY

plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
resplot = sns.residplot(boston_data_A['lnYfit'], boston_data_A['Res, lnY'], lowess=True,
color="darkcyan", scatter_kws={'s':2})
resplot.axes.set_title('Residual Plot for lnY')
resplot.axes.set_xlabel('Predicted Value')
resplot.axes.set_ylabel('Residuals')

#Normal QQ plot for lnY
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
plt.rc('lines', markersize = 2)

qqplot2 = sm.qqplot(boston_data_A['Res, lnY'], line = 'r')
plt.title('Normal QQ Plot for lnY')
plt.show()

#Standardized Residuals For Y Histogram
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
stdresidY = boston_fit_A.resid_pearson
sns.distplot(stdresidY, bins = 50, color = "darkcyan", hist_kws=dict(edgecolor="k", linewidth=1))
plt.rcParams["patch.force_edgecolor"] = True
plt.xlim([-5,5])
#plt.autoscale(enable=True, axis='y')
plt.title('Histogram for Normalized Residuals')
plt.show()
#The Histogram of Normalized Residuals reveals a normal distribution. This was using Y prior to the lnY
transformation. For the model, I will
#stick with using Y rather than lnY as there have not been improvements with residual normality after a
lnY transformation.

#Histogram for elite wave
plt.style.use('seaborn')
```

```
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
wave_e = boston_data_A[boston_data_A['X5, Wave: Elite']== 1]
wave_e_B = boston_data_B[boston_data_B['X5, Wave: Elite']== 1]
sns.distplot(wave_e["Y, Official Time"], bins = 35, color = "m", hist_kws=dict(edgecolor="k",
linewidth=1))
plt.rcParams["patch.force_edgecolor"] = True
#plt.xlim([-5,5])
#plt.autoscale(enable=True, axis='y')
plt.title('Histogram for Wave: Elite')
plt.show()

#OLS regression for ELITE
X = wave_e[['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender']]

## fit a OLS
X = sm.add_constant(X)
y = wave_e['Y, Official Time']
elite_OLS = sm.OLS(y, X.astype(float)).fit()

# fitted values (need a constant term for intercept)
elite_OLS_fit = elite_OLS.fittedvalues
wave_e['elite_fit'] = elite_OLS_fit

# residuals
elite_OLS_res = elite_OLS.resid
wave_e['Res, elite_OLS'] = elite_OLS_res

elite_OLS.summary()
#Normal QQ plot for Elite Wave
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
plt.rc('lines', markersize = 3)
qqplot2 = sm.qqplot(wave_e['Res, elite_OLS'], line = "r", color = "m")
plt.title('Normal QQ Plot for Wave: Elite')
plt.show()

#Histogram for wave 1
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
```

```
wave_1 = boston_data_A[boston_data_A['X6, Wave: One']== 1]
wave_1_B = boston_data_B[boston_data_B['X6, Wave: One']== 1]
sns.distplot(wave_1["Y, Official Time"], bins = 35, color = "coral", hist_kws=dict(edgecolor="k",
linewidth=1))
plt.rcParams["patch.force_edgecolor"] = True
#plt.xlim([-5,5])
#plt.autoscale(enable=True, axis='y')
plt.title('Histogram for Wave: One')
plt.show()

#OLS regression for wave 1
X = wave_1[['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender']]
y = wave_1['Y, Official Time']
## fit a OLS
X = sm.add_constant(X)
w1_OLS = sm.OLS(y, X.astype(float)).fit()

# fitted values (need a constant term for intercept)
w1_OLS_fit = w1_OLS.fittedvalues
wave_1['w1_fit'] = w1_OLS_fit

# residuals
w1_OLS_res = w1_OLS.resid
wave_1['Res, w1_OLS'] = w1_OLS_res

w1_OLS.summary()
#Normal QQ plot for Wave One
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
plt.rc('lines', markersize = 2)
qqplot2 = sm.qqplot(wave_1['Res, w1_OLS'], line = "r", color = "coral")
plt.title('Normal QQ Plot for Wave: One')
plt.show()

#Histogram for wave 2
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
wave_2 = boston_data_A[boston_data_A['X7, Wave: Two']== 1]
wave_2_B = boston_data_B[boston_data_B['X7, Wave: Two']== 1]
sns.distplot(wave_2["Y, Official Time"], bins = 35, color = "steelblue", hist_kws=dict(edgecolor="k",
linewidth=1))
plt.rcParams["patch.force_edgecolor"] = True
```

```
#plt.xlim([-5,5])
#plt.autoscale(enable=True, axis='y')
plt.title('Histogram for Wave: Two')
plt.show()

#OLS regression for wave 2
X = wave_2[['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender']]
y = wave_2['Y, Official Time']
## fit a OLS
X = sm.add_constant(X)
w2_OLS = sm.OLS(y, X.astype(float)).fit()

# fitted values (need a constant term for intercept)
w2_OLS_fit = w2_OLS.fittedvalues
wave_2['w2_fit'] = w2_OLS_fit

# residuals
w2_OLS_res = w2_OLS.resid
wave_2['Res, w2_OLS'] = w2_OLS_res

w2_OLS.summary()
#Normal QQ plot for Wave One
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
plt.rc('lines', markersize = 2)
qqplot2 = sm.qqplot(wave_2['Res, w2_OLS'], line = "r", color = "steelblue")
plt.title('Normal QQ Plot for Wave: Two')
plt.show()

#Histograms for wave 3
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
wave_3 = boston_data_A[boston_data_A['X8, Wave: Three']== 1]
wave_3_B = boston_data_B[boston_data_B['X8, Wave: Three']== 1]
sns.distplot(wave_3["Y, Official Time"], bins = 35, color = "seagreen", hist_kws=dict(edgecolor="k",
linewidth=1))
plt.rcParams["patch.force_edgecolor"] = True
#plt.xlim([-5,5])
#plt.autoscale(enable=True, axis='y')
plt.title('Histogram for Wave: Three')
plt.show()
```

```
#OLS regression for wave 3
X = wave_3[['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender']]
y = wave_3['Y, Official Time']
## fit a OLS
X = sm.add_constant(X)
w3_OLS = sm.OLS(y, X.astype(float)).fit()

# fitted values (need a constant term for intercept)
w3_OLS_fit = w3_OLS.fittedvalues
wave_3['w3_fit'] = w3_OLS_fit

# residuals
w3_OLS_res = w3_OLS.resid
wave_3['Res, w3_OLS'] = w3_OLS_res

w3_OLS.summary()
#Normal QQ plot for Wave One
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
plt.rc('lines', markersize = 2)
qqplot2 = sm.qqplot(wave_3['Res, w3_OLS'], line = "r", color = "seagreen")
plt.title('Normal QQ Plot for Wave: Three')
plt.show()

#Histogram for wave 4
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
wave_4 = boston_data_A[(boston_data_A['X8, Wave: Three'] + boston_data_A['X7, Wave: Two'] +
boston_data_A['X6, Wave: One'] + boston_data_A['X5, Wave: Elite']) == 0 ]
wave_4_B = boston_data_B[(boston_data_B['X8, Wave: Three'] + boston_data_B['X7, Wave: Two'] +
boston_data_B['X6, Wave: One'] + boston_data_B['X5, Wave: Elite']) == 0 ]
sns.distplot(wave_4["Y, Official Time"], bins = 35, color = "mediumpurple", hist_kws=dict(edgecolor="k",
linewidth=1))
plt.rcParams["patch.force_edgecolor"] = True
#plt.xlim([-5,5])
#plt.autoscale(enable=True, axis='y')
plt.title('Histogram for Wave: Four')
plt.show()

#OLS regression for wave 4
X = wave_4[['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender']]
y = wave_4['Y, Official Time']
```

```
## fit a OLS
X = sm.add_constant(X)
w4_OLS = sm.OLS(y, X.astype(float)).fit()

# fitted values (need a constant term for intercept)
w4_OLS_fit = w4_OLS.fittedvalues
wave_4['w4_fit'] = w4_OLS_fit

# residuals
w4_OLS_res = w4_OLS.resid
wave_4['Res, w4_OLS'] = w4_OLS_res

w4_OLS.summary()
#Normal QQ plot for Wave One
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
plt.rc('lines', markersize = 2)

qqplot2 = sm.qqplot(wave_4['Res, w4_OLS'], line = "r", color = "mediumpurple")
plt.title('Normal QQ Plot for Wave: Four')
plt.show()

"""# Preliminary Research"""

# Understanding the data
#Scatter Plot of 5k pace vs half marathon pace
cmap = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)

ax = sns.scatterplot(x = 'X2, 5K Pace', y = 'X3, Half Pace', sizes=(20, 200), palette=cmap, hue_norm=(0, 7),
legend="full", data = boston_data_A)

#Partial Regression Plots
fig = plt.figure(figsize=(12,17))
plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
plt.rc('lines', markersize = 2)
plt.rc('lines', linewidth = 1)
plt.rc('lines', color = 'forestgreen')
fig = sm.graphics.plot_partregress_grid( boston_fit_A, fig=fig)
#fig2 = boston_fit_A.plot_coefficients_of_determination(figsize=(8,2))

"""# Data Statistical Summary: Correlations, Matrix Plots, Scatter Plots"""
```

```
#@title Correlation Matrix Heatmap{ form-width: "120px" }
boston_select_A = boston_data_A.loc[:,['Y, Official Time','X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4,
Gender', 'X5, Wave: Elite', 'X6, Wave: One', 'X7, Wave: Two', 'X8, Wave: Three']]
corrMatrix = boston_select_A.corr()
display(corrMatrix)

mask = np.zeros_like(corrMatrix)
mask[np.triu_indices_from(mask)] = True
fig, ax = plt.subplots(figsize=(8,8))


with sns.axes_style("white"):
  sns.heatmap(corrMatrix, mask=mask, square= True,  annot=True, cmap='YlGnBu',  vmin=-1, vmax=1,
linewidths=1)
  ax.set_ylim(11, 0)

#@title Pairwise { form-width: "120px" }
display(boston_select_A.describe(include='all'))
#d.summary(stats='basic', columns='all', orientation='auto')

#@title Pairwise scatter plots for checking multicollinearity{ form-width: "120px" }
sns.pairplot(boston_select_A, height = 2.5)

#I'd like to investigate the negative coefficient for X4, Gender
sns.lmplot(y='X2, 5K Pace', x='X3, Half Pace',
        hue='X4, Gender', data=boston_select_A, lowess = True, palette = 'seismic', markers = "+",
scatter_kws={'s':6, 'alpha':0.2})
#plt.rc("lines", )
plt.title('Impact of Gender on Pace')
```

Remedial Measures

```
from scipy import stats
#influence & outliers

#GENERAL MODEL
#make new empty dataframe of outliers
oddcases_g = pd.DataFrame()

#INFLUENCE OLS Influence results
b_influences = boston_fit_A.get_influence()

###ti externally studentized deleted residuals
b_ti = b_influences.resid_studentized_external
boston_data_A['ti']=b_ti
###t test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10
#t(1-(alpha/2n); n-p-1)
```

```
#cases greater denote outliers in Y; should be added to outlier list
t_bon = stats.t.ppf(1-(.10/(2*len(boston_data_A))), (len(boston_data_A)-9-1))
#4.478348339712635
print("t value for test for Bonferroni simultaneous test procedure with a family significance level of alpha = .10: ", t_bon)
bad_ti = boston_data_A[abs(boston_data_A['ti']) > t_bon]
#71 outliers; not bad for sample size

#### leverage, from statsmodels internals
b_leverage = b_influences.hat_matrix_diag
boston_data_A['hii'] = b_leverage
#leverage that is greater than 3p/n denotes extreme outliers in X (I changed criteria from 2p/n to 3p/n to avoid deleting too many data points)
hii_marker = 3*9/len(boston_data_A)
print("Hii cutoff point for declaring outliers in X: ", hii_marker)
bad_hii = boston_data_A[abs(boston_data_A['hii']) > hii_marker]
#463 cases (greater than 2*(p/n)) --> 155 cases with tighter criteria (greater than 3*(p/n))

#### DFFITS, from statsmodels internals
###NOTE, due to large number of DFFITS greater than 2*sqrt(p/n), I changed the criteria of highly influential for the model to 4*sqrt(p/n) to avoid
#getting rid of too many data points
b_DFFITS = b_influences.dffits
boston_data_A['DFFITS'] = b_DFFITS[0]
#DFFITS greater than 4*sqrt(p/n) denotes very high influence; for a small n, DFFITS greater than 1 denotes high influence
DFFITS_marker = 4*math.sqrt(9/len(boston_data_A))
print("DFFITS cutoff point for declaring highly influential points: ", DFFITS_marker)
bad_DFFITS = boston_data_A[abs(boston_data_A['DFFITS']) > DFFITS_marker]
#607 cases (greater than 2*sqrt(p/n)) --> 127 cases with tighter criteria (greater than 4*sqrt(p/n))

# cook's distance, from statsmodels internals
b_cooks = b_influences.cooks_distance[0]
boston_data_A['Cooks distance'] = b_cooks
#Now we need the F statistic from F(p,n-p) distribution to see if the percentile is 50 percent or more. Find 50th percentile and see if DFFITS value exceeds
F50 = sp.stats.f.ppf(q=.5, dfn=9, dfd=(len(boston_data_A)-9))
print("F statistic from F(p,n-p) distribution at 50th percentile: ", F50)
bad_cooks = boston_data_A[abs(boston_data_A['Cooks distance']) > F50]
#0.9270286556620607
#No cases

#TABLE OF OUTLIERS AND INFLUENCERS
oddcases_g = pd.concat([bad_ti, bad_hii, bad_DFFITS, bad_cooks])
boston_data_A_g = boston_data_A[~boston_data_A['Bib'].isin(oddcases_g['Bib'])]
print("The number of outliers and influential points deleted from the data is: ", len(boston_data_A)-len(boston_data_A_g))
```

```
#Export to CSV
boston_data_A_g = shuffle(boston_data_A_g, random_state = 3)
boston_data_A_g.to_csv('boston_data_A_g.csv')
files.download('boston_data_A_g.csv')

boston_data_B= shuffle(boston_data_B, random_state = 3)
boston_data_B.to_csv('boston_data_B.csv')
files.download('boston_data_B.csv')

#WAVE Elite
#make new empty dataframe of outliers
oddcases_e = pd.DataFrame()

#INFLUENCE OLS Influence results
b_influences_e = elite_OLS.get_influence()

###ti externally studentized deleted residuals
b_ti_e = b_influences_e.resid_studentized_external
wave_e['ti']=b_ti_e
###t test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10
#t(1-(alpha/2n); n-p-1)
#cases greater denote outliers in Y; should be added to outlier list
t_bon_e = stats.t.ppf(1-(.10/(2*len(wave_e))), (len(wave_e)-9-1))
print("t value for test for Bonferroni simultaneous test procedure with a family significance level of
alpha = .10: ", t_bon_e)
bad_ti_e = wave_e[abs(wave_e['ti']) > t_bon_e]


#### leverage, from statsmodels internals
b_leverage_e = b_influences_e.hat_matrix_diag
wave_e['hii'] = b_leverage_e
#leverage that is greater than 3p/n denotes extreme outliers in X (I changed criteria from 2p/n to 3p/n
to avoid deleting too many data points)
hii_marker_e = 3*9/len(wave_e)
print("Hii cutoff point for declaring outliers in X: ", hii_marker_e)
bad_hii_e = wave_e[abs(wave_e['hii']) > hii_marker_e]


#### DFFITS, from statsmodels internals
###NOTE, due to large number of DFFITS greater than 2*sqrt(p/n), I changed the criteria of highly
influential for the model to 4*sqrt(p/n) to avoid
#getting rid of too many data points
b_DFFITS_e = b_influences_e.dffits
wave_e['DFFITS'] = b_DFFITS_e[0]
#DFFITS greater than 4*sqrt(p/n) denotes very high influence; for a small n, DFFITS greater than 1
denotes high influence
DFFITS_marker_e = 4*math.sqrt(9/len(wave_e))
print("DFFITS cutoff point for declaring highly influential points: ", DFFITS_marker_e)
```

```
bad_DFFITS_e = wave_e[abs(wave_e['DFFITS']) > DFFITS_marker_e]

# cook's distance, from statsmodels internals
b_cooks_e = b_influences_e.cooks_distance[0]
wave_e['Cooks distance'] = b_cooks_e
#Now we need the F statistic from F(p,n-p) distribution to see if the percentile is 50 percent or more.
Find 50th percentile and see if DFFITS value exceeds
F50_e = sp.stats.f.ppf(q=.5, dfn=9, dfd=(len(wave_e)-9))
print("F statistic from F(p,n-p) distribution at 50th percentile: ", F50_e)
bad_cooks_e = wave_e[abs(wave_e['Cooks distance']) > F50_e]


#TABLE OF OUTLIERS AND INFLUENCERS
oddcases_e = pd.concat([bad_ti_e, bad_hii_e, bad_DFFITS_e, bad_cooks_e])
#wave_e_g is "good" data that is left
wave_e_g = wave_e[~wave_e['Bib'].isin(oddcases_e['Bib'])]
print("The number of outliers and influential points deleted from the data is: ", len(wave_e)-
len(wave_e_g))

#WAVE One
#make new empty dataframe of outliers
oddcases_1 = pd.DataFrame()

#INFLUENCE OLS Influence results
b_influences_1 = w1_OLS.get_influence()
###ti externally studentized deleted residuals
b_ti_1 = b_influences_1.resid_studentized_external
wave_1['ti']=b_ti_1
###t test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10
#t(1-(alpha/2n); n-p-1)
#cases greater denote outliers in Y; should be added to outlier list
t_bon_1 = stats.t.ppf(1-(.10/(2*len(wave_1))), (len(wave_1)-9-1))
print("t value for test for Bonferroni sumultaneous test procedure with a family significance level of
alpha = .10: ", t_bon_1)
bad_ti_1 = wave_1[abs(wave_1['ti']) > t_bon_1]


#### leverage, from statsmodels internals
b_leverage_1 = b_influences_1.hat_matrix_diag
wave_1['hii'] = b_leverage_1
#leverage that is greater than 3p/n denotes extreme outliers in X (I changed criteria from 2p/n to 3p/n
to avoid deleting too many data points)
hii_marker_1 = 3*9/len(wave_1)
print("Hii cutoff point for declaring outliers in X: ", hii_marker_1)
bad_hii_1 = wave_1[abs(wave_1['hii']) > hii_marker_1]


#### DFFITS, from statsmodels internals
```

```
###NOTE, due to large number of DFFITS greater than 2*sqrt(p/n), I changed the criteria of highly influential for the model to 4*sqrt(p/n) to avoid
#getting rid of too many data points
b_DFFITS_1 = b_influences_1.dffits
wave_1['DFFITS'] = b_DFFITS_1[0]
#DFFITS greater than 4*sqrt(p/n) denotes very high influence; for a small n, DFFITS greater than 1 denotes high influence
DFFITS_marker_1 = 4*math.sqrt(9/len(wave_1))
print("DFFITS cutoff point for declaring highly influential points: ", DFFITS_marker_1)
bad_DFFITS_1 = wave_1[abs(wave_1['DFFITS']) > DFFITS_marker_1]

# cook's distance, from statsmodels internals
b_cooks_1 = b_influences_1.cooks_distance[0]
wave_1['Cooks distance'] = b_cooks_1
#Now we need the F statistic from F(p,n-p) distribution to see if the percentile is 50 percent or more. Find 50th percentile and see if DFFITS value exceeds
F50_1 = sp.stats.f.ppf(q=.5, dfn=9, dfd=(len(wave_1)-9))
print("F statistic from F(p,n-p) distribution at 50th percentile: ", F50_1)
bad_cooks_1 = wave_1[abs(wave_1['Cooks distance']) > F50_1]


#TABLE OF OUTLIERS AND INFLUENCERS
oddcases_1 = pd.concat([bad_ti_1, bad_hii_1, bad_DFFITS_1, bad_cooks_1])
#wave_e_g is "good" data that is left
wave_1_g = wave_1[~wave_1['Bib'].isin(oddcases_1['Bib'])]
print("The number of outliers and influential points deleted from the data is: ", len(wave_1)-len(wave_1_g))

#WAVE Two
#make new empty dataframe of outliers
oddcases_2 = pd.DataFrame()

#INFLUENCE OLS Influence results
b_influences_2 = w2_OLS.get_influence()
###ti externally studentized deleted residuals
b_ti_2 = b_influences_2.resid_studentized_external
wave_2['ti']=b_ti_2
###t test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10
#t(1-(alpha/2n); n-p-1)
#cases greater denote outliers in Y; should be added to outlier list
t_bon_2 = stats.t.ppf(1-(.10/(2*len(wave_2))), (len(wave_2)-9-1))
print("t value for test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10: ", t_bon_2)
bad_ti_2 = wave_2[abs(wave_2['ti']) > t_bon_2]


#### leverage, from statsmodels internals
b_leverage_2 = b_influences_2.hat_matrix_diag
```

```
wave_2['hii'] = b_leverage_2
#leverage that is greater than 3p/n denotes extreme outliers in X (I changed criteria from 2p/n to 3p/n
to avoid deleting too many data points)
hii_marker_2 = 3*9/len(wave_2)
print("Hii cutoff point for declaring outliers in X: ", hii_marker_2)
bad_hii_2 = wave_2[abs(wave_2['hii']) > hii_marker_2]


#### DFFITS, from statsmodels internals
###NOTE, due to large number of DFFITS greater than 2*sqrt(p/n), I changed the criteria of highly
influential for the model to 4*sqrt(p/n) to avoid
#getting rid of too many data points
b_DFFITS_2 = b_influences_2.dffits
wave_2['DFFITS'] = b_DFFITS_2[0]
#DFFITS greater than 4*sqrt(p/n) denotes very high influence; for a small n, DFFITS greater than 1
denotes high influence
DFFITS_marker_2 = 4*math.sqrt(9/len(wave_2))
print("DFFITS cutoff point for declaring highly influential points: ", DFFITS_marker_2)
bad_DFFITS_2 = wave_2[abs(wave_2['DFFITS']) > DFFITS_marker_2]

# cook's distance, from statsmodels internals
b_cooks_2 = b_influences_2.cooks_distance[0]
wave_2['Cooks distance'] = b_cooks_2
#Now we need the F statistic from F(p,n-p) distribution to see if the percentile is 50 percent or more.
Find 50th percentile and see if DFFITS value exceeds
F50_2 = sp.stats.f.ppf(q=.5, dfn=9, dfd=(len(wave_2)-9))
print("F statistic from F(p,n-p) distribution at 50th percentile: ", F50_2)
bad_cooks_2 = wave_2[abs(wave_2['Cooks distance']) > F50_2]


#TABLE OF OUTLIERS AND INFLUENCERS
oddcases_2 = pd.concat([bad_ti_2, bad_hii_2, bad_DFFITS_2, bad_cooks_2])
#wave_e_g is "good" data that is left
wave_2_g = wave_2[~wave_2['Bib'].isin(oddcases_2['Bib'])]
print("The number of outliers and influential points deleted from the data is: ", len(wave_2)-
len(wave_2_g))

#WAVE Three
#make new empty dataframe of outliers
oddcases_3 = pd.DataFrame()

#INFLUENCE OLS Influence results
b_influences_3 = w3_OLS.get_influence()
###ti externally studentized deleted residuals
b_ti_3 = b_influences_3.resid_studentized_external
wave_3['ti']=b_ti_3
###t test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10
#t(1-(alpha/2n); n-p-1)
```

```
#cases greater denote outliers in Y; should be added to outlier list
t_bon_3 = stats.t.ppf(1-(.10/(2*len(wave_3))), (len(wave_3)-9-1))
print("t value for test for Bonferroni sumultaneous test procedure with a family significance level of
alpha = .10: ", t_bon_3)
bad_ti_3 = wave_3[abs(wave_3['ti']) > t_bon_3]


#### leverage, from statsmodels internals
b_leverage_3 = b_influences_3.hat_matrix_diag
wave_3['hii'] = b_leverage_3
#leverage that is greater than 3p/n denotes extreme outliers in X (I changed criteria from 2p/n to 3p/n
to avoid deleting too many data points)
hii_marker_3 = 3*9/len(wave_3)
print("Hii cutoff point for declaring outliers in X: ", hii_marker_3)
bad_hii_3 = wave_3[abs(wave_3['hii']) > hii_marker_3]


#### DFFITS, from statsmodels internals
###NOTE, due to large number of DFFITS greater than 2*sqrt(p/n), I changed the criteria of highly
influential for the model to 4*sqrt(p/n) to avoid
#getting rid of too many data points
b_DFFITS_3 = b_influences_3.dffits
wave_3['DFFITS'] = b_DFFITS_3[0]
#DFFITS greater than 4*sqrt(p/n) denotes very high influence; for a small n, DFFITS greater than 1
denotes high influence
DFFITS_marker_3 = 4*math.sqrt(9/len(wave_3))
print("DFFITS cutoff point for declaring highly influential points: ", DFFITS_marker_3)
bad_DFFITS_3 = wave_3[abs(wave_3['DFFITS']) > DFFITS_marker_3]

# cook's distance, from statsmodels internals
b_cooks_3 = b_influences_3.cooks_distance[0]
wave_3['Cooks distance'] = b_cooks_3
#Now we need the F statistic from F(p,n-p) distribution to see if the percentile is 50 percent or more.
Find 50th percentile and see if DFFITS value exceeds
F50_3 = sp.stats.f.ppf(q=.5, dfn=9, dfd=(len(wave_3)-9))
print("F statistic from F(p,n-p) distribution at 50th percentile: ", F50_3)
bad_cooks_3 = wave_3[abs(wave_3['Cooks distance']) > F50_3]


#TABLE OF OUTLIERS AND INFLUENCERS
oddcases_3 = pd.concat([bad_ti_3, bad_hii_3, bad_DFFITS_3, bad_cooks_3])
#wave_e_g is "good" data that is left
wave_3_g = wave_3[~wave_3['Bib'].isin(oddcases_3['Bib'])]
print("The number of outliers and influential points deleted from the data is: ", len(wave_3)-
len(wave_3_g))

#WAVE Four
#make new empty dataframe of outliers
```

```
oddcases_4 = pd.DataFrame()

#INFLUENCE OLS Influence results
b_influences_4 = w4_OLS.get_influence()
###ti externally studentized deleted residuals
b_ti_4 = b_influences_4.resid_studentized_external
wave_4['ti']=b_ti_4
###t test for Bonferroni sumultaneous test procedure with a family significance level of alpha = .10
#t(1-(alpha/2n); n-p-1)
#cases greater denote outliers in Y; should be added to outlier list
t_bon_4 = stats.t.ppf(1-(.10/(2*len(wave_4))), (len(wave_4)-9-1))
print("t value for test for Bonferroni sumultaneous test procedure with a family significance level of
alpha = .10: ", t_bon_4)
bad_ti_4 = wave_4[abs(wave_4['ti']) > t_bon_4]


#### leverage, from statsmodels internals
b_leverage_4 = b_influences_4.hat_matrix_diag
wave_4['hii'] = b_leverage_4
#leverage that is greater than 3p/n denotes extreme outliers in X (I changed criteria from 2p/n to 3p/n
to avoid deleting too many data points)
hii_marker_4 = 3*9/len(wave_4)
print("Hii cutoff point for declaring outliers in X: ", hii_marker_4)
bad_hii_4 = wave_4[abs(wave_4['hii']) > hii_marker_4]


#### DFFITS, from statsmodels internals
###NOTE, due to large number of DFFITS greater than 2*sqrt(p/n), I changed the criteria of highly
influential for the model to 4*sqrt(p/n) to avoid
#getting rid of too many data points
b_DFFITS_4 = b_influences_4.dffits
wave_4['DFFITS'] = b_DFFITS_4[0]
#DFFITS greater than 4*sqrt(p/n) denotes very high influence; for a small n, DFFITS greater than 1
denotes high influence
DFFITS_marker_4 = 4*math.sqrt(9/len(wave_4))
print("DFFITS cutoff point for declaring highly influential points: ", DFFITS_marker_4)
bad_DFFITS_4 = wave_4[abs(wave_4['DFFITS']) > DFFITS_marker_4]

# cook's distance, from statsmodels internals
b_cooks_4 = b_influences_4.cooks_distance[0]
wave_4['Cooks distance'] = b_cooks_4
#Now we need the F statistic from F(p,n-p) distribution to see if the percentile is 50 percent or more.
Find 50th percentile and see if DFFITS value exceeds
F50_4 = sp.stats.f.ppf(q=.5, dfn=9, dfd=(len(wave_4)-9))
print("F statistic from F(p,n-p) distribution at 50th percentile: ", F50_4)
bad_cooks_4 = wave_4[abs(wave_4['Cooks distance']) > F50_4]
```

```
#TABLE OF OUTLIERS AND INFLUENCERS
oddcases_4 = pd.concat([bad_ti_4, bad_hii_4, bad_DFFITS_4, bad_cooks_4])
#wave_e_g is "good" data that is left
wave_4_g = wave_4[~wave_4['Bib'].isin(oddcases_4['Bib'])]
print("The number of outliers and influential points deleted from the data is: ", len(wave_4)-
len(wave_4_g))

wave_e_g= shuffle(wave_e_g, random_state = 3)
wave_e_g.to_csv('wave_e_g.csv')
files.download('wave_e_g.csv')

wave_e_B= shuffle(wave_e_B, random_state = 3)
wave_e_B.to_csv('wave_e_B.csv')
files.download('wave_e_B.csv')

wave_1_g= shuffle(wave_1_g, random_state = 3)
wave_1_g.to_csv('wave_1_g.csv')
files.download('wave_1_g.csv')

wave_1_B= shuffle(wave_1_B, random_state = 3)
wave_1_B.to_csv('wave_1_B.csv')
files.download('wave_1_B.csv')

wave_2_g= shuffle(wave_2_g, random_state = 3)
wave_2_g.to_csv('wave_2_g.csv')
files.download('wave_2_g.csv')

wave_2_B= shuffle(wave_2_B, random_state = 3)
wave_2_B.to_csv('wave_2_B.csv')
files.download('wave_2_B.csv')

wave_3_g= shuffle(wave_3_g, random_state = 3)
wave_3_g.to_csv('wave_3_g.csv')
files.download('wave_3_g.csv')

wave_3_B= shuffle(wave_3_B, random_state = 3)
wave_3_B.to_csv('wave_3_B.csv')
files.download('wave_3_B.csv')

wave_4_g= shuffle(wave_4_g, random_state = 3)
wave_4_g.to_csv('wave_4_g.csv')
files.download('wave_4_g.csv')

wave_4_B= shuffle(wave_4_B, random_state = 3)
wave_4_B.to_csv('wave_4_B.csv')
files.download('wave_4_B.csv')

#Plotting regression plots with confidence interval bands
```

```
#MODEL 1- general
X = boston_data_A_g[['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender', 'X5, Wave: Elite', 'X6, Wave: One', 'X7, Wave: Two', 'X8, Wave: Three']]
y = boston_data_A_g['Y, Official Time']
## fit a OLS
X = sm.add_constant(X)
m1fit = sm.OLS(y, X).fit()
m1fit.summary()

#MODEL 2- wave elite
X2 = wave_e_g[['X2, 5K Pace', 'X3, Half Pace']]
y2 = wave_e_g['Y, Official Time']
## fit a OLS
X2 = sm.add_constant(X2)
m2fit = sm.OLS(y2, X2).fit()
m2fit.summary()

#Plotting regression plots with confidence interval bands
#MODEL 3- wave 1
X = wave_1_g[['X1, Age', 'X2, 5K Pace', 'X3, Half Pace', 'X4, Gender']]
y = wave_1_g['Y, Official Time']
## fit a OLS
X = sm.add_constant(X)
m1fit = sm.OLS(y, X).fit()
m1fit.summary()

#Plotting regression plots with confidence interval bands
#MODEL 4- wave 2
X = wave_2_g[['X2, 5K Pace', 'X3, Half Pace', 'X4, Gender']]
y = wave_2_g['Y, Official Time']
## fit a OLS
X = sm.add_constant(X)
m1fit = sm.OLS(y, X).fit()
m1fit.summary()

#Plotting regression plots with confidence interval bands
#MODEL 5- wave 3
X = wave_3_g[['X2, 5K Pace', 'X3, Half Pace', 'X4, Gender']]
y = wave_3_g['Y, Official Time']
## fit a OLS
X = sm.add_constant(X)
m1fit = sm.OLS(y, X).fit()
m1fit.summary()

#Plotting regression plots with confidence interval bands
#MODEL 6- wave 4
X = wave_4_g[['X2, 5K Pace', 'X3, Half Pace', 'X4, Gender']]
y = wave_4_g['Y, Official Time']
```

```
## fit a OLS
X = sm.add_constant(X)
m1fit = sm.OLS(y, X).fit()
m1fit.summary()

#influence plot
fig3, ax = plt.subplots(figsize=(18,16))
fig3 = sm.graphics.influence_plot(boston_fit_A, ax=ax)

#residuals against X2

resplotvx4 = sns.residplot(boston_data_A['X2, 5K Pace'], boston_data_A['ABS RES'], lowess=True,
color="m")
resplotvx4.axes.set_title('Absolute Residuals against X2')
resplotvx4.axes.set_xlabel('X2, 5K Pace')
resplotvx4.axes.set_ylabel('Absolute Residuals')

#residuals against X3

resplotvx4 = sns.residplot(boston_data_A['X3, Half Pace'], boston_data_A['ABS RES'], lowess=True,
color="slateblue")
resplotvx4.axes.set_title('Absolute Residuals against X3')
resplotvx4.axes.set_xlabel('X3, Half Pace')
resplotvx4.axes.set_ylabel('Absolute Residuals')
```