ETH zürich

Realtime Election Result Forecast and FX Market Reaction

Author: Alex Huang

Supervisors: Ke Wu, Prof. Didier Sornette

Master Thesis

Chair of Entrepreneurial Risk

Department of Management, Technology and Economics

ETH Zürich

02/2018

Abstract

Currency exchange value fluctuates wildly during election time as different leaders have vastly different visions for the country and will lead the country down a different path. This master thesis investigates the FX market reaction during the night of US 2016 election and UK 2017 election. As the result of each state (US) or constituency (UK) is available, the likelihood of each outcome of the election is updated due to the new piece of information. An efficient market would adjust the currency exchange values according to the likelihood of the outcomes.

The main goal of this thesis is to build a predictive model of the election result in real-time during the recent US and UK election. The model should give a probability of each outcome at a given time between counting of votes begins and the full result is available. It uses polling data for each state or constituency as prior data and available results at each time t to predict the results in states or constituencies that are still unknown. When each piece of new information is available, a linear regression is performed on the available result against the polling data. Other demographic data is also used to improve prediction results. Using the available results and the model prediction, the probability of each outcome can be calculated at any time t during election night.

If the market is reacting efficiently, the currency exchange value of the relevant pairs of currency should move in sync with the true probability of the potential outcomes. For example, in the US 2016 election, if the probability of Donald Trump winning is high, the value of the Mexican peso against the US dollar should be low, otherwise arbitrage would exist and one can beat the market.

We observed that US market was working efficiently, the price movement reflected the calculated probability of possible outcomes. However, for the UK election, the result can be calculated much earlier on than the pricing of the British pound suggests, leading to arbitrage in the market.

Acknowledgement

I would like to extend my gratitude to my thesis supervisor Mr. Ke Wu for giving me the idea to do this thesis project and worked with me each step of the way from obtaining data, implementing analysis and analysing the result. From my work with him, I have expanded my knowledge in data analysis and finance. We met frequently to discuss results and his input has help me to correct mistakes and improve my model and results.

I would also like to thank Professor Didier Sornette for setting up this master thesis. His course Financial Market Risk at ETH has given me the key foundation knowledge in finance and statistics to work on this project.

Contents

Introduction

During the night of the 2016 US and 2017 UK elections, the currency exchange market fluctuated wildly as election results of each state or constituency were reported. The outcome of the elections will have a big impact on the future of the countries' policies, economics and politics, thus changing the value of their currencies. This paper will focus on the 2016 US election and 2017 UK election. The US election resulted in the victory of Donald Trump, and Mexican peso crashed against the US dollar. The UK election resulted in the victory of Theresa May and her conservative party, however she failed to secure majority in the parliament. The British pound crashed against the US dollar when the result became clear.

2016 US Election Recap:

The 45th US presidential election took place on the 8th of November 2016; the two candidates Hillary Clinton and Donald Trump fought hard against each other during long campaigns that took place many months before the voting took place. They each offered a different vision for the country in the next four years.

Democratic party nominee Hillary Clinton wanted to continue the path of her predecessor Barack Obama on important issues such as trade, immigration, and foreign policies. She would uphold the US commitment to the current world order and maintain US influence around the world. She appealed to voters by promoting her experience in politics, political correctness and that a female president would be a significant milestone in gender equality.

Republican party nominee Donald Trump saw the country going in a different way; he wanted to focus on the country's internal problems he perceived, primarily immigration and trade imbalances. He proposed ideas such as a building a wall on the US Mexico border to stop illegal immigrants and drug trafficking. He appealed to voters by promoting himself as a successful businessman and willing to speak about and tackle difficult problems such as job loss due to outsourcing and illegal immigration.

Both candidates were embroiled in controversies. Hillary Clinton had an E-mail scandal where she used a private E-mail server to handle sensitive work information. Donald Trump was suspected to be the benefactor of alleged Russian influence of the US election. Neither candidates were well liked, they were often mentioned as the most disliked candidates ever in the media [1].

Polls near the election day consistently suggested that Hillary Clinton will win the presidency with high probability, but the race was close. To many people's surprise, at the end, Donald Trump emerged victorious winning 309 electoral votes and became the 45th president of the United States of America. The Mexican peso crashed from buying $\frac{1}{18}$ US dollar before election night to $\frac{1}{20}$ US dollar after election night.

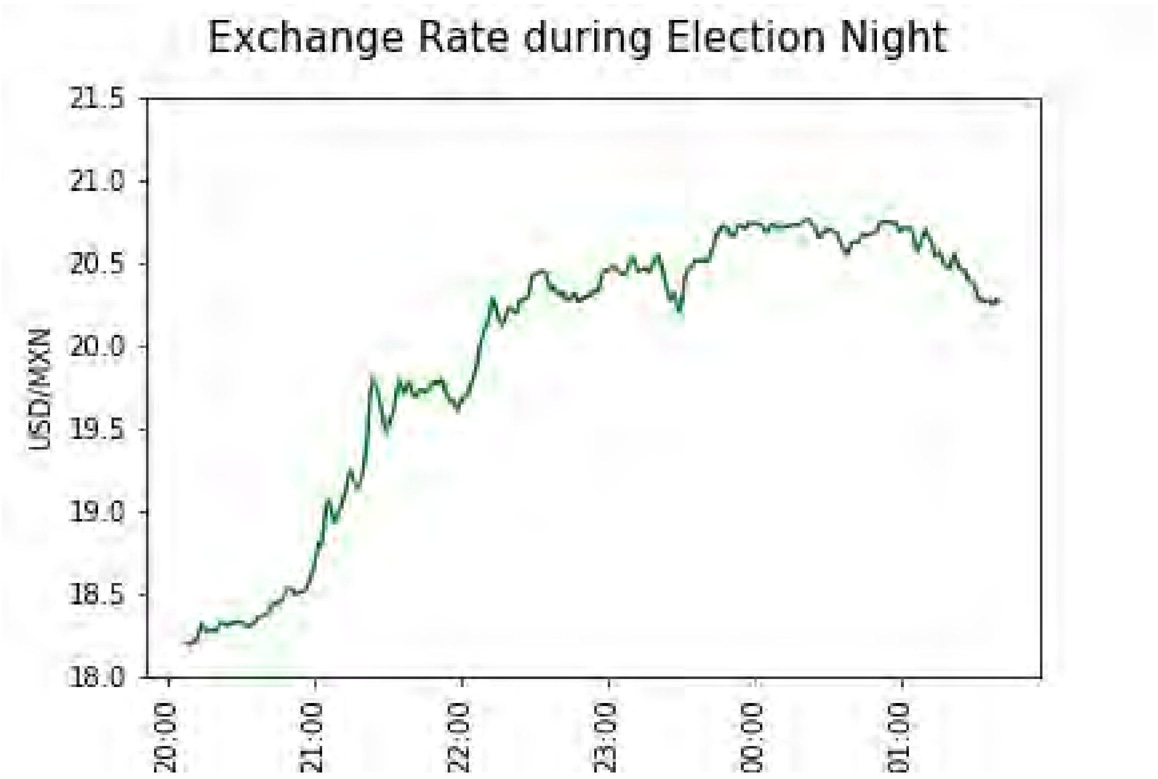


Figure 1. USD/MXN fluctuation during election night

2017 UK Election Recap:

British prime minister Theresa May called a snap election on the 18th of April 2017 in hopes to strengthen her position in the parliament and her position in the upcoming Brexit negotiation with the European Union. On 23rd of June 2016, the people of the United Kingdom voted in a referendum with a slim margin to leave the European Union. On 30th of March 2017, May's government triggered Article 50, the mechanism to end Britain's EU membership. May inherited the majority government from former prime minister David Cameron, however the margin of majority is slim. The snap election is how May hoped to achieve a stronger majority, allowing her a stronger position in the Brexit negotiation. May's main rival was the Labour Party lead by Jeremy Corbyn, who is on the left on the political spectrum and was looking to gain more seats after a poor performance in the 2015 UK general election.

The Conservative Party:

The Conservative Party of the UK campaigned to focus their effort on Brexit and lowering taxes. It's leader Theresa May envisioned UK to get a good deal during Brexit negotiation and lowering taxes to attract more business post Brexit. Theresa May did not participate in the national debate that took place in 31st of May 2017, and was criticised by rivals and media. [2]

The Labour Party:

The Labour Party of the UK was led by Jeremy Corbyn. During the campaign, he focused on issues such as reversing the cut to capital gain tax and increasing spending for public services. The Labour Party accepted the result of the Brexit referendum, but would focus negotiation on different priorities such as jobs and safeguarding British industries.

Other Parties:

Other major parties include the Liberal Party and the Scottish National Party. The Liberal Party is strongly pro-EU and campaigned for 'soft' Brexit or a second referendum. The Scottish National Party campaigned actively only in Scotland for a referendum on Scottish independence. Both parties have much lower support compared to the Conservatives or the Labour party.

The election resulted in a Conservative victory, Theresa May's party secured 317 seats, down from the 330 secured in the 2015 election. However, it did not secure the 326 seats needed for a majority in the parliament, resulting in a hung parliament. The biggest winner of the snap election was the Labour party, winning 262 seats, up 30 seats from 2015 election.

The British pound crashed as a result of the election, as Theresa May did not secure a stronger position in the upcoming Brexit negotiation, rather her position weakened. The pound crashed from 1.30 USD before voting took place to 1.26 USD after voting took place.

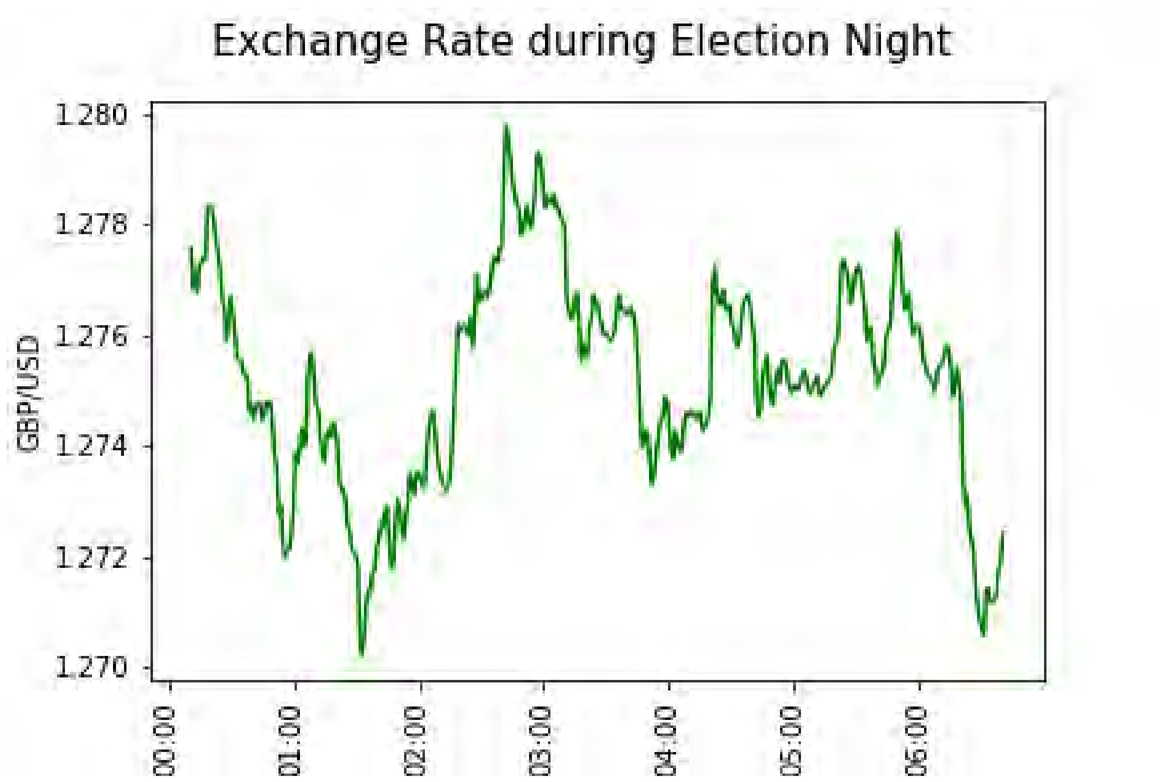


Figure 2. GBP/USD During 2017 Election Night

In this paper we will investigate whether the currency exchange rates during election nights when only partial results were available were consistent with the probability of the outcome of the election. We will discuss the efficient market hypothesis, existing election forecasts, pre-election bias, our model of predicting probabilities of outcomes and the consistency of currency exchange rates with election results.

Implied Probabilities and Expected Value

At any point in time during the election, we can determine the implied probabilities of each outcome, if we know the final values of the currency exchange for each possible outcome and the value of the currency exchange value at the current time. Conversely, we can determine the fair price (expected value) of the currency exchange value if we know the probability of each outcome and the resulting currency exchange value for each outcome. The following equations must hold when an asset is considered fairly priced.

$$v_t = \sum_{i=1}^{n} p_i \, v_i \quad (1)$$

$$\sum_{i=1}^{n} p_i = 1$$

Where v_t is the value of a currency time t. p_i is the probability of outcome i. v_i is the value of the currency if outcome i occurs.

Example:

Let us consider a hypothetical example. At time t, Donald Trump has a 20% probability of winning the election, and 80% probability of losing the election. If Trump wins, the US dollar will have a value of 20 Mexican pesos, and if he loses, the US dollar will have a value of 15 Mexican pesos. We can calculate the fair price (expected value) of the US dollar at time t as the following:

$$v_t = 80\% * 15 + 20\% * 20 = 16$$

Conversely, we can determine the implied probability of each outcome with the current price of the asset, and the price of the asset at each outcome. Continuing the example above, if we do not know the probability but we know the price of US dollar at time t' is 18 Mexican pesos, we can determine the probability of Trump winning as the following:

$$v_{t'} = (1 - p) * 15 + p * 20 = 18$$

$$p = 60\%$$

With the pricing data, we can determine what the market thinks the probability of each outcome is. Or if we can calculate the probability of each outcome, we can determine whether the current price is fair.

Efficient Market Hypothesis

The Efficient-Market Hypothesis (EMH) is a financial theory which states that prices of assets fully reflect all available information. There is no arbitrage available for anyone to exploit. In our case of currency exchange, the exchange rate at any time is always the fair price and reflect all the available information. If there is a discrepancy between the fair price of the asset and the actual price of the asset, traders would trade immediately and capitalise on the difference to drive the asset price to the fair price. There are three forms of Efficient Market Hypothesis discussed below.

Weak Form of Efficiency

The weak form of Efficient Market Hypotheses states that the future return of an asset cannot be predicted by analysing the price of the past. For example, we cannot look at the price movement of a stock in the past to predict the future performance. There is no pattern in the price of an asset. And future price movement is determined entirely on future information that is not contained in the past price movement. Technical analysis (looking at past price movement) of an asset price does not yield any excess return, i.e. beat the market.

Semi-Strong Form of Efficiency

The semi strong form of Efficient Market Hypothesis states that in addition to the weak form of efficiency, the price of an asset contains all public information available. Using the stock example again, this would mean that the price of the stock reflects the past prices and all the fundamental information regarding the company such as financial performance, information on CEO and etc. Fundamental analysis of an asset does not yield any excess return.

Strong Form Efficiency

The strong form of Efficient Market Hypothesis states that in addition to the semi strong form of efficiency, the price of asset contains all private information. Even trading with private information (insider trading, illegal in most countries) cannot yield any excess return.

In this thesis, we will examine particularly the semi strong form of the Efficient Market Hypothesis. If the semi strong form of efficiency holds, the price of a currency should reflect all past price movement information and publicly available information.

If we can use publicly available information to determine the probability of each outcome (using our predictive model discussed in part 4 and 5) and we can determine the value of the currency given an outcome (using forecasts), we can calculate the fair price of the currency using the equation (1). If there is any deviation of the currency price to the fair price, traders should be able to capitalise on this opportunity to drive the asset price to the fair price.

Literature Review – Existing Election Prediction

Pre-election Forecast:

A number of forecasters made pre-election forecasts, which were based on polls of candidates and previous election results for both the UK and the US. However, most of their results in these two elections (and the Brexit referendum case) were quite far off compared to the actual results. This was later largely attributed to the population that did not vote in previous elections and did not participate in polls turning out to vote for the Republicans in the US and Conservatives in the UK. Here we have the result of the US and UK pre-election forecasts vs the real result.

US Election:

Princeton Consortium

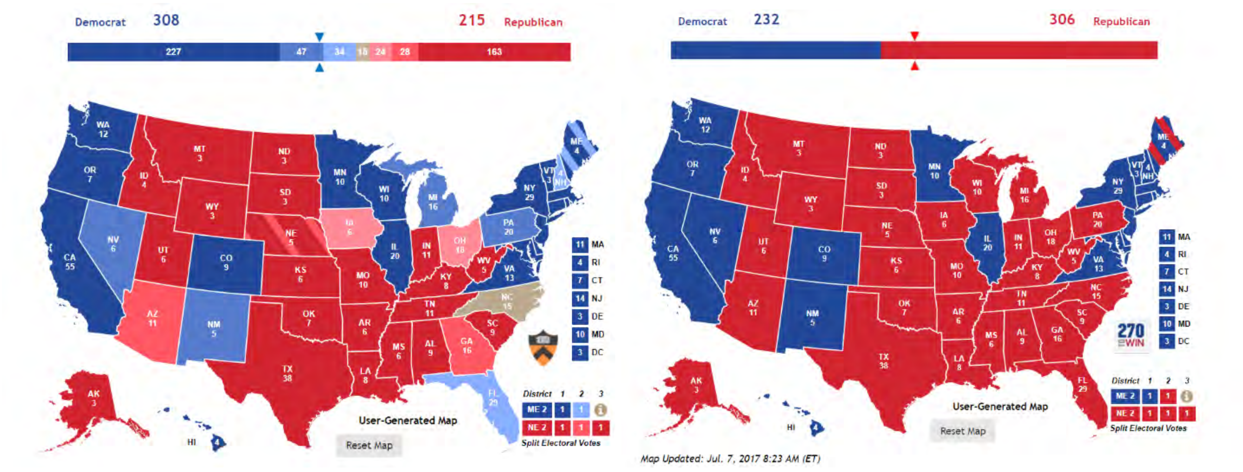


Figure 3: US election, prediction (left) vs actual result (right).

The Princeton Consortium, a forecasting body consisting of Princeton academic members, predicted Hillary Clinton will win the election by a wide margin as shown in the graph above on the left.

Their process of estimating the election result is the following:

1. Calculate the median and standard error of mean using the last three or four polls for each state.
2. The median and standard error of mean is converted to a z-score and then a probability of winning using a t-distribution is calculated.
3. Simulate each state to obtain the probability of all possible outcome. [3]

They acknowledged that if there are consistence biases in the polls, their results will be quite inaccurate. The polls were indeed biased in the 2016 election and the prediction was far off.

Huffington Post

Huffington Post's model predicted that Trump has 1.7 percent chance of winning the election and Clinton 98 percent of winning the election. In their research, Hillary Clinton won 9.8 million times out of 10 million simulations.

Their methodology is the following:

1. Simulate population according to the polling results. The undecided is split up into three equally sized groups: one third would not vote, one third swings to either candidate, one third is added to the uncertainty in their model.
2. Simulate result for each state based on:

   State-by-state averages calculated in the previous step
   The uncertainty in the average of national polls
   The way one-third of undecided voters may vote

   The model also takes into account state by state correlation during the simulation by examining the correlation between states in the election results from 1932. [4]

Traditionally, US polls are quite accurate with low margin of error. However, in the 2016 election, the polls consistently underestimated the support for Donald Trump. One likely reason is that there was a strong overlap between people that did not participate in the polls and people that voted for Donald Trump. The Pew Research Center proposed that 'It is possible that the frustration and anti-institutional feelings that drove the Trump campaign may also have aligned with an unwillingness to respond to polls. The result would be a strongly pro-Trump segment of the population that simply did not show up in the polls in proportion to their actual share of the population.' [5]

UK Election

UK polls have not been very accurate historically, also this time the polls missed the actual result However they did not miss more than they did previously. The average polls at the end of the campaign showed conservative winning by 6.4 points whereas at the end they won by 2 to 3. The polls missed by about 4 points which was quite consistent with past errors. YouGov, a polling agency and election forecaster, projected a hung parliament right before the election took place, however, it was dismissed as inaccurate.

While most polls did underestimate the support Labour had, the difference was not much more than the errors in the past. What was surprising is that on the betting market, people treated the polls to have a much smaller margin of error than they actually had and assumed that Conservative will certainly win majority. One possible explanation of this bias is the groupthink mentality. The Conservative started with a much stronger position than Labour when the snap election was announced, however, the support for Conservative quickly eroded after as the campaigns continued on. It was unthinkable that Conservative would lose their majority given the strong support they initially had and the mindset carried on even when new data was available. Most media disregarded the polls that showed strong Labour support as outliers. [6]

Chris Hanretty, an experienced election forecaster, predicted that the Conservative will win a majority with a major margin. Below is his prediction of constituencies won by each party and total seats for each party:

Party	Range (95% confidence Interval)
Conservative	395 - 468
Labour	110 – 206
Liberal	3 - 20

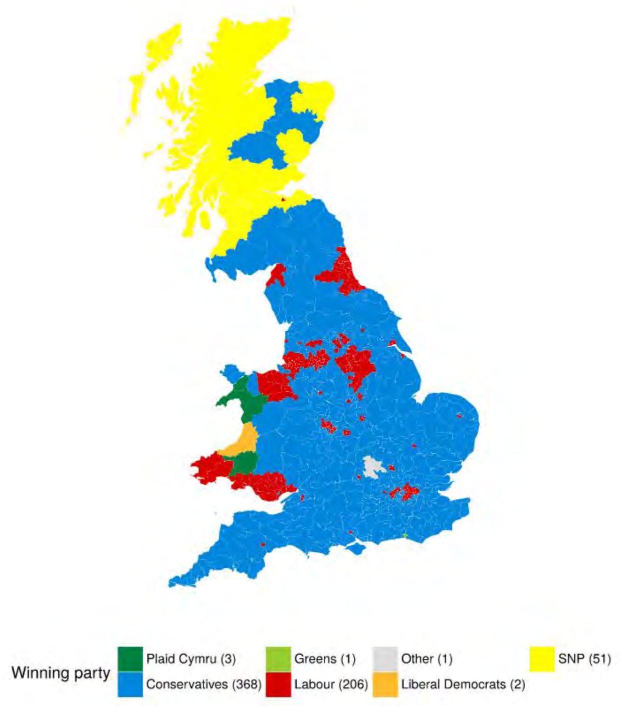


Figure 4: Hanretty UK 2017 election forecast

He noted that a large confidence interval is a design feature rather than a bug. However, even with a large confidence interval, the true result lied far away.

His model is based on information about past election results, current and historic national polling, individual polling, and information about constituencies.

The past election results were used to calibrate the prediction and eliminate some unrealistic predictions, pooled polling of individual constituencies was used to predict the results of the upcoming election, historical aggregate polling was used to predict and correct the systematic bias in the polls. [7]

Pre-election forecasts for both the UK and US elections have been wrong in these two instances due to an unprecedented candidate in the US election and a quick changing of support in the UK election. Both effects were difficult to predict in advance as they are relatively special cases. But what about real-time forecast during the result announcement period? When the results of some constituencies or states are available, can we find the systematic bias to accurately predict the true final result?

Real-time Forecast

Real-time forecast tries to predict the result of an election when part of the result is available. For example, in the US election, states published their results anytime from 8PM eastern time to the morning next day. Trump's victory was announced close to 2:.00AM EST by news media. From the time of the result first coming in, till the time that the result is certain, the likelihood of each candidate winning can be predicted by real-time forecast. Realtime forecast is the focus of this paper.

US election

FiveThirtyEight

FiveThirtyEight, an election forecast agency, had a real-time election forecast for the 2016 US election. They used a very simple model with only three factors, they are:

1. Their pre-election forecasts, largely based on polls.
2. States that are "called" for a candidate by their partners at ABC News.
3. The amount of time that has passed since the polls closed in a state, if it had not been called yet.

Why the first two factors are relevant for election prediction is self-explanatory. The third factor is important because if a long time has passed since the polls were closed without the state being called, one can infer that the race in that state is much closer than the others.

FiveThirtyEight did not use the use votes counted so far, margin of victory, or exit polls. This was quite a simple model, and we can see that they projected Clinton had a higher chance of victory at the beginning until 11PM EST and there was a high probability of Trump's victory after. [8]

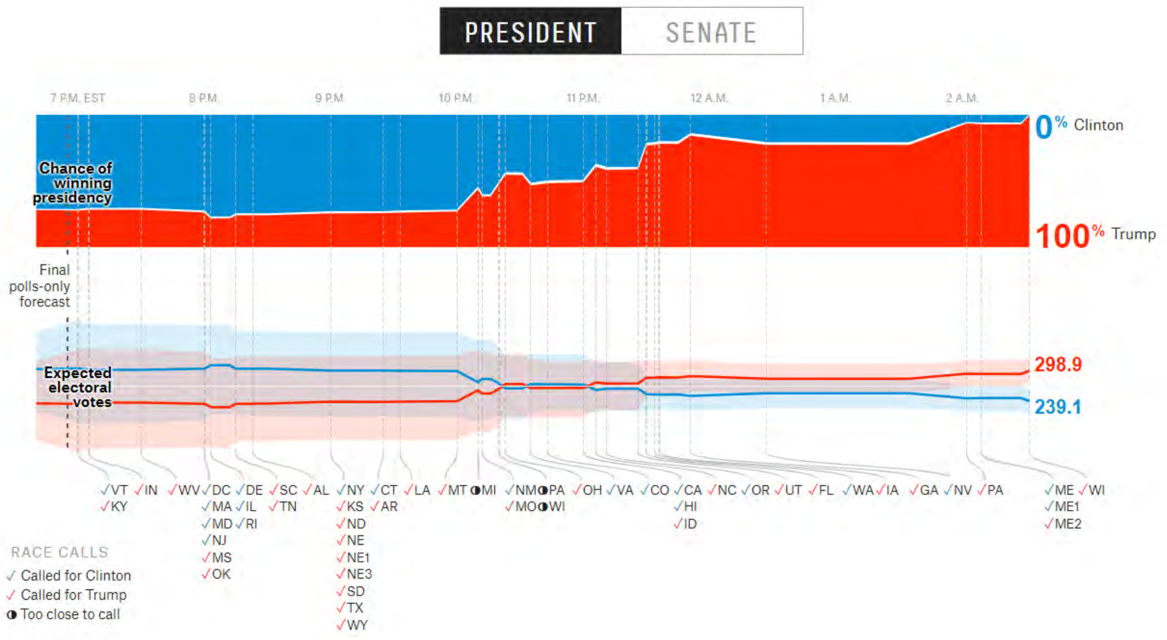


Figure 5: FiveThirtyEight election prediction

New York Times

The New York Times also made a real time forecast; the result was similar to the forecast completed by FiveThirtyEight, shown below in Figure 6. However, their methodology is not mentioned on their website.

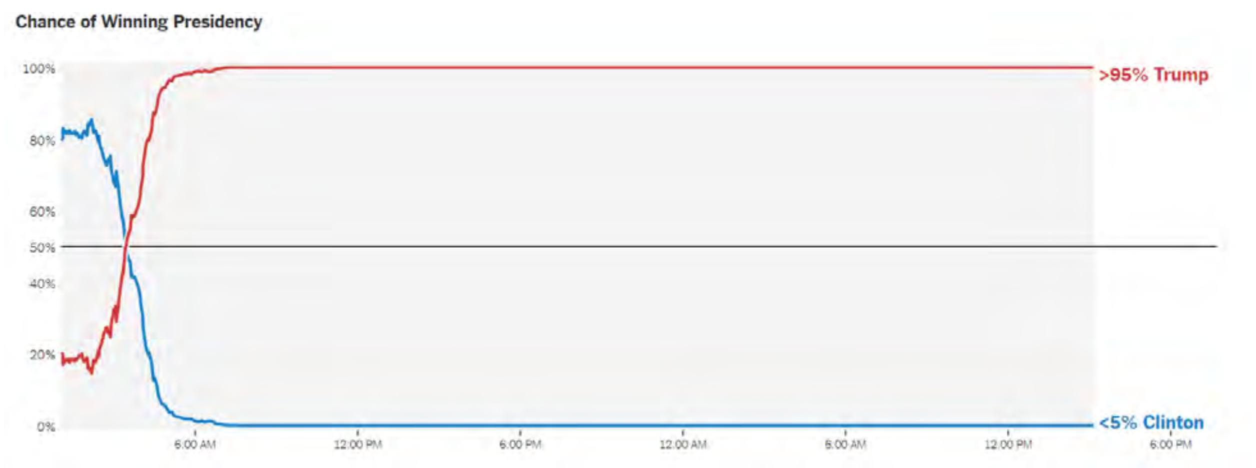
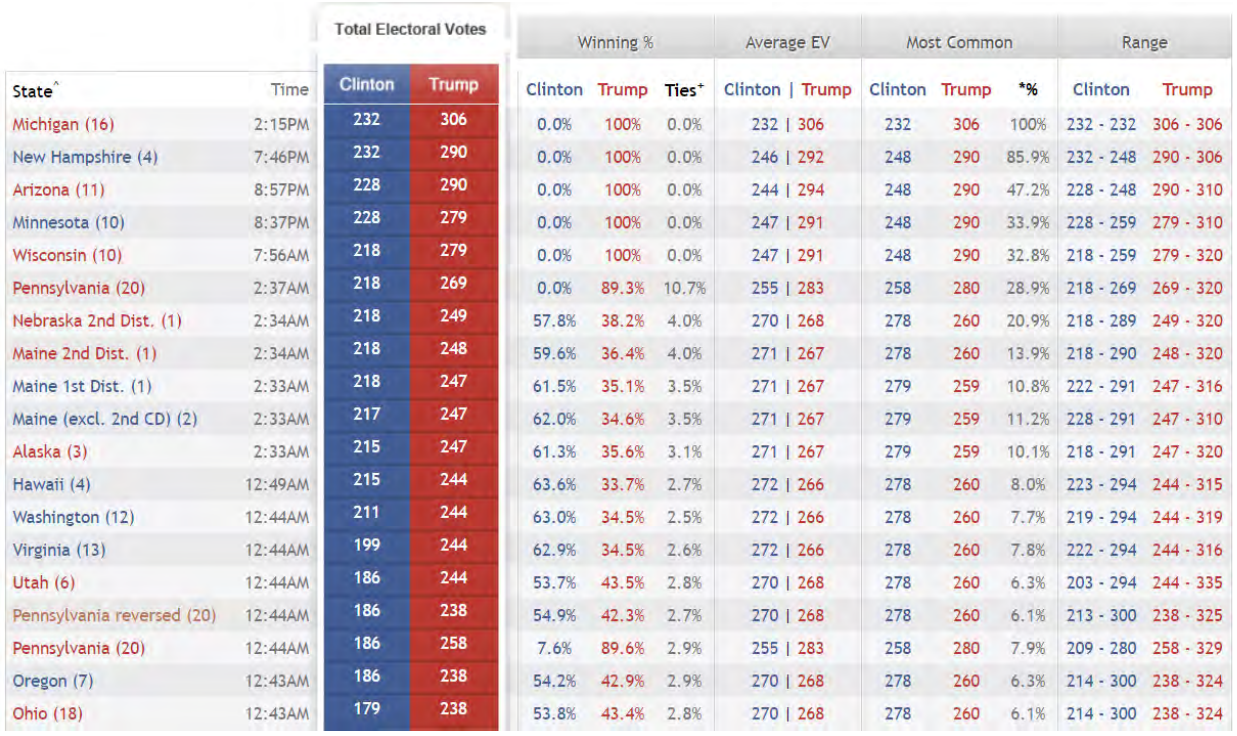


Figure 6: New York Times US election real-time prediction [9]

270TOWIN



		Total Electoral Votes		Winning %			Average EV	Most Common			Range	
State^	Time	Clinton	Trump	Clinton	Trump	Ties+	Clinton | Trump	Clinton	Trump	*%	Clinton	Trump
Michigan (16)	2:15PM	232	306	0.0%	100%	0.0%	232 | 306	232	306	100%	232 - 232	306 - 306
New Hampshire (4)	7:46PM	232	290	0.0%	100%	0.0%	246 | 292	248	290	85.9%	232 - 248	290 - 306
Arizona (11)	8:57PM	228	290	0.0%	100%	0.0%	244 | 294	248	290	47.2%	228 - 248	290 - 310
Minnesota (10)	8:37PM	228	279	0.0%	100%	0.0%	247 | 291	248	290	33.9%	228 - 259	279 - 310
Wisconsin (10)	7:56AM	218	279	0.0%	100%	0.0%	247 | 291	248	290	32.8%	218 - 259	279 - 320
Pennsylvania (20)	2:37AM	218	269	0.0%	89.3%	10.7%	255 | 283	258	280	28.9%	218 - 269	269 - 320
Nebraska 2nd Dist. (1)	2:34AM	218	249	57.8%	38.2%	4.0%	270 | 268	278	260	20.9%	218 - 289	249 - 320
Maine 2nd Dist. (1)	2:34AM	218	248	59.6%	36.4%	4.0%	271 | 267	278	260	13.9%	218 - 290	248 - 320
Maine 1st Dist. (1)	2:33AM	218	247	61.5%	35.1%	3.5%	271 | 267	279	259	10.8%	222 - 291	247 - 316
Maine (excl. 2nd CD) (2)	2:33AM	217	247	62.0%	34.6%	3.5%	271 | 267	279	259	11.2%	228 - 291	247 - 310
Alaska (3)	2:33AM	215	247	61.3%	35.6%	3.1%	271 | 267	279	259	10.1%	218 - 291	247 - 320
Hawaii (4)	12:49AM	215	244	63.6%	33.7%	2.7%	272 | 266	278	260	8.0%	223 - 294	244 - 315
Washington (12)	12:44AM	211	244	63.0%	34.5%	2.5%	272 | 266	278	260	7.7%	219 - 294	244 - 319
Virginia (13)	12:44AM	199	244	62.9%	34.5%	2.6%	272 | 266	278	260	7.8%	222 - 294	244 - 316
Utah (6)	12:44AM	186	244	53.7%	43.5%	2.8%	270 | 268	278	260	6.3%	203 - 294	244 - 335
Pennsylvania reversed (20)	12:44AM	186	238	54.9%	42.3%	2.7%	270 | 268	278	260	6.1%	213 - 300	238 - 325
Pennsylvania (20)	12:44AM	186	258	7.6%	89.6%	2.9%	255 | 283	258	280	7.9%	209 - 280	258 - 329
Oregon (7)	12:43AM	186	238	54.2%	42.9%	2.9%	270 | 268	278	260	6.3%	214 - 300	238 - 324
Ohio (18)	12:43AM	179	238	53.8%	43.4%	2.8%	270 | 268	278	260	6.1%	214 - 300	238 - 324

Figure 7: 270TOWIN US election real-time prediction

270TOWIN logged the expected electoral votes for each candidate as the results of states were announced. We can see their real-time prediction in the above log (Figure 7) that the expected electoral vote 'Average EV' changed in favour for Trump at 12:44AM when Pennsylvania and Utah's results were announced.

270TOWIN relied mostly on polling data and they updated their model parameters when battle ground states were called, or if there were large surprises, as stated on their website:

‘The underlying probabilities (for uncalled states) are based primarily on the final polling averages in each state. We take those averages and turn them into a probability that a candidate will win the state and its electoral votes. From there it is a mathematical calculation, one that is highly dependent on the accuracy of the state-level polling. Barring any surprises in the ‘safe’ states, the statistics (other than Total Electoral Votes') will not vary all that much until one or more battleground states is called.’ [10]

Forecast Companies working with Media

There are other forecasting companies working with the news medias to help them call the results of states. For example, CNN called New York for Clinton with only 15 percent of votes counted. When the forecasting companies believed they were sufficiently sure of the final result of a state based on the votes already counted, such information is relayed to the media.

These companies must also have a model for forecasting the whole election, however, these forecasts are not publicly available.

Other news agencies or forecasts probably had live forecasts as well, however, old results are often overwritten when new information becomes available. Thus, we were not able to find a nice graph that shows the forecast over time.

UK Election Real-time Forecast

There are a number of UK election real-time forecasters, for example, The Telegraph performed live prediction as the results came in. However, we can only see their final data as old data is overwritten when new data comes in. Thus, we cannot see how well the real-time forecasters did with partial information.

Pre-election Bias

Media

While polling suggested Hillary Clinton is more likely win the US election and Conservative will likely win majority in the UK election, the margin of the polls was small. The other outcome had a definite non-zero probability according to the polls and the standard errors on the polls. However, the media seems to have assumed that Trump's victory and Conservative minority is impossible to happen. Even after falsely predicting the event of Brexit, the media seemed to have taken a position that such events would not happen again. Below are some quotes from the media before the voting events took place.

UK:

'YouGov itself acknowledges that its methodology is "controversial". Indeed it is, for it is not an opinion poll in the traditional sense at all but a seat by seat "estimate" projected into a notional national result.' YouGov's poll predicting a hung parliament is certainly brave, The Gardian, May 31, 2017 [11]

YouGov correctly predicted a hung parliament multiple times, but mentioned their methodology is 'controversial'.

US:

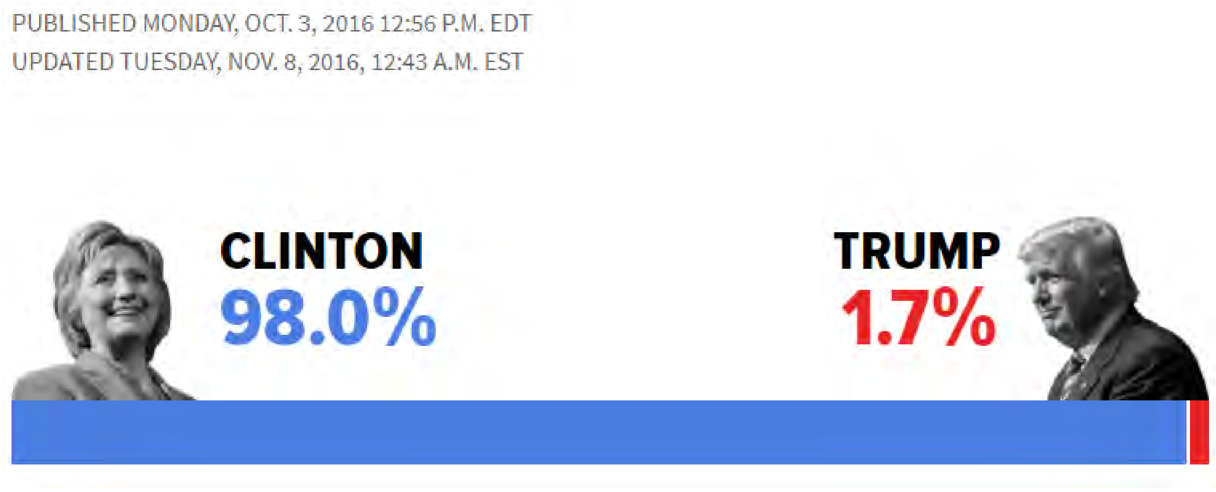
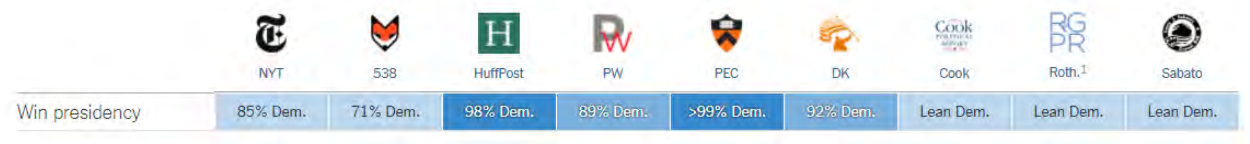

PUBLISHED MONDAY, OCT. 3, 2016 12:56 P.M. EDT
UPDATED TUESDAY, NOV. 8, 2016, 12:43 A.M. EST

Fig 8: Huffington posts, probability of US election [4]

In the US election, the Huffington post predicted the chance of a Trump victory is only 1.7 percent. Other election forecasters consistently favoured Clinton by a significant margin.



	NYT	538	HuffPost	PW	PEC	DK	Cook	Roth.1	Sabato
Win presidency	85% Dem.	71% Dem.	98% Dem.	89% Dem.	>99% Dem.	92% Dem.	Lean Dem.	Lean Dem.	Lean Dem.

Figure 9: forecast of the US election compiled by NYT[12]

Betting Market

Similar to the media, the betting market also significantly underestimated the probability of a Trump victory or Conservative minority putting them at around 20 percent and 3 percent respectively. But polls implied that the race is much tighter than race than the public is perceiving.

UK

'The bookmakers are certain that the Conservatives are going to win the General Election on June 8, so much so that the prices are prohibitively low to bet on. In some places the Tories are as low as 1/50 to win the most seats and 1/33 to gain an overall majority – unless you are betting gigantic sums then there is not a lot of point in investing at those odds.', Where to get the best odds on the Conservatives in the General Election, Metro, May 18,2017 [13]

US:

'The PredictIt market gave Clinton an 81 percent probability of winning the White House.' Betting sites see record wagering on U.S. presidential election, Reuters, Nov 7, 2016

Real-time US Election Forecast

Election Rules

There are 538 electoral votes in the US to be won by candidates. A candidate needs 270 electoral votes to win the election in the US and becomes the next president. The 538 electoral votes are distributed to the fifty states and District of Columbia, roughly according to the size of the population in each state. All electoral votes in a state is assigned to the candidate with most votes in the state for all the states except Maine and Nebraska, which use Congressional District Method where votes can be split between candidates. Below is a table of electoral votes for each of the fifty states.

US State	**Electoral Votes**	**US State**	**Electoral Votes**
Alabama	9	Montana	3
Alaska	3	Nebraska	5
Arizona	11	Nevada	6
Arkansas	6	New Hampshire	4
California	55	New Jersey	14
Colorado	9	New Mexico	5
Connecticut	7	New York	29
Delaware	3	North Carolina	15
Florida	29	North Dakota	3
Georgia	16	Ohio	18
Hawaii	4	Oklahoma	7
Idaho	4	Oregon	7
Illinois	20	Pennsylvania	20
Indiana	11	Rhode Island	4
Iowa	6	South Carolina	9
Kansas	6	South Dakota	3
Kentucky	8	Tennessee	11
Louisiana	8	Texas	38
Maine	4	Utah	6
Maryland	10	Vermont	3
Massachusetts	11	Virginia	13
Michigan	16	Washington	12
Minnesota	10	West Virginia	5
Mississippi	6	Wisconsin	10
Missouri	10	Wyoming	3

Figure 10: Electoral votes by state

Candidates

The main contestants for the presidency were Hillary Clinton of the Democratic Party and Donald Trump of the Republican Party. While independent candidates were allowed to compete, the winner of the election has always come from either the Democratic Party or the Republican Party, as the independent candidates usually lack the political backing and funding to compete with the two main candidates and have not won any election. We will be focusing our analysis on Trump and Clinton as they are the two candidates with real chances of winning.

Both candidates campaigned for more than a year and had three live debates against each other on national television prior to the election date. From the campaign information, we can obtain a good understanding of what they proposed for the country, which is important for understanding why the currency exchange market reacted as it did.

Trump

Since Donald Trump was the outsider to the election, we will focus on what he said during the campaign and his policy proposals.

Donald Trump, a real estate billionaire, announced that he will run the 45th US prudency on June 16th, 2015. He was quite critical of the Obama (44th president) administration and intend to change the direction of the country going forward. He was seen as a joke candidate at the beginning but his popularity soon surged by promoting himself as an outsider and connecting with people who are frustrated at the current political system. He soon defeated other republican candidates such as Ted Cruz, Jeb Bush and became the Republican presidential nominee on July 19th, 2016. His view on immigration and free trade was quite hostile particularly in regards to Mexico as he believed that outsourcing to Mexico and the Mexican immigrants are the root cause of loss of jobs and crime in the US. Here are some of his promises during the presidential campaign regarding Mexico:

"I would build a great wall, and nobody builds walls better than me, believe me, and I'll build them very inexpensively. I will build a great great wall on our southern border and I'll have Mexico pay for that wall." – Trump, January 16th, 2017

"We have at least 11 million people in this country that came in illegally. They will go out. They will come back — some will come back, the best, through a process. They have to come back legally. They have to come back through a process, and it may not be a very quick process, but I think that's very fair, and very fine." – Trump Feb 25, 2016

“A Trump administration will renegotiate NAFTA and if we don't get the deal we want, we will terminate NAFTA and get a much better deal for our workers and our companies. 100 percent." Trump Nov 7, 2016 [14]

From Donald Trump’s rhetoric during his campaign, one can tell that he is quite hostile towards Mexico and intended to punish Mexico economically for allegedly taking advantage of the US. It is no surprise that the Mexican peso fell when Trump’s Victory was announced.

Clinton:

Hillary Clinton, wife of previous president Bill Clinton, has been a politician most of her career. She ran as a Democratic Party candidate in 2008 but lost the primary to the 44th president Barack Obama. She then served in the Obama administration as secretary of state. She was mostly satisfied with the direction the country was heading under president Obama and promised to continue most of Obama’s policies. She did not voice opposition on trade with Mexico or Mexican immigrants during the campaign. Currency exchange market would have reacted very differently if Hillary Clinton won the 45th presidency.

Real-time Forecast Model

If there was a consistent poll bias on one of the candidate, we should be able to identify it in real-time when results of some states were available. We build a simple one factor regression model to identify the bias between polling data and vote count for each candidate. Then the results for the remaining unknown states are simulated. Using the available data and simulated data, we can predict the winner of the election with a confidence interval.

The algorithm:

At each time step t:

1. Perform weighted least square regression (more explanation in data source section) of actual voting results on polling data for each candidate $c \in [Trump, Clinton]$ for states with results according to formula

$$actual\ vote_c = \beta_{1,c} * poll_c + \epsilon_c$$

to obtain estimates of parameter $\widehat{\beta_{1,c}}$, $se_{1,c}$ and residual $resid_c$ as well as standard error of parameters. The parameters β_1 follows Normal distribution , $N(\widehat{\beta_1}, se_1)$. If the standard deviation of the residuals is denoted by σ, then the sum of squares of residuals divided by σ^2 has a chi-squared distribution with $k-1$ degrees of freedom.

2. Simulate the N times based on the distribution of the parameter $\beta_{1,c}$ to obtain N sets of parameters $\beta_{1,c,n}$, where $n = 1,2, \dots N$
3. For each group of parameters, simulate N times the voting percentage for each candidate for unknown states s as $vote_{c,n,s} = N(\ \hat{\beta}_{1,c,n,s} * poll_{c,n,s}\ ,\ \widehat{\sigma_n})$
4. For each unknown state s, compare the simulation result between Trump and Clinton. The electoral vote is awarded to the candidate with the higher simulated result.
5. The **expected electoral votes** for candidate c : $EEV_{c,t} = EV_{c,t} + \sum_n EV_s * \mathbf{1}_{(\boldsymbol{R_c}(vote_{c,n,s}) = 1)} / N^2$, where $EV_{c,t}$ is the electoral votes candidate c already obtained at time t, EV_s is the electoral votes for the remining state s, $\boldsymbol{R_c}(vote_{c,n,s})$ is the rank of candidate c for simulation n for state s, and $\mathbf{1}_{()}$ is the indicator function.
6. At each step t we plot the expected electoral votes of the candidate, confidence interval vs the currency exchange rate

Constant β_0 is removed from the regression as it is statistically insignificant.

Results

We obtained the following result using the algorithm mentioned above. In the following result n is set to 100.

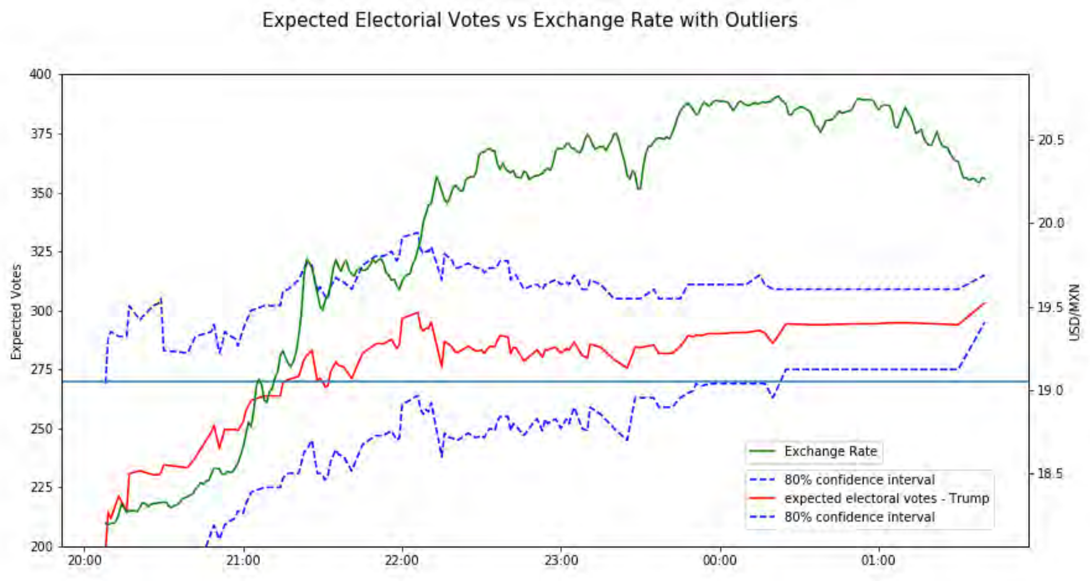


Figure 11: Expected electoral votes for Donald Trump and exchange rates

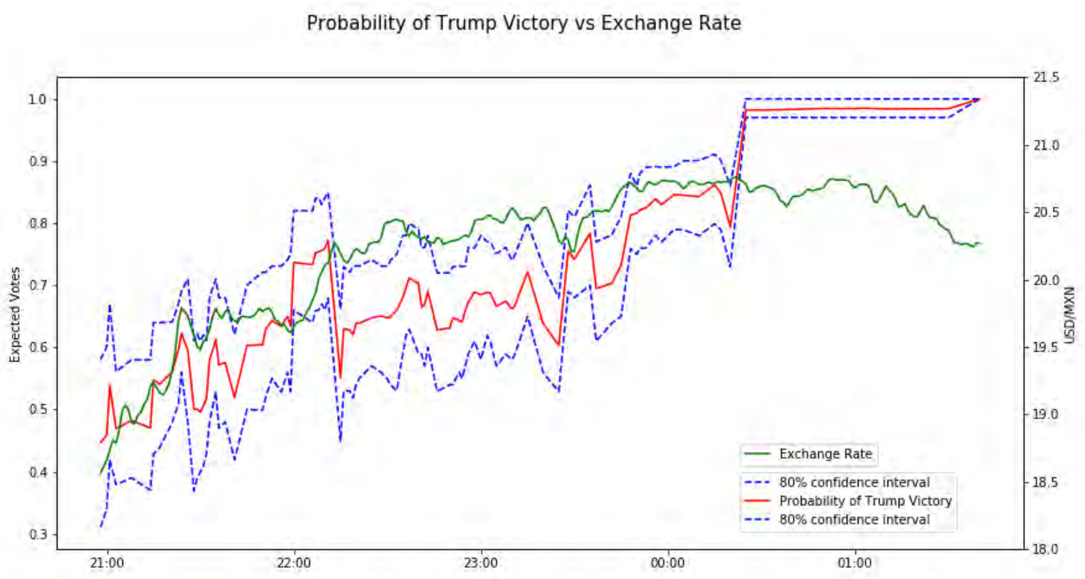


Figure 12: Probability of Trump Victory and Exchange Rates

We can see that our simple one factor model predicted a high ($> 50\%$) probability of Trump victory from around 21:15 onwards. And the probability of Trump winning moved quite closely with the currency exchange rate between USD/MXN. Comparing with the real-time prediction model from FiveThirtyEight, our model did well as we were able to predict Trump's victory with high probability earlier. However, our model was still not able to say with 90% certainty (bottom blue line in figure 12) that Trump will win until very late in the election night. Below we can see how the β_1 parameters evolved overtime.

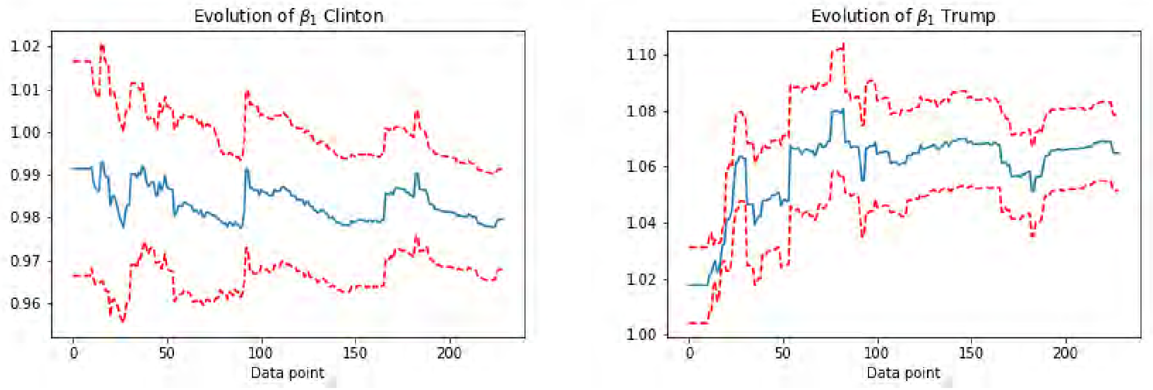


Figure 13: Evolution of $\beta_1$parameter

Donald Trump's β_1 parameter ended up at about 1.06, indicating that on average he received 6 percent more votes than he polled at. Hillary Clinton's β_1 ended at 0.98, indication that on average she received 2 percent fewer votes she polled at.

The parameters are relatively stable after 100 data point, which correspond to approximately 22:00. Thus, our expected electoral votes for Trump is relatively stable from 22:00 onwards.

Data Source

We obtained our real-time US election update by CNN, which had a full night election coverage. The whole election night video can be found on YouTube [15]. The election coverage contains two types of updates:

Winner announcement: CNN announced the winner of a state when it was confident that one candidate had won a state. In these announcements, the final percentage of each candidate is not available.

Intermediate results for states: CNN provided regular updates on states that are still counting but no clear winner had emerged yet. In these announcements, CNN announced the percentage for each candidate and the percentage of the votes counted.

We logged the information from CNN carefully, and used the intermediate information for our regression. We used the most up-to-date information of each state to perform a weighted least square (WLS) regression with the weights being the percentage of the votes counted. Therefore, the states that were early in the counting process had less significance in the regression compare to the states that were mostly finished counting, making our model more robust to early counting anomalies. When more information come in, the old information is overwritten with new percentage for each candidate and percentage of votes counted.

The polling data is taken from FiveThirtyEight which is an aggregate of polls of other polling companies. A poll aggregate is more robust compared to taking an individual poll, as sampling bias of an individual polling company are likely averaged out.

Outliers

We identified that New York, Montana, Texas, Utah and Iowa were outliers. New York, Montana, Texas, Utah were declared very early on for the election for either candidate, and voting percentage for each candidate were not changing anymore after the state winner is declared.

For example, New York was declared at Trump 21%, Clinton 77% when 15% of the votes were counted. The end result for New York was Trump 38%, Clinton 59%. The end result as not available until the whole counting was finished, much later than Trump's victory was announced. Including these intermediate points gave the model very biased data and skewed our result. In real life, one can quickly determine these were early bias data points that do not show much significance.

For Iowa, the regions that were pro Clinton reported in first leading to a very significant pro Clinton bias, at 50 percent reporting, we can see a more balanced view for the state. The result from Iowa was removed from the regression until 50 percent of the votes are counted.

Traders on the currency exchange market should be able to tell also that the very early data for these states are unreliable, thus ignoring or heavily penalising these data points when they made currency exchange decisions.

For completion, below (Figure 14) is a plot of the prediction without any of these outliers removed.

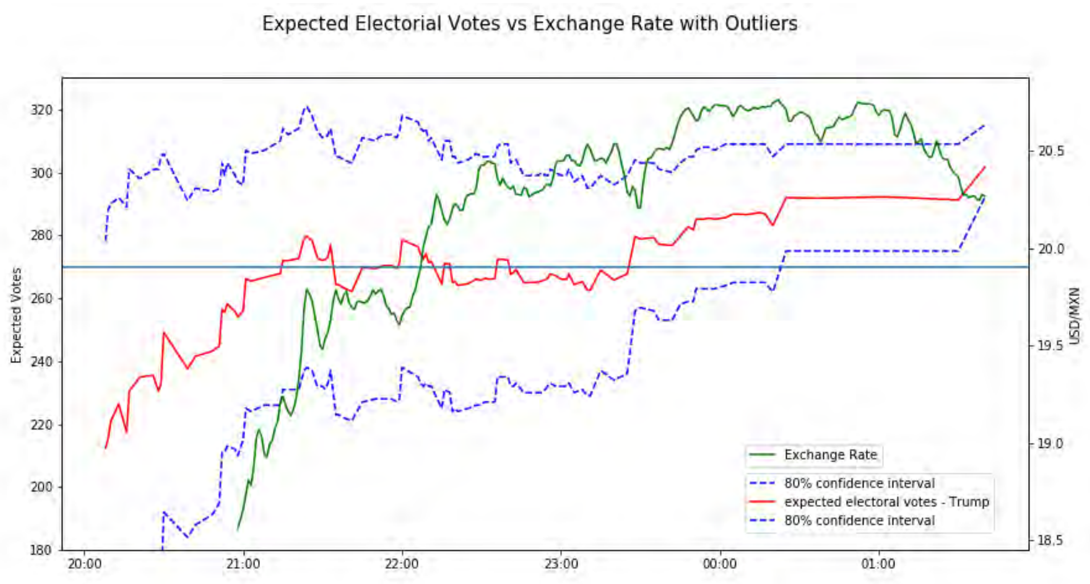


Figure 14: Result with outliers

These outliers did not change the overall result and the trend by a large amount. However, the expected electoral vote for Trump zigzagged around 270 for much longer.

Market Reaction

The USD/MXN rate rose significantly between 21:00 and 22:30, indicating that currency traders are increasing their bets to favour Trump. The exchange reached its peak at around 01:00 the next day before correcting back to the level at 23:00. The movement of the market matched the probability of a Trump victory calculated by our model very well. We can see the overall trend of the exchange rate moved in the same direction with our predicted probability in figure 10, and the small fluctuations in the exchange rate moved mostly in sync with our predicted probability of Trump victory.

Market Efficiency

Based on the movement of the currency exchange value of the USD/MXN, the market seemed to be quite efficient during the US election night. The semi-strong form of Efficient Market Hypothesis seemed to have held during the 2016 US election. Currency traders quickly caught on to the difference between the actual results during the counting process and the polling data, and updated their bets. At around 22:30, the value of the USD/MXN had risen to the level when Trump's victory was announced.

Further Model Improvements

While our model's prediction was not lagging the market movement, we were not able to say with high certainty that Trump will win until the very end. We believe that the model can be further improved upon to produce a more accurate prediction earlier on. However, due the limitation of time and data availability, these improvements have not been implemented.

Demographic data:

While poll was a good predictor of the actual result. Demographic data are also good predictors of election results given the poll. These demographic data include age, gender, income, ethnicity, education, etc. Donald Trump attracted the votes of a large number of white male with low level of education that the polls missed in the rural areas of the US which is a main reason of his surprising victory. The votes casted by this group of people are poorly reflected in the polls as they are often not reported or sampled due to not participating in previous elections.

The demographic data are quite heterogenous even within each state. For example, while Texas as a whole voted for Donald Trump with a significant margin, cities such as Austin, Dallas, and Huston voted for Clinton.

If one can find the data on the county level (poll, demographic, result update), it is quite possible that the predictive capability of the model will be improved and Trump's victory can be predicted earlier than our simple one factor model did.

Voter Turnout:

A significant reason that Donald Trump won is by convincing people that did not vote in the previous elections to vote for him in the 2016 election. By observing a state having a larger than usual turnout can give us clue that Donald Trump might be favoured in the results.

However, we were not able to find turnout data when the election was happening, as the intermediate turnout data might be overwritten by the final result.

Previous Election Results:

Many pundits use previous election results of each state as a factor to predict the next one. We decided not to include this factor as this was a very special case with very unique candidates and due to limited time availability. However, using this factor might improve our prediction.

If a better model is developed which can predict a Trump victory much earlier on, the market might not be as efficient as we thought. Such slightly more complex model can still be developed by traders with a large amount of capital and labour with relative ease. Then if the USD/MXN rates lags the predicted probability of Trump victory by a large amount, the market might not be as efficient as in our analysis.

Tools used

This project is written in Python. All statistical analysis is done using 'statsmodels' package. Code and data can be found here:

https://github.com/alexhuang1117/Data-Science-Portfolio/blob/master/FX_Analysis_During_US_Election/main.ipynb

Real-time UK Election Forecast

Election Rules

There are 650 constituencies in the UK, each constituency elects one member of parliament (MP) to the House of Commons. The party with the most votes in a constituency wins the constituency and gains a seat in the House of Commons. Parties that have seats in the House of Commons then form a government that will be in power for the next four to five years. If a party has more than fifty percent of the seats in the House of Commons, then this party governs with a *majority government*. If no party won more than fifty percent of the seats, two or more parties can establish a *coalition government* with more than fifty percent of the seats to govern. If such a coalition cannot be formed, the largest party will govern with a *minority government*. The opposition party is the party with the most seats that is not in the ruling government.

The major uncertainty in the 2017 election was whether the conservative government could obtain a majority government, and if so by how much. A party needs 326 seats in the House of Commons to become the majority party.

Prior to the 2017 election, Theresa May of the Conservative Party held a majority government and Jeremy Corbyn of the Labour Party was the leader of the opposition.

Brexit and Election Background

Prime minister Theresa May of the Conservative Party called the snap election strengthen her position in the British parliament and in the Brexit negotiation by hoping to get a stronger majority than the one the former prime minister David Cameron left her. The primary purpose of this snap election and the subsequent campaigns focused on Brexit, which is a referendum that took place in the UK in 2016 to decide whether the UK will stay in the European Union(EU). Before she called the election, polls suggested that she would win the election by a landslide. Theresa May decided to take advantage of the opportunity to strengthen her position in the House of Commons.

Brexit information:

In the general election in 2015, Conservative candidate and then prime minister David Cameron promised an in-or-out referendum to decide whether the UK will stay in the EU. David Cameron won the 2015 general election with a majority government with 330 seats in the House of Commons and stayed as the prime minister. He kept his promise on the referendum and campaigned for remaining in the EU.

The referendum was held on June 23, 2016. To his surprise, the voters in the UK voted to leave the EU with a narrow majority.

Since David Cameron campaigned for the UK staying in the EU, he believed that he has lost the mandate to govern and stepped down as the prime minister. Then home secretary Theresa May became the new prime minister in 2016 and inherited David Cameron's majority government. She was tasked with the mission to take the UK out of the EU.

In March 2017, Theresa May triggered Article 50, a clause in the EU's Lisbon treaty that outlines the steps a member country needs to take to exit the European Union. What followed was a two-year negotiation period between the UK and the EU to decide the relation between the two entities after the two-year transition period. The UK has been a member of the EU for more than forty years at this point and much of the UK economy was tightly integrated with other countries in the EU. Breaking away from the EU would require lengthy and difficult negotiation regarding which relations to keep and which relations to break up.

Theresa May's majority government began negotiations with the EU representatives at the end of March 2017. However, the Conservative Party and members of other parties are split between how Brexit should be implemented. Two options are possible now for the UK to leave the EU, either 'soft' Brexit or 'hard' Brexit.

'soft' Brexit: The UK will leave the European Union but maintain most economic ties with it, which includes the four cores freedom of the EU: The free movement of goods, people, services and capital over borders. In this event, UK will have a relationship with the EU similar to what Norway has with the EU. UK will have more power to make its own regulations but must still pay dues to the EU and follow most core EU principles.

'hard' Brexit: The UK will sever most ties with the EU, and become a country with little connection with the EU. In this event, the UK will regain most power to regulate its domestic affairs but losing trade ties with the EU might hurt its economy.

As most members in the parliaments are split between 'soft' and 'hard' Brexit, even in her own party, Theresa May decided to call a snap election to obtain a mandate for her own version of Brexit and strengthen her position in the negotiation with the EU.

Parties and Campaigns

Unlike the US election where there were only two major candidates with a real chance to win any state. The UK election consisted of multiple parties each with the goal of winning seats in the House of Commons.

The Conservative Party:

The Conservative Party led by Theresa May hoped to gain a stronger majority than they did in 2015 under the leadership of David Cameron. The Conservative Party was seen as business friendly and campaigned for a ‘hard’ Brexit. Theresa May said ‘Brexit means Brexit’ on the campaign trail, intending to leave EU’s custom union and be free of the EU’s laws and European Court of Justice.

The polls in the beginning of the campaign had suggested that Theresa May would win a majority with a large margin. Closer to the election day, the polls still mostly favoured her, but the margin was much thinner.

The Labour Party:

The Labour Party led by Jeremy Corbyn campaigned against Brexit during the referendum. However, they decided to respect the result of the referendum and go forward with Brexit. Comparing to the Conservatives, they preferred to keep closer ties the EU and maintain most relationships. They opposed a second referendum on a final deal with the EU.

The Labour Party polled very poorly in the beginning of the campaign. However, polls improved closer to the election day.

The Liberal Democrats:

The Liberal Democrats were strongly pro-EU, they promised to stop ‘hard’ Brexit as they see it as a disaster for the country. They campaigned to stay in the EU during the referendum and proposed a second referendum on the final deal with the EU. They had much less support compared to the Conservative Party and the Labour Party.

The Scottish National Party:

The Scottish National Party was primarily active in Scotland, they campaigned for Scotland to have a special status in the UK and maintain closer ties with the EU. The leader Nicola Sturgeon also proposed a referendum on Scotland leaving the UK.

Other Parties:

In addition to the four major parties, there are some smaller parties such as UKIP, Green, Plaid Cymru, DUP, and Sinn Fein. They each have different goals and stands on Brexit. Since they usually do not win a significant number of seats in the House of Commons, we will not discuss their political agenda in details here.

Value of the British Pound Sterling

The value of the British pound sterling crashed on the eve of the Brexit referendum. The UK faced very big uncertainty in its future outside of the EU. The negative risk on the UK economy was quite large given the UK's current connection and dependence on the EU.

Forecasters and political analysts believed that if the Conservative Party led by Theresa May was able to gain majority in the upcoming election with a large margin, she could negotiate from a stronger position for Britain in the upcoming transition period. This outcome would remove some negative risk on the UK economy going forward as Theresa May would likely be able to pursue a better deal with the EU. The value of the pound would like go up on brighter economic perspective for the UK.

However, if Theresa May failed to secure a majority in this election, the UK government would be even more divided on how to go forward with Brexit. The economic future of the UK would continue to be plagued with negative risks. The value of the pound will likely go down due to higher uncertainty for the future of the UK economy.

Single Factor Model

We first attempted to use a single factor to predict the UK election as it happened, similar to the what we did in the US election. We tried to identify the bias between the poll and the actual result for the three major parties, Conservative, Labour and SNP. The Liberal Party or other parties are assumed to win the remainder of the votes not taken by the three major parties. No regression is performed on the Liberal Party and other parties as there is limited data available.

The algorithm:

At each time step t:

1. Perform OLS regression of actual results of the available constituencies on polling data for each candidate $c \in [Con, Lib, SNP]$ with formula

$$actual\ vote_c = \beta_{0,c} + \beta_{1,c} * poll_c + \epsilon_c$$

   to obtain estimates of parameters $\widehat{\beta_{0,c}}, \widehat{\beta_{1,c}}, se_{0,c}, se_{1,c}$ and residual $resid_c$ as well as standard error of parameters. The parameters β_0, β_1 follow Normal distribution $N(\widehat{\beta_0}, se_0), N(\widehat{\beta_1}, se_1)$. If the standard deviation of the residuals is denoted by σ, then the sum of squares of residuals divided by σ^2 has a chi-squared distribution with $k - 2$ degrees of freedom.

2. Simulate $\beta_{0,c}$, $\beta_{1,c}$, from their normal distribution N times to obtain N sets of parameters $(\beta_{0,c,n}, \beta_{1,c,n})$ where $n = 1,2, \ldots N$
3. For each group of parameters, simulate N times the voting percentage for each candidate for unknown constituency s as $vote_{c,n,s} = N(\hat{\beta}_{0,c,n,s} + \hat{\beta}_{1,c,n,s} * poll_{c,n,s} , \widehat{\sigma_n})$ and all the remaining votes goes to an 'other' party $vote_{other,n,s} = 1 - \sum_c vote_{c,n,s}$
4. For each unknown constituency s, compare the simulation result between the parties. The seat is awarded to the candidate with the higher simulated result.
5. The expected total seats $ES_{c,t} = S_{c,t} + \sum_n \mathbf{1}_{(\boldsymbol{R_c}(vote_{c,n,s}) = 1)} \quad / N^2$, where $S_{c,t}$ is the seats a candidate obtained at time t, $\boldsymbol{R_c}(vote_{c,n,s})$ is the rank of candidate c for simulation n for constituency s, and $\mathbf{1}_{()}$ is the indicator function.
6. At each step t we plot the expected seat of the conservative, confidence interval vs the currency exchange rate.

Single Factor Model Results

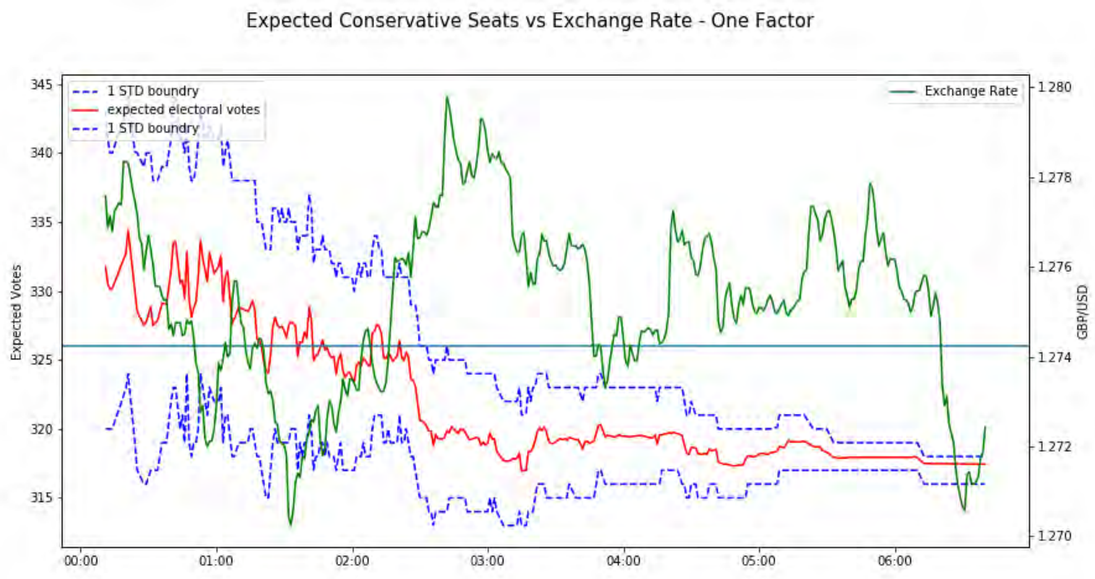


Figure 15: Expected Seats for Conservative and Exchange rate – One Factor

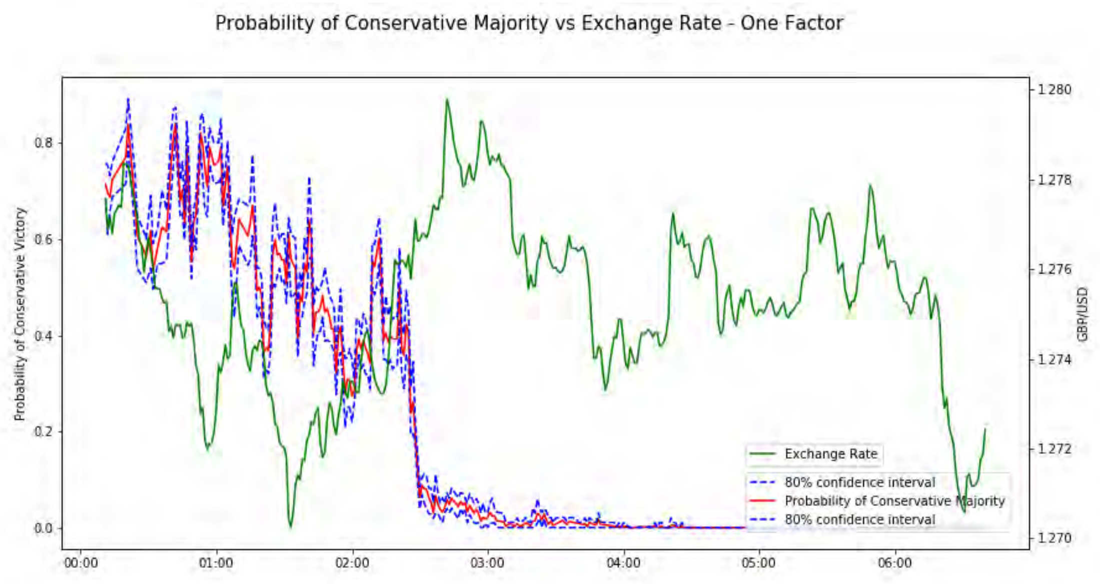


Figure 16: Probability of Conservative majority and exchange rate – One Factor

This model was quite indecisive about the outcome until about 2:30. And it predicted that there was still a small chance that the Conservative would win majority from 2:30 to 3:30. The performance of the model was good, as we can see that the probability of a Conservative majority crashed at 2:30 while the value of the British pound is still quite high. However, this model was quite simple as it only used one factor, namely polls. While polls were probably the best predictor of an election outcome, there existed other data we could use that might improve model performance. For example, election forecasters consistently analysed demographic data to improve their prediction. We will expand on this simple model in the next chapter to see if the model's predictive performance will improve.

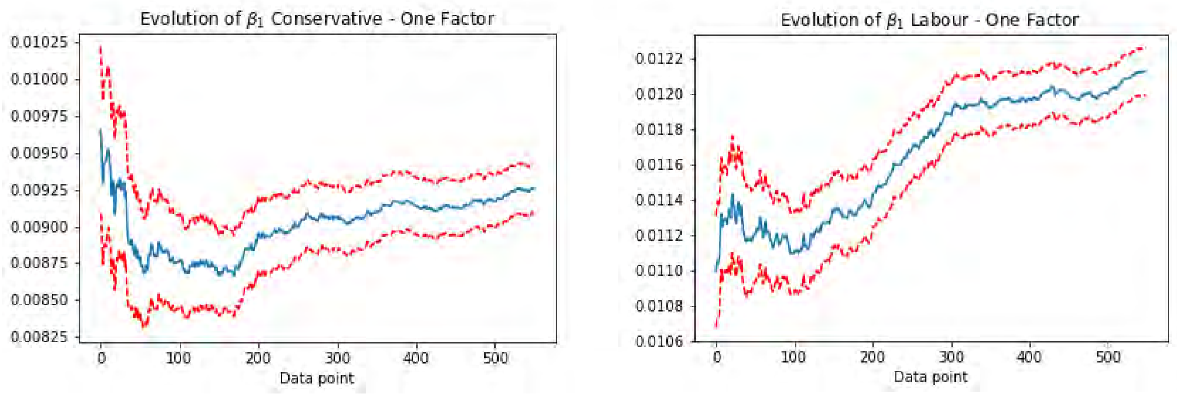


Figure 17: Evolution of β_1 parameters – One factor

Looking at the β_1 parameters for Conservative and Labour, we could observe that the parameters were not stable. For Labour, there was a large uptrend to the point where we had 300 data points and continued to increase ending up at 0.012 (note, in our data, poll is in percentage i.e. 42 for Conservative polling, and final result is in fraction, i.e. 42% = 0.42 voted for Conservative, therefore there is a factor of 100 difference.) For Conservative, we also see a growing trend from data point 200 and ending up at 0.0092.

This indicates our model has failed to take some key parameters into account. If we are able to find out the factors that are correlated with the growth with the parameters, we can improve our model's performance.

Three-factor Model

As we can see in the Figure 15, the single factor model predicted the Conservatives would win a majority with high probability at around 3:00. We wondered if the model performance could be improved with a more complex model. In this section, we will expand on our previous model and use a three-factor model to predict UK election results.

We included two additional factors to see whether the single factor model could be improved upon. The two additional factors were median wage level of each constituency and percent of population over 65. We picked these two additional factors as they are easily accessible (from 2011 UK census), and wage and age are usually correlated with election results.

Our algorithm becomes:

At each time step t:

1. Perform regression of actual results of the available constituencies on polling data for each candidate $c \in [Con, Lib, SNP]$ with formula

$$actual\ vote_c = \beta_{0,c} + \beta_{1,c} * poll_c + \beta_2 * wage + \beta_3 * Populationover65 + \epsilon_c$$

   to obtain estimates of parameters $\widehat{\beta_{0,c}}, \widehat{\beta_{1,c}}, \widehat{\beta_{2,c}}, \widehat{\beta_{3,c}}$ $se_{0,c}, se_{1,c}, se_{2,c}, se_{3,c}$ and residual $resid_c$ as well as standard error of parameters. The parameters $\beta_0, \beta_1, \beta_2, \beta_3$ follow Normal distribution $N(\widehat{\beta_0}, se_0), N(\widehat{\beta_1}, se_1), N(\widehat{\beta_2}, se_2)\ N(\widehat{\beta_3}, se_3)$. If the standard deviation of the residuals is denoted by σ, then the sum of squares of residuals divided by σ^2 has a chi-squared distribution with $k-4$ degrees of freedom.

2. Simulate N times based on the distribution of the parameters $\beta_{0,c}, \beta_{1,c}, \beta_{2,c}, \beta_{3,c},$ where $n = 1,2, \dots N$ to obtain N sets of parameters $(\beta_{0,c,n}, \beta_{1,c,n}, \beta_{3,c,n}, \beta_{4,c,n})$
3. For each group of parameters, simulate N times the voting percentage for each candidate for unknown constituency s as $vote_{c,n,s} = N(\hat{\beta}_{0,c,n,s} + \hat{\beta}_{1,c,n,s} * poll_{c,n,s} + \beta_{2,c,n,s} * wage + \beta_{3,c,n,s} * Populationover65\ ,\ \widehat{\sigma_n})$ and all the remaining votes goes to an 'other' party $vote_{other,n,s} = 1 - \sum_c vote_{c,n,s}$
4. For each unknown constituency s, compare the simulation result between the parties. The seat is awarded to the candidate with the higher simulated result.
5. The expected total seats $ES_{c,t} = S_{c,t} + \sum_n \mathbf{1}_{(\boldsymbol{R_c}(vote_{c,n,s}) = \ \mathbf{1})} \ \ /N^2$, where $S_{c,t}$ is the seats a candidate obtained at time t, $\boldsymbol{R_c}(vote_{c,n,s})$ is the rank of candidate c for simulation n for constituency s, and $\mathbf{1}_{()}$ is the indicator function.
6. At each step t we plot the expected seat of the conservative, confidence interval vs. the currency exchange rate.

Three-factor Model Result

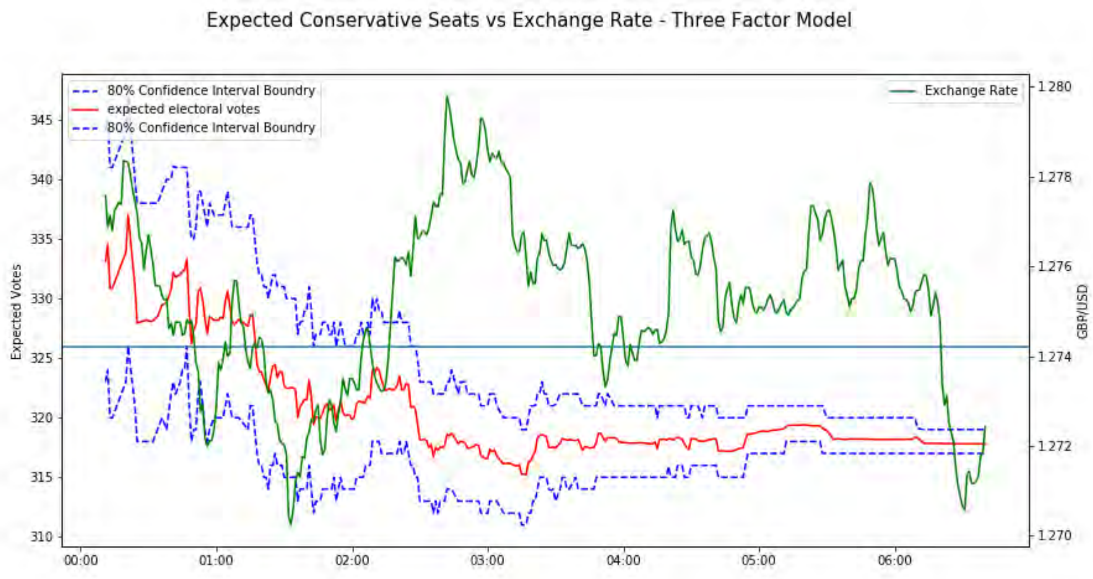


Figure 18: Expected Seats for Conservative and Exchange rate – Three Factor

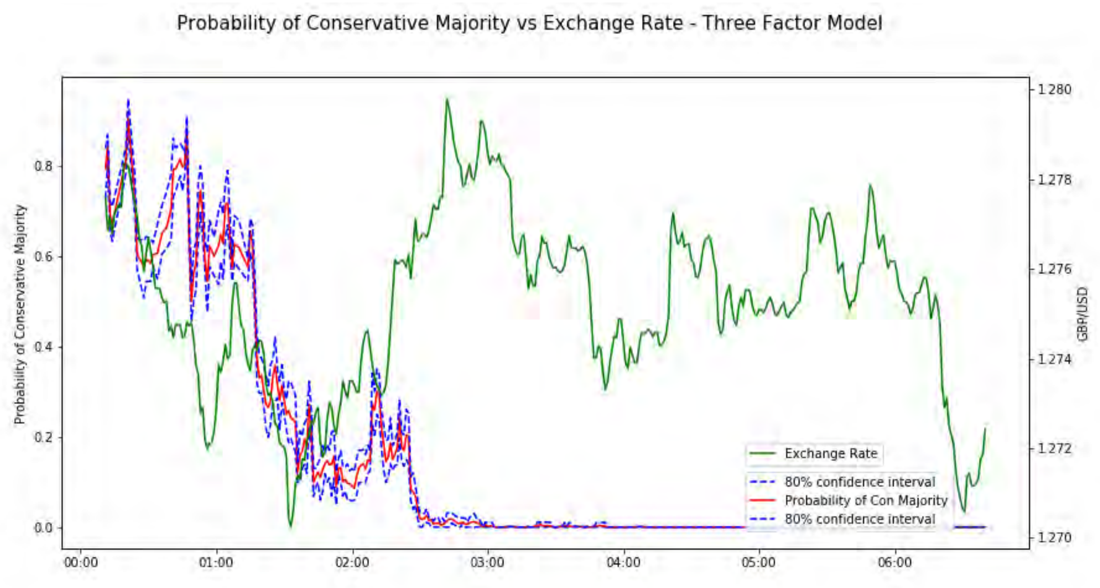


Figure 19: Probability of Conservative majority and exchange rate – Three Factor

Using the two additional factors improved our model significantly, we can see that our model predicted the probability of a conservative majority if virtually 0 at around 2:30. And the probability crashed around 1:00 to approximately 30%. This is quite a significant improvement over the single factor model.

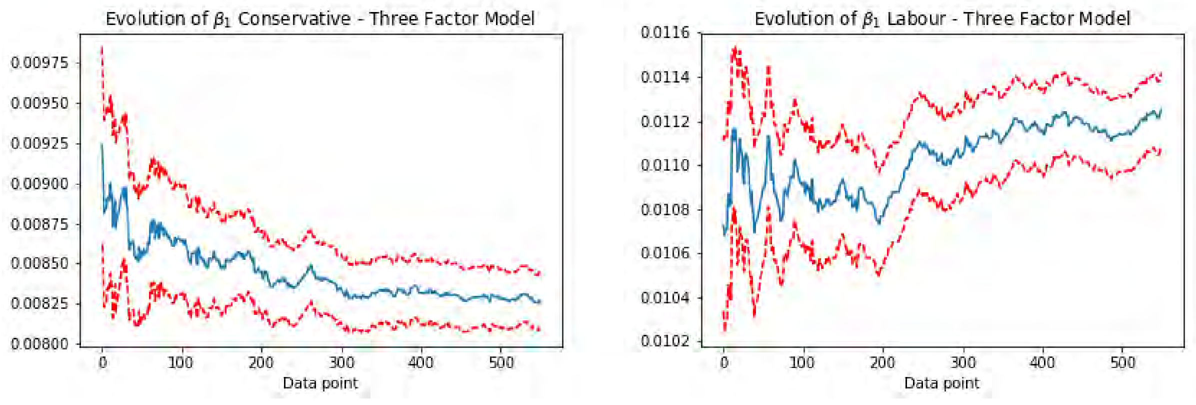


Figure 20: Evolution of β_1 parameters – Three factor

We investigated the β_1 parameter for both parties here again. While there were three β parameters, β_1, polls, remained the most important predictive parameter in our model, thus our interest continued to focus on β_1 . The β_1 parameters were much more stable for Conservative and Labour compared to our previous one factor model. However, there was still a big jump of the β_1 parameter for Labour around point 200 and a small growing trend afterwards. While our model performance improved drastically and the β_1 parameters were much more stable, we could further improve our model performance in the next section.

Three-factor Model with Regional Dummies

UK is consisting of twelve regions, they are West Midlands, Scotland, South East, Yorkshire and The Humber, Wales, South West, East of England, London, North West, North East, East Midlands, and Northern Ireland. By performing a three-factor model on each of the region, we identified that some regions have very different βs. The three regions that were significantly different are London, Scotland and Wales (Northern Ireland has no polling data for Conservative, Labour, SNP and is thus omitted). It made sense as London is a big city, Scotland and Wales are separate countries to England leading to different voter characteristics compared to most parts of England.

After identifying the difference in regions, we improve our model's prediction by adding a dummy variable for each: London, Scotland and Wales. Our previous three-factor model becomes a twelve-factor model.

We expanded the model for both the Conservative and Labour party, for SNP data, we kept the three-factor model as the number of data points for SNP is limited. Having thirteen βs requires at least thirteen data points to solve the equations. Otherwise the system is underdetermined.

Below is a table for the coefficient name for each variable.

Parameter Name	**Corresponding factor f_i**
β_0	1 (constant): f_0
β_1	Poll: f_1
β_2	Poll * Dummy for London: f_2
β_3	Poll * Dummy for Wales: f_3
β_4	Poll * Dummy for Scotland: f_4
β_5	Median Wage: f_5
β_6	Median Wage * Dummy for London: f_6
β_7	Median Wage * Dummy for Wales: f_7
β_8	Median Wage * Dummy for Scotland: f_8
β_9	Percent Population over 65: f_9
β_{10}	Percent Population over 65 * Dummy for London: f_{10}
β_{11}	Percent Population over 65 * Dummy for Wales: f_{11}
β_{12}	Percent Population over 65 * Dummy for Scotland: f_{12}

Figure 21: Parameters and factors

The dummy variable takes on the value of 1 if the constituency is in the region, 0 otherwise. Regression parameters for factors with a dummy variable can be interpreted as the additional effect of the region on the parameter. We can see this in the following simple example:

Example: Let's consider a simple model where

$$actual\ vote = \beta_0 + \beta_1 * poll + \beta_2 * poll * London + \epsilon$$

$$\beta_1 = 1$$

$$\beta_2 = 0.1$$

Here if we see an increase of δ in regions that are not London, we expect to see an increase of δ in the actually vote. However, if we see an increase of δ in London, we expect to see an increase of $1.1 * \delta$ in London. β_2 is the additional effect of London on the poll factor.

Our algorithm is mostly the same except the equation is changed.

At each time step t:

1. Perform regression of actual results of the available constituencies on polling data for each candidate $c \in [Con, Lib, SNP]$ with formula

$$actual\ vote_c = \sum_{i=0}^{12} \beta_{i,c} * \mathrm{f_{i,c}} + \epsilon_c$$

to obtain estimates of parameters $\widehat{\beta_{\iota,c}}$, $se_{i,c}$, $where\ i \in 0,1,2, \dots 12$, and residual $resid_c$ as well as standard error of parameters. The parameters β_i follow normal distribution $N(\widehat{\beta_\iota}, se_i)$. If the standard deviation of the residuals is denoted by σ, then the sum of squares of residuals divided by σ^2 has a chi-squared distribution with $k - 13$ degrees of freedom.
2. Simulate $\beta_{i,c}$ N times the based on the distribution of to obtain n sets of parameters $(\beta_{i,c,n})$ $where\ i \in 0,1,2, \dots 12$ and $n = 1,2, \dots N$
3. For each group of parameters, simulate N times the voting percentage for each candidate for unknown constituency s as $vote_{c,n,s} = N(\widehat{y_{c,n,s}}\ ,\ \widehat{\sigma_n})$, $where\ \hat{y} = \sum_{i=0}^{12} \widehat{\beta_{\iota,c,n,s}} * f_i$ and all the remaining votes goes to an 'other' party $vote_{other,n,s} = 1 - \sum_c vote_{c,n,s}$
4. For each unknown constituency s, compare the simulation result between the parties. The seat is awarded to the candidate with the higher simulated result.
5. The expected total seats $ES_{c,t} = S_{c,t} + \sum_n (\boldsymbol{R_c}(vote_{c,n,s}) = \mathbf{1})\ /N^2$, where $S_{c,t}$ is the seats a candidate obtained at time t, $\boldsymbol{R_c}(vote_{c,n,s})$ is the rank of candidate c for simulation n for constituency s, and $\mathbf{1}_{()}$ is the indicator function

6. At each step t we plot the expected seat of the conservative, confidence interval vs. the currency exchange rate.

Three-factor Model with Regional Dummies Results

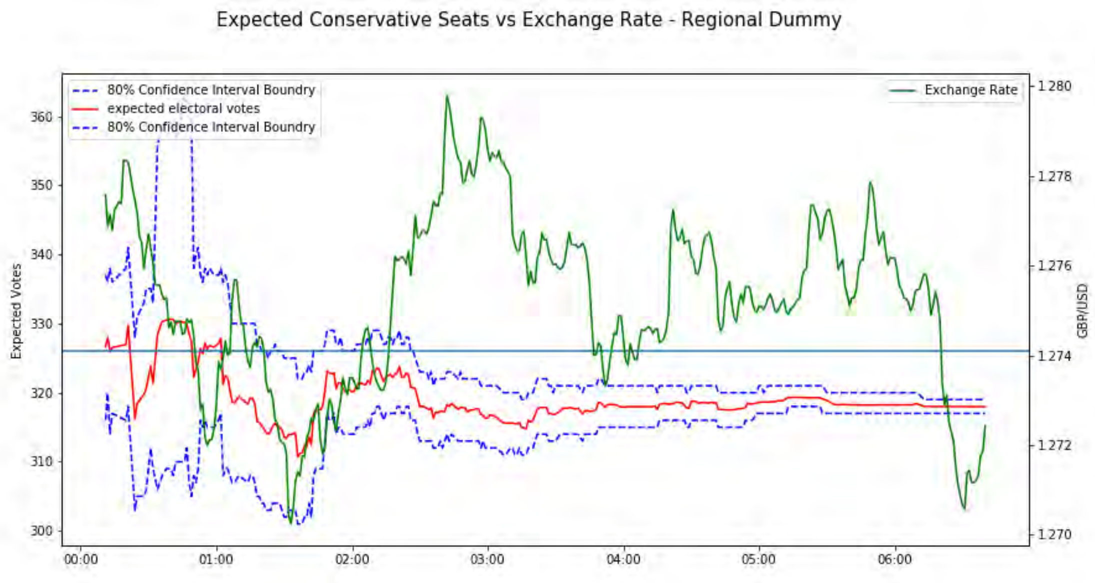


Figure 22: Expected Seats for Conservative and Exchange rate – Regional Dummy

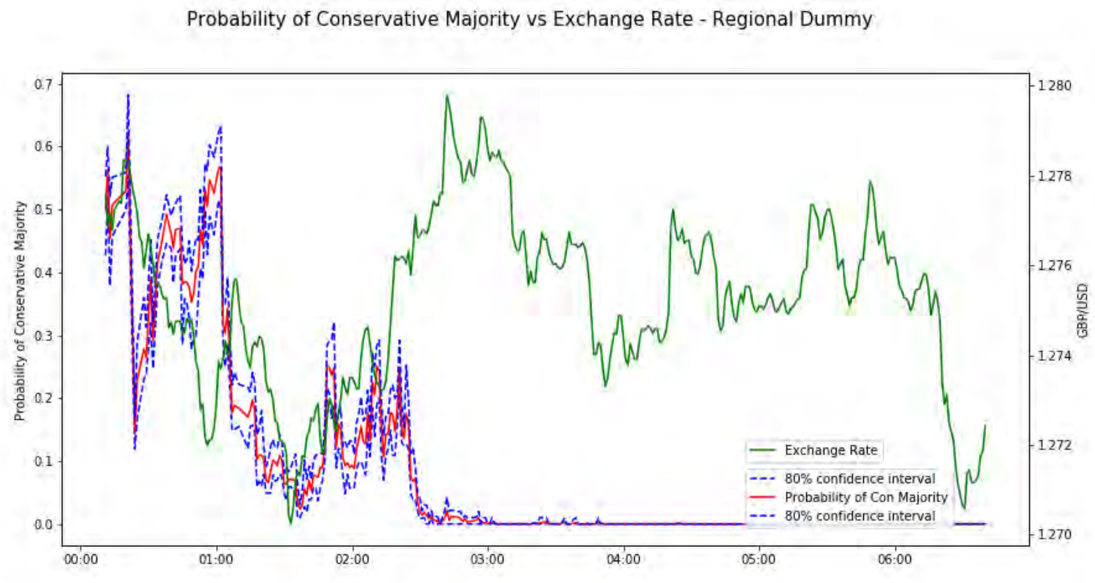


Figure 23: Probability of Conservative majority and exchange rate – Regional Dummy

Here we can see that our result improved further. We observed that there was high probability that the Conservative would not win majority stating at around 1:30. Although the probability increased slightly at around 2:00. At 2:30 we could tell that the Conservative Party had very little chance of winning majority, while the British pound was still at its highest point. Our model's movement preceded that of the currency exchange rate most of the time.

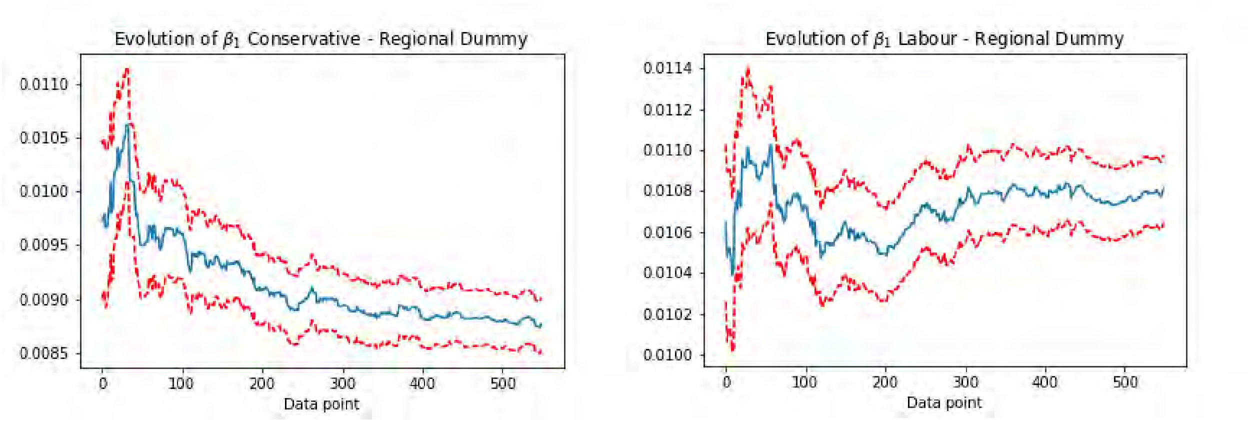


Figure 24: Evolution of β_1 parameters – Regional Dummy

Conservative Regression Parameter:

Parameter Name	Coefficient	Standard Error	P Value
β_0	-0.0530	0.018	0.004
β_1	0.0087	0.000	0.000
β_2	-0.0016	0.001	0.026
β_3	0.0008	0.001	0.337
β_4	0.0034	0.001	0.000
β_5	-1.078e-05	3.54e-05	0.761
β_6	-7.716e-05	3.67e-05	0.036
β_7	0.0002	7.73e-05	0.005
β_8	1.108e-05	5.86e-05	0.850
β_9	0.5654	0.060	0.000
β_{10}	0.7927	0.252	0.002
β_{11}	-0.3334	0.193	0.084
β_{12}	-0.3407	0.190	0.073

Figure 25: Regression parameter coefficients – Conservative – Regional Dummy

Labour Regression Parameter:

Parameter Name	Coefficient	Standard Error	P Value
β_0	0.1706	0.021	0.000
β_1	0.0108	0.000	0.000
β_2	0.0007	0.000	0.019
β_3	-0.0014	0.001	0.011
β_4	0.0004	0.001	0.476
β_5	-5.302e-05	2.63e-05	0.044
β_6	-3.145e-05	3.92e-05	0.423
β_7	0.0003	8.62e-05	0.003
β_8	-0.0001	6.01e-05	0.056
β_9	-0.4496	0.046	0.000
β_{10}	-0.0024	0.138	0.986
β_{11}	-0.4648	0.146	0.002
β_{12}	-0.0206	0.132	0.876

Figure 26: Regression parameter coefficients – Labour – Regional Dummy

The two tables (Figure 25 and 26) above show the coefficients for our parameters at the end, the highlighted values are not statistically significant.

Our β_1 parameters for Conservative and Labour stabilized even more, the slight uptrend in the end in β_1 for Labour had disappeared, and the jump around point 200 has decreased. Our β_1 parameters were quite stable around point 250 corresponding to around 2:30 in the evening, where the expected electoral votes for Conservative was stable.

Feature Selection with Best Subset Regression

Since some of the parameters were not statistically significant, we used best subsets regression with AIC (Akaike information criterion) as selection benchmark to find the optimal set of parameters. Best subsets regression uses all possible combinations of predictors to predict the result and evaluating the combinations based on a certain metric, in our case AIC where residual sum of squares and number of parameters are penalised. Using best subsets regression, we could select a combination of predictors that balanced model simplicity and predictive capability. The parameters that remained were:

Conservative: $\beta_0, \beta_1, \beta_2, \beta_4, \beta_6, \beta_7, \beta_9, \beta_{10}, \beta_{12}$

Labour: $\beta_0, \beta_1, \beta_2, \beta_3, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{11}$

With the statistically insignificant variable results removed, we had the following results:

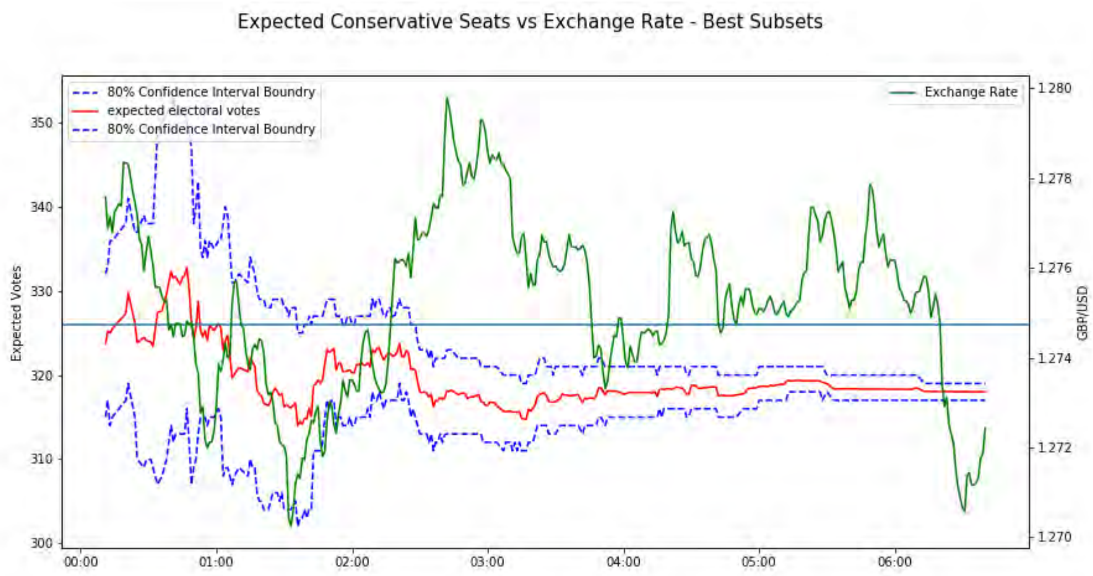


Figure 27: Expected Seats for Conservative and Exchange rate – Best subsets

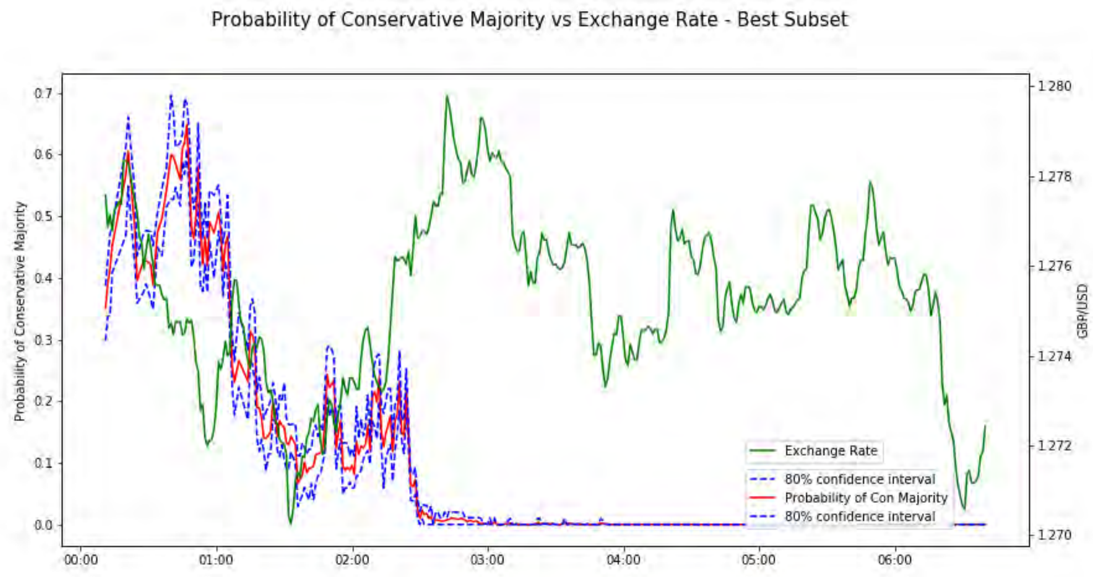


Figure 28: Probability of Conservative majority and exchange rate – Best Subset

Results remained similar to the results of the model without the statistically insignificant variables removed, but we reduced the number of parameters in our model and thus its complexity.

Conservative Regression Parameter:

Parameter Name	Coefficient	Standard Error	z-score	P Value
β_0	-0.0573	0.010	-5.830	0.000
β_1	0.0088	0.000	45.892	0.000
β_2	-0.0017	0.001	-2.338	0.019
β_4	0.0033	0.001	3.800	0.000
β_6	-8.188e-05	3.4e-05	-2.407	0.016
β_7	0.0001	1.85e-05	6.798	0.000
β_9	0.5456	0.057	9.560	0.000
β_{10}	0.8284	0.252	3.294	0.001
β_{12}	-0.3547	0.128	-2.769	0.006

Figure 29: Regression parameter coefficients – Conservative – Best subsets

Labour Regression Parameter:

Parameter Name	Coefficient	Standard Error	z-score	P Value
β_0	0.1696	0.021	8.065	0.000
β_1	0.0108	0.000	67.779	0.000
β_2	0.0007	0.000	2.516	0.012
β_3	-0.0014	0.001	-2.578	0.010
β_5	-5.149e-05	2.53e-05	-2.033	0.042
β_6	-3.16e-05	1.92e-05	-1.647	0.100
β_7	0.0003	8.6e-05	2.953	0.003
β_8	-0.0001	9.51e-06	-10.728	0.000
β_9	-0.4526	0.044	-10.184	0.000
β_{11}	-0.4594	0.146	-3.155	0.002

Figure 30: Regression parameter coefficients – Labour – Best subsets

β_1, polls, was the most important parameter in our regression, which was greater than 0.01 for Labour, and less than 0.01 for Conservative. This was the main reason Conservative lost majority.

β_2 the additional effect of London on polls was positive for Labour and negative for Conservative. This matched the strong Labour support we saw in London with more left leaning inhabitants, and the effect was even stronger than other regions than the polls suggested.

β_3 , the additional effect of Wales on polls was the opposite sign of β_2 for Labour. Wales was quite wealthy with low taxes, reduced support for Labour more was not surprising, polls in Wales for Labour was overestimated ($\beta_1 + \beta_3 < 1$). β_3 was removed during feature selection for Conservative

β_4, the additional effect of Scotland on polls, as removed during feature selection for Labour, but significant and positive for Conservative. In this election the SNP lost much ground to Conservative, which could explain why β_4 is quite large.

β_5, the wage's effect on results in 'other' region, negative for Labour and removed during feature selection for Conservative, matched the common political trend that the wealthy were less supportive of the left.

β_6, the additional effect of London on income was insignificant for Labour (but kept in the model according to the result of best subsets regression) but significant for Conservative, indicating wealthier voters were less likely to vote Conservative in London compared to other parts of the UK.

β_7, the additional effect of Wales on income, was significant and positive for both Labour and Conservative, implying wealthy voters in Wales prefer Labour or Conservative to other parties compared with other regions.

β_8, was removed for Conservative, but highly significant for labour, indicating wealthy individuals in Scotland prefer Labour less in Scotland compared to other parts of UK. This might be a reason why Labour failed to make big gains in Scotland.

β_9, age's effect on result in 'other' region, was negative for Labour and positive for Conservative, this also matches the common political trend that the elderly is less supportive of the left and more of the right.

β_{10} the additional effect of London on age was positive for Conservative, indicating that the elderly in London favoured Conservative more than other regions. This parameter was removed for Labour during feature selection.

β_{11} the additional effect of Wales on age was negative for Labour, indicating that the elderlies in Wales disliked Labour more than other regions. This parameter was removed for Conservative during feature selection.

β_{12} , the additional effect of Scotland on age was negative for Conservative, indicating that the elderlies in Scotland disliked Conservative more than other regions. This parameter was removed for Labour during feature selection.

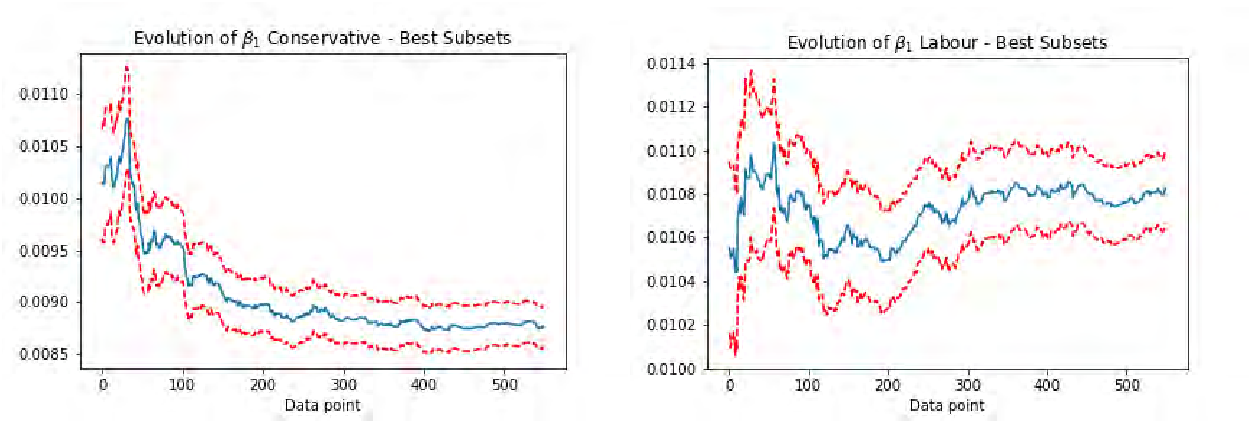


Figure 31: Evolution of β_1 parameters – Best subsets

Our β_1 parameters stabilised further. For Conservative, the slight down trend from point 200 in the previous model had mostly disappeared. However, were still not able to determine the cause of the jump in the β_1 for Labour around point 200.

Given the limited time we have, we were satisfied with the performance of this model.

Market Reaction

As we can observe, the market was quite late in reacting to the information in the UK election. Using our most sophisticated model, we observed that there was an arbitrage opportunity from 1:30 onwards, where our model predicts the chance of a conservative victory was very low (practically 0 after 2:30). We did not see the value of the pound drop to its lowest level until around 6:30.

Pound's Rise at 2:15

There was a sharp rise in the value of the pound at around 2:13 from 1.273 to 1.277. However, our model showed little movement this point in time as seen in figure 31. An investigation was conducted into the results in this time period (2:05 – 2:15).

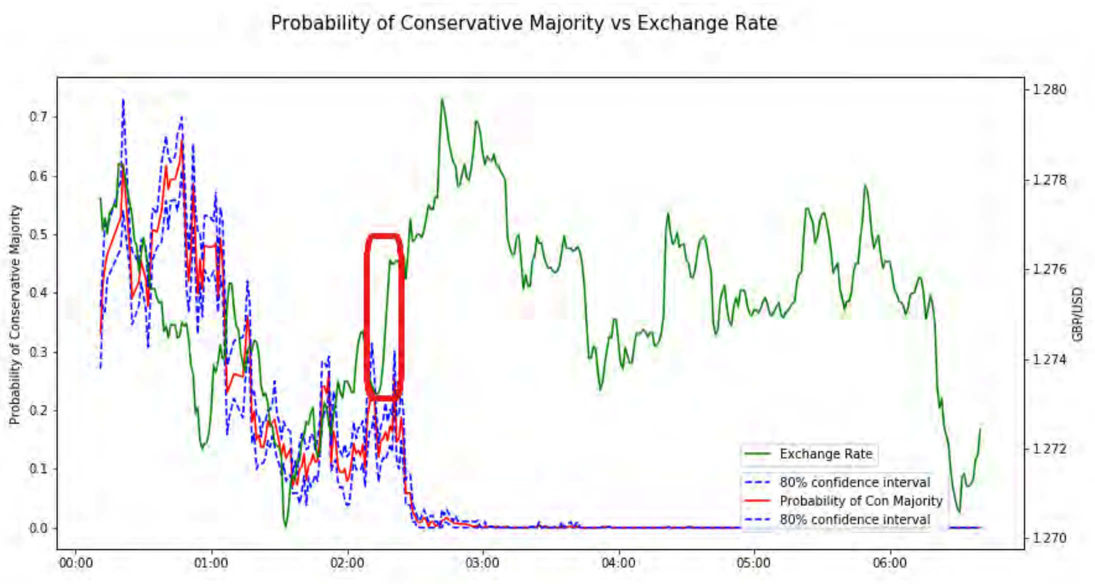


Figure 32: Jump in value of British Pound

We plotted the poll vs. actual results for Conservative during this time period to visualise the data and to spot any irregularities. Below (Figure 33) is a plot of actual results vs. polls.

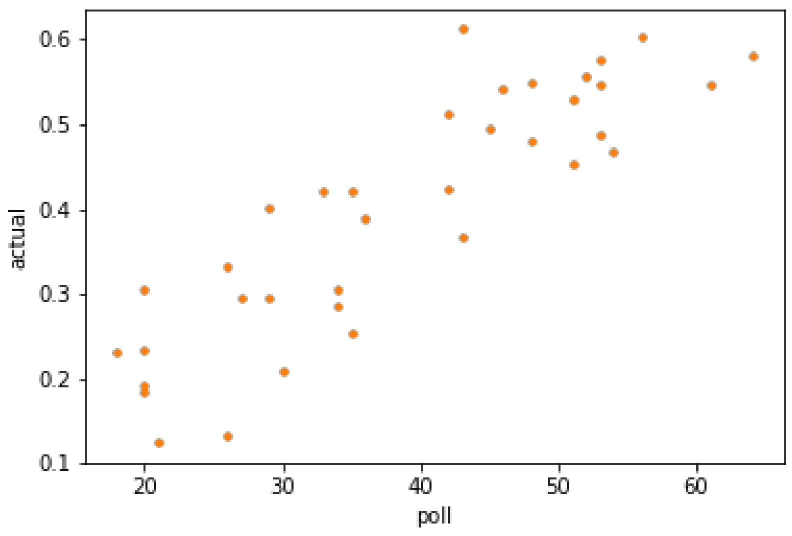


Figure 33: Actual Result vs Poll from 2:05 to 2:15

It seems that the point where Conservative polled at 43 percent but actually received 61 percent of the votes was causing traders to be more optimistic about a Conservative majority. This constituency was Clacton, a constituency that previously voted for UKIP but switched to Conservative. Our model predicted that Conservative would receive 50 percent of the vote with upper bound 95% confidence interval at 60 percent which is slightly lower than the actual result. Looking at the big picture, this was not considered to be an outlier data point, as it was the only data point above the 95% confidence interval bound and only by a small amount. By definition, we would expect 1 in 20 data point to be outside the 95% confidence interval bound. This one point did not change our model significantly.

The media, however, cheered for Conservative when the result became available, possibly causing the raise of the value of the British pound. Even though, it was not a big surprise when we look at the big picture.

''IT was third time lucky for Giles Watling as the Conservative swept Ukip aside to win the Clacton seat with a massive 15,828-vote majority." – Clacton Gazette [16]

Group Think Mentality

The market did not respond to the available information with the speed that we saw in the US election. One possible explanation for this phenomenon is the group think mentality.

Coming into the snap election, the conservatives were at a very strong position. They were expected to win the election by a landslide. Even as polls started to suggest the Conservative might lose majority close to the election, most forecasters refused to believe it.

Having such a large group of people believing that there is no possibility that the Conservative will not win the majority, investors realisation of the truth might have been delayed.

Such a phenomenon also occurred on the night of the Brexit referendum. The outcome of Brexit can be predicted very early on as the result of each constituency comes out. The drop in value of the British pound was delayed significantly though. [17]

Confirmation Bias

One other possible explanation of delay in market reaction can be the confirmation bias in finance. It occurs when we selectively collect evidence that support our previously held beliefs. When new evidence appears to contradict our beliefs, we tend to ignore it at first, only come to the realisation that our previous belief might be biased moments later. Such phenomenon is well documented in psychology where participants of studies only remember the data supporting their view points. [18]

This explanation can very well be at play here. When one data point arrives that was above our upper bound confidence interval of our predicted value (Clacton), the believers for Conservative majority cheered on and the pound's value rose significantly. However, this point was very close to our 95% confidence interval. And by definition, we expect 5% or 1 in 20 points to be outside our 95% confidence interval. And other points that support the model predicting a Conservative minority were ignored or weighted with less significance. Thus, leading a big jump in the value of the British pound but our model's prediction changed little.

Data Source

Polling data: obtained from http://www.electionforecast.co.uk. The polling data was a pool of polls compiled by experienced election forecaster Chris Hanretty. Using a pool of polls had the advantage of averaging out biases in each individual poll.

Result announcement and timestamp: We crawled BBC's Twitter account to get their announcements of election results of each region. The timestamp on Twitter was the time where the information became available for our model.

Demographic data: The demographic data used in our model was taken from the 2011 UK censes.

Outliers and Exceptions

We were not able to get polling data for the major parties in the region of Northern Ireland as Northern Ireland had its own popular parties such as the Democratic Unionist Party and Sinn Fein. Neither the Conservatives or Labour had a chance of winning a seat there in the 2017 election. Therefore, we assumed that the Conservatives would not win any seat in Northern Ireland in our model.

We used robust regression in our model to avoid outlier constituencies having a large effect on our model. Because there are 650 constituencies in the UK election. Investigating each point that seemed like an outlier to isolate the reason would be too time consuming.

Tools used

This project is written in Python. All statistical analysis is done using 'statsmodels' package. Code and data can be found here:

https://github.com/alexhuang1117/Data-Science-Portfolio/blob/master/FX_Analysis_during_UK_Election/main.ipynb

Conclusion and Future Work

In this paper, we investigated the 2016 US and 2017 UK election. Both elections had surprising results which were not predicted by most forecasted beforehand. We built a predictive model both for the US and UK election to see how fast we can predict the outcome of the result in real-time when only partial results of the election were available. We then compared the result with the currency exchange value between the US dollar and Mexican peso for the US election and the British pound Sterling and the US dollar during the UK election. The comparison gives us an idea about how fast the market was reacting to information and how efficient the market was during these two elections.

In the case of the US election, we observed that the market reacted quite quickly to the information. Our model was not able to outperform the market. Leading us to conclude the market was quite efficient during this period and reacted very quickly to information. However, our model was very simple due to time constraint and data availability. Given more data available and a more complex model, one might be able to find that currency exchange market was not behaving as efficiently as we proposed here.

In the case of the UK election, we started with a one-factor model whose performance is mediocre. We then looked into other factors that might influence the election and increased our model complexity. Our final model was able to predict the result of the election early and its parameters were stable early on during the night. Our model was able to beat the market by predicted the result quite early while the market seemed to be slow in digesting the result. We might attribute the markets slowness to adapt to new information by group think mentality and confirmation bias as most traders and forecasters have strong held believes that the conservatives would win majority in the UK election.

In the future, one can use this model for the 2020 US and 2021/2022 UK election to examine how well the model performs in real time.

References:

[1] Poll: Clinton, Trump most unfavorable candidates ever, USA Today, https://www.usatoday.com/story/news/politics/onpolitics/2016/08/31/poll-clinton-trump-most-unfavorable-candidates-ever/89644296/

[2] BBC debate: Rivals attack Theresa May over absence, BBC, http://www.bbc.com/news/election-2017-40105324

[3] About the Meta-Analysis, Princeton Election Consortium, http://election.princeton.edu/methods/

[4] How We're Forecasting The Presidential Election, Huffington Post https://www.huffingtonpost.com/entry/forecast-2016-president_us_57ee8eede4b0c2407cdd9155

[5] Why 2016 election polls missed their mark, Pew Research Center, http://www.pewresearch.org/fact-tank/2016/11/09/why-2016-election-polls-missed-their-mark/

[6] The U.K. Election Wasn't That Much Of A Shock, FiveThirtyEight, https://fivethirtyeight.com/features/uk-election-hung-parliament/

[7] A forecast of the 2017 UK general election, Medium, https://medium.com/@chrishanretty/a-forecast-of-the-2017-uk-general-election-8bb721a59eff

[8] Live Election Night Forecast, FiveThirtyEight, https://projects.fivethirtyeight.com/election-night-forecast-2016/

[9] Live Presidential Forecast, New York Times, https://www.nytimes.com/elections/forecast/president

[10] 2016 Presidential Election Live Results, 270TOWIN, https://www.270towin.com/live-2016-presidential-election-results/state-by-state/

[11] YouGov's poll predicting a hung parliament is certainly brave, The Guardian, https://www.theguardian.com/politics/2017/may/31/yougov-poll-predicting-hung-parliament-brave

[12] Who Will Be President?, New York Times, https://www.nytimes.com/interactive/2016/upshot/presidential-polls-forecast.html

[13] Where to get the best odds on the Conservatives in the General Election, Metro, http://metro.co.uk/2017/05/18/where-to-get-the-best-odds-on-the-conservatives-in-the-general-election-6646206/

[14] Truth-O-Meter, Politifact, http://www.politifact.com/truth-o-meter/promises/trumpometer/promise/1397/build-wall-and-make-mexico-pay-it/

[15] CNN Election Night 2016 Coverage Full Broadcast, YouTube, https://www.youtube.com/watch?v=w31B49W03io

[16] Conservative Giles Watling gains massive majority in Clacton as Ukip is beaten into third place, Clacton Gazette,

http://www.clactonandfrintongazette.co.uk/news/15337990.VIDEO___RESULT__Conservative_Giles_Watling_gains_massive_majority_in_Clacton_as_Ukip_is_beaten_into_third_place/

[17] The British Pound on Brexit Night: A Natural Experiment of Market Efficiency and Real-Time Predictability, Swiss Finance Institute Research Paper, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2940173##

[18] What Is Confirmation Bias?, Psychology Today, https://www.psychologytoday.com/blog/science-choice/201504/what-is-confirmation-bias

1 Appendix A: USD/MXN Exchange Rate during 2016 US Election Night

During the night of the 2016 US election night, we saw Donald Trump won the presidency while most polling agency predicted Hillary Clinton would be victorious. This came as a suprise to the most people. In this project I would like to how the financial market reacted during the election night.

1.1 Background:

During the campaign Donald Trump repeatedly mentioned building a wall between the US and Mexico boarder, putting high import tariff on imported goods from Mexico and renegotiate or abandon NAFTA. All such action are seen as negatives for the Mexican economy whose biggest export market is the US. It is no suprise that in the event of a trump victory, the Mexican Peso will depreciate against other currencies.

In this project I will run a realtime analysis on Trumps victory chance and see how fast the financial market reacted as the result came out from each state.

```
In [299]: # -*- coding: utf-8 -*-
          """
          Created on Wed Aug  2 16:32:27 2017

          @author: AlexH
          """

          #Priors 1: state, number


          #Updates & Results: state, time, number

          #Step 1, get all data (use just website time info for now)

          #Step 2, rolling regression

          #Step 3, find error margin at each stage

          #Step 4, compare with financial data


          import numpy as np
          import csv
          import matplotlib.pyplot as plt
          from sklearn import linear_model, datasets
          import statsmodels.api as sm
```

```
from statsmodels.regression.linear_model import RegressionResults
import random

import pandas as pd
from collections import OrderedDict
from datetime import date

from datetime import datetime
import matplotlib.dates as mdates
import pylab
```

1.1.1 Getting the Data

The data is collected from the internet, polling data is from Fivethrityeight 'prior-538.csv' and the election realtime data is recorded by watching BBC/CNN election live broadcast 'results.csv'.

```
In [300]: #%%
          #Reading in data


          # with open('data/prior_538.csv', newline='') as csvfile:
          #     data = csv.reader(csvfile, delimiter=',')
          #     prior=list(data)
          prior = pd.read_csv('data/prior_538.csv')

          prior = pd.DataFrame(prior)
          prior.columns = ['State', 'poll_hc', 'poll_dt', 'poll_gj']
          prior.State=[name.replace('-',' ') for name in prior.State]

          #with open('call-AP.csv', newline='') as csvfile:
          #    data = csv.reader(csvfile, delimiter=',')
          #    call=list(data)
          #
          #call = pd.DataFrame(call)
          #call.columns = ['State', 'result', 'time']

          with open('data/results CNN-2.csv', newline='') as csvfile:
              data = csv.reader(csvfile, delimiter=',')
```

```
            result=list(data)

        result = pd.DataFrame(result)
        result.columns = ['State', 'result', 'time', 'trump', 'clinton', 'EV', 'intermediate','rprpc']
        result.State=[name.lower() for name in result.State]


        #join the table by state
        jointb = prior.join(result.set_index('State'), on='State')
        #order by time




        for i in range(len(jointb)):
            try:
                jointb.time.iloc[i]=datetime.strptime(jointb.time.iloc[i], '%Y-%m-%d %H:%M')
            except:
                jointb.time.iloc[i]=datetime.strptime(jointb.time.iloc[i], '%Y-%m-%d %I:%M%p')

        jointb = jointb.sort_values(by='time').reset_index(drop=True)
        jointb.index = np.arange(0,len(jointb))
        jointb[[ 'EV','result', 'trump', 'clinton', 'poll_hc', 'poll_dt', 'poll_gj','rprpc']]=jointb[[ 'EV','result', 'trump', 'clinton','poll_hc
        #jointb.EV.fillna(0,inplace =True)
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\pandas\core\indexing.py:179: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

1.1.2 Linear Regression and Monte Carlos Simulation

```
In [301]: #storing result
          columns = ['time', 'state','low', 'ave', 'high','clinton']
          index= jointb.index
          pred=pd.DataFrame(index=index, columns=columns)
```

```
parameters_trump = np.empty((len(jointb),2))
stderr_trump = np.empty((len(jointb),2))
parameters_clinton = np.empty((len(jointb),2))
stderr_clinton = np.empty((len(jointb),2))

#number of simulation for (a,b) and (error)
sim_num = 200
for i in range(14,len(jointb)-4):

    #Trump regression
    #fit linear regression
    select=jointb[0:i]
    full_result=select[select.intermediate!='1'].drop_duplicates(subset='State', keep='last')
    intermediate = select[select.intermediate=='1'].drop_duplicates(subset='State', keep='last')
    #intermediate = intermediate[intermediate.rprpc>20]

    X_dt=intermediate['poll_dt']
    if len(X_dt)<4:
        continue
    Y_dt=intermediate['trump']
    #X_dt=sm.add_constant(X_dt)
    model_dt = sm.RLM(Y_dt.values, X_dt.values,weights=intermediate['rprpc']).fit()

    parameters_trump[i,:]=model_dt.params
    stderr_trump[i,:]=model_dt.bse

    #extract parameters
    params_dt = np.random.multivariate_normal(model_dt.params, RegressionResults.cov_params(model_dt), sim_num**2)
#     slope_dt = np.random.normal(model_dt.params[1], model_dt.bse[1], sim_num)

    #Clinton regression
    X_hc=intermediate['poll_hc']
    Y_hc=intermediate['clinton']
    #X_hc=sm.add_constant(X_hc)
    model_hc = sm.WLS(Y_hc.values, X_hc.values,weights=intermediate['rprpc']).fit()
    parameters_clinton[i,:]=model_hc.params
    stderr_clinton[i,:]=model_hc.bse
    params_hc = np.random.multivariate_normal(model_hc.params, RegressionResults.cov_params(model_hc), sim_num**2)


    #make predictions, use states with no results and intermediate results
    remain=jointb[~jointb.State.isin(full_result.State)].drop_duplicates(subset='State', keep='last')

        #trump
    #result_dt = np.dot(params_dt,sm.add_constant(np.array(remain['poll_dt'])).T)
    result_dt = np.dot(params_dt,np.matrix(remain['poll_dt']))
    #result_dt = np.random.normal(result_dt, np.std(model_dt.resid))
    err_dev = np.std(model_dt.resid)*(np.random.chisquare(len(model_dt.resid)-1)*1./(len(model_dt.resid)-1))**0.5
    result_dt = np.random.normal(result_dt, err_dev)


        #clinton
    #result_hc = np.dot(params_hc, sm.add_constant(np.array(remain['poll_hc'])).T)
    result_hc = np.dot(params_hc, np.matrix(remain['poll_hc']))
    #result_hc = np.random.normal(result_hc, np.std(model_hc.resid))
    err_dev = np.std(model_hc.resid)*(np.random.chisquare(len(model_hc.resid)-1)*1./(len(model_hc.resid)-1))**0.5
    result_hc =  np.random.normal(result_hc, err_dev)


    result = result_dt>result_hc


    EEV=sum(full_result['result'] * full_result['EV'])+ np.dot(result, remain['EV'])
    EEV_clinton=sum((1-full_result['result']) * full_result['EV'])+ np.dot((1-result), remain['EV'])
   # print(EEV+EEV_clinton)
#     print(jointb['time'][i],jointb['State'][i], EEV)

    #make table for plot

    pred.time[i]=jointb.time[i]
    pred.state[i]=jointb.State[i]
    pred.low[i]=np.percentile(EEV,10)
```

```
            pred.ave[i]=np.mean(EEV)
            pred.high[i]=np.percentile(EEV,90)
            pred.clinton[i]=np.mean(EEV_clinton)
            #print(pred.time[i])
            #print(model_dt.summary())

        # keep only the last value at a certain time. And remove NAs.
        pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
        pred.index = np.arange(0,len(pred))
```

```
In [302]: #%%
        # read the USD/MXN data
        # data found here http://www.histdata.com/download-free-forex-historical-data/?/metatrader/1-minute-bar-quotes/usdmxn/2016
        # my_data = np.genfromtxt('DAT_MT_USDMXN_M1_2016.csv', delimiter=',')


        with open('data/DAT_MT_USDMXN_M1_2016.csv',newline='') as csvfile:
            data = csv.reader(csvfile, delimiter=',')
            fx=list(data)

        fx = pd.DataFrame(fx)

        #the sixth column is 0, drop it
        fx = fx.drop(fx.columns[6], 1)

        # average the minute values.
        fx['mean'] = fx.ix[:,2:5].astype(float).mean(axis=1)

        # extract time value to datetime format
        fx['time'] = fx.ix[:,0]+fx.ix[:,1]
        fx['time'] = [datetime.strptime(v, '%Y.%m.%d%H:%M') for v in fx['time']]

        fx = fx.set_index(['time'])
        fx = fx.loc[pred.time[0]:pred.time[len(pred)-1]]
        fx = fx.drop(fx.columns[0:6],1)
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:17: DeprecationWarning:
.ix is deprecated. Please use
```

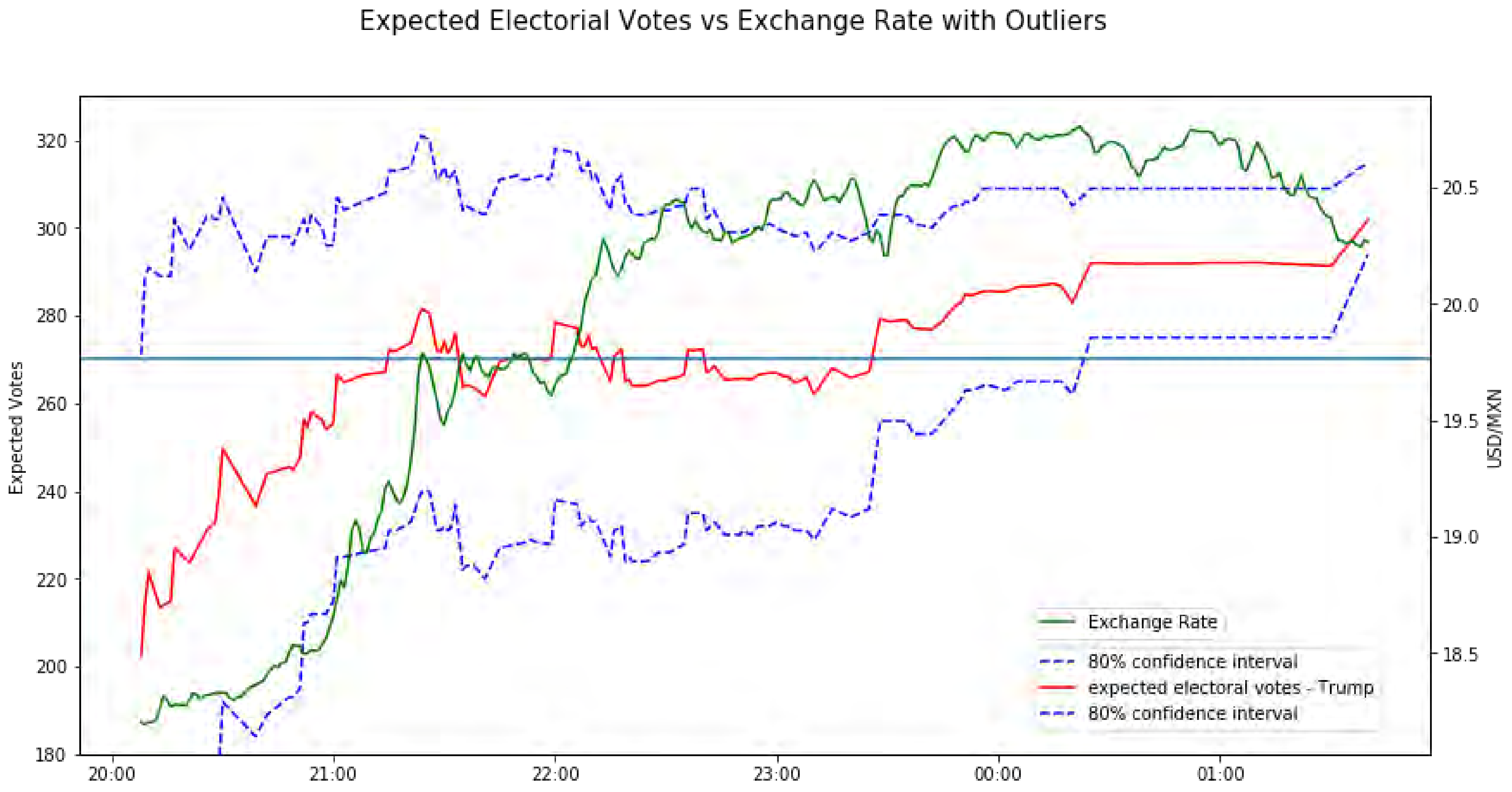
```
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate_ix
```

```
In [ ]:
```

1.1.3 Plots

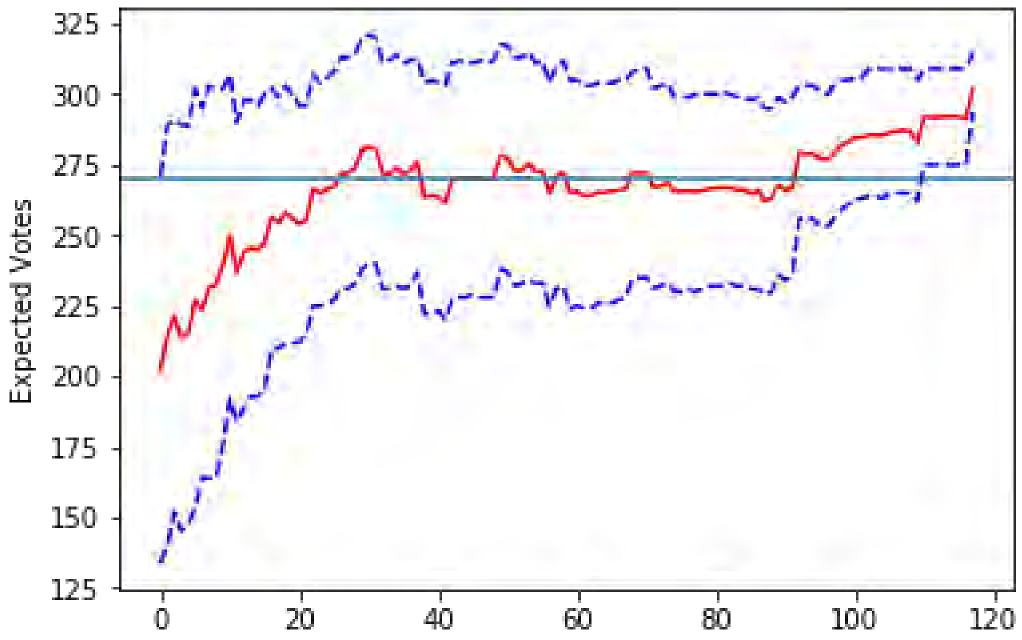
```
In [303]: #%%
          fig, ax1 = plt.subplots()
          fig.set_size_inches(14, 7)
          ax1.plot_date(pred.time, pred.low,'b--', label='80% confidence interval')
          ax1.plot_date(pred.time, pred.ave,'r-', label='expected electoral votes - Trump')
          #ax1.plot_date(pred.time, pred.clinton,'c-', label='expected electoral votes - Clinton')
          ax1.plot_date(pred.time, pred.high,'b--', label='80% confidence interval')
          plt.ylabel('Expected Votes')
          #pylab.legend(loc='lower right')
          plt.axhline(y=270)
          ax1.legend(loc='center left', bbox_to_anchor=(0.7, 0.1))
          ax1.set_ylim([180,330])

          ax2 = ax1.twinx()
          ax2.plot(fx['mean'],'g', label='Exchange Rate')

          HMFmt = mdates.DateFormatter('%H:%M')
          ax1.xaxis.set_major_formatter(HMFmt)
          _ = plt.xticks(rotation=90)

          plt.ylabel('USD/MXN')

          #pylab.legend(loc='upper right')
          ax2.legend(loc='center left', bbox_to_anchor=(0.7, 0.2))

          fig.suptitle('Expected Electorial Votes vs Exchange Rate with Outliers', fontsize=15)
```

```
        #plt.figure(figsize=(20,10))
        plt.show()
        fig.savefig('results.png')
```

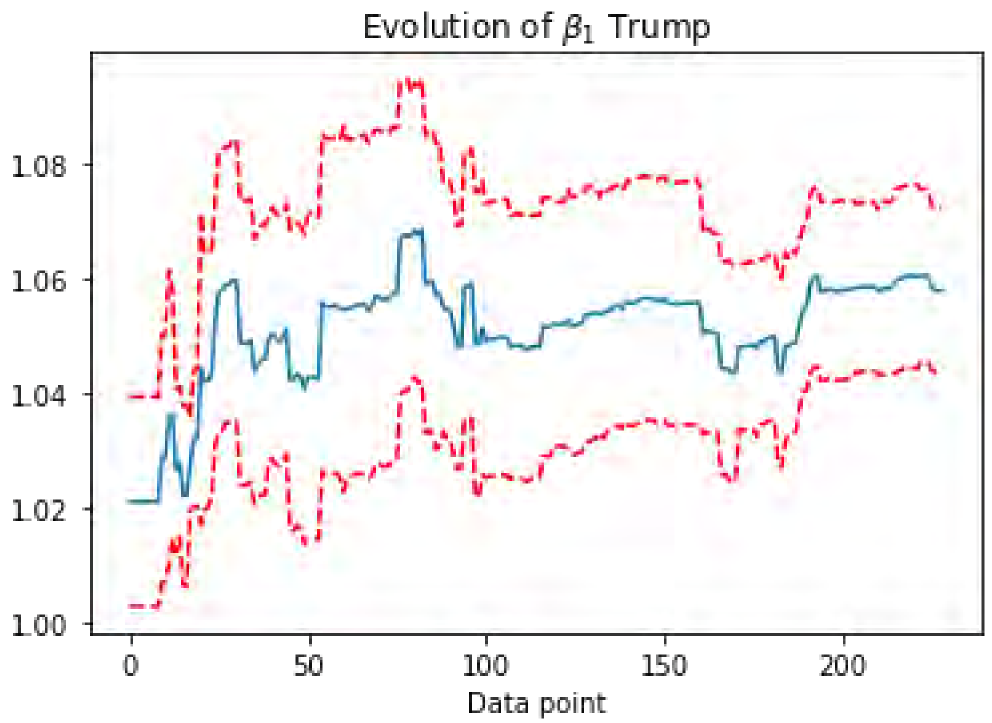


```
In [304]: #plot after each update
          fig, ax1 = plt.subplots()
          ax1.plot(pred.low,'b--', label='80% confidence')
```

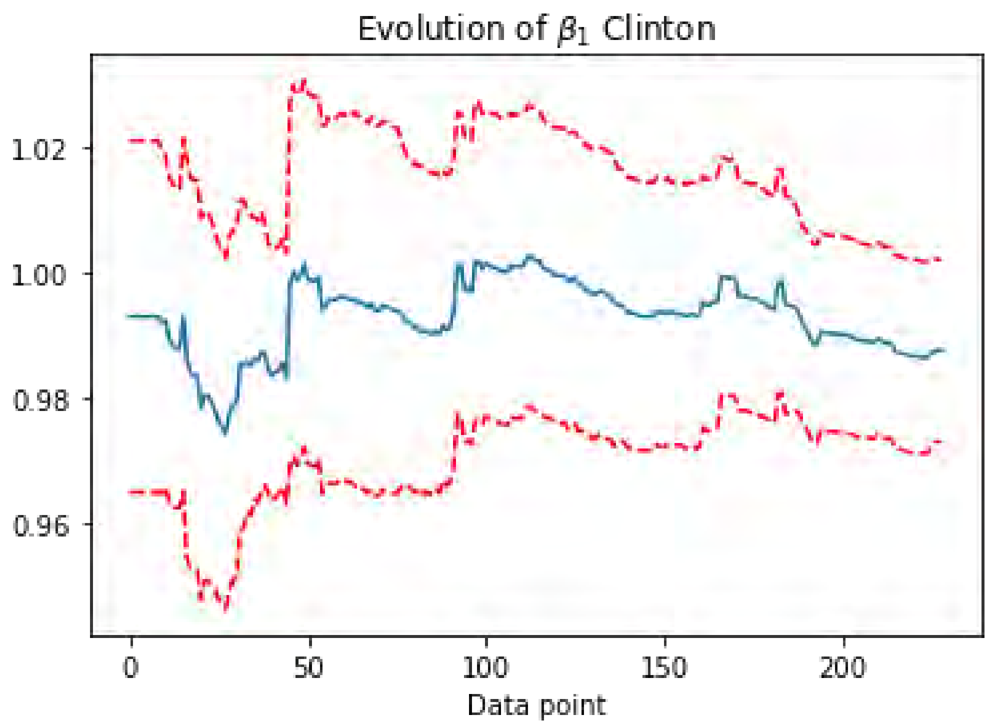
```
         ax1.plot(pred.ave,'r-', label='expected electoral votes')
         ax1.plot(pred.high,'b--', label='80% confidence')
         plt.ylabel('Expected Votes')
         plt.axhline(y=270)
         plt.show()
```



```
In [305]: plt.plot(parameters_trump[50:len(jointb)-20,1])
          plt.plot(parameters_trump[50:len(jointb)-20,1]-stderr_trump[50:len(jointb)-20,1],'r--')
          plt.plot(parameters_trump[50:len(jointb)-20,1]+stderr_trump[50:len(jointb)-20,1],'r--')
          plt.title(r'Evolution of $\beta_1 $ Trump')
          plt.xlabel('Data point')
          plt.savefig('trump b1.png')
          plt.show()
```



```
In [306]: plt.plot(parameters_clinton[50:len(jointb)-20,0])
          plt.plot(parameters_clinton[50:len(jointb)-20,1]-stderr_clinton[50:len(jointb)-20,1],'r--')
          plt.plot(parameters_clinton[50:len(jointb)-20,1]+stderr_clinton[50:len(jointb)-20,1],'r--')
          #plt.axhline(y=1)
          plt.title(r'Evolution of $\beta_1 $ Clinton')
          plt.xlabel('Data point')
          plt.savefig('clinton b1.png')
          plt.show()
```



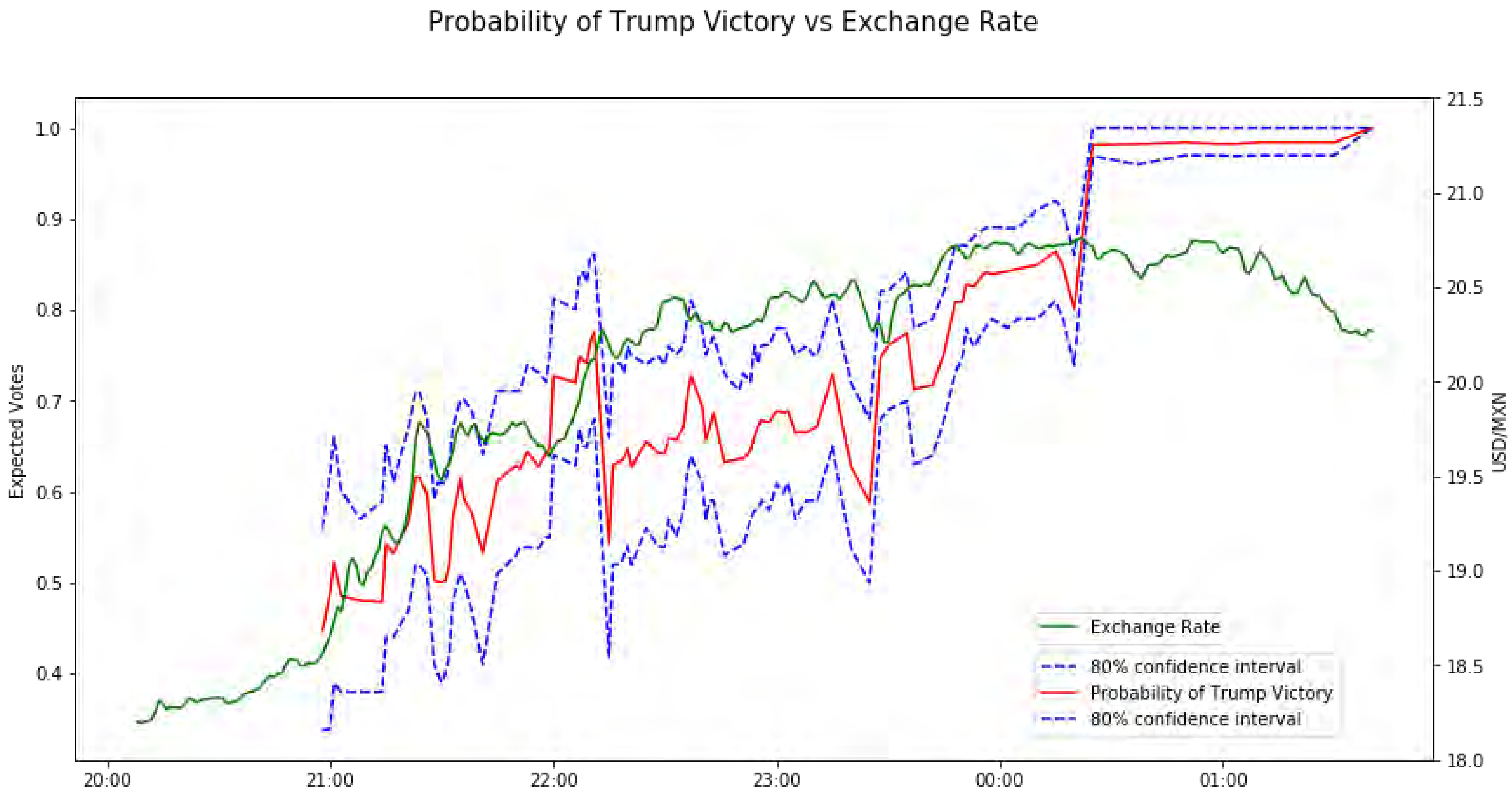
```
In [307]: #storing result
          columns = ['time', 'state','low', 'ave', 'high']
          index= jointb.index
          pred=pd.DataFrame(index=index, columns=columns)

          #number of simulation for (a,b) and (error)
          sim_num = 10


          for i in range(50,len(jointb)-4):
```

```
#Trump regression
#fit linear regression
select=jointb[0:i]
full_result=select[select.intermediate!='1'].drop_duplicates(subset='State', keep='last')
intermediate = select[select.intermediate=='1'].drop_duplicates(subset='State', keep='last')
#intermediate = intermediate[intermediate.rprpc>20]
intermediate = intermediate[intermediate.State!='new york']
intermediate = intermediate[intermediate.State!='montana']
intermediate = intermediate[intermediate.State!='utah']
intermediate = intermediate[intermediate.State!='texas']
intermediate = intermediate[np.bitwise_and(intermediate.State!='georgia' , intermediate.rprpc!=7)]

if not intermediate[intermediate.State=='iowa'].empty and (intermediate[intermediate.State=='iowa']['rprpc']<50).bool():
    intermediate = intermediate[intermediate.State!='iowa']

X_dt=intermediate['poll_dt']
if len(X_dt)<4:
    continue
Y_dt=intermediate['trump']
#X_dt=sm.add_constant(X_dt)
model_dt = sm.WLS(Y_dt.values, X_dt.values,weights=intermediate['rprpc']).fit()

#Clinton regression
X_hc=intermediate['poll_hc']
Y_hc=intermediate['clinton']
#X_hc=sm.add_constant(X_hc)
model_hc = sm.WLS(Y_hc.values, X_hc.values,weights=intermediate['rprpc']).fit()


prob = np.zeros(100)
for k in range(0,100):
#extract parameters
    params_dt = np.random.multivariate_normal(model_dt.params, RegressionResults.cov_params(model_dt), sim_num)
#    slope_dt = np.random.normal(model_dt.params[1], model_dt.bse[1], sim_num)
```

```
        #extract parameters
        params_hc = np.random.multivariate_normal(model_hc.params, RegressionResults.cov_params(model_hc), sim_num)
    #    slope_hc = np.random.normal(model_hc.params[1], model_hc.bse[1], sim_num)

        #make predictions, use states with no results and intermediate results
        remain=jointb[~jointb.State.isin(full_result.State)].drop_duplicates(subset='State', keep='last')

        result = np.zeros([sim_num, len(remain)])



        for j in range(0, sim_num):

            #trump
            #result_dt = np.dot(params_dt,sm.add_constant(np.array(remain['poll_dt'])).T)
            result_dt = np.dot(params_dt,np.matrix(remain['poll_dt']))
            err_dev = np.std(model_dt.resid)*(np.random.chisquare(len(model_dt.resid)-2)*1./(len(model_dt.resid)-2))**0.5
            result_dt = np.random.normal(result_dt, err_dev)


            #clinton
            #result_hc = np.dot(params_hc, sm.add_constant(np.array(remain['poll_hc'])).T)
            result_hc = np.dot(params_hc, np.matrix(remain['poll_hc']))
            err_dev = np.std(model_hc.resid)*(np.random.chisquare(len(model_hc.resid)-2)*1./(len(model_hc.resid)-2))**0.5
            result_hc =  np.random.normal(result_hc, err_dev)

            result_temp = result_dt>result_hc
            result = np.append(result, result_temp,axis=0)

        #remove the zeros during initialisation
        result = np.delete(result,range(0,sim_num),0)
        result = result.astype(int)


    #filling the result of intermediates states with current count
#    select.result[select.result.isnull()[0:i]]=(select['trump'][select.result.isnull()[0:i]]>=select['clinton'][select.result.isnull()[0
            EEV=sum(full_result['result'] * full_result['EV'])+ np.dot(result, remain['EV'])

            prob[k] = sum(EEV>270)/len(EEV)


        #make table for plot

        pred.time[i]=jointb.time[i]
        pred.state[i]=jointb.State[i]
        pred.low[i]=np.percentile(prob,10)
        pred.ave[i]=np.mean(prob)
        pred.high[i]=np.percentile(prob,90)

    # keep only the last value at a certain time. And remove NAs.
    pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
    pred.index = np.arange(0,len(pred))
```

In [308]:
```
#%%
fig, ax1 = plt.subplots()
fig.set_size_inches(14, 7)
ax1.plot_date(pred.time, pred.low,'b--', label='80% confidence interval')
ax1.plot_date(pred.time, pred.ave,'r-', label='Probability of Trump Victory')
ax1.plot_date(pred.time, pred.high,'b--', label='80% confidence interval')
plt.ylabel('Expected Votes')
#pylab.legend(loc='lower right')
#plt.axhline(y=270)
ax1.legend(loc='center left', bbox_to_anchor=(0.7, 0.1))
#ax1.set_ylim([0.5,0.9])

ax2 = ax1.twinx()
ax2.plot(fx['mean'],'g', label='Exchange Rate')
ax2.set_ylim([18,21.5])

HMFmt = mdates.DateFormatter('%H:%M')
ax1.xaxis.set_major_formatter(HMFmt)
_ = plt.xticks(rotation=90)

plt.ylabel('USD/MXN')
```

```
#pylab.legend(loc='upper right')
ax2.legend(loc='center left', bbox_to_anchor=(0.7, 0.2))

fig.suptitle('Probability of Trump Victory vs Exchange Rate', fontsize=15)

#plt.figure(figsize=(20,10))
plt.show()
fig.savefig('prob.png')
```



1 Appendix B: GBP/USD Exchange Rate during 2017 UK Election Night

```
In [186]: import numpy as np
          import csv
          import matplotlib.pyplot as plt
          from sklearn import linear_model, datasets
          import statsmodels.api as sm
          from statsmodels.regression.linear_model import RegressionResults
          import random

          import pandas as pd
          from collections import OrderedDict
          from datetime import date

          from datetime import datetime, timedelta
          import matplotlib.pyplot as plt
          import matplotlib.dates as mdates
          import pylab

          import tweepy

          import json

          import re
          from fuzzywuzzy import fuzz
          from fuzzywuzzy import process
```

```
In [187]: CONSUMER_KEY = '4BByuBKYk19fpSl5iMIkju3c0'
          CONSUMER_SECRET = '2EK91aT0s7uMJ1oWECBRUwkXrxGykigrsmtqtOIAvFBPXiucQq'
          ACCESS_TOKEN = '892729320736739328-E30nIY5dacqxeugxPoe3TXB2fIjITZB'
          ACCESS_TOKEN_SECRET = 'WMViNA7y1d1trkb5nt7L5dOAHdScmYBMHm33sLeUVZrWT'


          auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
          auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
          api = tweepy.API(auth)

          tweets_raw = api.user_timeline(screen_name = 'bbcelection', count = 200, include_rts = False)
```

```
        for i in range(0,5):
            oldest=tweets_raw[-1].id
            new_tweets = api.user_timeline(screen_name = 'bbcelection',count=200,max_id=oldest)
            tweets_raw.extend(new_tweets)

        data = [[tw.created_at.year, tw.created_at.month, tw.created_at.day,"%s.%s"%(tw.created_at.hour, tw.created_at.minute), tw.id_str, tw.tex
        tweets=pd.DataFrame(data, columns=['year','month','date','time','tweet_id','tweet'])
        tweets = tweets[tweets.year==2017]
```

```
In [188]: # Wikipedia data
          UKpoll = pd.read_csv('data/UK2017Poll.txt', sep='\t', header=0)
          UKpoll.columns=['ID','Con_poll', 'Lab_poll', 'Lib_poll','SNP_poll','Pla_poll','Greens_poll',\
                          'UKIP_poll', 'Other_poll', 'Seat','Region','2015']
          results = pd.read_csv("data/result.csv")
```

```
In [189]: tweets.tweet=tweets.tweet.astype(str)
          tweets_cleaned = tweets[tweets.tweet.str.contains('#GE2017')]
          tweets_cleaned['time_full'] = tweets_cleaned["year"].map(str)+ "/"+ tweets_cleaned["month"].map(str) + \
              "/"+ tweets_cleaned["date"].map(str)  + "/"+ tweets_cleaned["time"].map(str)
```

1.1 Merging result data with time

```
In [190]: tweets_cleaned['Constituency']=np.nan
          for i in range(len(tweets)):
              tweets_cleaned.Constituency[i] = tweets_cleaned.tweet[i][tweets_cleaned.tweet[i].find("\'")+1:tweets_cleaned.tweet[i].find(":")]
          tweets_cleaned.to_csv('data/tweets.csv',sep=',')
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\AlexH\Anaconda3\lib\site-packages\pandas\core\indexing.py:179: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

```
In [191]: #manual matching
          tw_matched = pd.read_csv('data/tweets_matched.csv')
          tw_matched = tw_matched.dropna(axis=0, how='any')
          tw_matched = tw_matched.drop_duplicates(subset='ID', keep='last')
          tw_matched.index = np.arange(len(tw_matched))
          tw_matched.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 642 entries, 0 to 641
Data columns (total 9 columns):
year            642 non-null int64
month           642 non-null int64
date            642 non-null int64
time            642 non-null float64
tweet_id        642 non-null float64
tweet           642 non-null object
time_full       642 non-null object
Constituency    642 non-null object
ID              642 non-null object
dtypes: float64(2), int64(3), object(4)
memory usage: 50.2+ KB
```

```
In [192]: results = results.merge( tw_matched[['ID','time_full']], how='left', left_on = 'ID', right_on='ID')
          results
          results[['Con[b]','Lab[c]','LD','SNP','UKIP','Grn[d]','DUP']]=(results[['Con[b]','Lab[c]','LD','SNP','UKIP','Grn[d]','DUP']].T/results['T
          results['time']=np.nan
          for i in range(0,len(results)):
              try:
                  results['time'].loc[i]= datetime.strptime(results.time_full[i],'%Y/%m/%d/%H.%M')
              except:
                  results['time'].loc[i] = np.nan
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\pandas\core\indexing.py:179: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

1.2 Merging poll with result

```
In [193]: full_table = pd.merge(results, UKpoll,  how='left', left_on='ID', right_on = 'ID')
          full_table = full_table.sort_values(by='time').reset_index(drop=True)
```

1.3 Other factors

taken from UK sensus

```
In [194]:     #    {nan,
              #'West Midlands',
              # 'Scotland',
              # 'South East',
              # 'Yorkshire and The Humber',
              # 'Wales',
              # 'South West',
              # 'East of England',
              # 'London',
              # 'North West',
              # 'North East',
              # 'East Midlands'}

          xl = pd.ExcelFile("data/Wages.xlsx")
          wages=xl.parse("Data")
          wages.head()
          full_table = pd.merge(full_table, wages[['ONSConstID','WageMedianConst']],  how='left', left_on='ID', right_on = 'ONSConstID')

          xl = pd.ExcelFile("data/Business-numbers.xlsx")
          business=xl.parse("Data")
          #full_table = pd.merge(full_table, business,  how='left', left_on='Constituency', right_on = 'ConstituencyName')

          xl = pd.ExcelFile("data/Population-by-age.xlsx")
          population=xl.parse("Data")
          full_table = pd.merge(full_table, population[['ONSConstID','Pop65ConstRate']],  how='left', left_on='ID', right_on = 'ONSConstID')

          full_table['islab'] = (full_table['Last Election']=='Lab').astype(int)
          full_table['iscon'] = (full_table['Last Election']=='Con').astype(int)
          full_table['islib'] = (full_table['Last Election']=='LD').astype(int)
```

```
        full_table['issnp'] = (full_table['Last Election']=='SNP').astype(int)
        full_table['london'] = (full_table['Region']=='London').astype(int)
        full_table['Turnout'] = [float(v.rstrip("%")) for v in full_table.turnout]
        full_table['iswales'] = (full_table['Region']=='Wales').astype(int)
        full_table['isscot'] = (full_table['Region']=='Scotland').astype(int)
```

```
In [ ]:
```

2 Analysis

2.1 Single Factor

```
In [195]: columns = ['time', 'Constituency','low', 'ave', 'high']
          index= full_table.index
          pred=pd.DataFrame(index=index, columns=columns)

          parameters_con = np.empty((650,2))
          stderr_con = np.empty((650,2))
          parameters_lab = np.empty((650,2))
          stderr_lab = np.empty((650,2))
          parameters_snp = np.empty((650,2))
          stderr_snp = np.empty((650,2))

          sim_num = 50
          for i in range(18,len(full_table)-14):

              #labour regression
             # data_lab=full_table[['Lab_poll','Lab[c]','Total','Last Election','WageMedianConst','Pop65ConstRate']]
              data_lab= pd.concat([full_table['Lab_poll'],\
                                   full_table['Lab[c]']], axis=1)
              #,full_table['Con_poll']*full_table['iscon'],
              X_lab = data_lab.drop('Lab[c]',axis=1)[0:i]
              Y_lab=data_lab['Lab[c]'][0:i]
              X_lab=sm.add_constant(X_lab, has_constant='add')
              model_lab = sm.RLM(Y_lab, X_lab,missing = 'drop').fit()

              parameters_lab[i,:]=model_lab.params
```

```
stderr_lab[i,:]=model_lab.bse

#Conservatives regression
#data_con=full_table[['Con_poll','Con[b]','Total','Last Election','WageMedianConst','Pop65ConstRate']][0:i].dropna(axis=0)
data_con= pd.concat([full_table['Con_poll'],\
                     full_table['Con[b]']], axis=1)
X_con = data_con.drop(['Con[b]'], axis = 1)[0:i]
Y_con=data_con['Con[b]'][0:i]
X_con=sm.add_constant(X_con, has_constant='add')
model_con = sm.RLM(Y_con, X_con,missing = 'drop').fit()

parameters_con[i,:]=model_con.params
stderr_con[i,:]=model_con.bse


data_snp=pd.concat([full_table['SNP_poll'], full_table['SNP']], axis=1)
X_snp = data_snp.drop(['SNP'],axis = 1)[0:i]
Y_snp=data_snp['SNP'][0:i]
X_snp=sm.add_constant(X_snp, has_constant='add')
try:
    model_snp = sm.RLM(Y_snp, X_snp,missing = 'drop').fit()
    parameters_snp[i,:]
    stderr_snp[i,:]=model_snp.bse
except:
    model_snp = 0




predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')
predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]
result = np.zeros([sim_num, len(predict_lab)])

#Sampling the regression parameters to generate predicted outcome
for j in range(0, sim_num):

    #Labour
```

```
param_lab = np.random.multivariate_normal(np.asarray(model_lab.params), RegressionResults.cov_params(model_lab), sim_num)
predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')


#fill NaN with mean
#predict_lab = predict_lab.fillna(predict_lab.mean())
predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]

result_temp =  np.matmul(param_lab,np.array(predict_lab).T)
#result_lab = np.random.normal(result_temp,np.std(model_lab.resid))
err_dev = np.std(model_lab.resid)*(np.random.chisquare(len(model_lab.resid)-2)*1./(len(model_lab.resid)-2))**0.5
result_lab = np.random.normal(result_temp,err_dev)


#Conservative
param_con = np.random.multivariate_normal(np.asarray(model_con.params), RegressionResults.cov_params(model_con), sim_num)
predict_con=sm.add_constant(data_con.drop(['Con[b]'], axis = 1)[i:], has_constant='add')


#predict_con = predict_con.fillna(predict_con.mean())
predict_con = predict_con[np.isfinite(predict_con['Con_poll'])]

result_temp = np.matmul(param_con,np.array(predict_con).T)
#result_con = np.random.normal(result_temp,np.std(model_con.resid))
err_dev = np.std(model_con.resid)*(np.random.chisquare(len(model_con.resid)-2)*1./(len(model_con.resid)-2))**0.5
result_con = np.random.normal(result_temp,err_dev)





#SNP
predict_snp=sm.add_constant(data_snp.drop(['SNP'],axis = 1)[i:], has_constant='add')
predict_snp = predict_snp[np.isfinite(predict_snp['SNP_poll'])]
if not model_snp==0 and len(model_snp.resid)>8:
    param_snp = np.random.multivariate_normal(np.asarray(model_snp.params), RegressionResults.cov_params(model_snp), sim_num)
```

```
            #predict_snp = predict_snp.fillna(predict_snp.mean())

            result_temp =  np.matmul(param_snp,np.array(predict_snp).T)
            #result_snp = np.random.normal(result_temp,np.std(model_snp.resid))
            err_dev = np.std(model_snp.resid)*(np.random.chisquare(len(model_snp.resid)-2)*1./(len(model_snp.resid)-2))**0.5
            result_snp = np.random.normal(result_temp,err_dev)
            #if the poll is 0, the resulting simulated data should also be 0
            ind=np.where(predict_snp.SNP_poll==0)
            result_snp[:,ind]=0

        else:
            #no data
            result_snp = np.tile(predict_snp['SNP_poll'],(sim_num,1))/100

        #result_pla = np.nan_to_num(np.tile(full_table['Pla_poll'][i:],(sim_num,1)))/100
        #result_ukip = np.nan_to_num(np.tile(full_table['UKIP_poll'][i:],(sim_num,1)))/100
        other = 1-result_con-result_lab-result_snp

        result_temp = (result_con>result_lab) & (result_con>result_snp) & (result_con>other)
        result = np.append(result, result_temp,axis=0)

    #remove the zeros during initialisation
    result = np.delete(result,range(0,sim_num),0)
    result = result.astype(int)

    #filling the result of intermediates states with current count
#    select.result[select.result.isnull()[0:i]]=(select['trump'][select.result.isnull()[0:i]]>select['clinton'][select.result.isnull()[0:

    EEV=len(full_table.Party[0:i][full_table.Party=='Con'])+ np.count_nonzero(result, axis=1)
#    print(jointb['time'][i],jointb['State'][i], EEV)

    #make table for plot
    pred.time[i]=full_table.time[i]
    pred.Constituency[i]=full_table.ID[i]
    pred.low[i]=np.percentile(EEV,10)
    pred.ave[i]=np.mean(EEV)
    pred.high[i]=np.percentile(EEV,90)
```

```
          # keep only the last value at a certain time. And remove NAs.
          pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
          pred.index = np.arange(0,len(pred))
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1353: RuntimeWarning: divide by zero encountered in double_scalar
  return np.dot(wresid, wresid) / self.df_resid
```

2.1.1 Reading Financial Data

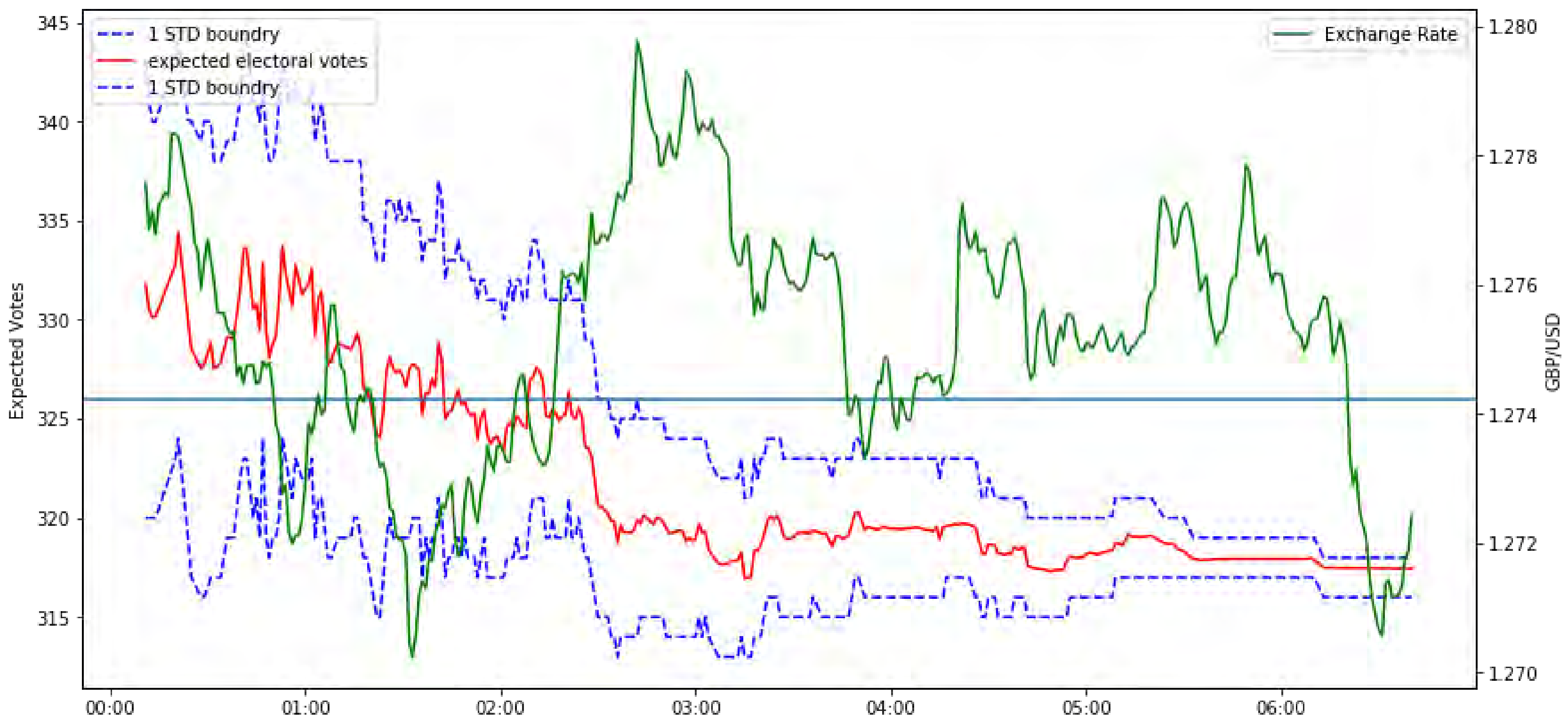
```
In [196]: fx = pd.read_csv('data/FX data.csv',header=None,sep='\;')

          #the sixth column is 0, drop it
          fx = fx.drop(fx.columns[5], 1)

          # average the minute values.
          fx['mean'] = fx.ix[:,2:4].astype(float).mean(axis=1)

          # extract time value to datetime format
          fx['time'] = fx.ix[:,0]
          fx['time'] = [datetime.strptime(v, '%Y%m%d %H%M%S') for v in fx['time']]

          #change time to UTC to match twitter
          fx['time'] = [v + timedelta(hours=5) for v in fx['time']]

          fx = fx.set_index(['time'])
          fx = fx.loc[pred.time[0]:pred.time[len(pred)-1]]
          fx = fx.drop(fx.columns[0:5],1)
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: ParserWarning: Falling back to the 'python' engine because the 'c' engine does 

C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
```

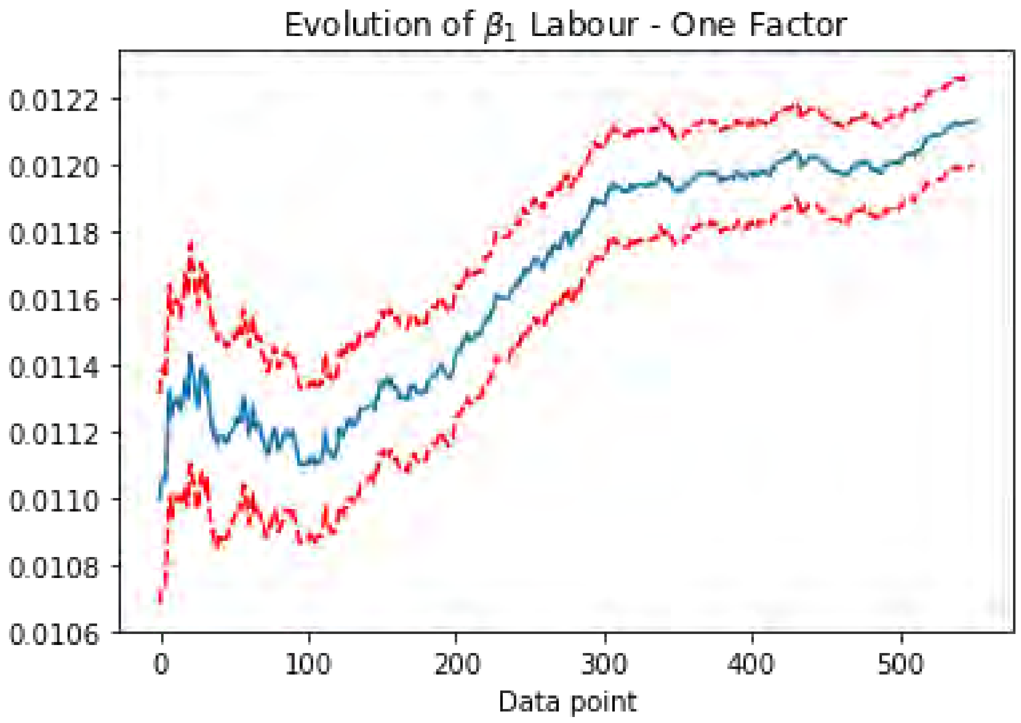
http://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate_ix

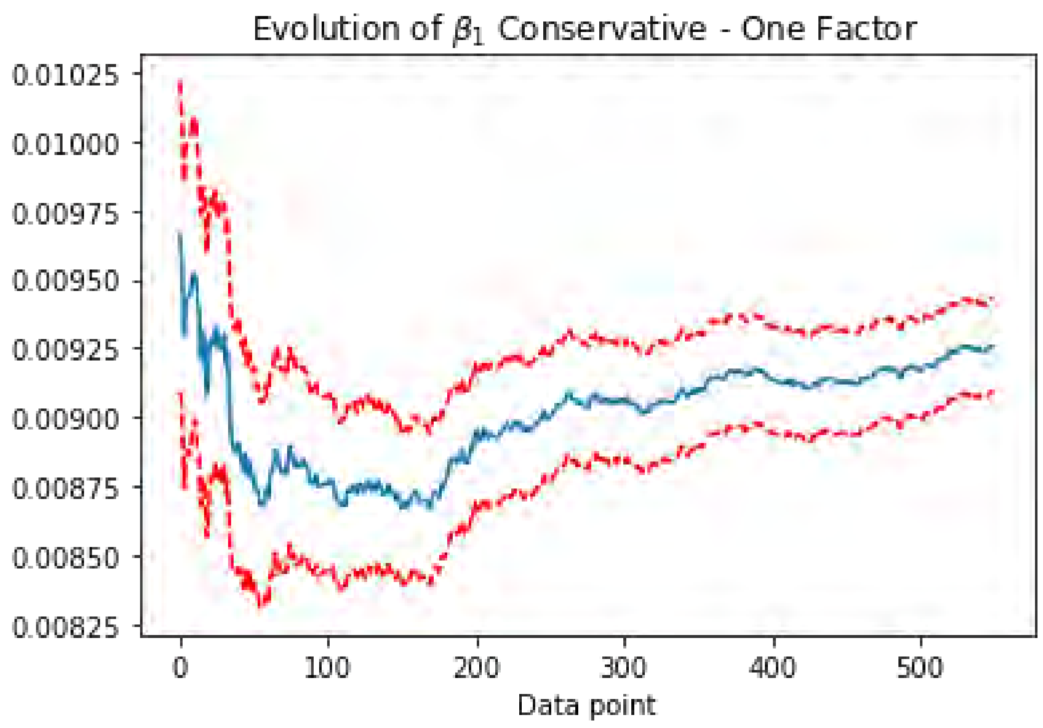
```
In [197]: fig, ax1 = plt.subplots()
          fig.set_size_inches(14, 7)
          ax1.plot_date(pred.time, pred.low,'b--', label='1 STD boundry')
          ax1.plot_date(pred.time, pred.ave,'r-', label='expected electoral votes')
          ax1.plot_date(pred.time, pred.high,'b--', label='1 STD boundry')
          plt.ylabel('Expected Votes')
          pylab.legend(loc='upper left')
          plt.axhline(y=326)
          #ax1.set_ylim([300,350])

          ax2 = ax1.twinx()
          ax2.plot(fx['mean'],'g', label='Exchange Rate')

          HMFmt = mdates.DateFormatter('%H:%M')
          ax1.xaxis.set_major_formatter(HMFmt)
          _ = plt.xticks(rotation=90)

          plt.ylabel('GBP/USD')
          plt.xlabel('UTC Time')
          pylab.legend(loc='upper right')

          fig.suptitle('Expected Conservative Seats vs Exchange Rate - One Factor', fontsize=15)
          fig.savefig('results-onefac.png')
          plt.show()
```

Expected Conservative Seats vs Exchange Rate - One Factor



```
In [198]: plt.plot(parameters_lab[50:600,1])
          plt.plot(parameters_lab[50:600,1]-stderr_lab[50:600,1],'r--')
          plt.plot(parameters_lab[50:600,1]+stderr_lab[50:600,1],'r--')
          plt.title(r'Evolution of $\beta_1 $ Labour - One Factor')
          plt.xlabel('Data point')
          plt.savefig('lab one factor b1.png')
          plt.show()
plt.plot(parameters_con[50:600,1])
plt.plot(parameters_con[50:600,1]-stderr_con[50:600,1],'r--')
plt.plot(parameters_con[50:600,1]+stderr_con[50:600,1],'r--')
plt.title(r'Evolution of $\beta_1 $ Conservative - One Factor')
plt.xlabel('Data point')
plt.savefig('con one factor b1.png')
plt.show()
```





```
In [199]: columns = ['time', 'Constituency','low', 'ave', 'high']
          index= full_table.index
          pred=pd.DataFrame(index=index, columns=columns)

          parameters_con = np.empty((650,2))
          stderr_con = np.empty((650,2))
          parameters_lab = np.empty((650,2))
          stderr_lab = np.empty((650,2))
          parameters_snp = np.empty((650,2))
          stderr_snp = np.empty((650,2))
```

```
sim_num = 10
for i in range(18,len(full_table)-14):
    #labour regression
   # data_lab=full_table[['Lab_poll','Lab[c]','Total','Last Election','WageMedianConst','Pop65ConstRate']]
    data_lab= pd.concat([full_table['Lab_poll'],\
                         full_table['Lab[c]']], axis=1)
    #,full_table['Con_poll']*full_table['iscon'],
    X_lab = data_lab.drop('Lab[c]',axis=1)[0:i]
    Y_lab=data_lab['Lab[c]'][0:i]
    X_lab=sm.add_constant(X_lab, has_constant='add')
    model_lab = sm.RLM(Y_lab, X_lab,missing = 'drop').fit()

    parameters_lab[i,:]=model_lab.params
    stderr_lab[i,:]=model_lab.bse

    #Conservatives regression
    #data_con=full_table[['Con_poll','Con[b]','Total','Last Election','WageMedianConst','Pop65ConstRate']][0:i].dropna(axis=0)
    data_con= pd.concat([full_table['Con_poll'],\
                        full_table['Con[b]']], axis=1)
    X_con = data_con.drop(['Con[b]'], axis = 1)[0:i]
    Y_con=data_con['Con[b]'][0:i]
    X_con=sm.add_constant(X_con, has_constant='add')
    model_con = sm.RLM(Y_con, X_con,missing = 'drop').fit()

    parameters_con[i,:]=model_con.params
    stderr_con[i,:]=model_con.bse

    data_snp=pd.concat([full_table['SNP_poll'], full_table['SNP']], axis=1)
    X_snp = data_snp.drop(['SNP'],axis = 1)[0:i]
    Y_snp=data_snp['SNP'][0:i]
    X_snp=sm.add_constant(X_snp, has_constant='add')
    try:
        model_snp = sm.RLM(Y_snp, X_snp,missing = 'drop').fit()
        parameters_snp[i,:]
        stderr_snp[i,:]=model_snp.bse
    except:
        model_snp = 0
```

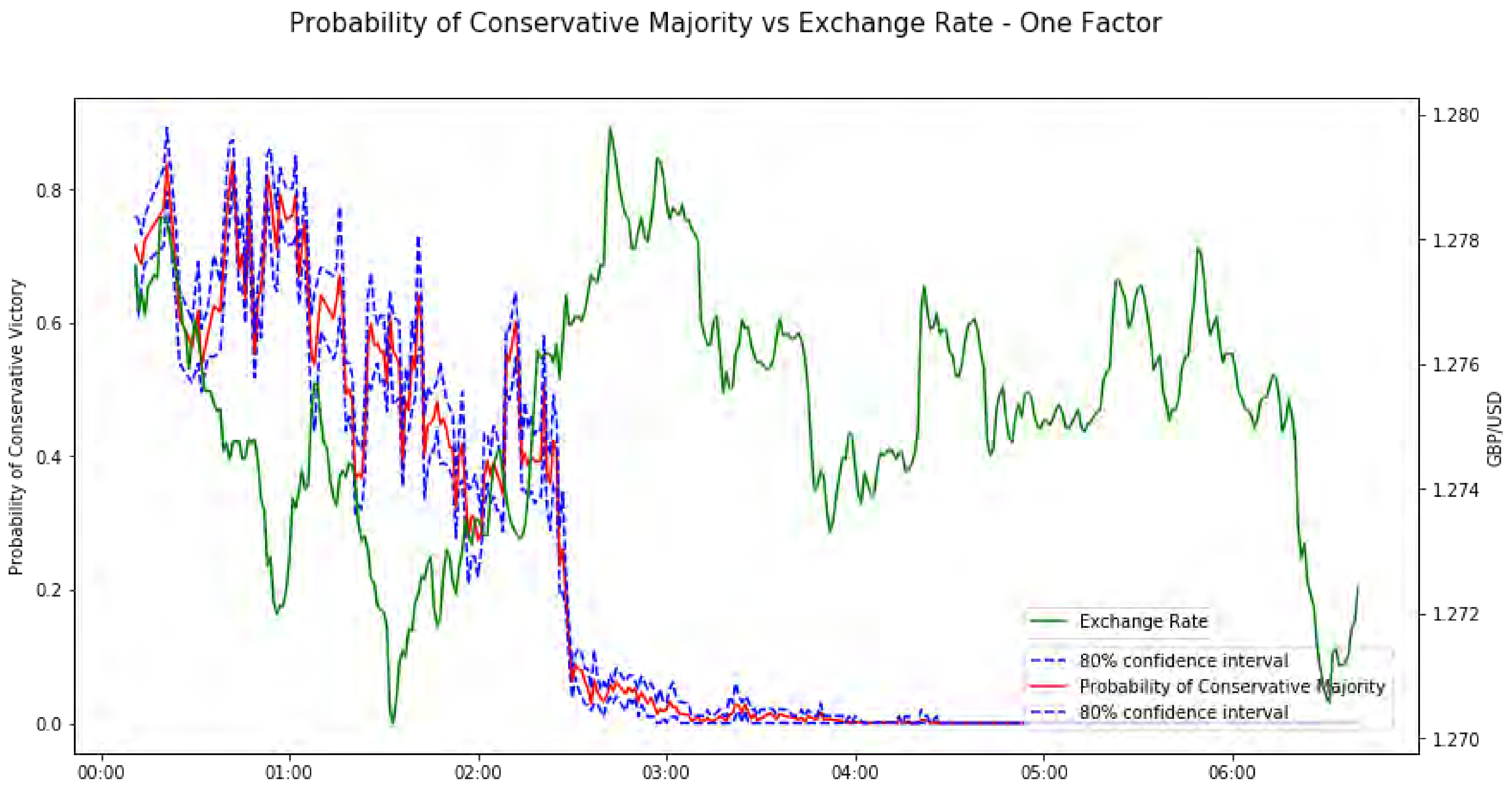
```
prob = np.zeros(10)
for k in range(0,10):

    predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')
    predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]
    result = np.zeros([sim_num, len(predict_lab)])

    #Sampling the regression parameters to generate predicted outcome
    for j in range(0, sim_num):

        #Labour
        param_lab = np.random.multivariate_normal(np.asarray(model_lab.params), RegressionResults.cov_params(model_lab), sim_num)
        predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')


        #fill NaN with mean
        #predict_lab = predict_lab.fillna(predict_lab.mean())
        predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]

        result_temp =  np.matmul(param_lab,np.array(predict_lab).T)
        #result_lab = np.random.normal(result_temp,np.std(model_lab.resid))
        err_dev = np.std(model_lab.resid)*(np.random.chisquare(len(model_lab.resid)-2)*1./(len(model_lab.resid)-2))**0.5
        result_lab = np.random.normal(result_temp,err_dev)


        #Conservative
        param_con = np.random.multivariate_normal(np.asarray(model_con.params), RegressionResults.cov_params(model_con), sim_num)
        predict_con=sm.add_constant(data_con.drop(['Con[b]'], axis = 1)[i:], has_constant='add')


        #predict_con = predict_con.fillna(predict_con.mean())
        predict_con = predict_con[np.isfinite(predict_con['Con_poll'])]

        result_temp = np.matmul(param_con,np.array(predict_con).T)
        #result_con = np.random.normal(result_temp,np.std(model_con.resid))
        err_dev = np.std(model_con.resid)*(np.random.chisquare(len(model_con.resid)-2)*1./(len(model_con.resid)-2))**0.5
        result_con = np.random.normal(result_temp,err_dev)



        #SNP
        predict_snp=sm.add_constant(data_snp.drop(['SNP'],axis = 1)[i:], has_constant='add')
        predict_snp = predict_snp[np.isfinite(predict_snp['SNP_poll'])]
        if not model_snp==0 and len(model_snp.resid)>8:
            param_snp = np.random.multivariate_normal(np.asarray(model_snp.params), RegressionResults.cov_params(model_snp), sim_num)

            #predict_snp = predict_snp.fillna(predict_snp.mean())

            result_temp =  np.matmul(param_snp,np.array(predict_snp).T)
            #result_snp = np.random.normal(result_temp,np.std(model_snp.resid))
            err_dev = np.std(model_snp.resid)*(np.random.chisquare(len(model_snp.resid)-2)*1./(len(model_snp.resid)-2))**0.5
            result_snp = np.random.normal(result_temp,err_dev)
            #if the poll is 0, the resulting simulated data should also be 0
            ind=np.where(predict_snp.SNP_poll==0)
            result_snp[:,ind]=0

        else:
            #no data
            result_snp = np.tile(predict_snp['SNP_poll'],(sim_num,1))/100

        #result_pla = np.nan_to_num(np.tile(full_table['Pla_poll'][i:],(sim_num,1)))/100
        #result_ukip = np.nan_to_num(np.tile(full_table['UKIP_poll'][i:],(sim_num,1)))/100
        other = 1-result_con-result_lab-result_snp

        result_temp = (result_con>result_lab) & (result_con>result_snp) & (result_con>other)
        result = np.append(result, result_temp,axis=0)

    #remove the zeros during initialisation
    result = np.delete(result,range(0,sim_num),0)
    result = result.astype(int)

    #filling the result of intermediates states with current count
```

```
#    select.result[select.result.isnull()[0:i]]=(select['trump'][select.result.isnull()[0:i]]>select['clinton'][select.result.isnull(

            EEV=len(full_table.Party[0:i][full_table.Party=='Con'])+ np.count_nonzero(result, axis=1)
#    print(jointb['time'][i],jointb['State'][i], EEV)

            prob[k] = sum(EEV>326)/len(EEV)


        #make table for plot
        pred.time[i]=full_table.time[i]
        pred.Constituency[i]=full_table.ID[i]
        pred.low[i]=np.percentile(prob,10)
        pred.ave[i]=np.mean(prob)
        pred.high[i]=np.percentile(prob,90)

    # keep only the last value at a certain time. And remove NAs.
    pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
    pred.index = np.arange(0,len(pred))
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1353: RuntimeWarning: divide by zero encountered in double_scalar
  return np.dot(wresid, wresid) / self.df_resid
```

```
In [200]: #%%
          fig, ax1 = plt.subplots()
          fig.set_size_inches(14, 7)
          ax1.plot_date(pred.time, pred.low,'b--', label='80% confidence interval')
          ax1.plot_date(pred.time, pred.ave,'r-', label='Probability of Conservative Majority')
          ax1.plot_date(pred.time, pred.high,'b--', label='80% confidence interval')
          plt.ylabel('Probability of Conservative Victory')
          #pylab.legend(loc='lower right')
          #plt.axhline(y=270)
          ax1.legend(loc='center left', bbox_to_anchor=(0.7, 0.1))
          #ax1.set_ylim([0.5,0.9])

          ax2 = ax1.twinx()
          ax2.plot(fx['mean'],'g', label='Exchange Rate')
          #ax2.set_ylim([18.5,21.5])
```

```python
HMFmt = mdates.DateFormatter('%H:%M')
ax1.xaxis.set_major_formatter(HMFmt)
_ = plt.xticks(rotation=90)

plt.ylabel('GBP/USD')

#pylab.legend(loc='upper right')
ax2.legend(loc='center left', bbox_to_anchor=(0.7, 0.2))

fig.suptitle('Probability of Conservative Majority vs Exchange Rate - One Factor', fontsize=15)

#plt.figure(figsize=(20,10))
plt.show()
fig.savefig('prob-onefac.png')
```



2.2 3 factor

```
In [201]: columns = ['time', 'Constituency','low', 'ave', 'high']
          index= full_table.index
          pred=pd.DataFrame(index=index, columns=columns)
```

```
parameters_con = np.empty((650,4))
stderr_con = np.empty((650,4))
parameters_lab = np.empty((650,4))
stderr_lab = np.empty((650,4))
parameters_snp = np.empty((650,4))
stderr_snp = np.empty((650,4))

sim_num = 50
for i in range(18,len(full_table)-14):

    #labour regression
   # data_lab=full_table[['Lab_poll','Lab[c]','Total','Last Election','WageMedianConst','Pop65ConstRate']]
    data_lab= pd.concat([full_table['Lab_poll'],\
                         full_table['WageMedianConst'] , \
                         full_table['Pop65ConstRate'] ,\
                          full_table['Lab[c]']], axis=1)
    #,full_table['Con_poll']*full_table['iscon'],
    X_lab = data_lab.drop('Lab[c]',axis=1)[0:i]
    Y_lab=data_lab['Lab[c]'][0:i]
    X_lab=sm.add_constant(X_lab, has_constant='add')
    model_lab = sm.RLM(Y_lab, X_lab,missing = 'drop').fit()

    parameters_lab[i,:]=model_lab.params
    stderr_lab[i,:]=model_lab.bse

    #Conservatives regression
    #data_con=full_table[['Con_poll','Con[b]','Total','Last Election','WageMedianConst','Pop65ConstRate']][0:i].dropna(axis=0)
    data_con= pd.concat([full_table['Con_poll'],\
                         full_table['WageMedianConst'],  \
                         full_table['Pop65ConstRate'] ,\
                         full_table['Con[b]']], axis=1)
    X_con = data_con.drop(['Con[b]'], axis = 1)[0:i]
    Y_con=data_con['Con[b]'][0:i]
    X_con=sm.add_constant(X_con, has_constant='add')
    model_con = sm.RLM(Y_con, X_con,missing = 'drop').fit()

    parameters_con[i,:]=model_con.params
```

```
stderr_con[i,:]=model_con.bse


data_snp=pd.concat([full_table['SNP_poll'],  full_table['WageMedianConst'],
                 full_table['Pop65ConstRate'], full_table['SNP']], axis=1)
X_snp = data_snp.drop(['SNP'],axis = 1)[0:i]
Y_snp=data_snp['SNP'][0:i]
X_snp=sm.add_constant(X_snp, has_constant='add')
try:
    model_snp = sm.RLM(Y_snp, X_snp,missing = 'drop').fit()
    parameters_snp[i,:]
    stderr_snp[i,:]=model_snp.bse
except:
    model_snp = 0




predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')
predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]
result = np.zeros([sim_num, len(predict_lab)])

#Sampling the regression parameters to generate predicted outcome
for j in range(0, sim_num):

    #Labour
    param_lab = np.random.multivariate_normal(np.asarray(model_lab.params), RegressionResults.cov_params(model_lab), sim_num)
    predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')


    #fill NaN with mean
    #predict_lab = predict_lab.fillna(predict_lab.mean())
    predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]

    result_temp =  np.matmul(param_lab,np.array(predict_lab).T)
    #result_lab = np.random.normal(result_temp,np.std(model_lab.resid))
    err_dev = np.std(model_lab.resid)*(np.random.chisquare(len(model_lab.resid)-4)*1./(len(model_lab.resid)-4))**0.5
```

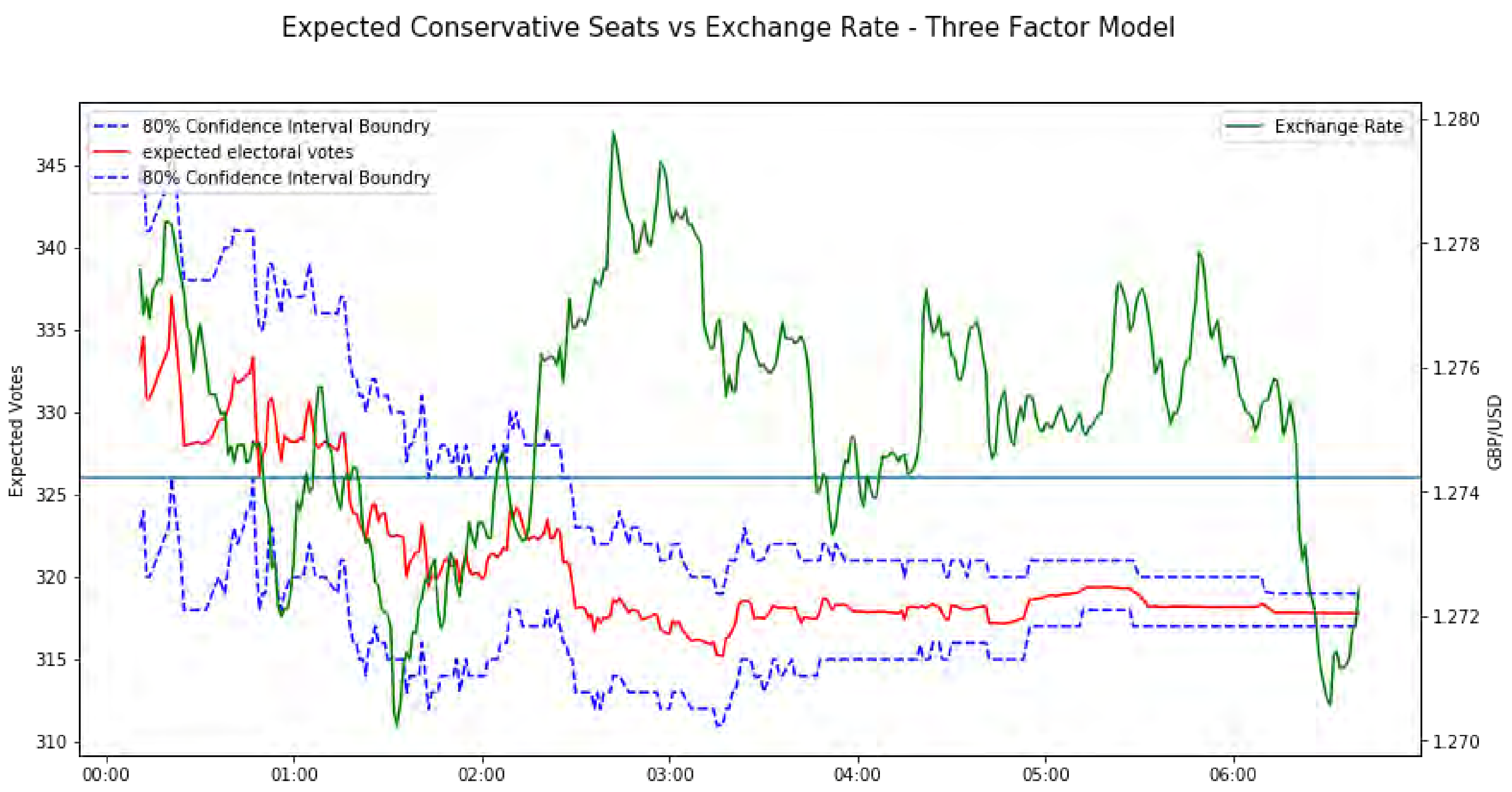
```
result_lab = np.random.normal(result_temp,err_dev)


#Conservative
param_con = np.random.multivariate_normal(np.asarray(model_con.params), RegressionResults.cov_params(model_con), sim_num)
predict_con=sm.add_constant(data_con.drop(['Con[b]'], axis = 1)[i:], has_constant='add')


#predict_con = predict_con.fillna(predict_con.mean())
predict_con = predict_con[np.isfinite(predict_con['Con_poll'])]

result_temp = np.matmul(param_con,np.array(predict_con).T)
#result_con = np.random.normal(result_temp,np.std(model_con.resid))
err_dev = np.std(model_con.resid)*(np.random.chisquare(len(model_con.resid)-4)*1./(len(model_con.resid)-4))**0.5
result_con = np.random.normal(result_temp,err_dev)




#SNP
predict_snp=sm.add_constant(data_snp.drop(['SNP'],axis = 1)[i:], has_constant='add')
predict_snp = predict_snp[np.isfinite(predict_snp['SNP_poll'])]
if not model_snp==0 and len(model_snp.resid)>8:
    param_snp = np.random.multivariate_normal(np.asarray(model_snp.params), RegressionResults.cov_params(model_snp), sim_num)

    #predict_snp = predict_snp.fillna(predict_snp.mean())

    result_temp =  np.matmul(param_snp,np.array(predict_snp).T)
    #result_snp = np.random.normal(result_temp,np.std(model_snp.resid))
    err_dev = np.std(model_snp.resid)*(np.random.chisquare(len(model_snp.resid)-4)*1./(len(model_snp.resid)-4))**0.5
    result_snp = np.random.normal(result_temp,err_dev)
    #if the poll is 0, the resulting simulated data should also be 0
    ind=np.where(predict_snp.SNP_poll==0)
    result_snp[:,ind]=0

else:
    #no data
```

```
                result_snp = np.tile(predict_snp['SNP_poll'],(sim_num,1))/100

            other = 1-result_con-result_lab-result_snp

            result_temp = (result_con>result_lab) & (result_con>result_snp) & (result_con>other)
            result = np.append(result, result_temp,axis=0)

        #remove the zeros during initialisation
        result = np.delete(result,range(0,sim_num),0)
        result = result.astype(int)

        #filling the result of intermediates states with current count
#       select.result[select.result.isnull()[0:i]]=(select['trump'][select.result.isnull()[0:i]]>select['clinton'][select.result.isnull()[0:

        EEV=len(full_table.Party[0:i][full_table.Party=='Con'])+ np.count_nonzero(result, axis=1)
#       print(jointb['time'][i],jointb['State'][i], EEV)

        #make table for plot
        pred.time[i]=full_table.time[i]
        pred.Constituency[i]=full_table.ID[i]
        pred.low[i]=np.percentile(EEV,10)
        pred.ave[i]=np.mean(EEV)
        pred.high[i]=np.percentile(EEV,90)

    # keep only the last value at a certain time. And remove NAs.
    pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
    pred.index = np.arange(0,len(pred))
```

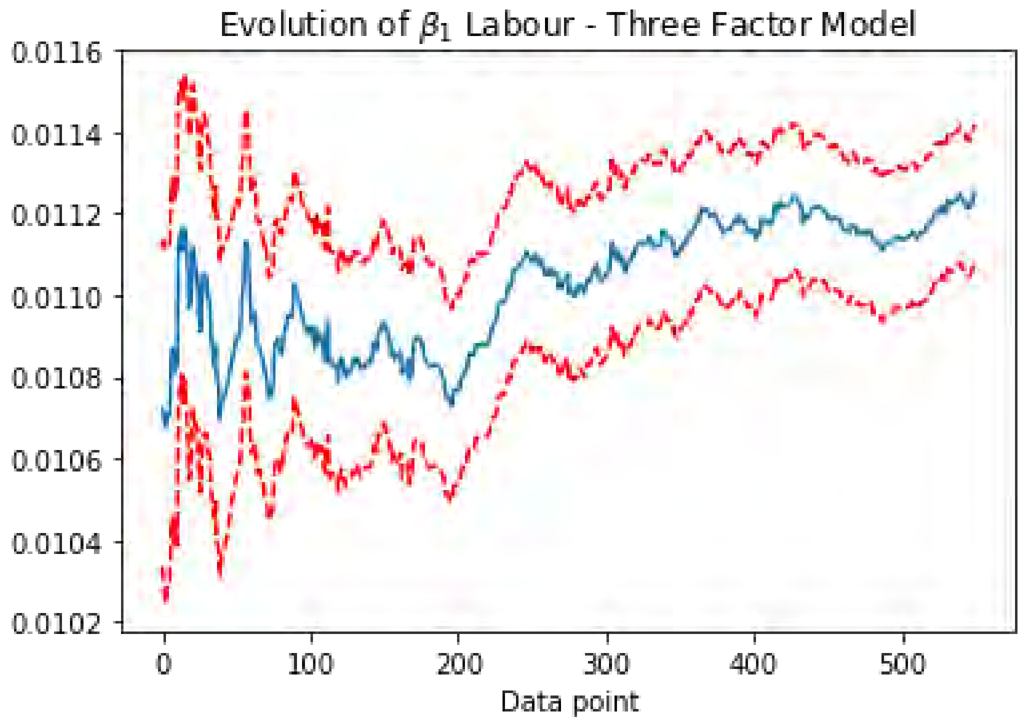
```
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:140: RuntimeWarning: invalid value encountered in greater
C:\Users\AlexH\Anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1353: RuntimeWarning: divide by zero encountered in double_scalar
  return np.dot(wresid, wresid) / self.df_resid
```

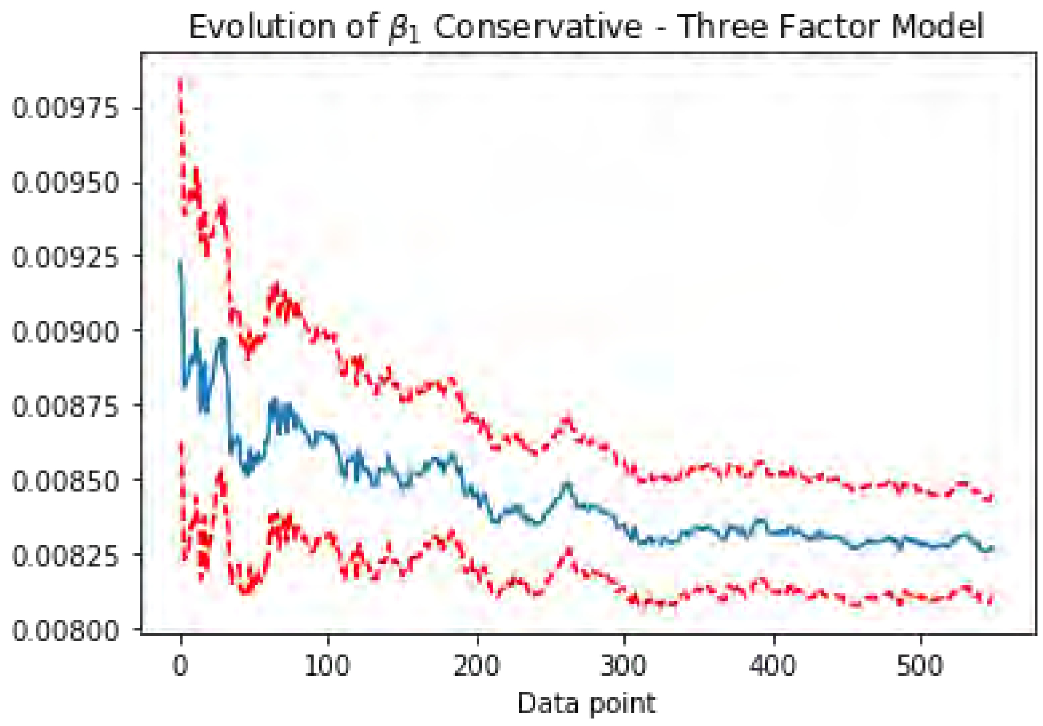
```
In [202]: fig, ax1 = plt.subplots()
          fig.set_size_inches(14, 7)
          ax1.plot_date(pred.time, pred.low,'b--', label='80% Confidence Interval Boundry')
          ax1.plot_date(pred.time, pred.ave,'r-', label='expected electoral votes')
          ax1.plot_date(pred.time, pred.high,'b--', label='80% Confidence Interval Boundry')
```

```
plt.ylabel('Expected Votes')
pylab.legend(loc='upper left')
plt.axhline(y=326)
#ax1.set_ylim([280,380])

ax2 = ax1.twinx()
ax2.plot(fx['mean'],'g', label='Exchange Rate')

HMFmt = mdates.DateFormatter('%H:%M')
ax1.xaxis.set_major_formatter(HMFmt)
_ = plt.xticks(rotation=90)

plt.ylabel('GBP/USD')
plt.xlabel('UTC Time')
pylab.legend(loc='upper right')

fig.suptitle('Expected Conservative Seats vs Exchange Rate - Three Factor Model', fontsize=15)
fig.savefig('results-threefac.png')
plt.show()
```



```
In [203]: plt.plot(parameters_lab[50:600,1])
          plt.plot(parameters_lab[50:600,1]-stderr_lab[50:600,1],'r--')
          plt.plot(parameters_lab[50:600,1]+stderr_lab[50:600,1],'r--')
          plt.title(r'Evolution of $\beta_1 $ Labour - Three Factor Model')
          plt.xlabel('Data point')
          plt.savefig('lab threefac b1.png')
          plt.show()
plt.plot(parameters_con[50:600,1])
plt.plot(parameters_con[50:600,1]-stderr_con[50:600,1],'r--')
plt.plot(parameters_con[50:600,1]+stderr_con[50:600,1],'r--')
plt.title(r'Evolution of $\beta_1 $ Conservative - Three Factor Model')
plt.xlabel('Data point')
plt.savefig('con treefac b1.png')
plt.show()
```





```
In [204]: columns = ['time', 'Constituency','low', 'ave', 'high']
          index= full_table.index
          pred=pd.DataFrame(index=index, columns=columns)

          parameters_con = np.empty((650,4))
          stderr_con = np.empty((650,4))
          parameters_lab = np.empty((650,4))
          stderr_lab = np.empty((650,4))
          parameters_snp = np.empty((650,4))
          stderr_snp = np.empty((650,4))
```

```
sim_num = 10
for i in range(18,len(full_table)-14):


    #labour regression
   # data_lab=full_table[['Lab_poll','Lab[c]','Total','Last Election','WageMedianConst','Pop65ConstRate']]
    data_lab= pd.concat([full_table['Lab_poll'],\
                         full_table['WageMedianConst'] , \
                         full_table['Pop65ConstRate'] ,\
                          full_table['Lab[c]']], axis=1)
    #,full_table['Con_poll']*full_table['iscon'],
    X_lab = data_lab.drop('Lab[c]',axis=1)[0:i]
    Y_lab=data_lab['Lab[c]'][0:i]
    X_lab=sm.add_constant(X_lab, has_constant='add')
    model_lab = sm.RLM(Y_lab, X_lab,missing = 'drop').fit()

    parameters_lab[i,:]=model_lab.params
    stderr_lab[i,:]=model_lab.bse

    #Conservatives regression
    #data_con=full_table[['Con_poll','Con[b]','Total','Last Election','WageMedianConst','Pop65ConstRate']][0:i].dropna(axis=0)
    data_con= pd.concat([full_table['Con_poll'],\
                         full_table['WageMedianConst'],  \
                         full_table['Pop65ConstRate'] ,\
                         full_table['Con[b]']], axis=1)
    X_con = data_con.drop(['Con[b]'], axis = 1)[0:i]
    Y_con=data_con['Con[b]'][0:i]
    X_con=sm.add_constant(X_con, has_constant='add')
    model_con = sm.RLM(Y_con, X_con,missing = 'drop').fit()

    parameters_con[i,:]=model_con.params
    stderr_con[i,:]=model_con.bse


    data_snp=pd.concat([full_table['SNP_poll'],  full_table['WageMedianConst'],
                     full_table['Pop65ConstRate'], full_table['SNP']], axis=1)
    X_snp = data_snp.drop(['SNP'],axis = 1)[0:i]
```

```python
Y_snp=data_snp['SNP'][0:i]
X_snp=sm.add_constant(X_snp, has_constant='add')
try:
    model_snp = sm.RLM(Y_snp, X_snp,missing = 'drop').fit()
    parameters_snp[i,:]
    stderr_snp[i,:]=model_snp.bse
except:
    model_snp = 0



prob = np.zeros(10)
for k in range(0,10):

    predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')
    predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]
    result = np.zeros([sim_num, len(predict_lab)])

    #Sampling the regression parameters to generate predicted outcome
    for j in range(0, sim_num):

        #Labour
        param_lab = np.random.multivariate_normal(np.asarray(model_lab.params), RegressionResults.cov_params(model_lab), sim_num)
        predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')


        #fill NaN with mean
        #predict_lab = predict_lab.fillna(predict_lab.mean())
        predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]

        result_temp =  np.matmul(param_lab,np.array(predict_lab).T)
        #result_lab = np.random.normal(result_temp,np.std(model_lab.resid))
        err_dev = np.std(model_lab.resid)*(np.random.chisquare(len(model_lab.resid)-4)*1./(len(model_lab.resid)-4))**0.5
        result_lab = np.random.normal(result_temp,err_dev)


        #Conservative
```

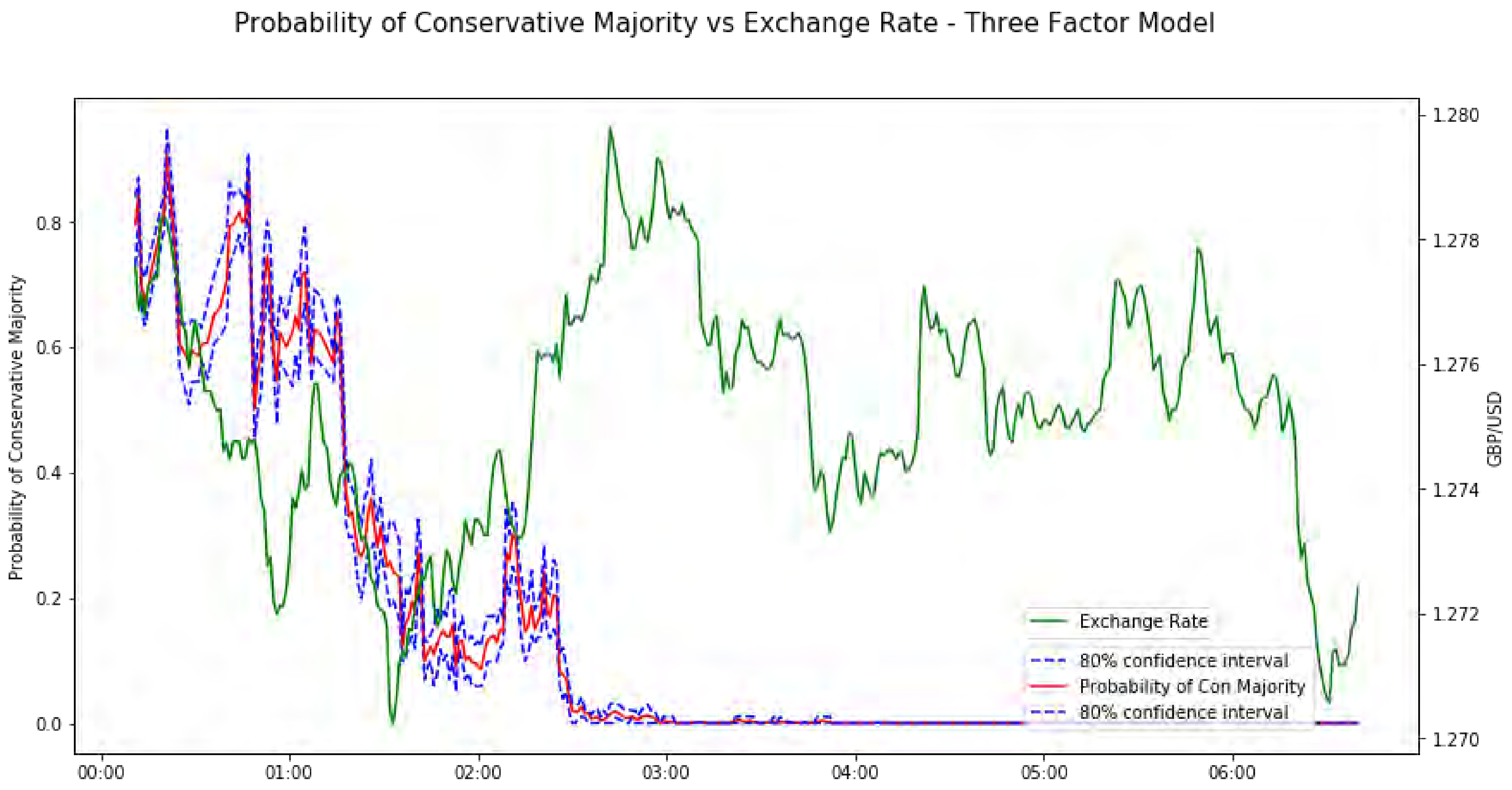
```
param_con = np.random.multivariate_normal(np.asarray(model_con.params), RegressionResults.cov_params(model_con), sim_num)
predict_con=sm.add_constant(data_con.drop(['Con[b]'], axis = 1)[i:], has_constant='add')


#predict_con = predict_con.fillna(predict_con.mean())
predict_con = predict_con[np.isfinite(predict_con['Con_poll'])]

result_temp = np.matmul(param_con,np.array(predict_con).T)
#result_con = np.random.normal(result_temp,np.std(model_con.resid))
err_dev = np.std(model_con.resid)*(np.random.chisquare(len(model_con.resid)-4)*1./(len(model_con.resid)-4))**0.5
result_con = np.random.normal(result_temp,err_dev)



#SNP
predict_snp=sm.add_constant(data_snp.drop(['SNP'],axis = 1)[i:], has_constant='add')
predict_snp = predict_snp[np.isfinite(predict_snp['SNP_poll'])]
if not model_snp==0 and len(model_snp.resid)>8:
    param_snp = np.random.multivariate_normal(np.asarray(model_snp.params), RegressionResults.cov_params(model_snp), sim_num)

    #predict_snp = predict_snp.fillna(predict_snp.mean())

    result_temp =  np.matmul(param_snp,np.array(predict_snp).T)
    #result_snp = np.random.normal(result_temp,np.std(model_snp.resid))
    err_dev = np.std(model_snp.resid)*(np.random.chisquare(len(model_snp.resid)-2)*1./(len(model_snp.resid)-2))**0.5
    result_snp = np.random.normal(result_temp,err_dev)
    #if the poll is 0, the resulting simulated data should also be 0
    ind=np.where(predict_snp.SNP_poll==0)
    result_snp[:,ind]=0

else:
    #no data
    result_snp = np.tile(predict_snp['SNP_poll'],(sim_num,1))/100

#result_pla = np.nan_to_num(np.tile(full_table['Pla_poll'][i:],(sim_num,1)))/100
#result_ukip = np.nan_to_num(np.tile(full_table['UKIP_poll'][i:],(sim_num,1)))/100
other = 1-result_con-result_lab-result_snp
```

```
                result_temp = (result_con>result_lab) & (result_con>result_snp) & (result_con>other)
                result = np.append(result, result_temp,axis=0)

            #remove the zeros during initialisation
            result = np.delete(result,range(0,sim_num),0)
            result = result.astype(int)

            #filling the result of intermediates states with current count
        #    select.result[select.result.isnull()[0:i]]=(select['trump'][select.result.isnull()[0:i]]>select['clinton'][select.result.isnull(

            EEV=len(full_table.Party[0:i][full_table.Party=='Con'])+ np.count_nonzero(result, axis=1)
        #    print(jointb['time'][i],jointb['State'][i], EEV)

            prob[k] = sum(EEV>326)/len(EEV)


        #make table for plot
        pred.time[i]=full_table.time[i]
        pred.Constituency[i]=full_table.ID[i]
        pred.low[i]=np.percentile(prob,10)
        pred.ave[i]=np.mean(prob)
        pred.high[i]=np.percentile(prob,90)

    # keep only the last value at a certain time. And remove NAs.
    pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
    pred.index = np.arange(0,len(pred))
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:143: RuntimeWarning: invalid value encountered in greater
C:\Users\AlexH\Anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1353: RuntimeWarning: divide by zero encountered in double_scalar
  return np.dot(wresid, wresid) / self.df_resid
```

```
In [205]: #%%
    fig, ax1 = plt.subplots()
    fig.set_size_inches(14, 7)
    ax1.plot_date(pred.time, pred.low,'b--', label='80% confidence interval')
    ax1.plot_date(pred.time, pred.ave,'r-', label='Probability of Con Majority')
```

```
ax1.plot_date(pred.time, pred.high,'b--', label='80% confidence interval')
plt.ylabel('Probability of Conservative Majority')
#pylab.legend(loc='lower right')
#plt.axhline(y=270)
ax1.legend(loc='center left', bbox_to_anchor=(0.7, 0.1))
#ax1.set_ylim([0.5,0.9])

ax2 = ax1.twinx()
ax2.plot(fx['mean'],'g', label='Exchange Rate')
#ax2.set_ylim([18.5,21.5])

HMFmt = mdates.DateFormatter('%H:%M')
ax1.xaxis.set_major_formatter(HMFmt)
_ = plt.xticks(rotation=90)

plt.ylabel('GBP/USD')

#pylab.legend(loc='upper right')
ax2.legend(loc='center left', bbox_to_anchor=(0.7, 0.2))

fig.suptitle('Probability of Conservative Majority vs Exchange Rate - Three Factor Model', fontsize=15)

#plt.figure(figsize=(20,10))
plt.show()
fig.savefig('prob-threefac.png')
```



2.3 Regional Dummies Model

```
In [259]: columns = ['time', 'Constituency','low', 'ave', 'high']
          index= full_table.index
          pred=pd.DataFrame(index=index, columns=columns)
```

```
parameters_con = np.empty((650,13))
stderr_con = np.empty((650,13))
parameters_lab = np.empty((650,13))
stderr_lab = np.empty((650,13))
parameters_snp = np.empty((650,4))
stderr_snp = np.empty((650,4))

sim_num = 50
for i in range(18,len(full_table)-14):

    #labour regression
   # data_lab=full_table[['Lab_poll','Lab[c]','Total','Last Election','WageMedianConst','Pop65ConstRate']]
    data_lab= pd.concat([full_table['Lab_poll'],\
                         full_table['Lab_poll']*full_table['london'],full_table['Lab_poll']*full_table['iswales'],\
                         full_table['Lab_poll']*full_table['isscot'],
                         full_table['WageMedianConst']  , \
                         full_table['WageMedianConst']*full_table['london'], full_table['WageMedianConst']*full_table['iswales'],\
                         full_table['WageMedianConst']*full_table['isscot'],\
                         full_table['Pop65ConstRate']  ,\
                         full_table['Pop65ConstRate']*full_table['london'], full_table['Pop65ConstRate']*full_table['iswales'],\
                         full_table['Pop65ConstRate']*full_table['isscot'],\
                          full_table['Lab[c]']], axis=1)
    #,full_table['Con_poll']*full_table['iscon'],
    X_lab = data_lab.drop('Lab[c]',axis=1)[0:i]
    Y_lab=data_lab['Lab[c]'][0:i]
    X_lab=sm.add_constant(X_lab, has_constant='add')
    model_lab = sm.RLM(Y_lab, X_lab,missing = 'drop').fit()

    parameters_lab[i,:]=model_lab.params
    stderr_lab[i,:]=model_lab.bse

    #Conservatives regression
    #data_con=full_table[['Con_poll','Con[b]','Total','Last Election','WageMedianConst','Pop65ConstRate']][0:i].dropna(axis=0)
    data_con= pd.concat([full_table['Con_poll'],\
                         full_table['Con_poll']*full_table['london'],full_table['Con_poll']*full_table['iswales'],\
                         full_table['Con_poll']*full_table['isscot'],
                         full_table['WageMedianConst'],  \
                     full_table['WageMedianConst']*full_table['london'], full_table['WageMedianConst']*full_table['iswales'],\
                     full_table['WageMedianConst']*full_table['isscot'],\
                     full_table['Pop65ConstRate'] ,\
                     full_table['Pop65ConstRate']*full_table['london'], full_table['Pop65ConstRate']*full_table['iswales'],\
                     full_table['Pop65ConstRate']*full_table['isscot'],\
                     full_table['Con[b]']], axis=1)
X_con = data_con.drop(['Con[b]'], axis = 1)[0:i]
Y_con=data_con['Con[b]'][0:i]
X_con=sm.add_constant(X_con, has_constant='add')
model_con = sm.RLM(Y_con, X_con,missing = 'drop').fit()

parameters_con[i,:]=model_con.params
stderr_con[i,:]=model_con.bse

data_snp=pd.concat([full_table['SNP_poll'], full_table['WageMedianConst'],
                 full_table['Pop65ConstRate'], full_table['SNP']], axis=1)
X_snp = data_snp.drop(['SNP'],axis = 1)[0:i]
Y_snp=data_snp['SNP'][0:i]
X_snp=sm.add_constant(X_snp, has_constant='add')
try:
    model_snp = sm.RLM(Y_snp, X_snp,missing = 'drop').fit()
    parameters_snp[i,:]
    stderr_snp[i,:]=model_snp.bse
except:
    model_snp = 0




predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')
predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]
result = np.zeros([sim_num, len(predict_lab)])

#Sampling the regression parameters to generate predicted outcome
for j in range(0, sim_num):

    #Labour
```

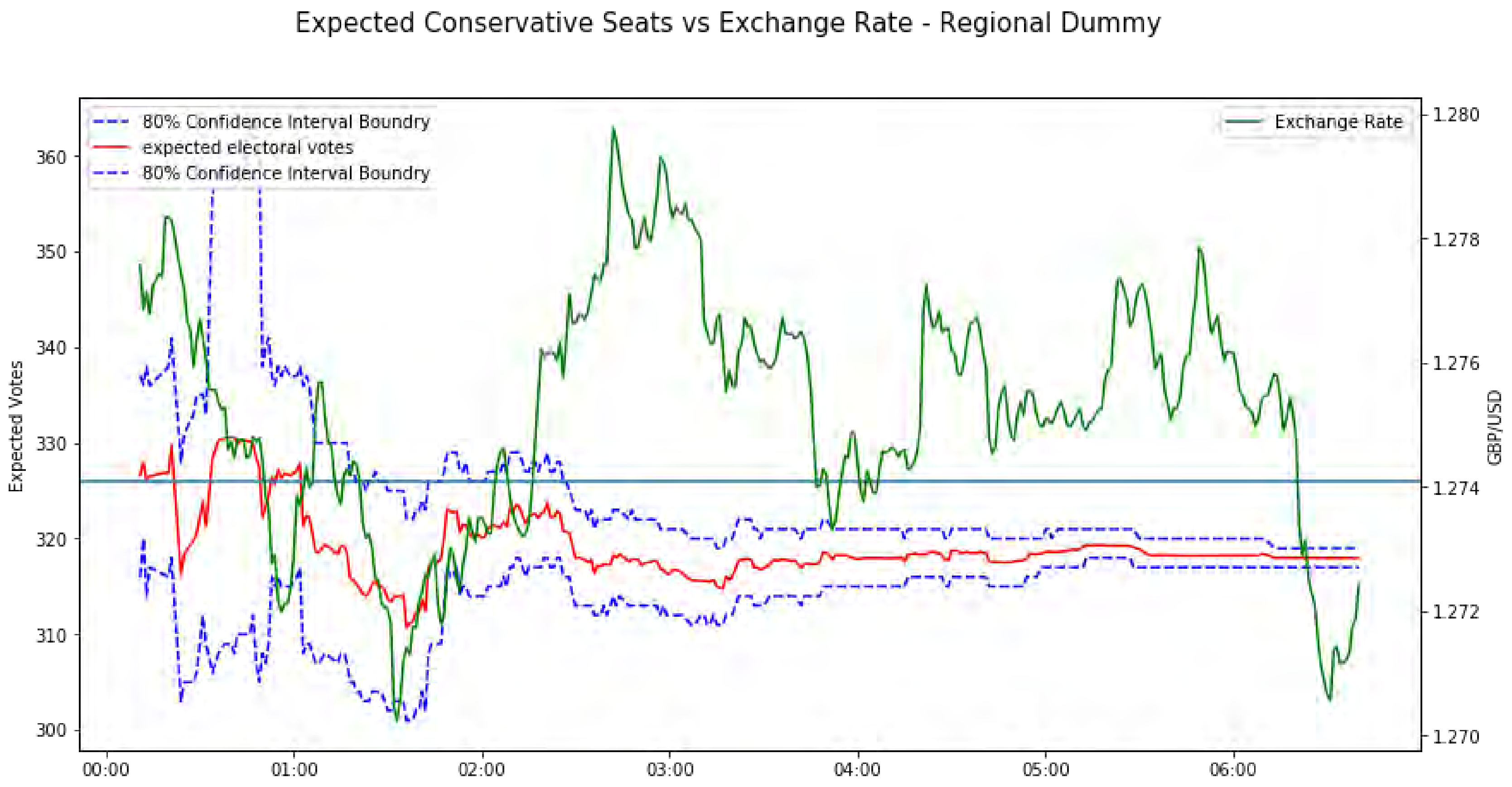
```
param_lab = np.random.multivariate_normal(np.asarray(model_lab.params), RegressionResults.cov_params(model_lab), sim_num)
predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')


#fill NaN with mean
#predict_lab = predict_lab.fillna(predict_lab.mean())
predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]

result_temp =  np.matmul(param_lab,np.array(predict_lab).T)
#result_lab = np.random.normal(result_temp,np.std(model_lab.resid))
err_dev = np.std(model_lab.resid)*(np.random.chisquare(len(model_lab.resid)-13)*1./(len(model_lab.resid)-13))**0.5
result_lab = np.random.normal(result_temp,err_dev)


#Conservative
param_con = np.random.multivariate_normal(np.asarray(model_con.params), RegressionResults.cov_params(model_con), sim_num)
predict_con=sm.add_constant(data_con.drop(['Con[b]'], axis = 1)[i:], has_constant='add')


#predict_con = predict_con.fillna(predict_con.mean())
predict_con = predict_con[np.isfinite(predict_con['Con_poll'])]

result_temp = np.matmul(param_con,np.array(predict_con).T)
#result_con = np.random.normal(result_temp,np.std(model_con.resid))
err_dev = np.std(model_con.resid)*(np.random.chisquare(len(model_con.resid)-13)*1./(len(model_con.resid)-13))**0.5
result_con = np.random.normal(result_temp,err_dev)


#SNP
predict_snp=sm.add_constant(data_snp.drop(['SNP'],axis = 1)[i:], has_constant='add')
predict_snp = predict_snp[np.isfinite(predict_snp['SNP_poll'])]
if not model_snp==0 and len(model_snp.resid)>8:
    param_snp = np.random.multivariate_normal(np.asarray(model_snp.params), RegressionResults.cov_params(model_snp), sim_num)

    #predict_snp = predict_snp.fillna(predict_snp.mean())

    result_temp =  np.matmul(param_snp,np.array(predict_snp).T)
```

```
            #result_snp = np.random.normal(result_temp,np.std(model_snp.resid))
            err_dev = np.std(model_snp.resid)*(np.random.chisquare(len(model_snp.resid)-4)*1./(len(model_snp.resid)-4))**0.5
            result_snp = np.random.normal(result_temp,err_dev)
            #if the poll is 0, the resulting simulated data should also be 0
            ind=np.where(predict_snp.SNP_poll==0)
            result_snp[:,ind]=0

        else:
            #no data
            result_snp = np.tile(predict_snp['SNP_poll'],(sim_num,1))/100

        #result_pla = np.nan_to_num(np.tile(full_table['Pla_poll'][i:],(sim_num,1)))/100
        #result_ukip = np.nan_to_num(np.tile(full_table['UKIP_poll'][i:],(sim_num,1)))/100
        other = 1-result_con-result_lab-result_snp

        result_temp = (result_con>result_lab) & (result_con>result_snp) & (result_con>other)
        result = np.append(result, result_temp,axis=0)

    #remove the zeros during initialisation
    result = np.delete(result,range(0,sim_num),0)
    result = result.astype(int)

    #filling the result of intermediates states with current count
#    select.result[select.result.isnull()[0:i]]=(select['trump'][select.result.isnull()[0:i]]>select['clinton'][select.result.isnull()[0:

    EEV=len(full_table.Party[0:i][full_table.Party=='Con'])+ np.count_nonzero(result, axis=1)
#    print(jointb['time'][i],jointb['State'][i], EEV)


    #make table for plot
    pred.time[i]=full_table.time[i]
    pred.Constituency[i]=full_table.ID[i]
    pred.low[i]=np.percentile(EEV,10)
    pred.ave[i]=np.mean(EEV)
    pred.high[i]=np.percentile(EEV,90)

# keep only the last value at a certain time. And remove NAs.
```

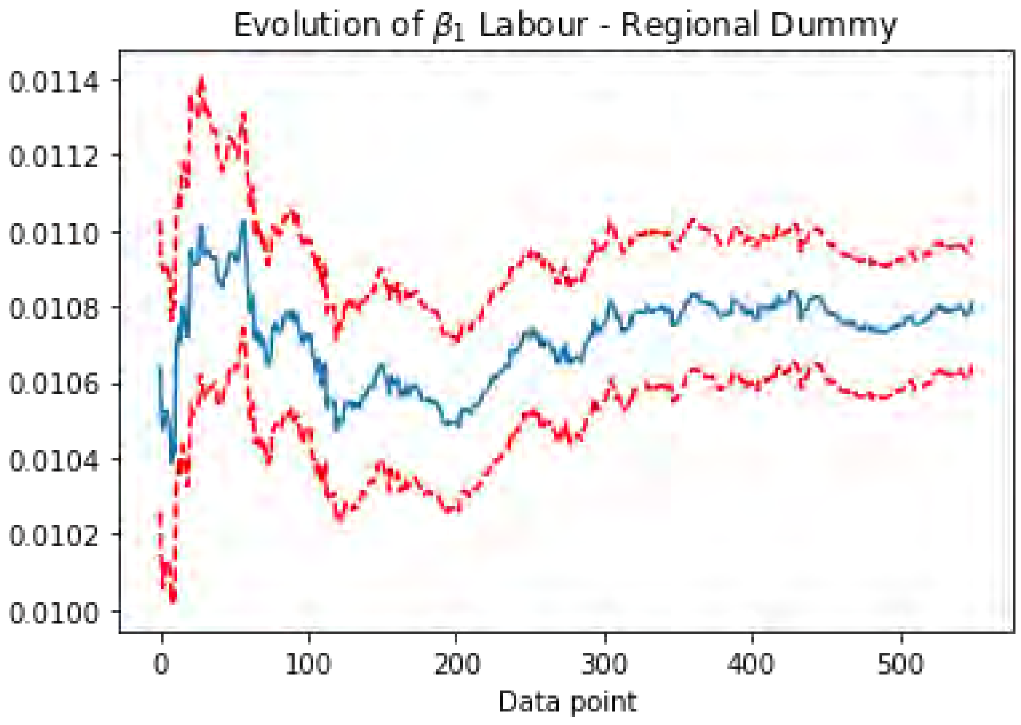
```
        pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
        pred.index = np.arange(0,len(pred))
```

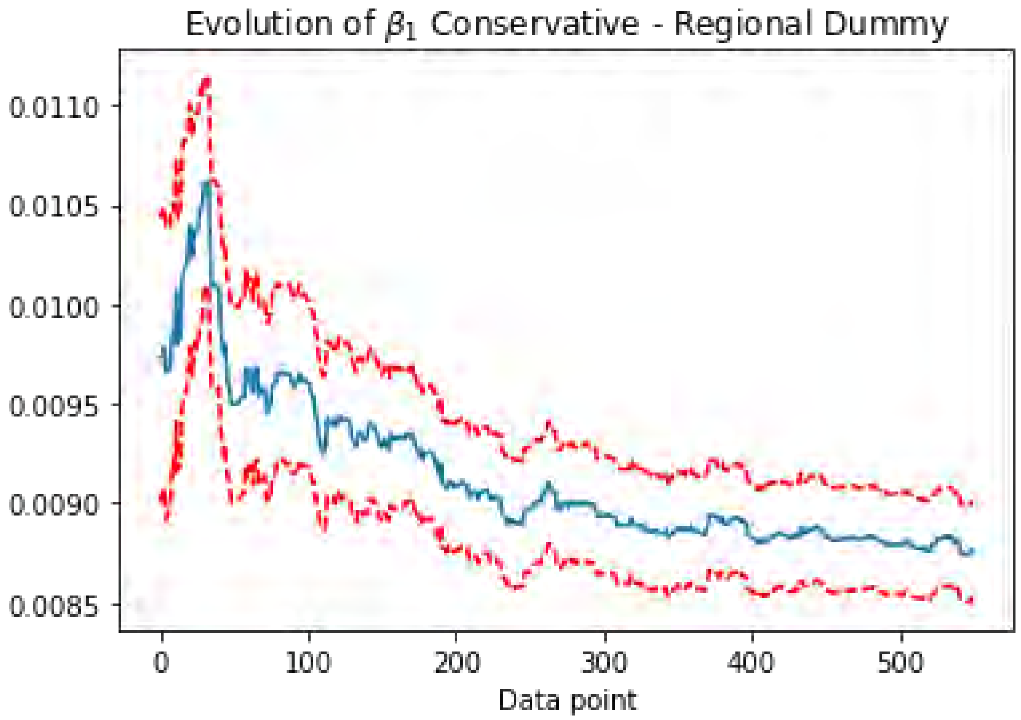
```
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:75: RuntimeWarning: covariance is not positive-semidefinite.
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:90: RuntimeWarning: covariance is not positive-semidefinite.
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:127: RuntimeWarning: invalid value encountered in greater
C:\Users\AlexH\Anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1353: RuntimeWarning: divide by zero encountered in double_scalar
  return np.dot(wresid, wresid) / self.df_resid
```

```
In [252]: fig, ax1 = plt.subplots()
          fig.set_size_inches(14, 7)
          ax1.plot_date(pred.time, pred.low,'b--', label='80% Confidence Interval Boundry')
          ax1.plot_date(pred.time, pred.ave,'r-', label='expected electoral votes')
          ax1.plot_date(pred.time, pred.high,'b--', label='80% Confidence Interval Boundry')
          plt.ylabel('Expected Votes')
          pylab.legend(loc='upper left')
          plt.axhline(y=326)
          #ax1.set_ylim([280,380])

          ax2 = ax1.twinx()
          ax2.plot(fx['mean'],'g', label='Exchange Rate')

          HMFmt = mdates.DateFormatter('%H:%M')
          ax1.xaxis.set_major_formatter(HMFmt)
          _ = plt.xticks(rotation=90)

          plt.ylabel('GBP/USD')
          plt.xlabel('UTC Time')
          pylab.legend(loc='upper right')

          fig.suptitle('Expected Conservative Seats vs Exchange Rate - Regional Dummy', fontsize=15)
          fig.savefig('results-dummy.png')
          plt.show()
```



```
In [208]: plt.plot(parameters_lab[50:600,1])
          plt.plot(parameters_lab[50:600,1]-stderr_lab[50:600,1],'r--')
          plt.plot(parameters_lab[50:600,1]+stderr_lab[50:600,1],'r--')
          plt.title(r'Evolution of $\beta_1 $ Labour - Regional Dummy')
          plt.xlabel('Data point')
          plt.savefig('lab dummy b1.png')
          plt.show()
plt.plot(parameters_con[50:600,1])
plt.plot(parameters_con[50:600,1]-stderr_con[50:600,1],'r--')
plt.plot(parameters_con[50:600,1]+stderr_con[50:600,1],'r--')
plt.title(r'Evolution of $\beta_1 $ Conservative - Regional Dummy')
plt.xlabel('Data point')
plt.savefig('con dummy b1.png')
plt.show()
```





```
In [209]: columns = ['time', 'Constituency','low', 'ave', 'high']
          index= full_table.index
          pred=pd.DataFrame(index=index, columns=columns)

          parameters_con = np.empty((650,13))
          stderr_con = np.empty((650,13))
          parameters_lab = np.empty((650,13))
          stderr_lab = np.empty((650,13))
          parameters_snp = np.empty((650,4))
          stderr_snp = np.empty((650,4))
```

```
sim_num = 10
for i in range(18,len(full_table)-14):

    #labour regression
   # data_lab=full_table[['Lab_poll','Lab[c]','Total','Last Election','WageMedianConst','Pop65ConstRate']]
    data_lab= pd.concat([full_table['Lab_poll'],\
                         full_table['Lab_poll']*full_table['london'],full_table['Lab_poll']*full_table['iswales'],\
                         full_table['Lab_poll']*full_table['isscot'],
                         full_table['WageMedianConst'] , \
                         full_table['WageMedianConst']*full_table['london'], full_table['WageMedianConst']*full_table['iswales'],\
                         full_table['WageMedianConst']*full_table['isscot'],\
                         full_table['Pop65ConstRate'] ,\
                         full_table['Pop65ConstRate']*full_table['london'], full_table['Pop65ConstRate']*full_table['iswales'],\
                         full_table['Pop65ConstRate']*full_table['isscot'],\
                          full_table['Lab[c]']], axis=1)
    #,full_table['Con_poll']*full_table['iscon'],
    X_lab = data_lab.drop('Lab[c]',axis=1)[0:i]
    Y_lab=data_lab['Lab[c]'][0:i]
    X_lab=sm.add_constant(X_lab, has_constant='add')
    model_lab = sm.RLM(Y_lab, X_lab,missing = 'drop').fit()

    parameters_lab[i,:]=model_lab.params
    stderr_lab[i,:]=model_lab.bse

    #Conservatives regression
    #data_con=full_table[['Con_poll','Con[b]','Total','Last Election','WageMedianConst','Pop65ConstRate']][0:i].dropna(axis=0)
    data_con= pd.concat([full_table['Con_poll'],\
                         full_table['Con_poll']*full_table['london'],full_table['Con_poll']*full_table['iswales'],\
                         full_table['Con_poll']*full_table['isscot'],
                         full_table['WageMedianConst'],  \
                         full_table['WageMedianConst']*full_table['london'], full_table['WageMedianConst']*full_table['iswales'],\
                         full_table['WageMedianConst']*full_table['isscot'],\
                         full_table['Pop65ConstRate'] ,\
                         full_table['Pop65ConstRate']*full_table['london'], full_table['Pop65ConstRate']*full_table['iswales'],\
                         full_table['Pop65ConstRate']*full_table['isscot'],\
                         full_table['Con[b]']], axis=1)
    X_con = data_con.drop(['Con[b]'], axis = 1)[0:i]
```

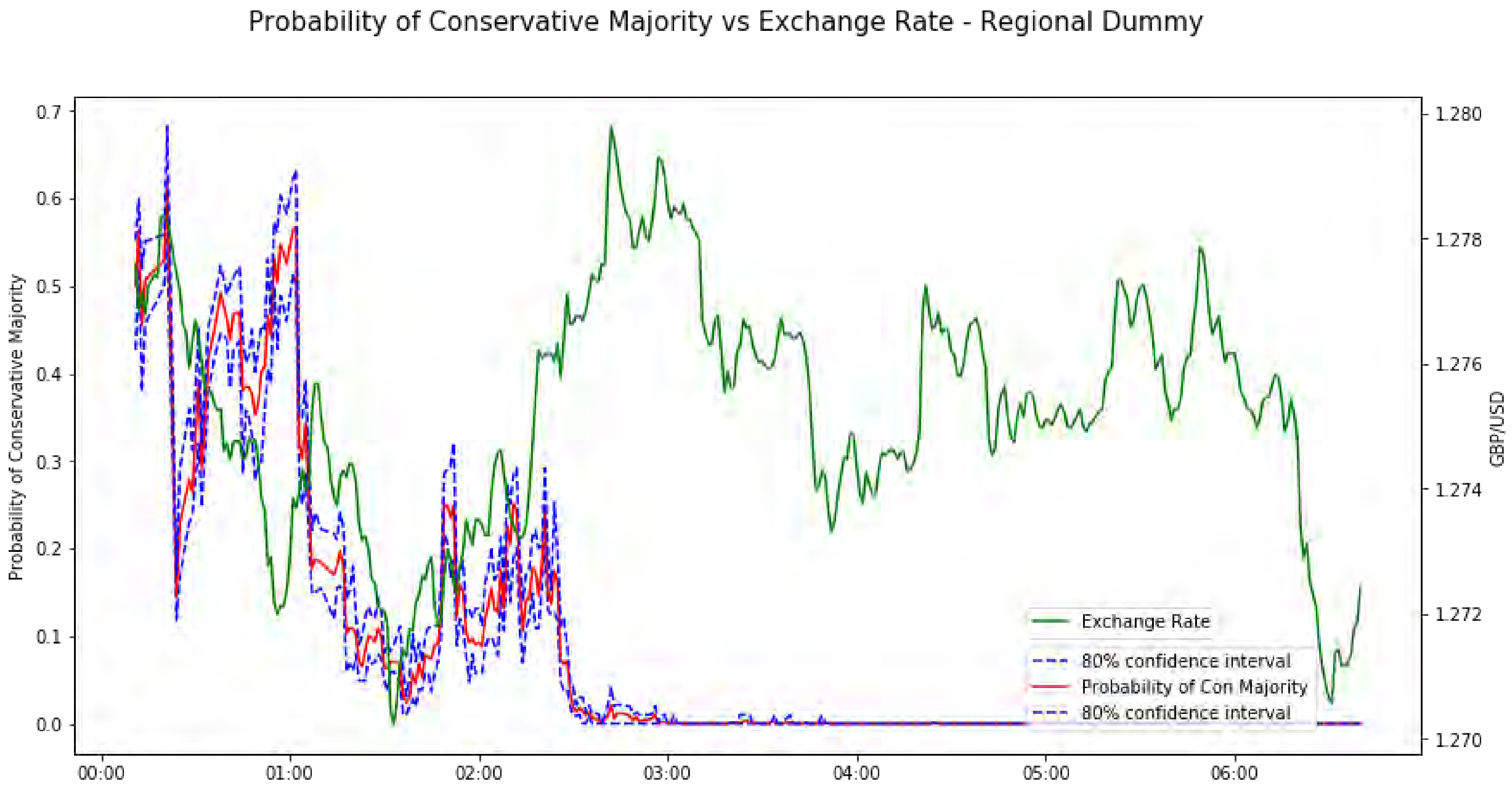
```
Y_con=data_con['Con[b]'][0:i]
X_con=sm.add_constant(X_con, has_constant='add')
model_con = sm.RLM(Y_con, X_con,missing = 'drop').fit()

parameters_con[i,:]=model_con.params
stderr_con[i,:]=model_con.bse


data_snp=pd.concat([full_table['SNP_poll'], full_table['WageMedianConst'],
                 full_table['Pop65ConstRate'], full_table['SNP']], axis=1)
X_snp = data_snp.drop(['SNP'],axis = 1)[0:i]
Y_snp=data_snp['SNP'][0:i]
X_snp=sm.add_constant(X_snp, has_constant='add')
try:
    model_snp = sm.RLM(Y_snp, X_snp,missing = 'drop').fit()
    parameters_snp[i,:]
    stderr_snp[i,:]=model_snp.bse
except:
    model_snp = 0


prob = np.zeros(10)
for k in range(0,10):

    predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')
    predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]
    result = np.zeros([sim_num, len(predict_lab)])


    #Sampling the regression parameters to generate predicted outcome
    for j in range(0, sim_num):

        #Labour
        param_lab = np.random.multivariate_normal(np.asarray(model_lab.params), RegressionResults.cov_params(model_lab), sim_num)
        predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')
```

```
#fill NaN with mean
#predict_lab = predict_lab.fillna(predict_lab.mean())
predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]

result_temp =  np.matmul(param_lab,np.array(predict_lab).T)
#result_lab = np.random.normal(result_temp,np.std(model_lab.resid))
err_dev = np.std(model_lab.resid)*(np.random.chisquare(len(model_lab.resid)-13)*1./(len(model_lab.resid)-13))**0.5
result_lab = np.random.normal(result_temp,err_dev)


#Conservative
param_con = np.random.multivariate_normal(np.asarray(model_con.params), RegressionResults.cov_params(model_con), sim_num)
predict_con=sm.add_constant(data_con.drop(['Con[b]'], axis = 1)[i:], has_constant='add')


#predict_con = predict_con.fillna(predict_con.mean())
predict_con = predict_con[np.isfinite(predict_con['Con_poll'])]

result_temp = np.matmul(param_con,np.array(predict_con).T)
#result_con = np.random.normal(result_temp,np.std(model_con.resid))
err_dev = np.std(model_con.resid)*(np.random.chisquare(len(model_con.resid)-13)*1./(len(model_con.resid)-13))**0.5
result_con = np.random.normal(result_temp,err_dev)




#SNP
predict_snp=sm.add_constant(data_snp.drop(['SNP'],axis = 1)[i:], has_constant='add')
predict_snp = predict_snp[np.isfinite(predict_snp['SNP_poll'])]
if not model_snp==0 and len(model_snp.resid)>8:
    param_snp = np.random.multivariate_normal(np.asarray(model_snp.params), RegressionResults.cov_params(model_snp), sim_num)

    #predict_snp = predict_snp.fillna(predict_snp.mean())

    result_temp =  np.matmul(param_snp,np.array(predict_snp).T)
    #result_snp = np.random.normal(result_temp,np.std(model_snp.resid))
    err_dev = np.std(model_snp.resid)*(np.random.chisquare(len(model_snp.resid)-4)*1./(len(model_snp.resid)-4))**0.5
```

```
            result_snp = np.random.normal(result_temp,err_dev)
            #if the poll is 0, the resulting simulated data should also be 0
            ind=np.where(predict_snp.SNP_poll==0)
            result_snp[:,ind]=0

        else:
            #no data
            result_snp = np.tile(predict_snp['SNP_poll'],(sim_num,1))/100

        #result_pla = np.nan_to_num(np.tile(full_table['Pla_poll'][i:],(sim_num,1)))/100
        #result_ukip = np.nan_to_num(np.tile(full_table['UKIP_poll'][i:],(sim_num,1)))/100
        other = 1-result_con-result_lab-result_snp

        result_temp = (result_con>result_lab) & (result_con>result_snp) & (result_con>other)
        result = np.append(result, result_temp,axis=0)

    #remove the zeros during initialisation
    result = np.delete(result,range(0,sim_num),0)
    result = result.astype(int)

    #filling the result of intermediates states with current count
#    select.result[select.result.isnull()[0:i]]=(select['trump'][select.result.isnull()[0:i]]>select['clinton'][select.result.isnull(

    EEV=len(full_table.Party[0:i][full_table.Party=='Con'])+ np.count_nonzero(result, axis=1)
#    print(jointb['time'][i],jointb['State'][i], EEV)



    prob[k] = sum(EEV>326)/len(EEV)


#make table for plot
pred.time[i]=full_table.time[i]
pred.Constituency[i]=full_table.ID[i]
pred.low[i]=np.percentile(prob,10)
pred.ave[i]=np.mean(prob)
pred.high[i]=np.percentile(prob,90)
```

```
        # keep only the last value at a certain time. And remove NAs.
        pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
        pred.index = np.arange(0,len(pred))

C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:92: RuntimeWarning: covariance is not positive-semidefinite.
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:107: RuntimeWarning: covariance is not positive-semidefinite.
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:156: RuntimeWarning: invalid value encountered in greater
C:\Users\AlexH\Anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1353: RuntimeWarning: divide by zero encountered in double_scalar
  return np.dot(wresid, wresid) / self.df_resid


In [210]: #%%
        fig, ax1 = plt.subplots()
        fig.set_size_inches(14, 7)
        ax1.plot_date(pred.time, pred.low,'b--', label='80% confidence interval')
        ax1.plot_date(pred.time, pred.ave,'r-', label='Probability of Con Majority')
        ax1.plot_date(pred.time, pred.high,'b--', label='80% confidence interval')
        plt.ylabel('Probability of Conservative Majority')
        #pylab.legend(loc='lower right')
        #plt.axhline(y=270)
        ax1.legend(loc='center left', bbox_to_anchor=(0.7, 0.1))
        #ax1.set_ylim([0.5,0.9])

        ax2 = ax1.twinx()
        ax2.plot(fx['mean'],'g', label='Exchange Rate')
        #ax2.set_ylim([18.5,21.5])

        HMFmt = mdates.DateFormatter('%H:%M')
        ax1.xaxis.set_major_formatter(HMFmt)
        _ = plt.xticks(rotation=90)

        plt.ylabel('GBP/USD')

        #pylab.legend(loc='upper right')
        ax2.legend(loc='center left', bbox_to_anchor=(0.7, 0.2))

        fig.suptitle('Probability of Conservative Majority vs Exchange Rate - Regional Dummy', fontsize=15)
```

```
#plt.figure(figsize=(20,10))
plt.show()
fig.savefig('prob-dummy.png')
```



2.4 Best Subset Regression

```
In [253]: X_con = data_con.drop('Con[b]',axis=1)
          Y_con=data_con['Con[b]']
          predictorcols=X_con
          target=Y_con
          train=X_con
```

```
In [254]: import itertools

          AICs = {}
          for k in range(1,len(predictorcols)+1):
          #    print(k)
              for variables in itertools.combinations(predictorcols, k):
                  predictors = train[list(variables)]
                  predictors['Intercept'] = 1
                  res = sm.OLS(target, predictors, missing='drop').fit()
                  AICs[variables] = 2*(k+1) - 2*res.llf
          pd.Series(AICs).idxmin()
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

```
Out[254]: ('Con_poll', 0, 2, 3, 4, 'Pop65ConstRate', 6, 8)
```

```
In [255]: X_lab = data_lab.drop('Lab[c]',axis=1)
          Y_lab=data_lab['Lab[c]']
          predictorcols=X_lab
          target=Y_lab
          train=X_lab
```

```
In [256]: AICs = {}
          for k in range(1,len(predictorcols)+1):
          #    print(k)
```

```
            for variables in itertools.combinations(predictorcols, k):
                predictors = train[list(variables)]
                predictors['Intercept'] = 1
                res = sm.OLS(target, predictors, missing='drop').fit()
                AICs[variables] = 2*(k+1) - 2*res.llf
        pd.Series(AICs).idxmin()
```

```
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

```
Out[256]: ('Lab_poll', 0, 1, 'WageMedianConst', 3, 4, 5, 'Pop65ConstRate', 7)
```

2.5 Final

```
In [263]: columns = ['time', 'Constituency','low', 'ave', 'high']
          index= full_table.index
          pred=pd.DataFrame(index=index, columns=columns)

          parameters_con = np.empty((650,9))
          stderr_con = np.empty((650,9))
          parameters_lab = np.empty((650,10))
          stderr_lab = np.empty((650,10))
          parameters_snp = np.empty((650,4))
          stderr_snp = np.empty((650,4))

          sim_num = 50
          for i in range(18,len(full_table)-14):

              #labour regression
             # data_lab=full_table[['Lab_poll','Lab[c]','Total','Last Election','WageMedianConst','Pop65ConstRate']]
              data_lab= pd.concat([full_table['Lab_poll'],\
                                   full_table['Lab_poll']*full_table['london'],full_table['Lab_poll']*full_table['iswales'],
                                   full_table['WageMedianConst'] , \
                     full_table['WageMedianConst']*full_table['london'], full_table['WageMedianConst']*full_table['iswales'],\
                     full_table['WageMedianConst']*full_table['isscot'],\
                     full_table['Pop65ConstRate'] ,\
                     full_table['Pop65ConstRate']*full_table['iswales'],\
                      full_table['Lab[c]']], axis=1)
#,full_table['Con_poll']*full_table['iscon'],
X_lab = data_lab.drop('Lab[c]',axis=1)[0:i]
Y_lab=data_lab['Lab[c]'][0:i]
X_lab=sm.add_constant(X_lab, has_constant='add')
model_lab = sm.RLM(Y_lab, X_lab,missing = 'drop').fit()

parameters_lab[i,:]=model_lab.params
stderr_lab[i,:]=model_lab.bse

#Conservatives regression
#data_con=full_table[['Con_poll','Con[b]','Total','Last Election','WageMedianConst','Pop65ConstRate']][0:i].dropna(axis=0)
data_con= pd.concat([full_table['Con_poll'],\
                     full_table['Con_poll']*full_table['london'],\
                     full_table['Con_poll']*full_table['isscot'],
                     full_table['WageMedianConst']*full_table['london'], full_table['WageMedianConst']*full_table['iswales'],\
                     full_table['Pop65ConstRate'] ,\
                     full_table['Pop65ConstRate']*full_table['london'],\
                     full_table['Pop65ConstRate']*full_table['isscot'],\
                     full_table['Con[b]']], axis=1)
X_con = data_con.drop(['Con[b]'], axis = 1)[0:i]
Y_con=data_con['Con[b]'][0:i]
X_con=sm.add_constant(X_con, has_constant='add')
model_con = sm.RLM(Y_con, X_con,missing = 'drop').fit()

parameters_con[i,:]=model_con.params
stderr_con[i,:]=model_con.bse

data_snp=pd.concat([full_table['SNP_poll'], full_table['WageMedianConst'],
                 full_table['Pop65ConstRate'], full_table['SNP']], axis=1)
X_snp = data_snp.drop(['SNP'],axis = 1)[0:i]
Y_snp=data_snp['SNP'][0:i]
X_snp=sm.add_constant(X_snp, has_constant='add')
```

```
try:
    model_snp = sm.RLM(Y_snp, X_snp,missing = 'drop').fit()
    parameters_snp[i,:]
    stderr_snp[i,:]=model_snp.bse
except:
    model_snp = 0




predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')
predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]
result = np.zeros([sim_num, len(predict_lab)])

#Sampling the regression parameters to generate predicted outcome
for j in range(0, sim_num):

    #Labour
    param_lab = np.random.multivariate_normal(np.asarray(model_lab.params), RegressionResults.cov_params(model_lab), sim_num)
    predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')


    #fill NaN with mean
    #predict_lab = predict_lab.fillna(predict_lab.mean())
    predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]

    result_temp =  np.matmul(param_lab,np.array(predict_lab).T)
    #result_lab = np.random.normal(result_temp,np.std(model_lab.resid))
    err_dev = np.std(model_lab.resid)*(np.random.chisquare(len(model_lab.resid)-13)*1./(len(model_lab.resid)-13))**0.5
    result_lab = np.random.normal(result_temp,err_dev)


    #Conservative
    param_con = np.random.multivariate_normal(np.asarray(model_con.params), RegressionResults.cov_params(model_con), sim_num)
    predict_con=sm.add_constant(data_con.drop(['Con[b]'], axis = 1)[i:], has_constant='add')
```

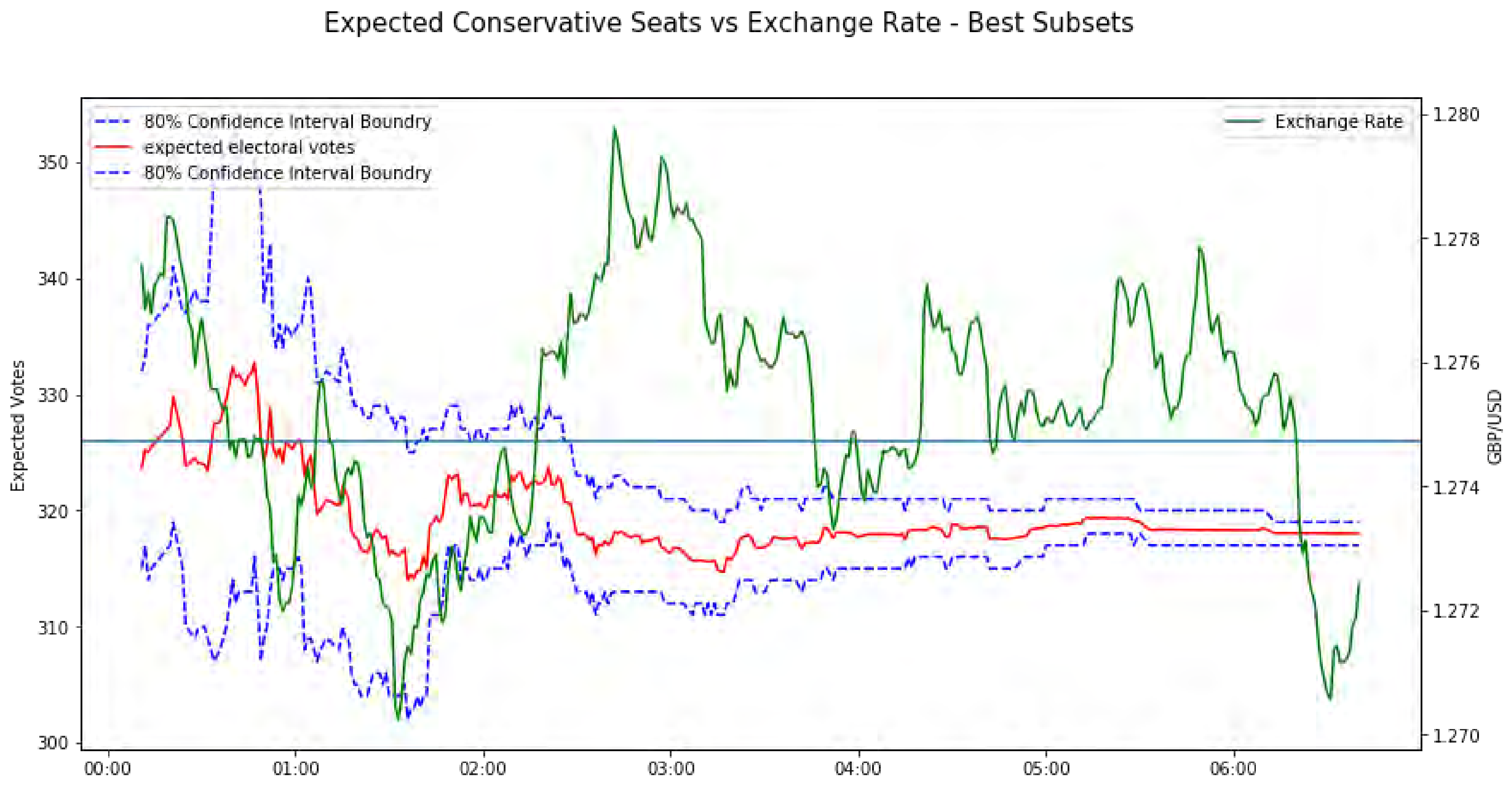
```
#predict_con = predict_con.fillna(predict_con.mean())
predict_con = predict_con[np.isfinite(predict_con['Con_poll'])]

result_temp = np.matmul(param_con,np.array(predict_con).T)
#result_con = np.random.normal(result_temp,np.std(model_con.resid))
err_dev = np.std(model_con.resid)*(np.random.chisquare(len(model_con.resid)-13)*1./(len(model_con.resid)-13))**0.5
result_con = np.random.normal(result_temp,err_dev)




#SNP
predict_snp=sm.add_constant(data_snp.drop(['SNP'],axis = 1)[i:], has_constant='add')
predict_snp = predict_snp[np.isfinite(predict_snp['SNP_poll'])]
if not model_snp==0 and len(model_snp.resid)>8:
    param_snp = np.random.multivariate_normal(np.asarray(model_snp.params), RegressionResults.cov_params(model_snp), sim_num)

    #predict_snp = predict_snp.fillna(predict_snp.mean())

    result_temp =  np.matmul(param_snp,np.array(predict_snp).T)
    #result_snp = np.random.normal(result_temp,np.std(model_snp.resid))
    err_dev = np.std(model_snp.resid)*(np.random.chisquare(len(model_snp.resid)-4)*1./(len(model_snp.resid)-4))**0.5
    result_snp = np.random.normal(result_temp,err_dev)
    #if the poll is 0, the resulting simulated data should also be 0
    ind=np.where(predict_snp.SNP_poll==0)
    result_snp[:,ind]=0

else:
    #no data
    result_snp = np.tile(predict_snp['SNP_poll'],(sim_num,1))/100

#result_pla = np.nan_to_num(np.tile(full_table['Pla_poll'][i:],(sim_num,1)))/100
#result_ukip = np.nan_to_num(np.tile(full_table['UKIP_poll'][i:],(sim_num,1)))/100
other = 1-result_con-result_lab-result_snp

result_temp = (result_con>result_lab) & (result_con>result_snp) & (result_con>other)
result = np.append(result, result_temp,axis=0)
```

```
              #remove the zeros during initialisation
              result = np.delete(result,range(0,sim_num),0)
              result = result.astype(int)

              EEV=len(full_table.Party[0:i][full_table.Party=='Con'])+ np.count_nonzero(result, axis=1)


              #make table for plot
              pred.time[i]=full_table.time[i]
              pred.Constituency[i]=full_table.ID[i]
              pred.low[i]=np.percentile(EEV,10)
              pred.ave[i]=np.mean(EEV)
              pred.high[i]=np.percentile(EEV,90)

          # keep only the last value at a certain time. And remove NAs.
          pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
          pred.index = np.arange(0,len(pred))
```

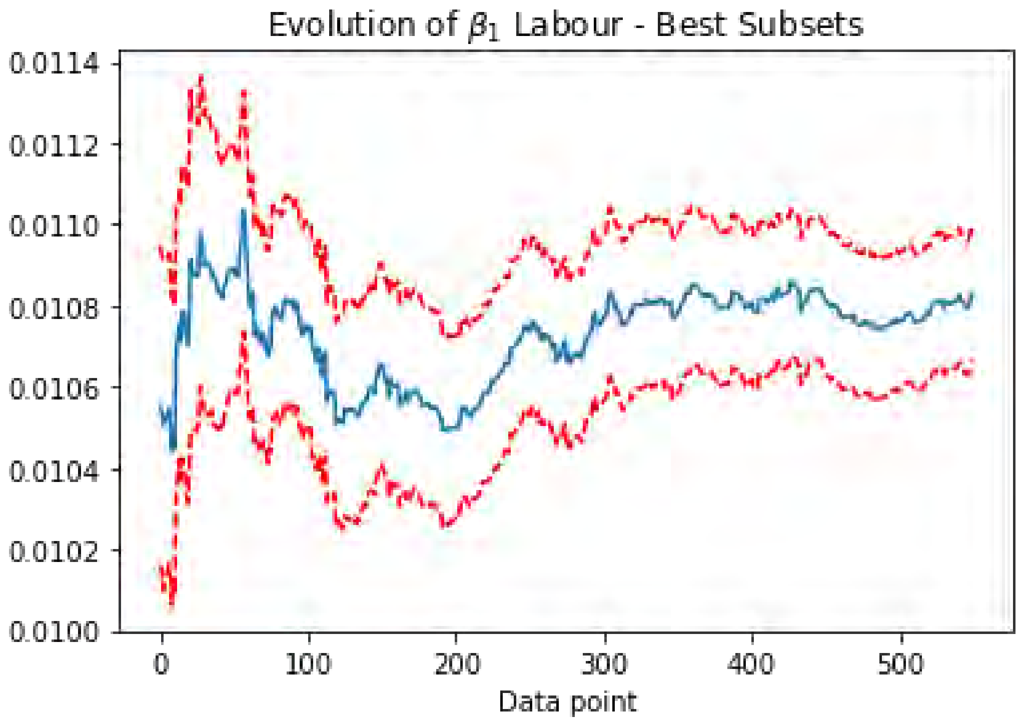
```
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:72: RuntimeWarning: covariance is not positive-semidefinite.
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:87: RuntimeWarning: covariance is not positive-semidefinite.
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:126: RuntimeWarning: invalid value encountered in greater
C:\Users\AlexH\Anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1353: RuntimeWarning: divide by zero encountered in double_scalar
  return np.dot(wresid, wresid) / self.df_resid
```

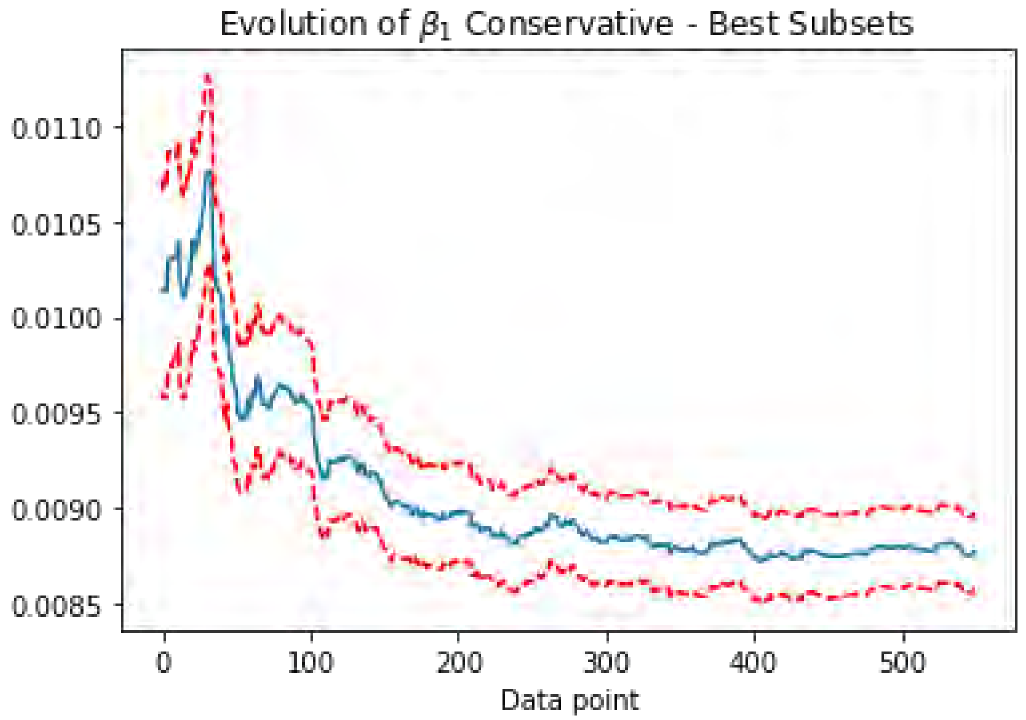
```
In [212]: fig, ax1 = plt.subplots()
          fig.set_size_inches(14, 7)
          ax1.plot_date(pred.time, pred.low,'b--', label='80% Confidence Interval Boundry')
          ax1.plot_date(pred.time, pred.ave,'r-', label='expected electoral votes')
          ax1.plot_date(pred.time, pred.high,'b--', label='80% Confidence Interval Boundry')
          plt.ylabel('Expected Votes')
          pylab.legend(loc='upper left')
          plt.axhline(y=326)
          #ax1.set_ylim([280,380])

          ax2 = ax1.twinx()
          ax2.plot(fx['mean'],'g', label='Exchange Rate')
```

```
HMFmt = mdates.DateFormatter('%H:%M')
ax1.xaxis.set_major_formatter(HMFmt)
_ = plt.xticks(rotation=90)

plt.ylabel('GBP/USD')
plt.xlabel('UTC Time')
pylab.legend(loc='upper right')

fig.suptitle('Expected Conservative Seats vs Exchange Rate - Best Subsets', fontsize=15)
fig.savefig('results-final.png')
plt.show()
```



```
In [213]: plt.plot(parameters_lab[50:600,1])
          plt.plot(parameters_lab[50:600,1]-stderr_lab[50:600,1],'r--')
          plt.plot(parameters_lab[50:600,1]+stderr_lab[50:600,1],'r--')
          plt.title(r'Evolution of $\beta_1 $ Labour - Best Subsets')
          plt.xlabel('Data point')
          plt.savefig('lab final b1.png')
          plt.show()
plt.plot(parameters_con[50:600,1])
plt.plot(parameters_con[50:600,1]-stderr_con[50:600,1],'r--')
plt.plot(parameters_con[50:600,1]+stderr_con[50:600,1],'r--')
plt.title(r'Evolution of $\beta_1 $ Conservative - Best Subsets')
plt.xlabel('Data point')
plt.savefig('con final b1.png')
plt.show()
```





```
In [214]: columns = ['time', 'Constituency','low', 'ave', 'high']
          index= full_table.index
          pred=pd.DataFrame(index=index, columns=columns)

          parameters_con = np.empty((650,9))
          stderr_con = np.empty((650,9))
          parameters_lab = np.empty((650,10))
          stderr_lab = np.empty((650,10))
          parameters_snp = np.empty((650,4))
          stderr_snp = np.empty((650,4))
```

```
sim_num = 10
for i in range(18,len(full_table)-14):

    #labour regression
   # data_lab=full_table[['Lab_poll','Lab[c]','Total','Last Election','WageMedianConst','Pop65ConstRate']]
    data_lab= pd.concat([full_table['Lab_poll'],\
                         full_table['Lab_poll']*full_table['london'],full_table['Lab_poll']*full_table['iswales'],
                         full_table['WageMedianConst'] , \
                         full_table['WageMedianConst']*full_table['london'], full_table['WageMedianConst']*full_table['iswales'],\
                         full_table['WageMedianConst']*full_table['isscot'],\
                         full_table['Pop65ConstRate'] ,\
                         full_table['Pop65ConstRate']*full_table['iswales'],\
                          full_table['Lab[c]']], axis=1)
    #,full_table['Con_poll']*full_table['iscon'],
    X_lab = data_lab.drop('Lab[c]',axis=1)[0:i]
    Y_lab=data_lab['Lab[c]'][0:i]
    X_lab=sm.add_constant(X_lab, has_constant='add')
    model_lab = sm.RLM(Y_lab, X_lab,missing = 'drop').fit()

    parameters_lab[i,:]=model_lab.params
    stderr_lab[i,:]=model_lab.bse

    #Conservatives regression
    #data_con=full_table[['Con_poll','Con[b]','Total','Last Election','WageMedianConst','Pop65ConstRate']][0:i].dropna(axis=0)
    data_con= pd.concat([full_table['Con_poll'],\
                         full_table['Con_poll']*full_table['london'],\
                         full_table['Con_poll']*full_table['isscot'],
                         full_table['WageMedianConst']*full_table['london'], full_table['WageMedianConst']*full_table['iswales'],\
                         full_table['Pop65ConstRate'] ,\
                         full_table['Pop65ConstRate']*full_table['london'],\
                         full_table['Pop65ConstRate']*full_table['isscot'],\
                         full_table['Con[b]']], axis=1)
    X_con = data_con.drop(['Con[b]'], axis = 1)[0:i]
    Y_con=data_con['Con[b]'][0:i]
    X_con=sm.add_constant(X_con, has_constant='add')
    model_con = sm.RLM(Y_con, X_con,missing = 'drop').fit()
```

```
parameters_con[i,:]=model_con.params
stderr_con[i,:]=model_con.bse




data_snp=pd.concat([full_table['SNP_poll'], full_table['WageMedianConst'],
                 full_table['Pop65ConstRate'], full_table['SNP']], axis=1)
X_snp = data_snp.drop(['SNP'],axis = 1)[0:i]
Y_snp=data_snp['SNP'][0:i]
X_snp=sm.add_constant(X_snp, has_constant='add')
try:
    model_snp = sm.RLM(Y_snp, X_snp,missing = 'drop').fit()
    parameters_snp[i,:]
    stderr_snp[i,:]=model_snp.bse
except:
    model_snp = 0


prob = np.zeros(10)
for k in range(0,10):

    predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')
    predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]
    result = np.zeros([sim_num, len(predict_lab)])


    #Sampling the regression parameters to generate predicted outcome
    for j in range(0, sim_num):

        #Labour
        param_lab = np.random.multivariate_normal(np.asarray(model_lab.params), RegressionResults.cov_params(model_lab), sim_num)
        predict_lab=sm.add_constant(data_lab.drop('Lab[c]',axis=1)[i:], has_constant='add')


        #fill NaN with mean
        #predict_lab = predict_lab.fillna(predict_lab.mean())
```

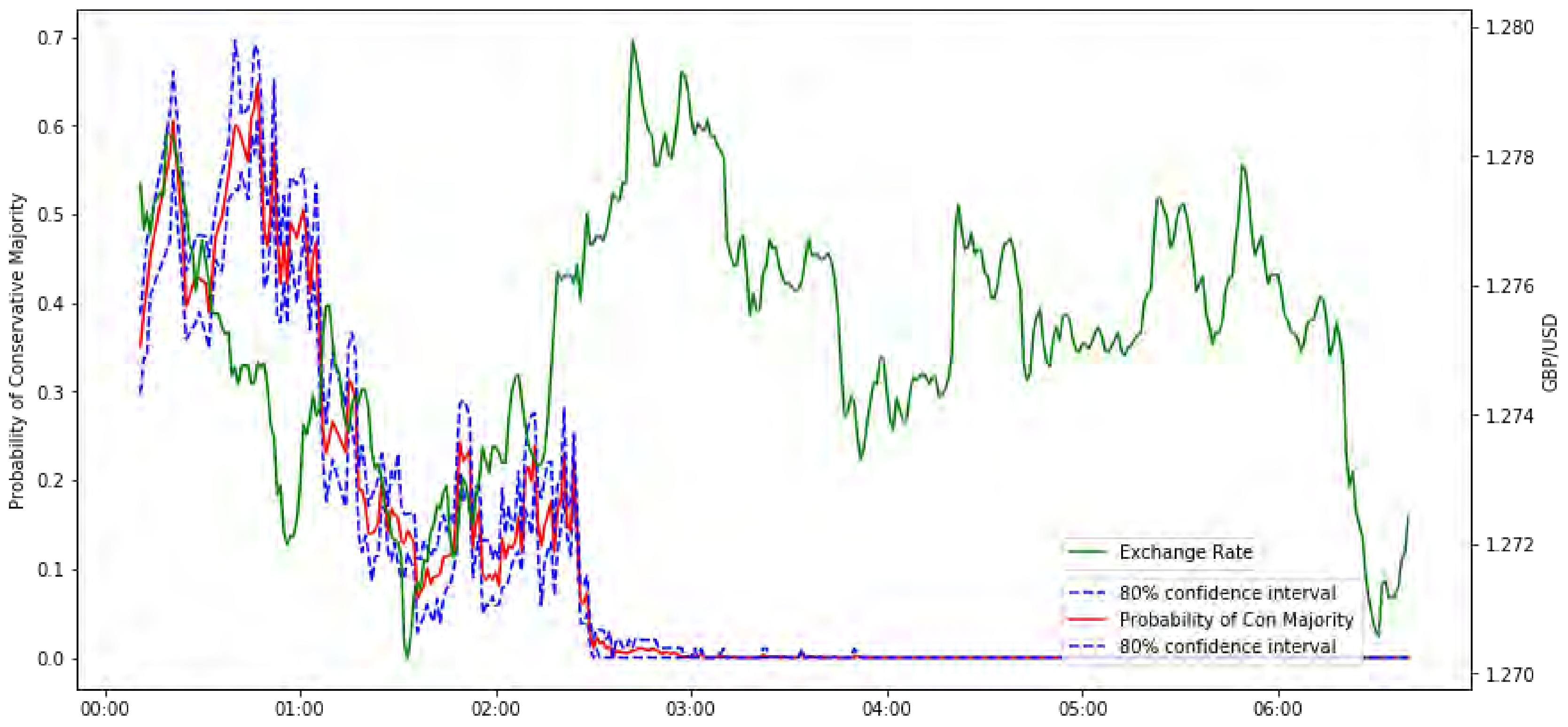
```
predict_lab=predict_lab[np.isfinite(predict_lab['Lab_poll'])]

result_temp =  np.matmul(param_lab,np.array(predict_lab).T)
#result_lab = np.random.normal(result_temp,np.std(model_lab.resid))
err_dev = np.std(model_lab.resid)*(np.random.chisquare(len(model_lab.resid)-13)*1./(len(model_lab.resid)-13))**0.5
result_lab = np.random.normal(result_temp,err_dev)


#Conservative
param_con = np.random.multivariate_normal(np.asarray(model_con.params), RegressionResults.cov_params(model_con), sim_num)
predict_con=sm.add_constant(data_con.drop(['Con[b]'], axis = 1)[i:], has_constant='add')


#predict_con = predict_con.fillna(predict_con.mean())
predict_con = predict_con[np.isfinite(predict_con['Con_poll'])]

result_temp = np.matmul(param_con,np.array(predict_con).T)
#result_con = np.random.normal(result_temp,np.std(model_con.resid))
err_dev = np.std(model_con.resid)*(np.random.chisquare(len(model_con.resid)-13)*1./(len(model_con.resid)-13))**0.5
result_con = np.random.normal(result_temp,err_dev)



#SNP
predict_snp=sm.add_constant(data_snp.drop(['SNP'],axis = 1)[i:], has_constant='add')
predict_snp = predict_snp[np.isfinite(predict_snp['SNP_poll'])]
if not model_snp==0 and len(model_snp.resid)>8:
    param_snp = np.random.multivariate_normal(np.asarray(model_snp.params), RegressionResults.cov_params(model_snp), sim_num)

    #predict_snp = predict_snp.fillna(predict_snp.mean())

    result_temp =  np.matmul(param_snp,np.array(predict_snp).T)
    #result_snp = np.random.normal(result_temp,np.std(model_snp.resid))
    err_dev = np.std(model_snp.resid)*(np.random.chisquare(len(model_snp.resid)-4)*1./(len(model_snp.resid)-4))**0.5
    result_snp = np.random.normal(result_temp,err_dev)
    #if the poll is 0, the resulting simulated data should also be 0
    ind=np.where(predict_snp.SNP_poll==0)
```

```
                result_snp[:,ind]=0

            else:
                #no data
                result_snp = np.tile(predict_snp['SNP_poll'],(sim_num,1))/100

            #result_pla = np.nan_to_num(np.tile(full_table['Pla_poll'][i:],(sim_num,1)))/100
            #result_ukip = np.nan_to_num(np.tile(full_table['UKIP_poll'][i:],(sim_num,1)))/100
            other = 1-result_con-result_lab-result_snp

            result_temp = (result_con>result_lab) & (result_con>result_snp) & (result_con>other)
            result = np.append(result, result_temp,axis=0)

        #remove the zeros during initialisation
        result = np.delete(result,range(0,sim_num),0)
        result = result.astype(int)

        #filling the result of intermediates states with current count
    #    select.result[select.result.isnull()[0:i]]=(select['trump'][select.result.isnull()[0:i]]>select['clinton'][select.result.isnull(

        EEV=len(full_table.Party[0:i][full_table.Party=='Con'])+ np.count_nonzero(result, axis=1)
    #    print(jointb['time'][i],jointb['State'][i], EEV)



        prob[k] = sum(EEV>326)/len(EEV)


    #make table for plot
    pred.time[i]=full_table.time[i]
    pred.Constituency[i]=full_table.ID[i]
    pred.low[i]=np.percentile(prob,10)
    pred.ave[i]=np.mean(prob)
    pred.high[i]=np.percentile(prob,90)

# keep only the last value at a certain time. And remove NAs.
pred=pred.drop_duplicates(subset='time', keep='last').dropna(axis=0, how='all')
```

```
         pred.index = np.arange(0,len(pred))

C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:89: RuntimeWarning: covariance is not positive-semidefinite.
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:104: RuntimeWarning: covariance is not positive-semidefinite.
C:\Users\AlexH\Anaconda3\lib\site-packages\ipykernel_launcher.py:153: RuntimeWarning: invalid value encountered in greater
C:\Users\AlexH\Anaconda3\lib\site-packages\statsmodels\regression\linear_model.py:1353: RuntimeWarning: divide by zero encountered in double_scalar
  return np.dot(wresid, wresid) / self.df_resid
```

```
In [215]: #%%
          fig, ax1 = plt.subplots()
          fig.set_size_inches(14, 7)
          ax1.plot_date(pred.time, pred.low,'b--', label='80% confidence interval')
          ax1.plot_date(pred.time, pred.ave,'r-', label='Probability of Con Majority')
          ax1.plot_date(pred.time, pred.high,'b--', label='80% confidence interval')
          plt.ylabel('Probability of Conservative Majority')
          #pylab.legend(loc='lower right')
          #plt.axhline(y=270)
          ax1.legend(loc='center left', bbox_to_anchor=(0.7, 0.1))
          #ax1.set_ylim([0.5,0.9])

          ax2 = ax1.twinx()
          ax2.plot(fx['mean'],'g', label='Exchange Rate')
          #ax2.set_ylim([18.5,21.5])

          HMFmt = mdates.DateFormatter('%H:%M')
          ax1.xaxis.set_major_formatter(HMFmt)
          _ = plt.xticks(rotation=90)

          plt.ylabel('GBP/USD')

          #pylab.legend(loc='upper right')
          ax2.legend(loc='center left', bbox_to_anchor=(0.7, 0.2))

          fig.suptitle('Probability of Conservative Majority vs Exchange Rate - Best Subset', fontsize=15)

          #plt.figure(figsize=(20,10))
          plt.show()
          fig.savefig('prob-final.png')
```

Probability of Conservative Majority vs Exchange Rate - Best Subset



```
In [264]: model_con.summary()

Out[264]: <class 'statsmodels.iolib.summary.Summary'>
          """
                               Robust linear Model Regression Results
          ==============================================================================
          Dep. Variable:                 Con[b]   No. Observations:                  615
```

```
Model:                            RLM   Df Residuals:                      606
Method:                          IRLS   Df Model:                            8
Norm:                          HuberT
Scale Est.:                       mad
Cov Type:                          H1
Date:                Sun, 25 Feb 2018
Time:                        14:31:38
No. Iterations:                    20
=================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
const            -0.0573      0.010     -5.830      0.000      -0.077      -0.038
Con_poll          0.0088      0.000     45.892      0.000       0.008       0.009
0                -0.0017      0.001     -2.338      0.019      -0.003      -0.000
1                 0.0033      0.001      3.800      0.000       0.002       0.005
2             -8.188e-05    3.4e-05     -2.407      0.016      -0.000   -1.52e-05
3                 0.0001   1.85e-05      6.798      0.000    8.95e-05       0.000
Pop65ConstRate    0.5456      0.057      9.560      0.000       0.434       0.658
4                 0.8284      0.252      3.294      0.001       0.335       1.321
5                -0.3547      0.128     -2.769      0.006      -0.606      -0.104
=================================================================================

If the model instance has been used for another fit with different fit
parameters, then the fit options might not be the correct ones anymore .
"""
```

```
In [265]: model_lab.summary()
```

```
Out[265]: <class 'statsmodels.iolib.summary.Summary'>
"""
                    Robust linear Model Regression Results
==============================================================================
Dep. Variable:                 Lab[c]   No. Observations:                  615
Model:                            RLM   Df Residuals:                      605
Method:                          IRLS   Df Model:                            9
Norm:                          HuberT
Scale Est.:                       mad
Cov Type:                          H1
```

```
Date:                Sun, 25 Feb 2018
Time:                        14:31:38
No. Iterations:                    22
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
const               0.1696      0.021      8.065      0.000       0.128       0.211
Lab_poll            0.0108      0.000     67.779      0.000       0.010       0.011
0                   0.0007      0.000      2.516      0.012       0.000       0.001
1                  -0.0014      0.001     -2.578      0.010      -0.003      -0.000
WageMedianConst -5.149e-05   2.53e-05     -2.033      0.042      -0.000   -1.85e-06
2                -3.16e-05   1.92e-05     -1.647      0.100   -6.92e-05       6e-06
3                   0.0003    8.6e-05      2.953      0.003    8.54e-05       0.000
4                  -0.0001   9.51e-06    -10.728      0.000      -0.000   -8.34e-05
Pop65ConstRate     -0.4526      0.044    -10.184      0.000      -0.540      -0.366
5                  -0.4594      0.146     -3.155      0.002      -0.745      -0.174
===================================================================================

If the model instance has been used for another fit with different fit
parameters, then the fit options might not be the correct ones anymore .
"""
```