PREDICTING LONDON'S HOUSING PRICES

A Brief Study On London's Housing Market
Evolution From 1995 To 2020

By

David Pacheco Aznar, 1565824

Autonomous University of Barcelona

Barcelona

Complex Data Analysis

Bachelor's degree in

Computational Mathematics And Data Analytics

2020-2021

Contents

PREDICTING LONDON'S HOUSING PRICES

A Brief Study On London's Housing Market
Evolution From 1995 To 2020

Abstract

The following work pretends to calculate a forecast of London's housing prices just by studying the time series involved. To do so, a monthly actualized index has been created weighting each neighborhood according to the number of houses sold each month. Hence, the assumption that there is a correlation between the values in the series. Finally, the idea is to prove that there is autocorrelation indeed and to find a linear model that describes the data best in order to get a proper forecast for the next ten months with the given data.

1 Motivation

Price fluctuations are often difficult to predict and model. However, the utility linked to a great prediction is invaluable. When predicting the future value of a time series, usually time series analysis is a good enough way to go; since it is often thought that past values of the series say a lot about future ones. A lot of research has been done about these type of series involving its stationarity and ways to forecast them.

Since housing is one of the most expensive markets and also one of the most stable, despite having had a recent crash, the following study will involve a study of the housing prices across time in London. The fact that there is so much into play when buying properties, makes a study about their future crucial.

The reason London city has been selected is that is one of the biggest cities in the world and at the same the top 4 city in the world by GDP. The population gets to about 9.4 million people as of 2021.

It is worth noting that depending on the district, housing prices will be higher or lower. However, the main interest of the study is to get the average price a regular citizen will have to pay in order to say he or she lives in London. Hence, no specific district research is going to be done. This way, a fair approximation property price will account for most buyers and, as a consequence, be more helpful in order to figure the expected cost in the upcoming months.

2 The data-set

In order to do the research proposed in the *abstract* and *motivation*. The data-set that will be used can be found in the following link: `https://www.kaggle.com/justinas/housing-in-london`.

The one we will focus on is *housing_in_london_monthly_variables.csv*. There we can find 7 different columns which, in short are: `date`, `area`, `average_price`, `code`, `houses_sold`, `no_of_crimes` and `borough_flag`. Briefly describing each of these columns we get that the `date` represents each month across time (from 1995 to 2020); `average_price` is the average price houses were sold in the specific district defined by the column `area` per month. In turn, each `area` has its own `code`. In addition to that, we have a column representing the number of `houses_sold`. The next column, that is: `no_of_crimes` represents the number of crimes per month per area. We will not take this column into account for our research, though it could lead to some extensive studies about its relation to the number of houses sold and their average price. Finally, the last column: `borough_flag` represents whether an `area` belongs to London or is a bigger place like *england*, *outer london* or *east midlands*.

After having presented the columns of the data-set we are going to be working with, we move on to explain how the series that we are going to be using has been constructed. So, in short, the series we are going to be using is a weighted average of the average house prices per area in London, i.e. `borugh_flag`=1. The weights are recalculated each month so the average price is always as close to reality as possible and they are based on the number of houses sold. That is, the more houses sold in an area in a month, the higher the weight that area will have that month when calculating the index.

The following plot shows how the housing index looks like:

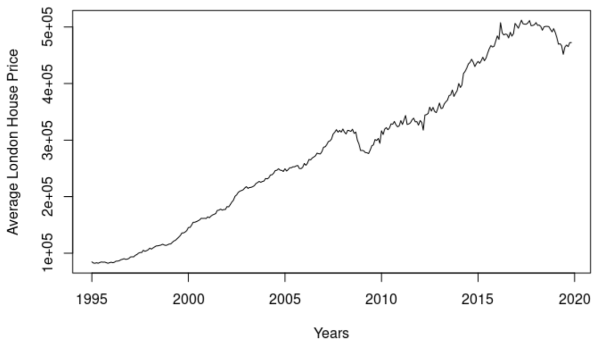

Figure 1: London housing index from 1995 to 2020

3 Data Analysis and Forecast

Now that we have our data-set, the first things that we are going to be doing are checking for break-points, seasonality and trend. Hence, we are going to perform a Chow test, to see if any break-point is detected, a permutation test to see whether there is autocorrelation in the time series and a dickey fuller test also to make sure we have stationary data after correcting all the needed adjustments.

The first thing we are going to do is a permutation test on our time series. The hypothesis under which we are going to do the test are the following:

$\mathcal{H}_0$: There is no autocorrelation in the time series. That is, observations follow a white noise: $X_i = \mu + Z_i$, where Z_i are iid such that $\mathbb{E}(Z_i) = 0$.

$\mathcal{H}_1$: There is indeed autocorrelation in the time series and observations do not follow a white noise, but rather they follow some trend.

Performing the permutation test for the first 5 coefficients, that is, checking for the first 5 orders of autocorrelation, we obtain the following p-values:

Order	1	2	3	4	5
Permutation test p-value	0	0	0	0	0

Table 1: p-values before applying transformations to make the series stationary

and we find that all the p-values are way below 0.05, they are actually 0. Hence,, we reject our null hypothesis and can affirm that there is autocorrelation within our data-set. We can make it more visual by plotting the decomposition of our time series and the ACF (autocorrelation function).

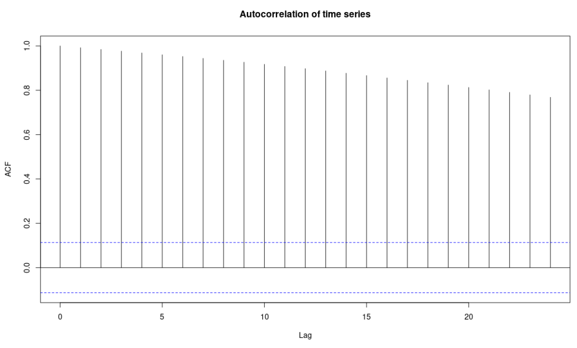

Figure 2: ACF plot

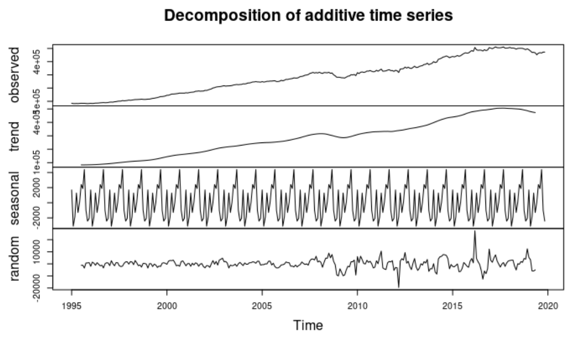

Figure 3: Time series decomposition

Analyzing figures [3] we can see that autocorrelation decreases as the order of the ACF grows, but nonetheless, even after 20 orders, the correlation is of about 0.8. If we now look at the time series decomposition, see can clearly see an up trend and seasonality in our data.

3.1 Making our time series stationary

In order to remove the trend and seasonality, we will have to apply the log transformation to our datset, and then one differentiation in order to have a stationary time series to be able to comfortably work with.

After stabilizing the mean and variance, we get that our time series looks as follows:

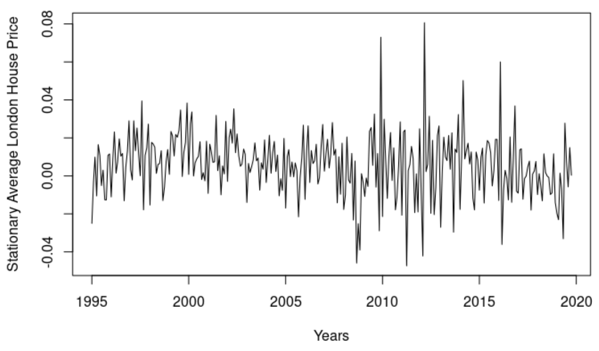

Figure 4: Time series after applying log and differentiation

We can now make sure our series is stationary by applying the dickey fuller test, which uses the following hypothesis: $\mathcal{H}_0$: a unit root is present in a time series sample; $\mathcal{H}_1$: Series is stationary. The result we are going to focus on is its p-value and it has a value of 0.01. Therefore, we reject the null hypothesis and we can safely say that our time series is not stationary

If we now execute our permutation test with this data, we get the following table:

Order	1	2	3	4	5
Permutation test p-value	0.9915	0.0614	3e-04	0.0896	0.1886

Table 2: p-values after applying transformations to make the series stationary

Since a permutation test is not exact, we will calculate confidence intervals on each of these p-values by using the following formula for confidence intervals:

$$\hat{x} = \overline{x} \pm z_{\alpha/2} Se_x$$

where $\overline{x}$ is the sample mean of x and $z_{\alpha/2}$ is the critical value, as we are looking for a 95% confidence interval, this value is 1.96. Se_x denotes the standard error, which is represented by: $\sqrt{\frac{\hat{x}(1-\hat{x})}{n}}$. If we check for the order 2 and 4 autocorrelations, since they are the closest to the 0.05 critical value, we get that 0.0614 ± 0.004705226 and 0.0896 ± 0.005597914 and none of them falls below the critical value set. We now accept $\mathcal{H}_0$ for all orders except for order 3. That is, there is still some autocorrelation of order 3. We will take this number into account for when we model the series and try to do a forecast.

3.2 Checking for break-points

As we mentioned at the beginning of the section, we now will look for break-points in our time series. The reason for that is that they will allow us to make a better forecast. To detect whether there are any break-points in our time series, we will use the Chow test. This tests uses the following hypothesis:

$\mathcal{H}_0$: There are no break-points in the time series, i.e. that the data set can be represented with a single regression line.

$\mathcal{H}_1$: There are break-points in the time series.

After applying it to our original time series, the p-value is 0.003972, which is lower than 0.05, and hence the null hypothesis is rejected. As a result, there are break-points in the time series. After computing them, we get the following indices: Since if we use the last break-point we would

break-point index	55	117	172	229
Date of break-point	1999-07-01	2004-09-01	2009-04-01	2014-01-01

Table 3: proposed breakpoints by software

get very few samples left, we use the one before that. That is from april 1st of 2009[1]. If we look at the plot, we see that the break-point selection is fairly acceptable:

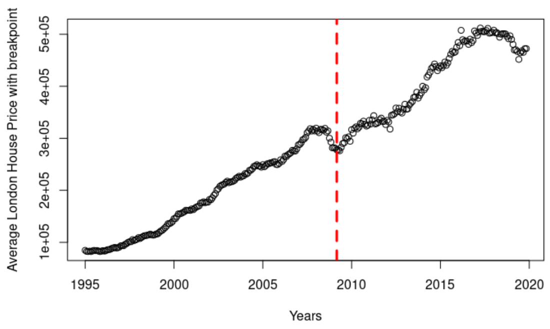

Figure 5: Time series with break-point

[1]From a historical point of view, it represents mainly the worse part of the 2008 recession.

Hence, the part of the data-set we are going to be using for our linear model is going to be the right one. It is going to give us a more accurate forecast, as it will contain only after the recession data. This way, the model will be capable of fitting better the time series.

3.3 The Forecast

Now that we have everything set up, we can proceed to perform our forecast. For that, we are going to be using an ARIMA (Autoregressive integrated moving average) model. Looking back at the *Data Exploration* section, we got that there was a persistent autocorrelation of order 3, and we had to make the time series stationary by differentiating and applying the logarithm[2] to remove seasonality and trend. As a result of that, we could suggest an ARIMA model with autoregression parameter equal 3: $p = 3$; differentiation parameter of 1, $d = 1$ and moving average parameter equal to 0. As a result, our model is an ARIMA(3, 1, 0).

By making use of the `auto.arima` function with `ic='aic'`, we indeed get that the best predictive model for the data is the ARIMA(3, 1, 0). Hence, the following plot shows the forecast of the model:

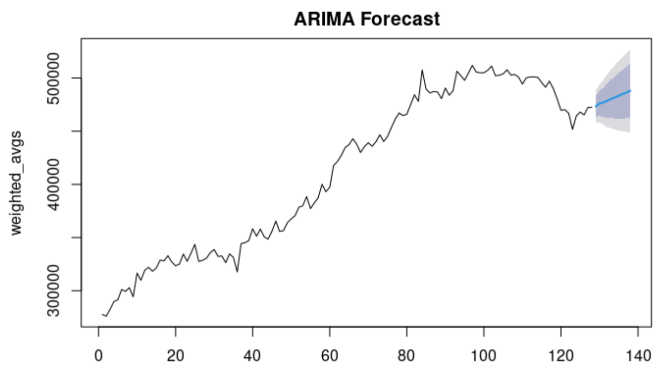

Figure 6: ARIMA(3, 1, 0) forecast

Where the darker bounds represent an 80% confidence interval and the lighter ones a 95% confidence interval on the forecast. The blue line represents the most likely set of points for the forecast. The last observation of the index accounts for November 1st, 2019th.

[2]A Box-Cox transformation could have also been used.

As a result, the forecast values are:

Date	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
"2019-12-01"	473323.4	463756.9	482890.0	458692.6	487954.3
"2020-01-01"	476053.3	464247.0	487859.5	457997.2	494109.3
"2020-02-01"	477094.6	463375.0	490814.1	456112.4	498076.8
"2020-03-01"	478585.6	462324.7	494846.5	453716.7	503454.5

Table 4: London housing predictions for the first trimester of 2020

The forecast looks reasonable. There is not a very significant difference between the 80% confidence interval and the 95% one. Now, we will take a look at how the model fitted our time series and then we will move on to analysing the residuals of the model.

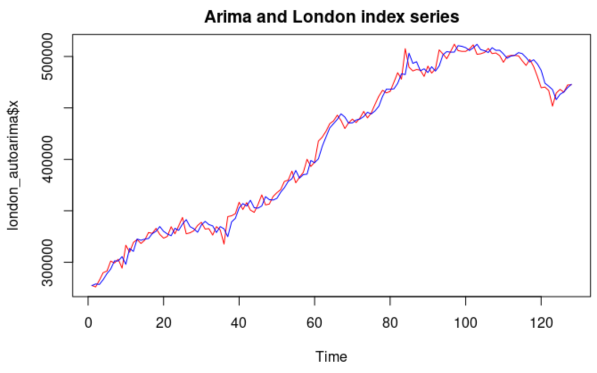

Figure 7: ARIMA model (red) onto time series data(blue)

For a linear model, we can say the model represents fairly well our data-set. However, it is interesting to see how our model has performed in reality when fitting the curve. For that, we will study the errors the model has made. [3] We can see that the MPE (mean percentage errror) is of

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-288.6474	5372.831	3700.571	-0.1968881	1.289505	0.9283399	0.06029688

Table 5: Error metrics

the order of 0.2 and the ME (mean error) is about 288, which taking into account the series takes values to up to 5×10^5, is low.

[3]The following table shows the Mean Error, the Root Mean Square Error, the Mean Percentage Error, the Mean Absolute Percentage Error, the Mean Absolute Standard Error and the First-Order Autocorrelation Coefficient.

3.4 Residual analysis

Now that we have our linear model set up and the forecast, we will examine the residuals of our model and see if they are valid or not. We are looking to get normally distributed residuals. To do so, we will perform a Shapiro-Wilk test, and then a Jackknife to check whether our residuals are normal. We will also check if they are stationary, performing the Dickey-Fuller test. Finally, we will perform a non-parametric bootstrap on the residual to get confidence intervals on the ARIMA parameters and mean of the model.

3.4.1 Check Stationary Residuals

Firstly, we will check at whether model residuals are stationary or not. In this case, we will just take a look at the Dickey-fuller test, used in the *Data Exploration* section.

The result is that the p-value of that is 0.01, which rejects the null hypothesis and hence, the residuals of the are stationary.

3.4.2 Check Normality in residuals

One important step now is check whether our residuals are normally distributed. We can plot them:

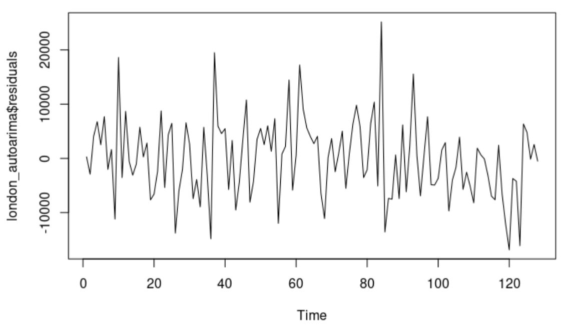

Figure 8: Residuals line plot

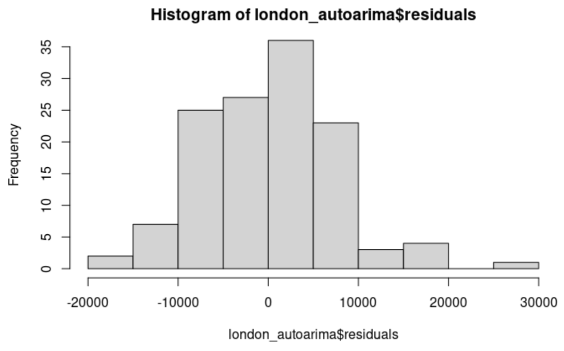

Figure 9: Residuals histogram

At first glance it does not seem trivial to confirm whether these residuals follow a normal distribution or not. So we execute the Shapiro test, with hypothesis:

$\mathcal{H}_0$: data is normally distributed;

$\mathcal{H}_1$: data is not normally distributed. At a 95% confidence interval.

The results show a p-value of 0.08298. And we can say the data is indeed normally distributed, however, is close to the 0.05 limit.

We can now perform a Jackknife test on the residuals to see whether their skewness and kurtosis correspond to those of a normal distribution. If residuals followed a normal distribution, their skewness should be 0, and their kurtosis, 3. The results of the jackknife are the following:

	Skewness	Se Skewness	Kurtosis	Se Kurtosis
values	0.4171544	0.2878775	3.823241	0.6411394

Table 6: Skewness, Kurtosis and their standard errors

Since our jackknife computation is not exact, we can derive confidence intervals for our statistics of interest with the following formulas:

$$\hat{s} = \overline{s} \pm z_{\alpha/2} Se_s; \quad \hat{k} = \overline{k} \pm z_{\alpha/2} Se_k$$

where $\overline{s}$ and $\overline{k}$ are the estimates of the Skewness and Kurtosis in the table, Se_s and Se_k their respective errors and $z_{\alpha/2}$ the critical value (in this case 1.96 for a 95% confidence interval).

As a result, our confidence intervals end up as follows:

$$\hat{s} \in (0.4171544 \pm 1.96 \cdot 0.2878775) = (-0.1470855, 0.9813943)$$
$$\hat{k} \in (3.823241 \pm 1.96 \cdot 0.6411394) = (2.566608, 5.079874)$$

Since $\hat{s} = 0$ belongs to its interval and $\hat{k} = 3$, belongs its interval, we cannot reject the null hypothesis. And we can say with a 95% confidence, that residuals follow a normal distribution.

3.4.3 Non-Parametric Bootstrap on the residuals

Ending the residuals study, we are going to perform a non parametric bootstrap to get confidence intervals on the ARIMA(3, 1, 0) parameters and mean.

The confidence intervals are the following:

	$\hat{\mu}$	$\hat{\phi}_1$	$\hat{\phi}_2$	$\hat{\phi}_3$
lower 2.5%	-0.09533518	-1.1001690	-0.8710888	-0.6716845
higher 97.5%	0.06830208	0.5242646	0.2834079	0.4531948

Table 7: 95% confidence intervals on ARIMA(3, 1, 0) coefficients

The coefficients calculated by the `auto.arima` are:

$\phi_1 : -0.27679613$, $\phi_2 : -0.06930736$ i $\phi_3 : 0.16474590$.

The mean of the coefficients calculated by bootstrap:

$\hat{\phi}_1 = -0.2879522$; $\hat{\phi}_2 = -0.2938405$ and $\hat{\phi}_3 = -0.1092449$

And their intervals defined by:

$\hat{\phi}_1 \in (-0.2879522 \pm 0.8122168)$; $\hat{\phi}_2 \in (-0.2938405 \pm 0.5772484)$; $\hat{\phi}_3 \in (-0.1092449 \pm 0.5624397)$

We can observe that the first coefficient was calculated almost to perfection, however, the other two, got a higher index of error. Despite that, real coefficients belong to all the intervals, so we can say non parametric bootstrap did not do bad at all. Probably including some drift to the ARIMA in the non parametric bootstrap would help to get a better prediction of the coefficients, but we will not cover it here.

Finally, we will plot the four histograms representing the values obtained from the non parametric bootstrap and the `auto.arima` calculated value in red.

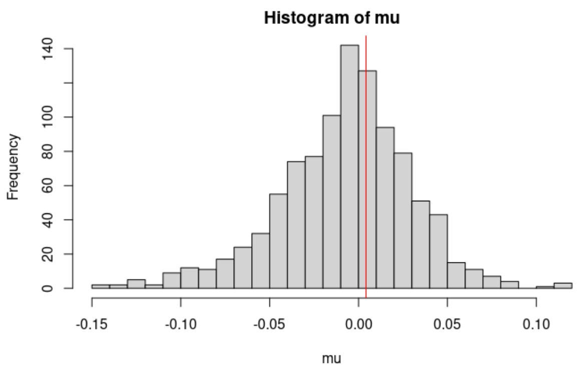

Figure 10: μ histogram

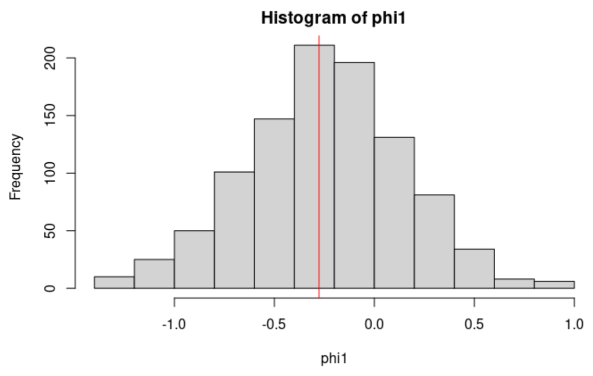

Figure 11: ϕ_1 histogram

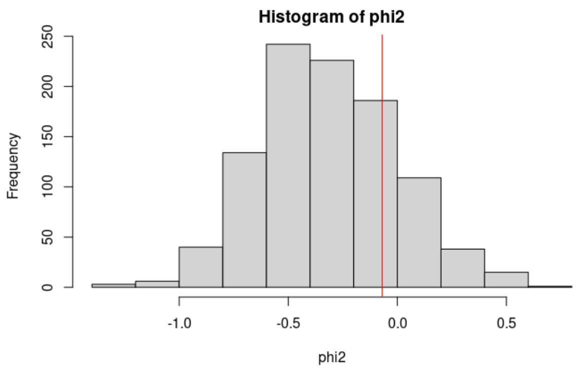

Figure 12: ϕ_2 histogram

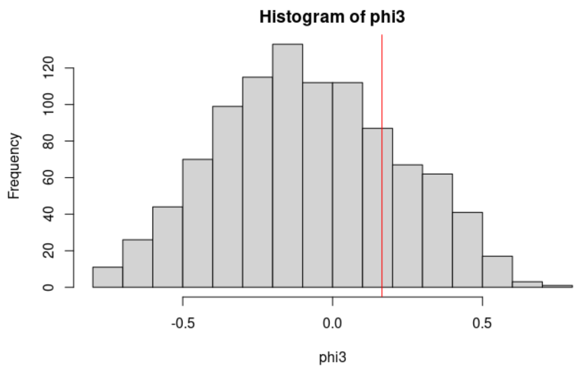

Figure 13: ϕ_3 histogram

We can see that the first two plots show that the most likely value as indeed very close to the real value calculated by the `auto.arima` model, which was using ARIMA(3, 1, 0). The next two values represented by ϕ_2 and ϕ_3 are among one of the most probable values, yet, they do not the most likely one. That could be due to a lack of iterations, since calculating the ARIMA is quite slow, it is only doing 1000 rearrangements. Nonetheless, they are still within the 95% confidence interval provided by our non-parametric bootstrap. It is also worth mentioning that to do this bootstrap the stationary data-set was used, since when using ARIMA is more convenient to get the desired results.

4 Conclusions

To conclude with this project, it is worth mentioning that we have accomplished the goals set at the beginning. It was proven that housing time-series did follow a trend and had autocorrelation. In order to get the ARIMA parameters, it was a requirement to make the series stationary and check for the order of autocorrelation in the series. The fit was surprisingly accurate showing a Mean Percentage Error of about just -0.2.

We studied the residuals of the ARIMA model that we constructed, since they were one of the most important things involving the model. We used a jackknifing techniques to estimate the skewness and kurtosis of the residuals along with the Shapiro-Wilk test to check for normality. We also made use of the Chow test to check for the presence of break-points in our time series. The Dickey-Fuller test accompanied with a permutation test to check for stationarity in our data-set. In addition, we performed a 4 months forecast for the prices of December of 2019 and the first trimester of 2020.

The results of the non-parametric bootstrap were rather surprising, since they were fairly accurate, taking into account the small number of iterations. Also, the blocks for the bootstrap where of length 25, and they were computed by using the ARIMA(3, 1, 0).

Finally, just note that the data-set had a London series which reported the average price of the houses sold in the city. But, since there was no specification about which neighborhoods where included in that series, the decision to build a weighted index was made. That way, only the neighborhoods represented in the data-set had representation and hence, the average cost of property was adjusted to the data given.

5 Statistical Methods Employed

5.1 Permutation test on acf

This method has been employed in section 3 to get whether the p-values of the acf indicated some kind of autocorrelation between observations or not. Supposing n pairs of observations $(X_1, Y_1), \dots, (X_n, Y_n)$, the steps are the following:

1. The null hypothesis is that X and Y are not correlated, that is, we could interchange the X_i between them, constructing new equivalent pairs. The more usual alternative is to assume a linear correlation. In our case, the alternative is that there is autocorrelation.
2. To consider an appropiate statistic. In this case, the autocorrelation coefficient function.
3. The next step is to randomly permute the Y column leaving the X fixed
4. Finally, we have to compare that the observed statistic to the distribution of the statistic when the Y column is permuted
5. It is important to note that these are approximations and that they should be accompanied with a confidence interval, usually of 95% significance.

5.2 Confidence interval calculation

A confidence interval tells us the probability that a parameter will fall between values around the mean. It has been used in this work to compute the confidence intervals in section 3 and 4.1.2.

In short, the steps to compute it are the following:

1. Compute the sample mean
2. In case the standard deviation is know, we have the following confidence interval: $(\overline{x} - z_{\alpha/2}\sigma/\sqrt{n}, \quad \overline{x} + z_{\alpha/2}\sigma/\sqrt{n})$. Where $z_{\alpha/2}$ follows a standard normal distribution. This confidence interval provides a $(1-\alpha)\%$ confidence interval.
3. In case the standard deviation is unknown, we have the following confidence interval: $(\overline{x} - t_\alpha(r)s/\sqrt{n}, \quad \overline{x} + t_\alpha(r)s/\sqrt{n})$.
   Where r represents the degrees of freedom and $\alpha = \frac{1-C}{2}$, for C confidence level. s is just an etimation of the standard deviation.s

5.3 Jackknife

It has been used in this work to check for normality in the residuals. It has been used to calculate confidence intervals of the coefficients of skewness and kurtosis to check for test for normality goodness of fit.

A jackknife estimate can be used when we are interested in estimated some population parameter.

1. We consider the estimation of the parameter (in this case skewness (s) and kurtosis (k)) based uppon iid observations and let $X = (X_1, X_2, \ldots, X_n)$ denote a sample of size n. The estimations of this sample are $\hat{s}$ and $\hat{k}$.
2. The i-th jackknife sample is defined as the sample of X when the i-th data point has been removed. That is, $X_(i) = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n$.
3. That will give us the estimation of the i-th jackknife estimation $\hat{s}_{(i)}$ and $\hat{k}_{(i)}$ in our case.
4. The pseudo value is going to be defined as follows: $\hat{s*}_{(i)} = n\hat{s} - (n-1)\hat{s}_{(i)}$, the same for k.
5. Finally, the jackknife estimator of s is given by: $\overline{s*} = \frac{1}{n}\sum_{i=1}^{n} \hat{s*}_{(i)}$. As before, the same for k, that is, just the mean of all the pseudo values calculated.
6. The standard error of the jackknife estimate can be calculated by just the sample standard error of the pseudo values. Can also be defined by doing: $\hat{se}_{jack} = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}\left(\hat{s}_{(i)} - \hat{s}_{(.)}\right)^2}$, where $\hat{s}_{(.)}$ is the mean of the pseudo values.

5.4 Non-Parametric Bootstrap for linear time series residuals

In our case, to do the boostrap method in ARIMA(p, d, q) time series, we first made them stationary, so that $E(Y_t) = \mu$, where μ can be estimated by the sample mean. In our case, the ARIMA has the form: $Y_t = Y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} + \phi_3 \Delta Y_{t-3} + \varepsilon_t$,
where ΔY_{t-k} represents $Y_{t-k} - Y_{t-k-1}$

In order to build a confidence interval with the bootstrap-t method, we don not have to assume nay probability distribution in our observations. We assume that the data-set is representative of the probability distribution.

1. First, estimate the population distribution from the data-set.
2. Then, simulate re-samplings from the given data-set with replacement. And will calculate the desired statistic for each of them.

3. We compute the statistic of interest for each sampling and store the value.

In our case, we computed the ARIMA(3, 1, 0) for each sample and store the value relative to each one.

5.5 ARIMA model

As suggested in the prvious subsection, the ARIMA (Autoregressive Integrated Moving Average) model has three different parameters: p, d and q.

the first one corresponds the number of time lags of the autoregressive model (AR(p)), the second one corresponds to the degree of differencing. That is, the number of times data has had past values substracted. Finally, the last component refers to the order of the moving average model.

The generalized ARIMA(p, d, q) is:

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right)(1-L)^d X_t = \delta + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

which includes the drift: $\frac{\delta}{1-\sum \varphi_i}$. The drift is often described as the μ when differentiation is 1.

The forecasts using arima models can be computed as a cascade of these two models:

$$Y_t = (1-L)^d X_t$$
$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) Y_t = \left(1 + \sum_{i=1} \theta_i L^i\right) \varepsilon_t$$

5.6 Chow test

The chow test is a test statistic which tests whether coefficients of two linear regressions in two sets of data are equal. It is usually used to find structural changes in time series. In our case, we used it for section 3.2. The statistic is defined as:

$$\frac{\left(S_C - (S_1 + S_2)\right)/k}{(S_1 + S_2)/(N_1 + N_2 + 2k)}$$

Where S_C is the sum of squared residuals of the original series; S_1 is the sum of squared residuals of the first group and S_2 of the second group. N_i represents the number of observation of each group and k is the number of parameters.

In short, the idea is that under the null hypothesis, given the following regression lines:

$$y_t = a_1 + b_1 x_{1t} + c_1 x_{2t} + \varepsilon$$
$$y_t = a_2 + b_2 x_{1t} + c_2 x_{2t} + \varepsilon$$

$a_1 = a_2, b_1 = b_2$ and $c_1 = c_2$.

5.7 Shapiro-Wilk test

The test is used to check for normality in a set of data. The null hypothesis is that given a sample $X_1, \ldots, X_n$, it comes from a normally distributed population. The statistic will oscillate between 0 and 1, and if it is lower than, say 0.05, the null hypothesis is rejected and hence the data is not normally distributed. The Shapiro-Whilk formula for the statistic can be found here.

5.8 Dickey-Fuller test

This test basically asserts that in the generation of the next value or i-th next value of a time series there is no autocorrelation factor. That is checked by calculating the unit roots. If there are none, then the series is stationary else, it is not, since there are unit roots.

In short, what the Dickey-Fuller tests is looking for is that we don't have a ρ term multiplying the previous observation, since that would imply there is autocorrelation. See this article for more information.

6 References

[1] Jesse Hemerik, Jelle Goeman. *Exact testing with random permutations.* (English), 2017 Nov 30

[2] Zhang, Yan, *Jackknife Empirical Likelihood Inferences for the Skewness and Kurtosis.* Thesis, Georgia State University, 2014.

[3] Härdle, Wolfgang; Horowitz, Joel L.; Kreiss, Jens-Peter (2001) : Bootstrap methods for time series, SFB 373 Discussion Paper, No. 2001,59, Humboldt University of Berlin, Interdisciplinary Research Project 373: Quantification and Simulation of Economic Processes, Berlin.

[4] G.Mélard, J.M.Pasteels *Automatic ARIMA modeling including interventions, using time series expert software* ISRO CP 210 (bldg NO room 2.O.9.300), Campus Plaine, Université Libre de Bruxelles, Bd du Triomphe, B-1050 Bruxelles, Belgium

[5] TY - JOUR AU - Luitel, Hari AU - Mahar, Gerry PY - 2015/09/25 T1 - *A Short Note on the Application of Chow Test of Structural Break in US GDP* VL - 8 DO - 10.5539/ibr.v8n10p112 JO - International Business Research

[6] Shapiro, S. S., and M. B. Wilk. *An Analysis of Variance Test for Normality (Complete Samples).* Biometrika, vol. 52, no. 3/4, 1965, pp. 591–611. JSTOR, www.jstor.org/stable/2333709.

[7] TY - JOUR AU - Dickey, D. AU - Fuller, Wayne PY - 1979/06/01 T1 - *Distribution of the Estimators for Autoregressive Time Series With a Unit Root* VL - 74 DO - 10.2307/2286348 JO - JASA. Journal of the American Statistical Association

7 Appendix: R-Script

7.1 Importing data-set and libraries

```
### IMPORTS AND LOADING DATA-SET
require('combinat')
require('tseries')
require('ggplot2')
require('reshape2')
require('outliers')
require('forecast')
require('dplyr')
require('boot')
require('MSwM')
require('strucchange')
require('zoo')
require('moments')
monthly_housing_data <- read.csv(file='housing_in_london_monthly_variables.csv')
View(monthly_housing_data)
```

7.2 Build weighted index

```
### BUILDING WEIGHTED INDEX WITH NEIGHBORHOODS OF LONDON
x <- monthly_housing_data[monthly_housing_data['borough_flag'] == 1, c('area')]
df <- monthly_housing_data[monthly_housing_data['area'] == x,
                           c('date', 'area', 'houses_sold', "average_price")]
df <- df %>% group_by(date) %>% filter (!is.na(houses_sold))



houses_by_month <- aggregate(houses_sold ~ date, df, as.integer)
houses_by_month <- matrix(houses_by_month)[2][[1]]
london_houses_sold <- monthly_housing_data[monthly_housing_data['area']
                                           == 'london', c('houses_sold')]

london_houses_sold <- london_houses_sold[!is.na(london_houses_sold)]
weight <- houses_by_month / london_houses_sold
w <- melt(weight)
w <- subset(w, select=-Var2)
w <- w[order(w$Var1),]
w <- subset(w, select=-Var1)
```

```
df <- df[order(df$date), ]
london_index <- df
london_index$weighted_average <- df$average_price * w

cols <- c('date', 'weighted_average')
london_index <- london_index[, cols]
london_index <- aggregate(london_index$weighted_average,
                          by=list(date=london_index$date), FUN=sum)

# set index and remove unnecessary columns
rownames(london_index) <- london_index$date
london_index <- subset(london_index, select= -date)
colnames(london_index) <- c('weighted_average')
# View(london_index)

### SECTION 2 PLOT
plot(london_index$weighted_average ~ as.Date(rownames(london_index)),
     xlab="Years", ylab="Average London House Price", type='l')
```

7.3 Autocorrelation Tests

```
### PERMUTATION TEST FOR TABLE 1 AND 2
### FOR TABLE 2 JUST CHANGE london_index$weighted_average for the stationary time
    series, in this case avg_price_diff
nr <- 10000
st <- numeric(nr)
st2 <- numeric(nr)
st3 <- numeric(nr)
st4 <- numeric(nr)
st5 <- numeric(nr)

sttrue <- acf(london_index$weighted_average, plot=FALSE)$acf[2]
sttrue2 <- acf(london_index$weighted_average, plot=FALSE)$acf[3]
sttrue3 <- acf(london_index$weighted_average, plot=FALSE)$acf[4]
sttrue4 <- acf(london_index$weighted_average, plot=FALSE)$acf[5]
sttrue5 <- acf(london_index$weighted_average, plot=FALSE)$acf[6]

n <- length(london_index$weighted_average)
for (i in 1:nr) {
  d <- sample(london_index$weighted_average,  n)
  st[i] <- acf(d, plot=FALSE)$acf[2]
  st2[i] <- acf(d, plot=FALSE)$acf[3]
  st3[i] <- acf(d, plot=FALSE)$acf[4]
  st4[i] <- acf(d, plot=FALSE)$acf[5]
  st5[i] <- acf(d, plot=FALSE)$acf[6]
}

length(st[st >= sttrue])/nr
length(st2[st2 >= sttrue2])/nr
length(st3[st3 >= sttrue3])/nr
length(st4[st4 >= sttrue4])/nr
length(st5[st5 >= sttrue5])/nr

### TRANSFORM TO TIME SERIES TO DECOMPOSE
### SECTION 3 FIGURE 3
london_ts <- ts(london_index$weighted_average, frequency = 12, start = c(1995, 1))
plot(decompose(london_ts))

### ACF PLOT SECTION 3 FIGURE 2
ac <- acf(london_index$weighted_average, plot=FALSE)
plot(ac, main="Autocorrelation of time series")
```

7.4 Making data stationary

```
# Stabilize variance:
# check whether we have heterocedasticity or not
# plot(rollapply(london_index$weighted_average, width=3, FUN=sd))
log_avg <- log(london_index$weighted_average)  # removal of heterocedasticity
# Box-Cox transformation can also work
# log_avg

# Stabilize mean:
avg_price_diff <- diff(log_avg)  # Linear tendency, then just one difference

# DICKEY-FULLER TEST FOR SECTION 3.1
dfuller_test <- adf.test(avg_price_diff)
dfuller_test
### FIGURE 4 SECTION 3.1
plot(avg_price_diff ~ as.Date(head(aux$date, length(avg_price_diff))),
     xlab="Years", ylab="Stationary Average London House Price", type='l')
```

7.5 Break-points and Chow Test

```
aux <- london_index
aux$date <- rownames(london_index)
nums <- seq(1, length(aux$date), 1)
# Chow test
sctest(aux$weighted_average ~ nums, type="Chow", point=10)

l <- length(london_index$weighted_average)
tt <- 1:(l-1)

# SECTION 3.2 TABLE
brk <- breakpoints(ts(london_index$weighted_average[2:299]) ~
                     london_index$weighted_average[1:298] + tt, h=55)
summary(brk)

### SECTION 3.2 PLOT
plot(london_index$weighted_average ~ as.Date(rownames(london_index)),
     xlab="Years", ylab="Average London House Price with breakpoint")
abline(v=c(as.Date(rownames(london_index)[171])), col="red", lwd=3, lty=2)
```

7.6 Forecast

```
### FOR ALL THE PLOTS IN SECTION 4. CALCULATES AUTOARIMA AND FORECAST
# After the acf and pacf plots, we can deduce a d=2 might work well for our
# dataset.

n <- length(tail(london_index$weighted_average, -171))
london_autoarima <- auto.arima(tail(london_index$weighted_average, -171),
                               ic='aic')  # d=1 based on acf and pacf

summary(london_autoarima)  # ARIMA(3, 1, 0)

# london_autoarima
plot(london_autoarima$residuals)

# Check residuals are stationary with Dickey fuller test:
dfuller_test_residuals <- adf.test(london_autoarima$residuals)
dfuller_test_residuals  # < 0.05, hence stationary, reject null-hypothesis

# Normality test of the residuals:
```

```
shapiro.test(london_autoarima$residuals)  # > 0.05, hence, normally distributed
                                          # accept Null-Hypothesis

# Forecast based on autoarima
london_pred <- forecast(london_autoarima)
# mean prediction
london_pred$mean

# prediction residuals
plot(london_pred$residuals)

plot(london_pred, main="ARIMA Forecast", ylab='weighted_avgs')

hist(london_autoarima$residuals, breaks=15)

plot(london_autoarima$x, col='red', main='Arima and London index series')
lines(fitted(london_autoarima), col='blue')

### FOR ERRORS TABLE
refit <- Arima(london_index$weighted_average, model=london_autoarima)
accuracy(refit)  # MPE of -0.1968881 looks good
```

7.7 Jackknife and Shapiro tests

```
jackknife_x <- function(x) {
    sk <- skewness(x)
    ku <- kurtosis(x)
    n <- length(x)

    thetask <- numeric(n)
    pseusk <- numeric(n)
    thetaku <- numeric(n)
    pseuku <- numeric(n)

    for (j in 1:n) {
      thetask[j] <- skewness(x[-j])
      pseusk[j] <- n*sk -(n-1)*thetask[j]

      thetaku[j] <- kurtosis(x[-j])
      pseuku[j] <- n*ku -(n-1)*thetaku[j]
    }
    values <- list("skewness" = mean(pseusk),
                   "se_skewness" = sd(pseusk)/sqrt(n),

                   "kurtosis" = mean(pseuku),
                   "se_kurtosis" = sd(pseuku)/sqrt(n))
    return(values)
}
x <- london_autoarima$residuals
jackknife_x(x)

shapiro.test(x)
```

7.8 Non parametric bootstrap for ARIMA

```
# mean of dataset: mean(london_index$weighted_average)
# arima residuals: london_autoarima$residuals
# arima model: Y_t = \beta_1Y_t-1 + ... + \beta_0Y_0 + \eps_t
# mean is called drift when d=1
# (1 + thetaB_1 + --- + theta_qB^q)eps_t
# X_i+1 = X_i + eps_i
# hat(y_t)  = yt-1 + drift + ar1*inc(y_t-1) + ar2*inc(yt-2) + ar3*inc(yt-3) +
    eps_t

x <- tail(london_index$weighted_average, -171)
log_x <- log(x)  # removal of heterocedasticity
                 # Box-Cox transformation can also work

# Stabilize mean:
diff_x <- diff(log_x)  # Linear tendency, then just one difference

dfuller_test <- adf.test(diff_x)

diff_x <- tail(london_index$weighted_average, -171)
fit <- Arima(diff_x, order=c(3, 1, 0), method="ML")
res <- fit$residuals
# res <- fit$resid
# fit$coef[1]: ar1
# fit$coef[2]: ar2
# fit$coef[3]: ar3
# fit$coef[4]: drift
```

```r
nb <- 1000
mu <- numeric(nb)
phi1 <- numeric(nb); phi2 <- numeric(nb); phi3 <- numeric(nb)
iters <- 30
xb <- numeric(iters)
print(xb)

for(j in 1:nb) {
  xb[1] <- diff_x[1];  xb[2] <- diff_x[2];
  xb[3] <- diff_x[3]; xb[4] <- diff_x[4]
  for (i in 5:iters) {
    r <- sample(res[5:iters], 1, replace=T)

    xb[i] <- xb[i-1] +
             fit$coef["ar1"]*(xb[i-1] - xb[i-2]) +
             fit$coef["ar2"]*(xb[i-2] - xb[i-3]) +
             fit$coef["ar3"]*(xb[i-3] - xb[i-4]) +
             r
    fit <- Arima(xb, order=c(3, 1, 0), method="ML")
    mu[j] <- mean(xb)
    phi1[j] <- fit$coef["ar1"]; phi2[j] <- fit$coef["ar2"];
    phi3[j] <- fit$coef["ar3"];
  }
}

### BOOTSTRAP PLOTS
quantile(mu,   c(0.025, 0.975))
hist(mu, breaks=20)
abline(v=mean(diff_x), col='red')
quantile(phi1, c(0.025, 0.975))
hist(phi1)
abline(v=london_autoarima$coef["ar1"], col='red')
quantile(phi2, c(0.025, 0.975))
hist(phi2)
abline(v=london_autoarima$coef["ar2"], col='red')
quantile(phi3, c(0.025, 0.975))
hist(phi3)
abline(v=london_autoarima$coef["ar3"], col='red')
london_autoarima$coef
```