

Statistical Analysis of Music Data

Written Report for
STAT 4799 Statistics Project

by

ZHAO DAWEI

Supervised by: Dr. Gilbert, C.S. Lui

Department of Statistics and Actuarial Science
University of Hong Kong

May, 2016

Abstraction

In this report, we mainly focus on the prediction of some future amplitude of a certain piece of music with the help of time series models and data mining methods. A total of 4 models were deployed, namely Self Threshold Autoregressive Model (SETAR Model), Continuous Autoregressive Model (CAR Model), Functional Time Series Model (FTSM) and Neural Network Time Series Forecasts (NNETAR Model). We first compare and find the best parameters within each model. Through this process, we get to learn the strength and weakness of models. Then we conduct parallel comparison to determine, for our dataset, which one works the best and try to analyze the reasons behind. Finally, we offer some suggestions for future research.

Contents

1.	Introduction.....	4
2.	Sound Analysis	4
2.1	Basic Features of Music.....	4
2.2	Digitalization	4
2.2.1	Sampling	4
2.2.2	Quantisation	5
2.3	Discrete Time Fourier Transform.....	5
3.	Problem Definition and Data specification	6
3.1	Problem Definition	6
3.2	Data Specification.....	6
4.	Model Training.....	7
4.1.1	Self Threshold Autoregressive Model (SETAR Model)	7
4.1.2	Results	7
4.1.3	Conclusion	8
4.2.1	Continuous Autoregressive Model (CAR Model)	9
4.2.2	Results	9
4.2.3	Conclusion.....	11
4.3.1	Functional Time Series Model (FTSM)	12
4.3.2	Results	12
4.3.3	Conclusion.....	14
4.4.1	Neural Network Time Series Forecasts (NNETAR Model)	14
4.4.2	Results	16
4.4.3	Conclusion.....	17
5.	Model Comparison	17
6.	Conclusion	19
7.	Acknowledgement.....	20
	Reference.....	21
	Appendix: result of model comparison.....	22
1.	SETAR Model.....	22
2.	CAR Model.....	23
3.	FTSM	23
4.	NNETAR	24

1. Introduction

When we are listening to music, our brain makes sense of music by constructing detailed models in real time. In fact, it turns out, the procedure of listening is an act of neural prediction. And with the help of music prediction, we are able to better evaluate some music theories to better understand the music language. In the following sections, I will start by introducing the basic background of sound analysis. And for each of the models, we train the data and discuss the advantages and disadvantages of the model. Finally, parallel comparisons were made to better elaborate on the characteristics of all the models.

2. Sound Analysis

2.1 Basic Features of Music

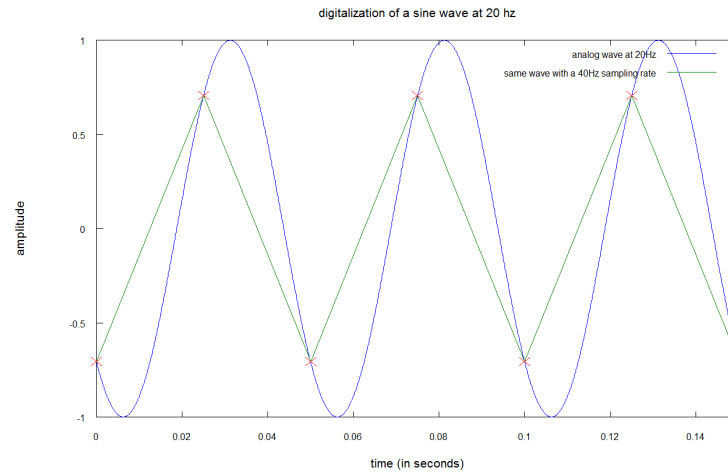
A sound is a vibration that propagates through air (or water) and can be decrypted by the ears. Characterized by its own frequency and amplitude, a sinusoidal wave is considered the most primitive model of sound also referred to as pure tone^{2,14}. Frequency is defined as the number of cycles per second and amplitude is the size of each cycle or the wavelength of wave. Pure tone doesn't naturally exist but every sound in the world is the sum of a multiple pure tones at different amplitudes. Based on the crude sinusoidal model, there are three features of music. According to basic physics law $speed = frequency * wavelength$, the wavelength together with the speed of the wave determine the frequency commonly referred to as pitch^{2,9}. The amplitude determines how loud a sound will be. Greater amplitude means the sound will be louder. Another important characteristic is timbre. The same note does not sound exactly the same if it's played by a flute, a drum, a violin or a human singer¹¹. The reason is that each instrument has its own timbre for a given note. For each instrument, the sound, with both its own fundamental frequency and various overtones, produced is a multitude of frequencies that sounds like a given pitch^{2,9}.

2.2 Digitalization

2.2.1 Sampling

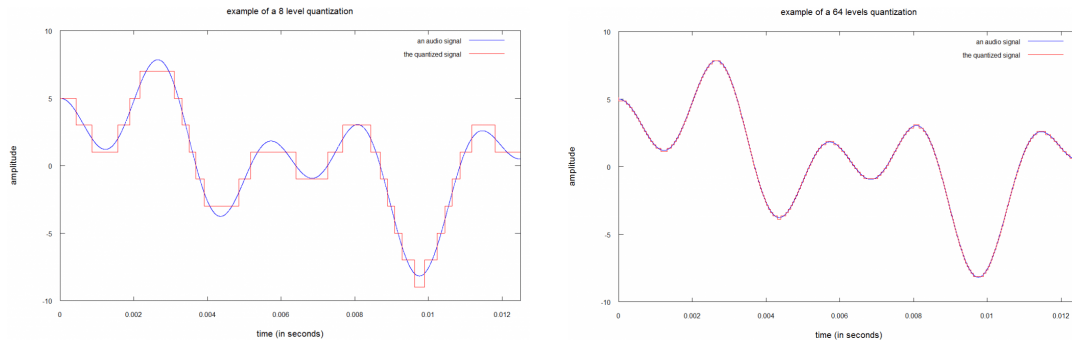
When artists produce music, vibrations and sounds are manufactured continuously. However, in most cases, the music is digitalized in order to be stored and played by electronic devices. In the digital world, you cannot afford to store an infinite amount of information and that's why we need analog signals where you cut music into small pieces. This problem is known as sampling. The standard unit of time in digital music is 44 100 units (or samples) per second which is named as sampling rate or sampling frequency^{11,14}. Humans can hear sounds from 20Hz to 20kHz^{11,14}. A theorem from Nyquist and Shannon states that if you want to

digitalize a signal from 0Hz to 20kHz you need at least 40,000 samples per second^{11,14}. The main idea is that a sine wave signal at a frequency F needs at least 2 points per cycle to be identified.



2.2.2 Quantisation

Another important parameter of digitization is the process of quantisation that consists in assigning a numerical value to each sample according to its amplitude. These numerical values are attributed according to a bit scale. The standard quantization is coded on 16 bits, which means 65536 levels. With a 16 bits quantization, the quantization noise is low enough for human ears^{2,9}.



2.3 Discrete Time Fourier Transform

With the help of above procedures, the digital sounds are already available in your computer. However, it is of great importance to figure out the frequency, which is one of the fundamental features, because many popular algorithms work with frequencies. Here the method of Discrete Time Fourier Transform is introduced.

$$X(n) = \sum_{k=0}^{N-1} x_k e^{-j\left(\frac{2\pi kn}{N}\right)}$$

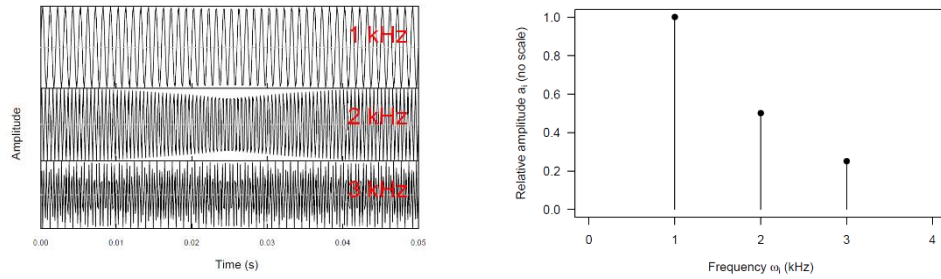
In this formula:

N is the size of the window: the number of samples that composed the signal.

$X(n)$ represents the n -th bin of frequencies.

x_k is k -th sample of the audio signal.

The DTFT applies to discrete signals and returns a discrete spectrum. Information related to both time and frequency are provided in the spectrum^{2,9}. We need to grab a small segment from the long audio signal to process. Such procedure can be viewed as multiply the original audio signal with an indicator function, and that signal is named as a rectangular window^{2,9}.



3. Problem Definition and Data specification

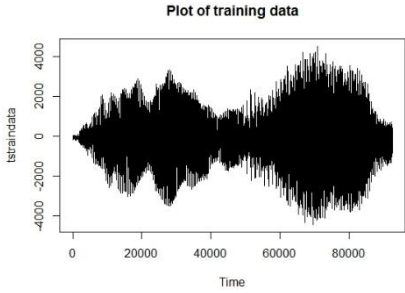
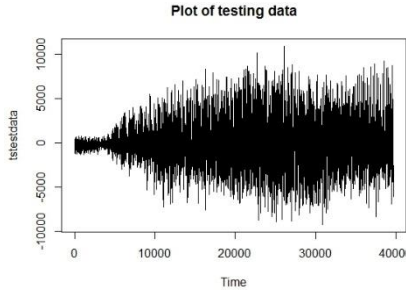
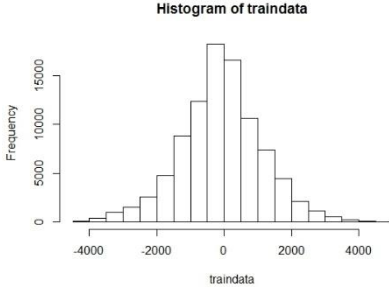
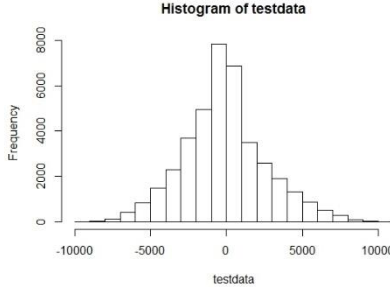
3.1 Problem Definition

In this study, I will be predicting some future amplitude of a certain piece of music with the help of time series models and data mining methods.

3.2 Data Specification

We draw a random piece of music that lasts for 60 seconds (randomly generated) from Violin Concerto No. 1 in D major Op 6 (Paganini) in WAV format. The total number of samples is 2,646,000 with a sample rate of 44.1k Hz¹¹. Simple summary statistics show that mean of the data is -174.5 and the median is -140. Since the sample size is more than big for some models, I selected 3.5% as training data where the sample size is 926,100 and 1.5% as testing data where the sample size is 396,858.

Time series plots and histograms of the training and testing data showed that they exhibit similar patterns and should be good for prediction.

	Training Data	Testing Data
Plot		
Histogram		
N	926,100	396,858

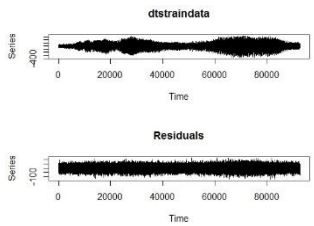
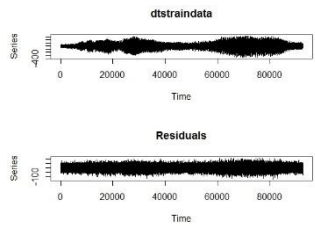
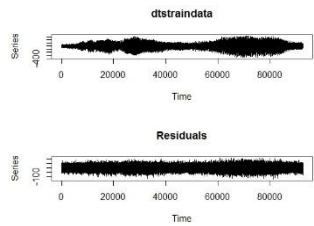
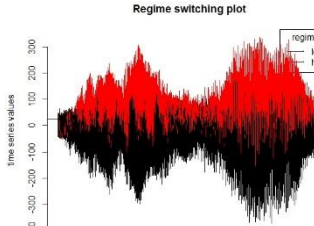
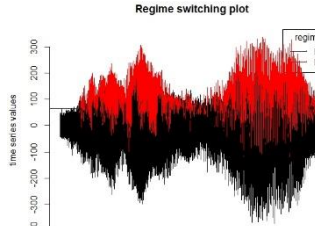
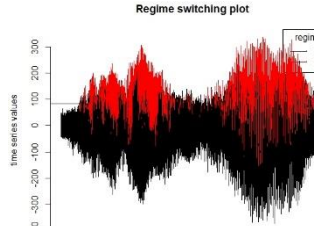
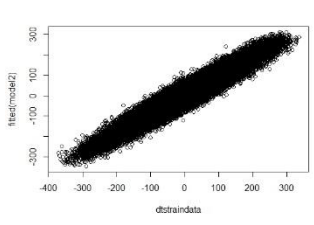
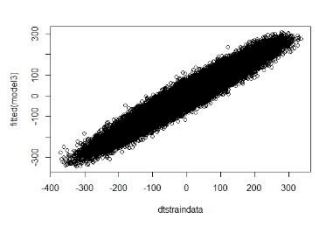
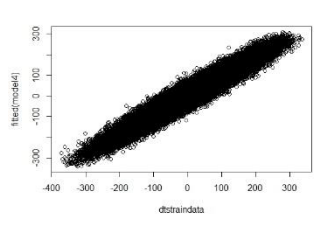
4. Model Training

4.1.1 Self Threshold Autoregressive Model (SETAR Model)

$$Y_t = \begin{cases} \phi_{1,0} + \phi_{1,1}Y_{t-1} + \dots + \phi_{1,p_1} + \sigma_1 e_t & \text{if } Y_{t-d} \leq r \\ \phi_{2,0} + \phi_{2,1}Y_{t-1} + \dots + \phi_{2,p_2} + \sigma_2 e_t & \text{if } Y_{t-d} > r \end{cases}$$

Where the ϕ' s are autoregressive parameters, σ' s are noise standard deviations, d is the number of lags, r is the threshold parameter, and $\{e_t\}$ is a sequence of independent and identically distributed random variables with zero mean and unit variance. Thus, if the lag d value is not greater than the threshold, the conditional distribution of Y_t is the same as that of an $AR(p_1)$ process with intercept $\phi_{1,0}$ and autoregressive coefficient $\phi_{1,1}, \dots, \phi_{1,p_1}$ and error variance σ_1^2 . On the other hand, when the lag d value exceeds the threshold r, the $AR(p_2)$ process is operational. In other words, this is a model which switches to different procedures based on the position of lag d value. And we use the conditional least squares (CLS) approach to estimate the parameters.^{1,8,10} This method achieves the estimated parameters by minimizing the predictive sum of squared errors or equivalently maximizing the log-likelihood function given the variances are constant.^{1,8,10}

4.1.2 Results

m=2, thDelay=1	m=3 thDelay=1	m=3 thDelay=2
		
		
		
residuals variance = 779.4	residuals variance = 732.4	residuals variance = 732.2
AIC = 616659	AIC = 610898	AIC = 610874
MAPE = 102.4%	MAPE = 99.02%	MAPE = 99.06%

We first observe from the residuals plot that we got constant variance residuals and we can see that in each one of the models, the fitted data vs original data plot they lie almost on the $y=x$ line. From model 1 to model 2, we observe a significant decrease in residuals variance, AIC, and MAPE. However, though we make the model more complicated from model 2 to 3, it does not seem to improve the overall performance. We can also prove this point in the regime switching plot where it is quite obvious that the third plot's performance is not as good as the second one. And it is very similar when we further increase the complexity of the model. So, the best model fit is model 2 where we have two AR(2) models on the two regimes.

4.1.3 Conclusion

Traditionally, in order to avoid the effect of 0 amplitude or very low amplitude points, it was naturally to directly cut those little pieces off and connect the rest of the pieces together. I do not think this method is good enough because those low or zero amplitude points are a coherent part of music. I intend to use this model to handle the 0 amplitude points in the music data. However, after doing basic statistical analysis of various pieces of music, I found that 0 amplitude usually accounts for less than 0.1% of the data. So the model might not be accurate if we only want to predict low amplitude points.

Generally the model performs well for the training part. However, when it comes to prediction, it might not be so accurate. Since we are only using 60 seconds of a piece music, where the periodical trend of the music is not fully displayed. Furthermore, the number of predictions we are making is too large leading to relatively high standard errors of the predictions thus affecting the accuracy.

4.2.1 Continuous Autoregressive Model (CAR Model)

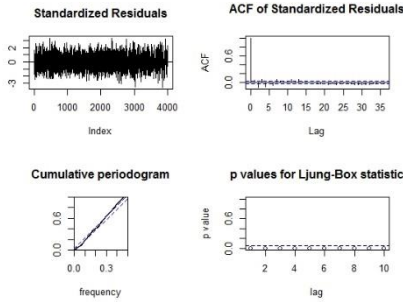
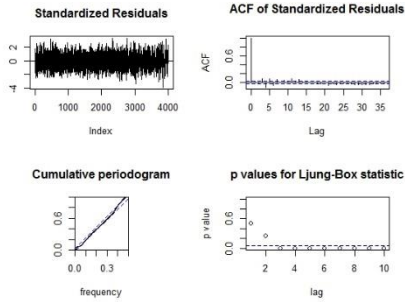
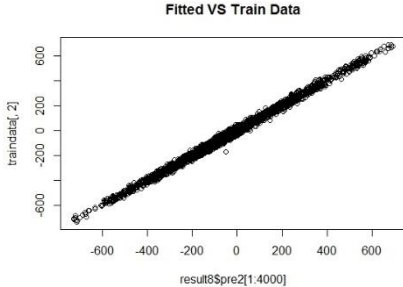
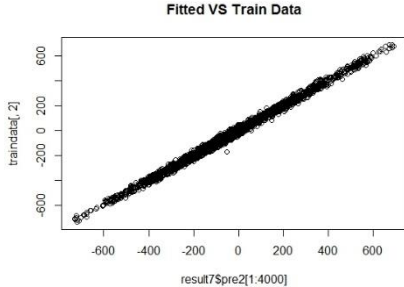
Suppose we have data x_k observed on time t_k for $k = 1, 2, \dots, n$. We assume noise-affected observations $x_k = y(t_k) + \eta_k$ and $y(t_k)$ follows a p -th order continuous time autoregressive process $Y(t)$ satisfying

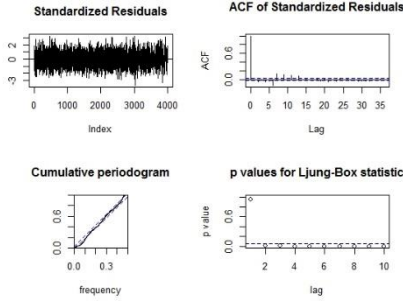
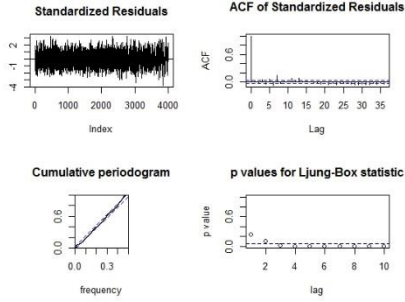
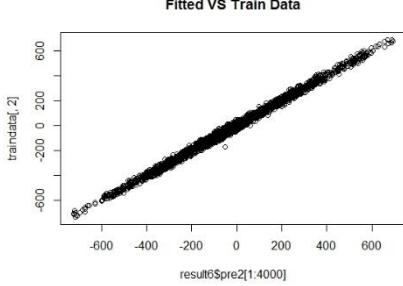
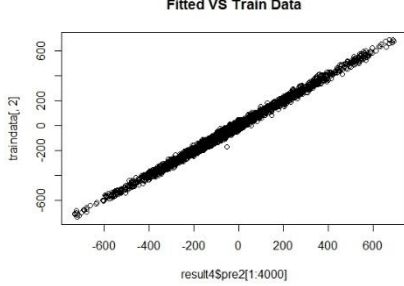
$$Y^{(p)}(t) + \alpha_1 Y^{(p-1)}(t) + \dots + \alpha_{p-1} Y^{(1)}(t) + \alpha_p Y(t) = \epsilon(t),$$

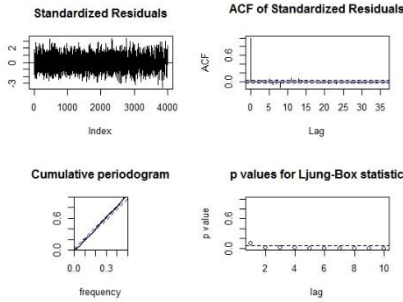
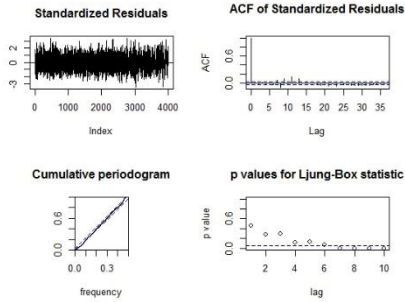
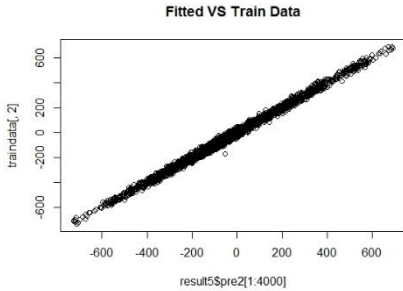
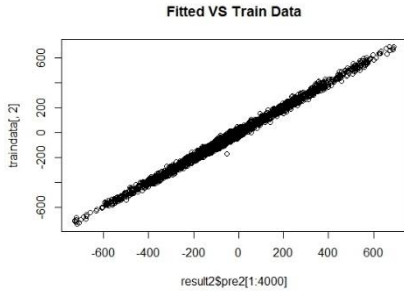
where $Y^{(i)}(t)$ is the i -th derivative of $Y(t)$ and $\epsilon(t)$ is the formal derivative of a Brownian process $B(t)$ with variance parameter $\sigma^2 = \text{var}\{B(t+1) - B(t)\}$. In addition, it will be assumed that η_k is a normally distributed random variable representing observational error, uncorrelated with $\epsilon(t)$, and $E(\eta_k) = 0$; $E(\eta_j \eta_k) = 0$, for $j \neq k$.¹⁵ Kalman filtering is applied to estimate the parameters of the above model¹⁵.

4.2.2. Results

Here I am comparing 6 models in total. We start from order equals 5 because for smaller orders, which means the model is less sophisticated, it would be less likely to fit well. After some trials, it was indeed the case. So, starting from 5, I increase the order of CAR model by one each time. We observe that at first the sigma square is incredibly large. It begins to decrease as the model becomes more complicated as expected and reached the lowest at order equals 7. The situation turns worse when we continue to increase the order. So, model 3 is selected as the best choice. As can be seen, the residuals are good enough for all the models. There seems to be a similar pattern for the “Fitted VS Train Data” graph with sigma square. What is interesting to note is that, however, the Ljung-Box statistics is better for model 6 rather than model 3. I think the reason is, though the more complex model can better reduce the serial correlation, it fails to identify the overall trending due to overfitting.

Model number	Model 1	Model 2
Model Specification	Order = 5	Order = 6
Summary Result		
Fitted VS Train Data		
Sigma Square	5051296	20752

Model number	Model 3	Model 4
Model Specification	Order = 7	Order = 8
Summary Result		
Fitted VS Train Data		
Sigma Square	2638	17301

Model number	Model 5	Model 6
Model Specification	Order = 9	Order = 10
Summary Result		
Fitted VS Train Data		
Sigma Square	62348	230064

4.2.3 Conclusion

Since CAR model consider time series as a kind of continuous process, it performs good enough for the closely observed samples. However, the relative computational burden involved is high. Due to the complexity, there are more limitations, like non-stationary structures and failure of convergence, than simpler models. The algorithm I implemented supports only 5000 samples and when smoothing functions are added, we must limit the control parameters used in optimization procedure to get a final result.

The prediction process is also hard for this model. Due to the limited number of training data, it would be normal to fail to capture long term periodical trend. Music data, especially our data, that have about three million sample points for only 60 seconds, takes long periods to display periodical patterns. And though the results or prediction is very accurate at the begging part, it cannot detect some long term variations.

4.3.1 Functional Time Series Model (FTSM)

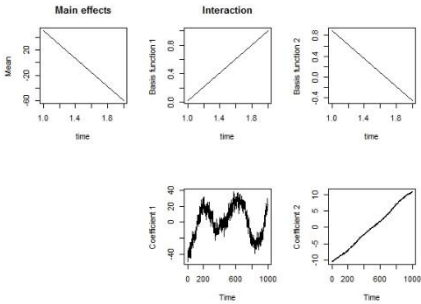
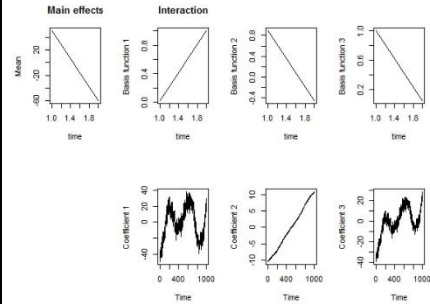
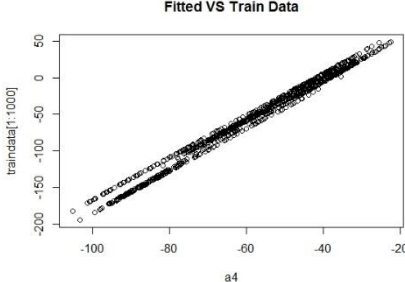
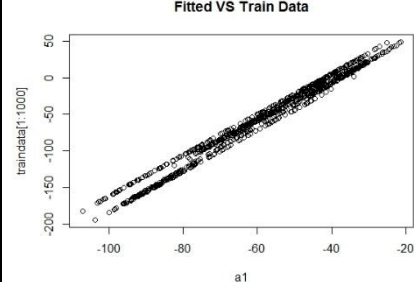
Let $\{Z_\omega, \omega \in [1, N]\}$ be a seasonal univariate time series which has been observed at N equispaced times. When the seasonal pattern is strong, one way to model the time series nonparametrically is to use ideas from functional data analysis^{3,4,5}. We divide the observed time series into n trajectories, and then consider each trajectory of length p as a curve rather than as p distinct points. The functional time series is then given by

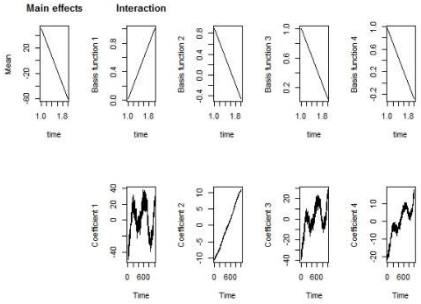
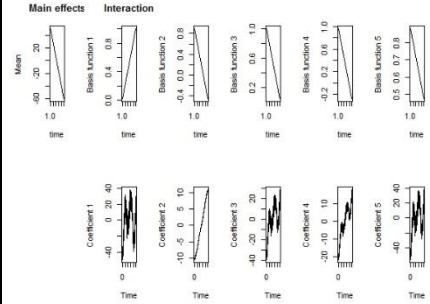
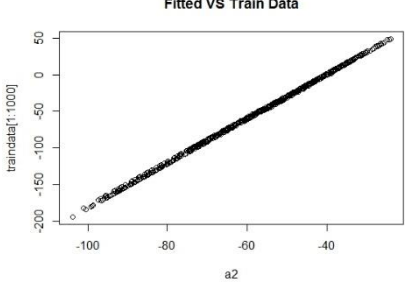
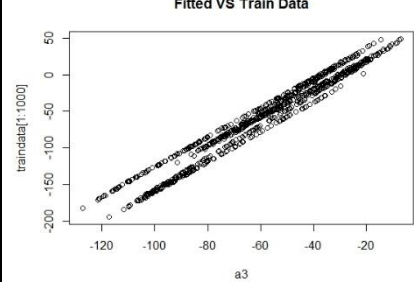
$$y_t(x) = \{Z_\omega, \omega \in (p(t-1), pt]\}, \quad t = 1, \dots, n.$$

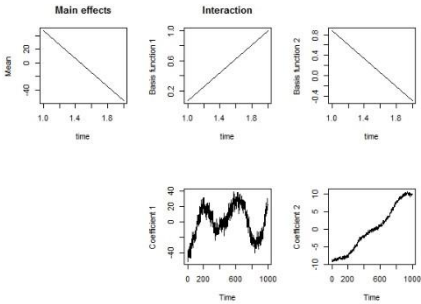
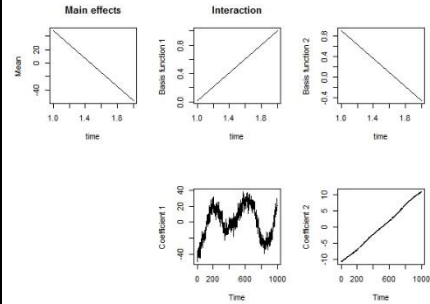
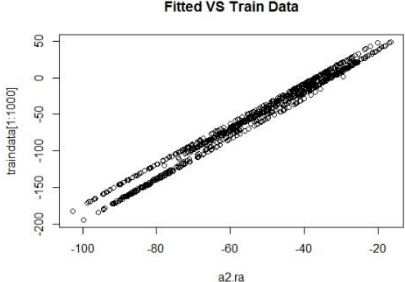
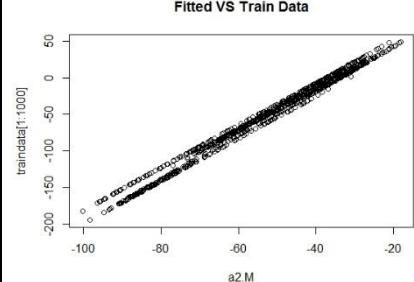
The usual problem of interest is to forecast $y_{n+h}(x)$, the data in year $n+h$, from the history data $\{y_1(x), \dots, y_n(x)\}$. When $N = np$, all trajectories are complete with some straightforward available methods⁴. We are mainly using functional principal component analysis (FPCA) because FPCA allows a decomposition of the observed data into a number of functional principal components and their uncorrelated principal component scores^{7,12}.

4.3.2 Results

Similar with previous methods, we observe that the MSE and ISE first decrease and then increase when the order of the model grows. The best performance was achieved at order equals 4. The “Fitted VS Train Data” plot also supports that we get the best model when order equals 4. Here we also deploy two other algorithm for FTSM model. We can see that among “classical”, “rapca” and “M”, “rapca” has the best performance. We can observe the interactive terms and their coefficients from the PC graph.

Model number		Model 1		Model 2	
Model Specification		Order = 2, method = classical		Order = 3, method = classical	
Summary Result					
Fitted VS Train Data					
MSE	ISE	1015	595	909	422

Model number		Model 3		Model 4	
Model Specification		Order = 4, method = classical		Order = 5, method = classical	
Summary Result					
Fitted VS Train Data					
MSE	ISE	890	390	944	330

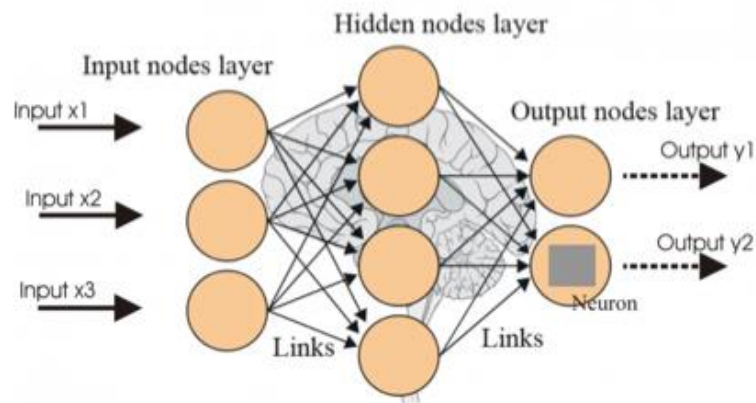
Model number		Model 5		Model 6	
Model Specification		Order = 2, method = rapca		Order = 2, method = M	
Summary Result					
Fitted VS Train Data					
MSE	ISE	1010	580	1032	603

4.3.3 Conclusion

Similar with the CAR model, though this model regards a smaller time series as continuous, it might not be able to capture future variations of music. Though it is not so capable of predicting a long time ahead, it gains the highest accuracy up to now because varies smoothing and weighting functions were added to improve the performance. And since this method generates more precise small period predictions, it is highly computational expensive.

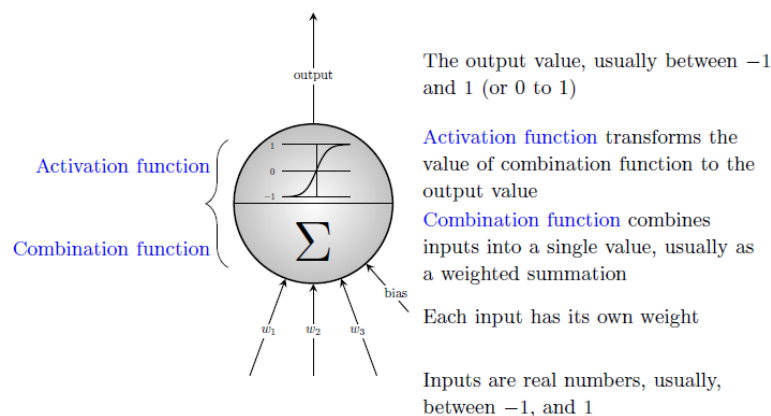
4.4.1 Neural Network Time Series Forecasts (NNETAR Model)

Inspired by the working manner of biological nervous system, Artificial Neural Network (ANN) is devised for processing information. Neural networks are typically organized in layers, which are composed of a number of interconnected 'nodes'. The 'input layer' read in data and communicate with the 'hidden layer', which works like a black box to inspect hidden association. The hidden layer then links to the 'output layer' and generates output results by some decision rules. The connection arrangements are decided by some weight functions. Such multiple layer networks are sometimes referred to as MLPs.



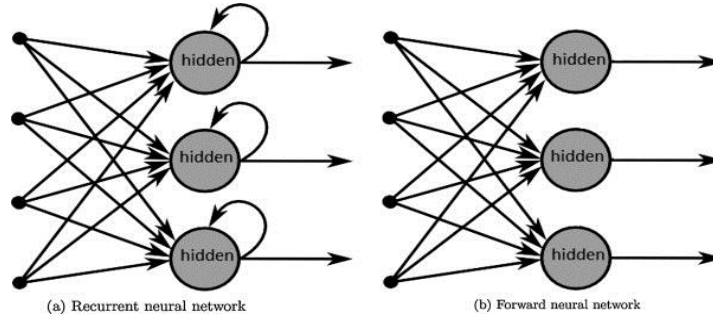
The ANN works in the following way. It takes several input as the input layer. To determine the output, we test if the weighted sum of inputs is greater than some threshold value. By varying the weights and the threshold, we can get different models of decision making and it should seem plausible that a complex network of perceptrons could make quite subtle decisions. However, we need to train our data to ensure that a small change in the weights of any single perceptron in the network will not cause the output of that perceptron to completely flip. In order to train a neural network to perform some task, we want to minimize the error between the calculated value and the actual value through attuning the weight functions. The system must calculate how the error changes as each weight is increased or decreased slightly. During the training period, the backpropagation algorithm is the most widely used method for determining the error derivative of the weights.

Artificial Neuron



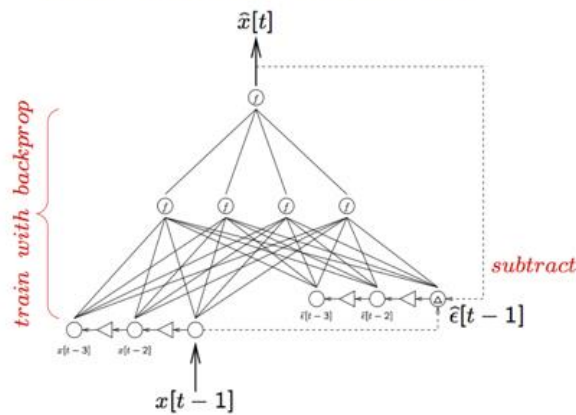
When it's operating normally (after being trained), patterns of information are fed into the network via the input units, which trigger the layers of hidden units, and these in turn arrive at the output units. This common design is known as a feed-

forward network. Another common design is recurrent networks which is more dynamic and powerful yet complicated compared with feed-forward approach. Recurrent networks allow values to travel both forward and back by introducing loops in the network.



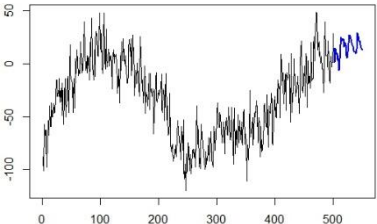
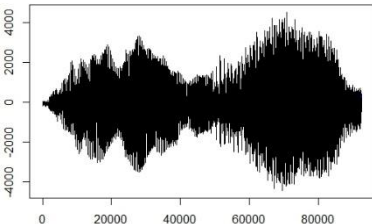
In our model, a feed-forward neural network is fitted with lagged values of x as inputs and a single hidden layer with size nodes. A total of *repeats* network are fitted, each with random starting weights. These are then averaged when computing forecasts. This is analogous to an AR(p) model but with nonlinear functions^{6,13}.

Nonlinear ARMA[p, q] Models



4.4.2 Results

Model number	Model 1	Model 2
Model Specification	NNAR(19,10) 500 data	NNAR(5,3) 92610 data

Summary Result		<div><p>Forecasts from NNAR(19,10)</p></div>		<div><p>Forecasts from NNAR(5,3)</p></div>	
RMSE	MAE	5	4	24	19

Since it needs a lot of computational power to train 100 thousand data points, we first used 500 data as a trial and it already reached an average of 20 networks, each of which is a 19-10-1 network with 211 weights options^{6,13}. By the same ratio, there might have been 3420 hidden unit in only one single layer. Due to limited computing power, I only tried a NNAR(5,3)^{6,13}. Furthermore, it would be not plausible to train all those hidden units with only one hidden layer. As can be observed from table, the RMSE figure is not large at all.

4.4.3 Conclusion

Neural Network methods can overcome the problems of the models mentioned above because it is capable of capturing long term alterations. However, the problem is that, though our model can be considered as one of the most basic ANN models, it still takes high time and space cost to train the data for the results. Of course the results will be far more improved if we were able to carry our MLP or recurrent networks to train the data. We still need to be careful about the trade-off between correctness and time, space costs.

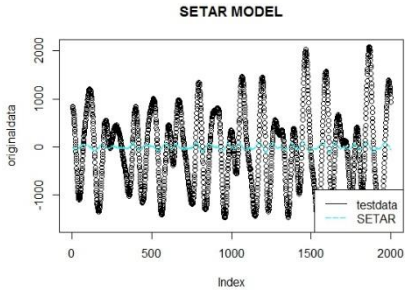
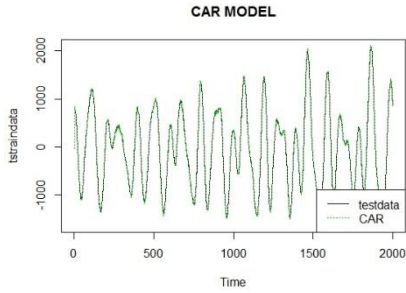
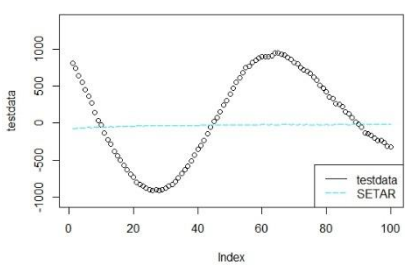
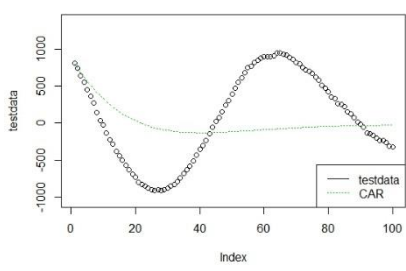
5. Model Comparison

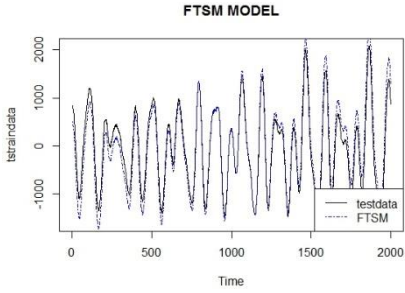
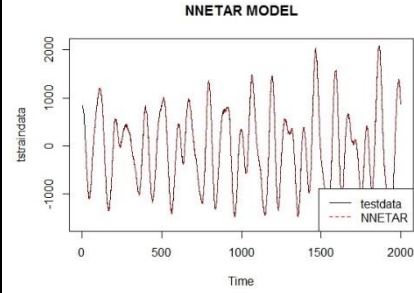
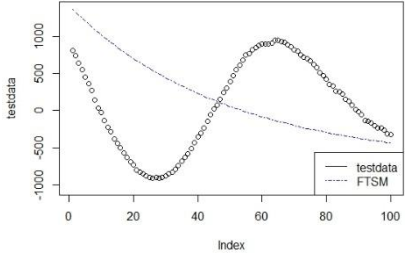
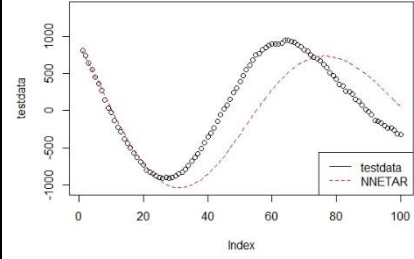
Due to some limitation of the algorithms of the models we have chosen, we used 2000 samples as training data and 100 as testing data. And achieved the following results.

We can see that the neural network model performs the best in the sense that it has the lowest MSE in fitting. However, we observe that CAR model and FTSM model are also doing quite a good job, because the fitted result curve is good enough. The only problem is that the SETAR model's accuracy is really low. The reason is that it has a large confidence interval with substantial errors, that's why we observe the range of fitted SETAR model is always around -200 to 200.

As for the exactness in prediction, according to the plots, neural network is the best choice for the reason that it catches the additional variations after 30 points. Though CAR and FTSM models successfully predict the trend among the first 30 points, their results in the long run is not good enough which corresponds perfectly with the analysis we made in the model section above. All the specification of the models can be found in the appendix.

Hence, traditional time series methods are sophisticated and complicated in the sense that they are very well studied and we can apply various smoothing curves or weights to get better results. The problem is that even if they captures the periodical trend, it is still not so meaningful in terms of long time prediction. Their advantage lies in short term predictions, like forecasting the stock price of tomorrow. This makes sense in our daily life, as no one will possibly believe the weather forecast of a year later. Instead, people rather trust in the report about the coming few hours. Data mining techniques counters the disadvantages of traditional time series perfectly, because they are able to train on a huge amount of data, given enough time and space. And it is apparent that the more data you invested in training, the more precise your result will be. The recent success of AlphaGo is a typical example. However, time and space invested into training the model might be more than costly. We need to balance the equation of exactness and time cost.

Model number	SETAR MODEL	CAR MODEL
Model Specification	m=3, thDelay = 2	Order = 2
Fitted Result		
Prediction Result		
MSE in fitting	524	1118

Model number	FTSM	NNTAR
Model Specification	Order = 5	NNAR(19,10)
Fitted Result		
Prediction Result		
MSE in fitting	80153	345

6. Conclusion

After training music data with the above four models, it is quite clear to notice the huge difference between traditional time series methods and modern data mining skills. Based on sophisticated theories and with rich literature, models of traditional time series can be very precise in both fitting and prediction, keeping a relatively low computational cost at the same time. However, these models might no longer be valid if we want a huge number of predictions. On the other hand, data mining techniques can forecasts larger periods provided enough training data. The biggest drawback of data mining is, in most cases, it takes much time, space and computational power.

Lots of aspects are open to further research. One of the problems that catch the most of my attention is that how we can reduce the sample size of music data while keeping the music still identifiable. To predict a single second of music, we need to calculate 44,100 values given a 44.1 kHz sample rate. Is it still meaningful to predict so many data at the same time? Another approach of music prediction can also be taken into account. It might be interesting if we could calculate the fundamental frequencies of the coming notes, and transform them back to amplitudes.

7. Acknowledgement

The author would like to thank Dr. Gilbert for his valuable suggestions and comments about the project.

Reference

1. Antonio, Fabio Di Narzo, Jose Luis Aznarte and Matthieu Stigler (2009). tsDyn: Time series analysis based on dynamical systems theory
2. Eronen, Antti. "Automatic musical instrument recognition." Mémoire de DEA, Tempere University of Technology (2001): 178.
3. Grunwald, Gary K., and Richard H. Jones. "Markov models for time series with mixed distribution." *Environmetrics* 11.3 (2000): 327-339.
4. Han Lin Shang and Rob J Hyndman (2016). rainbow: Rainbow Plots, Bagplots and Boxplots for Functional Data. R package version 3.4. <https://CRAN.R-project.org/package=rainbow>
5. Hyndman, Rob J., and Gary K. Grunwald. "Applications: Generalized Additive Modelling of Mixed Distribution Markov Models with Application to Melbourne's Rainfall." *Australian & New Zealand Journal of Statistics* 42.2 (2000): 145-158.
6. Hyndman RJ and Khandakar Y (2008). "Automatic time series forecasting: the forecast package for R." *_Journal of Statistical Software_*, *26*(3), pp. 1-22. <URL:<http://www.jstatsoft.org/article/view/v027i03>>.
7. Hyndman RJ and Shang HL (2016). *_ftsa: Functional Time Series Analysis_*. R package version 4.7, <URL: <http://cran.r-project.org/package=ftsa>>.
8. Jonathan, D. C., and Chan Kung-Sik. "Time Series Analysis With Applications in R." (2008).
9. Li, Tao, Mitsunori Ogihara, and George Tzanetakis, eds. *Music data mining*. CRC Press, 2011.
10. Matthieu Stigler (2010). tsDyn: Threshold cointegration: overview and implementation in R
11. Uwe Ligges, Sebastian Krey, Olaf Mersmann, and Sarah Schnackenberg (2013). tuneR: Analysis of music. URL: <http://r-forge.r-project.org/projects/tuner/>.
12. Shang HL (2013). "ftsa: An R Package for Analyzing Functional Time Series." *_The R Journal_*, *5*(1), pp. 64-72. <URL: <http://journal.r-project.org/archive/2013-1/shang.pdf>>.
13. Shang, Han Lin, and Rob J. Hyndman. "Nonparametric time series forecasting with dynamic updating." *Mathematics and Computers in Simulation* 81.7 (2011): 1310-1324.
14. Weihs, Claus, Uwe Ligges, and Katrin Sommer. "Analysis of music time series." *Compstat 2006-Proceedings in Computational Statistics*. Physica-Verlag HD, 2006. 147-159.
15. Zhu Wang (2013). cts: An R Package for Continuous Time Autoregressive Models via Kalman Filter. *Journal of Statistical Software*, 53(5), 1-19. URL: <http://www.jstatsoft.org/v53/i05/>.

Appendix: result of model comparison

1. SETAR Model

```
> summary(result.setar3)
```

Non linear autoregressive model

SETAR model (2 regimes)

Coefficients:

Low regime:

const.L	phiL.1	phiL.2	phiL.3
-1.6109943	0.2005245	0.2706600	0.4511048

High regime:

const.H	phiH.1	phiH.2	phiH.3
2.2483512	0.2370523	0.2846178	0.4106119

Threshold:

-Variable: $Z(t) = + (0) X(t) + (0) X(t-1) + (1) X(t-2)$

-Value: 16

Proportion of points in low regime: 62.12% High regime: 37.88%

Residuals:

Min	1Q	Median	3Q	Max
-78.933232	-14.589151	0.052356	14.723765	75.346879

Fit:

residuals variance = 523.9, AIC = 12534, MAPE = 105.5%

Coefficient(s):

	Estimate	Std. Error	t value	Pr(> t)
const.L	-1.610994	0.953354	-1.6898	0.09122 .
phiL.1	0.200524	0.025466	7.8743	5.584e-15 ***
phiL.2	0.270660	0.024971	10.8391	< 2.2e-16 ***
phiL.3	0.451105	0.028752	15.6896	< 2.2e-16 ***
const.H	2.248351	1.910927	1.1766	0.23951
phiH.1	0.237052	0.031859	7.4406	1.482e-13 ***
phiH.2	0.284618	0.031277	9.0998	< 2.2e-16 ***
phiH.3	0.410612	0.039635	10.3598	< 2.2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Threshold

Variable: $Z(t) = + (0) X(t) + (0) X(t-1) + (1) X(t-2)$

Value: 16

2. CAR Model

```
> summary(result.car2)
```

Call:

```
car(x = btraindata, order = 2, ctrl = car_control(n.ahead = 100))
```

Order of model = 2, $\sigma^2 = 537$

Estimated coefficients (standard errors):

	phi_1	phi_2
coef	-1.872	0.879
S.E.	0.012	0.012

Estimated mean (standard error):

[1]	-30.505
[1]	123.546

3. FTSM

```
> summary(result.ftsm6)
```

Functional time series model

Call: `ftsm(y = ftest, order = 5, method = "classical", weight = FALSE)`

Main effects: Mean

Order of interaction: 5

y: amplitude

x: time

Averages across x:

ME	MSE	MPE	MAPE
0.00	80153.31	NaN	Inf

Averages across time:

IE	ISE	IPE	IAPE
0.00000	30322.20929	0.88919	1.44359

4. NNETAR

```
> result.nn
```

```
Series: tstraindata
```

```
Model:  NNAR(19,10)
```

```
Call:    nnetar(x = tstraindata)
```

Average of 20 networks, each of which is
a 19-10-1 network with 211 weights
options were - linear output units

σ^2 estimated as 345