FORECASTING TIME SERIES WITH NEURAL NETS

by

Yoshio Kajitani

Graduate Program
in
Statistics

Submitted in partial fulfillment
of the requirements for the degree of
Master of Science

Faculty of Graduate Studies
The University of Western Ontario
London, Ontario
April 1999

THE UNIVERSITY OF WESTERN ONTARIO
FACULTY OF GRADUATE STUDIES

CERTIFICATE OF EXAMINATION

Chief Advisor

Examining Board

Advisory Committee

The thesis by
Yoshio Kajitani

entitled
FORECASTING TIME SERIES WITH NEURAL NETS

is accepted in partial fulfillment of the
requirements for the degree of
Master of Science

Date

Chair of Examining Board

ABSTRACT

FFNN (Feed-Forward Neural Nets) are one of the most widely used neural nets. In this thesis the FFNN architecture is examined and compared with statistical time series models for a variety of time series prediction problems. FFNN do not assume any probability models, while statistical models are based on the probability model. Therefore, if the goal of the modelling is rigorous quantification of uncertainty, statistical models are more suitable. However if the goal is merely prediction, we demonstrate that neural nets have a lot to offer. Widely different parameter settings in the neural net approach often lead to models which make virtually the same predictions. Neural net offer a more flexible approach to model building which is especially helpful in nonlinear and nonGaussian situations. In this thesis, the performance by NN models and statistical models for prediction is examined by using visualization techniques and statistical tests.

Keywords: Feed-Forward Neural Nets, Linear and nonlinear time series models, Forecasting, Nonlinear time series, Visualization

Dedicated to my family

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to my supervisor Dr. A. Ian McLeod for his great advice, encouragement, support and friendship.

I wish to thank Dr. Okada in Kyoto University for recommending me to study in this wonderful place.

TABLE OF CONTENTS

ARIMA model for such data.

Neural networks are also being applied in such emerging fields of study as: *Data Mining*, *Knowledge Discovery in Databases*, and *Machine Learning*.

Many excellent textbooks on the subject of neural networks are available, such as Kasabov (1998), Menrotra, Mohan & Ranka (1997), Freeman (1994) and Hertz, Krogh & Palmer (1991). Introductions to neural computing in the statistical literature have been given by Murtagn, (1999), Faraway & Chatfield (1998), Venables & Ripley (1997), Stern (1996), Warner & Misra (1996), and Chen & Titterington (1994). The theory and application of neural networks have also been further advanced by statisticians including De Veux et al. (1998), Ripley (1996) and Park et al. (1992).

The history of the development of neural computation is the subject of the books by Anderson and Rosenfeld (1998) and Johnson & Brown (1988).

Neural nets have been featured in the mass-circulation popular magazines as such as *Maclean's* (Kurzweil, 1999). Dyson (1997) provides an entertaining and speculative look at the future of neural computation and its impact on the *World-Wide-Web*. The book edited by Graubard (1990) contains several essays on the impact of neural nets in the field of *Artificial Intelligence*.

Software for fitting neural networks is widely available. In our research we have found the Splus functions developed by Ripley (1999) most useful. Ripley's functions were adapted by Faraway and Chatfield (1998) for the time series prediction problem and we have implemented their Splus functions. We also examined the SPSS (1998) package, *Neural Connection*. This package offers a visual modeling approach to the development of a neural network architecture and provides many more options and capabilities than the neural net functions based on Ripley (1999). However in our work we found no advantage in *Neural Connection* over Ripley's Splus functions.

1.1 Feed-Forward Neural Nets (FFNN)

FFNN are also often called *Multilayer Perceptrons* or *Backpropagation Networks*. In this thesis the FFNN architecture is examined for a variety of time series prediction

problems. The FFNN architecture is perhaps the most widely used neural net and it is in fact the only architecture supported by Ripley's software. Ripley (1996) gives a thorough treatment of the use of FFNN for pattern recognition.

The FFNN takes inputs, $x_i, i = 0, \ldots p$ and produces an output y. We take $x_0 = 1$. This x_0 is sometimes called the bias or dummy input node. In some applications, multiple outputs could be produced instead of just one output. The weighted linear sum of the inputs is fed forward to each of q hidden nodes where it is transformed using a sigmoid-like function, typically the logistic function,

$$\phi_h(z) = \frac{1}{1 + e^{-z}}.$$

Another popular sigmoid function is the hyperbolic tangent (tanh). Figure **??** compares these functions. Both functions are equally as good but the logistic is more tractable and is hence more widely used.

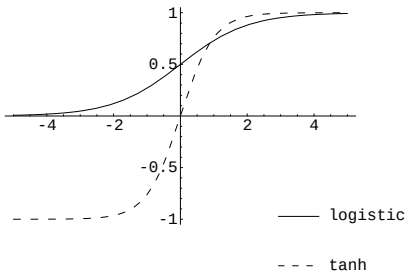


Figure 1: *The logistic,* $1/(1 + e^{-z})$*, and hyperbolic tangent,* $\tanh(z)$*, activation functions.*

The weighted linear sum of the hidden nodes is then fed to the output node where it is again transformed by a function $\phi_o(z)$. Popular forms for $\phi_o(z)$ include the threshold step function for classification problems as well as the logistic and linear function. For time series prediction the linear function, $\phi_o(z) = z$ is used. The weights are called the *connection strength* for the nodes and $\phi_h(z)$ and $\phi_o(z)$ are the node *activation functions*.

Figure **??** shows the structure of a FFNN with inputs $x(t-1)$ and $x(t-2)$ and with two hidden nodes. The output, $y(t)$, could be the predicted value of $x(t)$ or an

estimate of some other quantity. Given the number of hidden nodes, the output $y(t)$ only depends on the weights using in the linear sums.

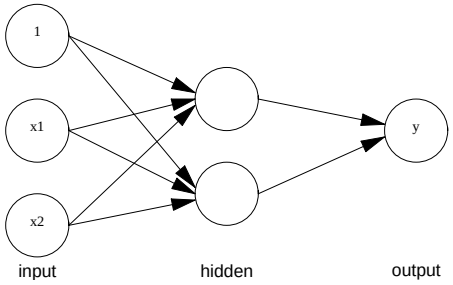


Figure 2: *FFNN diagram for time series prediction problem using the last two observations* $x1 = x(t-1)$ *and* $x2 = x(t-2)$ *and two hidden nodes. Depending on the weights,* y *could be an* ℓ*-step-ahead forecast.*

In general, given p inputs, $x_i, i = 0, \ldots p$ and q hidden nodes, the input to the j-th hidden node is

$$V_j = \sum_{i=0}^{p} w_{j,i} x_i.$$

The j-th hidden node transforms this to obtain the node output $\phi_h(V_j)$. The weighted sum of these hidden node outputs,

$$U = \sum_{j=1}^{p} W_j \ \phi_h(V_j),$$

is received at the output node and is transformed to obtain the output $y = \phi_o(U)$. The entire process can be summarized in a single equation,

$$y = \phi_o(\sum_{j=1}^{q} W_j \ \phi_h(\sum_{i=0}^{p} w_{i,j} x_i)).$$

The capability of this network is determined by the weights $w_{i,j}; i = 0, \ldots, p; j = 1, \ldots, q$ and $W_j, j = 1, \ldots, q$. These weights comprise $(p+1)q + q$ parameters which must be calibrated from available data. The number of hidden nodes, q, determines the architecture. The larger q is, the larger the number of connection weights which must be calibrated from data. On the other hand, with more hidden nodes the FFNN is better able to approximate an arbitrary continuous function. It

has been shown that for a continuous function of the input, $f(x_1, \ldots, x_p)$ and for any small $\epsilon > 0$ we can make the error uniformly small, that is, $|y - f(x_1, \ldots, x_p)| < \epsilon$ uniformly for all $(x_1, \ldots, x_p) \in \mathcal{R}^p$ (Mehrotra, Monan & Ranka, 1997, §3.5.3). Discontinuous functions can also be approximated by adding a second hidden layer to the architecture.

In time series modeling using the FFNN we shall for brevity denote the model by NN($\mathcal{L}; h$), where $\mathcal{L}$ is the set of lags and h is the number of hidden nodes. For example, the FFNN in Figure **??** would be denoted by NN$(1, 2; 2)$.

1.1.1 Calibration, Training and Backpropagation

Given data comprised of N input sequences $x_1^{(\mu)}, \ldots, x_p^{(\mu)}; \mu = 1, \ldots, N$ with corresponding target values $Y^{(\mu)}; \mu = 1, \ldots, N$ and after specifying q the number of hidden nodes, the problem is to calibrate or estimate the weights. In neural net terminology this calibration or estimation is called *training*.

Typically the data is divided into two portions, $N = N_1 + N_2$ where N_1 of the data are used for training and N_2 are used as test data to check the out-of-sample predictions. In neural net terminology, the out-of-sample predictions test how well FFNN *generalizes*. A common problem in FFNN design is choosing too many hidden nodes so the FFNN fits the training portion extremely well but it fails to generalize. When this occurs, the FFNN model is sometimes said to be overtrained.

The weights are chosen to minimize some *energy* function and the usual choice is,

$$E = \sum_{\mu=1}^{N_1} (y^{(\mu)} - Y^{(\mu)})^2,$$

where

$$y^{(\mu)} = \phi_o(\sum_{j=1}^{q} W_j \phi_h(\sum_{i=0}^{p} w_{i,j} x_i^{(\mu)})).$$

A widely used method in FFNN for minimizing E is the gradient descent method which is known as backpropagation. The term backpropagation arises because the gradient descent computations can be implemented very efficiently in terms of propagating the errors backwards through the network (Mehrotra, Monan & Ranka, 1997,

§3.3). However the convergence of the backpropagation algorithm can be very slow and require a lot of computer time. Various improvements to the basic gradient descent optimization algorithm have been suggested.

It should also be noted that for FFNN, E typically has many local minima and so the global minimum is nearly impossible to determine in many cases. However, in the spirit of the analogy with human learning, we may be content to halt the training when the performance is good enough. Worrying too much about the optimization algorithm used is probably not going to lead to any real improvement in the generalizability of the FFNN.

One obvious remedy to the problem of overtraining is to use fewer hidden nodes in the architecture. However in practice another approach using a validation sample is more convenient if the data are plentiful. In this approach the data are divided into three portions $N = N_1 + N_2 + N_3$ corresponding respectively to the training, validation and test data. Backpropagation is used iteratively and the performance of the FFNN is monitored on the validation sample. When performance of the validation sample starts to deteriorate, training is halted.

1.1.2 Other Applications of FFNN

The FFNN architecture has been widely used. As briefly indicated above, in the statistics arena, FFNN methods proved useful in problems handled by traditional methods for regression, classification, discrimination and time series prediction.

We highlight a few of the interesting other applications which are discussed in more detail in Hertz, Krogh and Palmer (1991, §6.3):

- Machine speech. *NETtalk* a FFNN that was taught to pronounce 1024 words using 80 hidden nodes.

- Hyphenation. Complicated hyphenation algorithms were developed for English. No algorithms were available for other languages such as Danish and German. FFNN approach has been used to obtain a quick and satisfactory solution.

- Sonar target detection. A FFNN network with two hidden layers was trained to distinguish metal cylinders on the ocean bottom from rock with reflected sonar signals.

- Car navigation. A FFNN was trained to drive a car on a winding road.

- Image compression. Here a FFNN has the targets identical to the input image. The image is compressed by using relatively fewer hidden nodes. The input-to-hidden connections perform the encoding and the hidden-to-output connections perform decoding.

- Backgammon. This is a popular and ancient game of skill. A variety of approaches have been tried to develop a computer program to play this game. *Neurogammon* which was developed by training a FFNN won the gold medal at the London Olympiad for computer backgammon in 1989.

- Postal Code Recognition. An FFNN has been developed to recognize postal codes.

- Speech Recognition. This is a very difficult problem in *Artificial Intelligence*. FFNN have been useful to solve some aspects such as distinguishing from a given set of words.

1.2 Comparison of Forecast and Actuality

In this thesis we will be interested in comparing the out-of-sample forecasting performance of FFNN with ARIMA and other time series models. We focus primarily on one-step forecasting. There are several possible methods of implementing multi-step forecasts with neural nets which are discussed in §**??**. A naive view is that since in the long-run all reasonable forecasting methods for a stationary process are the same, it is often the case that one-step forecasts discriminate the most between inadequate and adequate models. In most previous forecasting experiments with time series one-step forecasts have been used. However, in actual practice, some models

perform depending on the lead time of the forecast and further work should be done with multi-step forecasts with the FFNN.

For some time series data, $z_t, t = 1, \ldots, N$, suppose that we use a portion N_1 for training and N_2 as test data for out-of-sample forecasts. We wish to compare the performance of two models which generate forecast errors $e_t^{(1)}$ and $e_t^{(2)}$ respectively from specified target values Y_t. For convenience we renumber these errors and corresponding target values, $t = 1, \ldots, N_2$.

A simple comparison of the performance of the methods is given by just comparing the mean-square error or root-mean-square error of the forecasts. Median absolute error, mean absolute percentage error and other criteria have been used in previous forecasting studies (Hipel & McLeod (1994, Ch.15) and there are many other possibilities. A new criterion which does not appear to have been used in previous forecast studies is based on the Pitman Measure of Closeness (PMC) for comparing statistical estimators. For comparing competing forecasts from models 1 and 2 the PMC is estimated by

$$\text{PMC}\,(1,2) = \hat{p} = \sharp\{|e_t^{(1)}| < |e_t^{(2)}|\}/N_2,$$

where $\sharp\{\bullet\}$ denotes the number of elements in the set. If model 1 is better than model 2, $\text{PMC}\,(1,2) > 0.5$. To test if the improvement is statistically significant we can use a normal approximation to the binomial. So the two-sided significance level is given by $2(1 - \Phi(|Z|))$ where

$$Z = \sqrt{N_2}\frac{\hat{p} - 0.5}{\sqrt{\hat{p}(1-\hat{p})}}.$$

The PMC criterion measures which forecast is most often nearest to the truth. Using the the RMSE to compare the forecasts on the other hand is relevant when the magnitude of the forecast error is of concern.

With one-step forecasts it is also possible to use a simple statistical test to check if there is a statistically significant difference in mean-square error performance.

1.2.1 Pitman's test

Since the one-step forecast errors under a valid model are approximately uncorrelated we can use Pitman's test for testing the equality of variances in the paired sample case. (Hipel & McLeod, 1994, §8.3.2). This test can tell us if the forecasting performance as measured by mean square error is statistically significantly different between the two models. Pitman's test is easy to implement. Let $S_t = e_t^{(1)} + e_t^{(2)}$ and $D_t = e_t^{(1)} - e_t^{(2)}$ where $t = 1, \ldots, N_2$. Then under $\mathcal{H}_0 :$ Var $\{e_t^{(1)}\} =$ Var $\{e_t^{(1)}\}$, S_t and D_t should be uncorrelated. So to test the null hypothesis $\mathcal{H}_0$ we can compute the Pearson correlation coefficient,

$$r = \frac{\sum(S_t - \bar{S})(D_t - \bar{D})}{\sqrt{\sum(S_t - \bar{S})^2 \sum(D_t - \bar{D})^2}}.$$

The two-sided significance level is given $\Pr\{T > |\hat{t}|\}$, where

$$\hat{t} = r\sqrt{\frac{N_2 - 2}{(1 - r) \times r}},$$

and T denotes a random variable from the t-distribution with $N_2 - 2$ degrees of freedom.

1.2.2 Visualization

Visualization and graphical methods often provide insight superior to significance tests by indicating the magnitude of differences and often revealing unexpected outliers and other features of the data. Cleveland (1993) stresses the use of visualization methods as diagnostic methods to check the statistical assumptions behind fitting and also for exploratory data analysis where no explicit probability model is entertained. We adapt some of the methods discussed by Cleveland (1993) to compare the out-of-sample forecasting performance.

First we look at time series plots showing the observed and forecast. This is an obvious plot to provide a quick examination of the forecasting capability. However the time series plot is not very good at revealing differences between competing models.

The residual-fit (RF) spread plot introduced by Cleveland (1993) provides a visual summary of the amount of variability accounted for by a model and is readily adapted to out-of-sample forecast comparison. The RF-spread plot is comprised of two panels. Both panels are constructed from data quantile plots. Given some data, say $X_1, \ldots, X_m$, the quantile plot is a plot of $X_{(i)}$ vs $(i - 1/2)/m$ where $X_{(i)}$ denotes the i-th smallest data value. The left panel shows the quantile plot of the targets minus their mean, $Y_t - \bar{Y}$. The right panel of the RF-spread plots is a quantile plot of the forecast errors. The horizontal and vertical scales are identical on both plots. Keeping the vertical scale the same means the amount of variability of the data in the plot can be compared. For comparing the forecasting performance of two models we keep the scales identical on both pairs of RF-spread plots.

Another plot that we have adapted for comparing the forecasting performance is the Tukey mean-difference plot which is also discussed in Cleveland (1993). The Tukey mean-difference plots is generally useful when we want to compare paired data, $(A_i, B_i), i = 1, \ldots, m$. If we look at a scatter plot of B vs A then the differences between the pairs is indicated by the vertical displacement from the 45° line. Tukey, noting that it is much easier to visualize vertical displacements from a horizontal line, transformed the data by a planar rotation of 45°. Thus the new coordinates, (C, D) are given by

$$\begin{aligned} C &= \cos(45^\circ)A + \sin(45^\circ)B \\ D &= -\cos(45^\circ)A + \sin(45^\circ)B \end{aligned}$$

Since $\sin(45^\circ) = 1/\sqrt{2}$ and $\cos(45^\circ) = 1/\sqrt{2}$ we see that

$$\begin{aligned} C &= (A + B)/\sqrt{2} \\ D &= (-A + B)/\sqrt{2} \end{aligned}$$

Since the linear scaling is arbitrary, Tukey mean-difference plot consists of plotting $B - A$ vs $(B - A)/2$. For our forecasting experiments we take A and B to be the target and forecast respectively. Our display then is comprised of two Tukey mean-difference plots with the same scaling. The Tukey mean-difference plot was

found helpful in visualizing the difference in forecast performance between competing models. Essentially the Tukey mean-difference plot used in this way is similar to the standard model diagnostic plot of the residual vs the fit.

1.3 Statistical Models vs Neural Nets

Statistical models are based on probability assumptions whereas in neural nets there is no probability assumption. This feature has its benefits and costs for both approaches. With a probability model the tools of statistical inference can be used to make precise the amount of uncertainty involved in the parameter estimates, for giving probability bounds to predictions and for checking the model adequacy by checking how well the assumptions of the probability model are satisfied. Model comparisons of statistical models can be made by the use of likelihood-ratio tests and more generally using the Akaike or Bayes information criterion. Statistical diagnostic checking is also another advantage of statistical models.

In some statistical models, such as regression, the parameters estimated may have some scientific or technological interest. This is not often relevant for ARIMA time series forecasting but it is certainly relevant in other ARIMA modelling situations such as in *Intervention Analysis* and more generally regression and dynamic regression with autocorrelated errors.

The parameters in the neural net approach do not have any particular scientific meaning. In fact widely different parameter settings in the neural net approach may lead to models which make virtually the same predictions. The lack of a probability model makes the neural net seem comparatively like a purely black-box approach. But by not specifying such rigorous probability and model assumptions the neural net offers a more flexible approach to model building. This flexibility is especially helpful in nonlinear and nonGaussian situations.

If the goal of the modeling is rigorous understanding, explication and quantification of uncertainty, statistical models have a lot to offer. Sometimes the goal is merely prediction and here the neural net approach is often helpful.

In this thesis the FFNN model is compared with statistical time series models for

several monthly time series datasets and for the annual lynx series. The usefulness and limitations of the FFNN model for several types of Markovian dependence is illustrated. We conclude with an application to half-hourly streamflow forecasting where it is difficult to posit a suitable statistical model.

2 Airline Data

2.1 Introduction

The airline data were modeled by Box & Jenkins (1976) and also by previous authors. This time series consists of the total number of airline passengers every month on flights between U.S.A. and England over the period 1949–1960. A time series plot of this data is shown in Figure **??**.

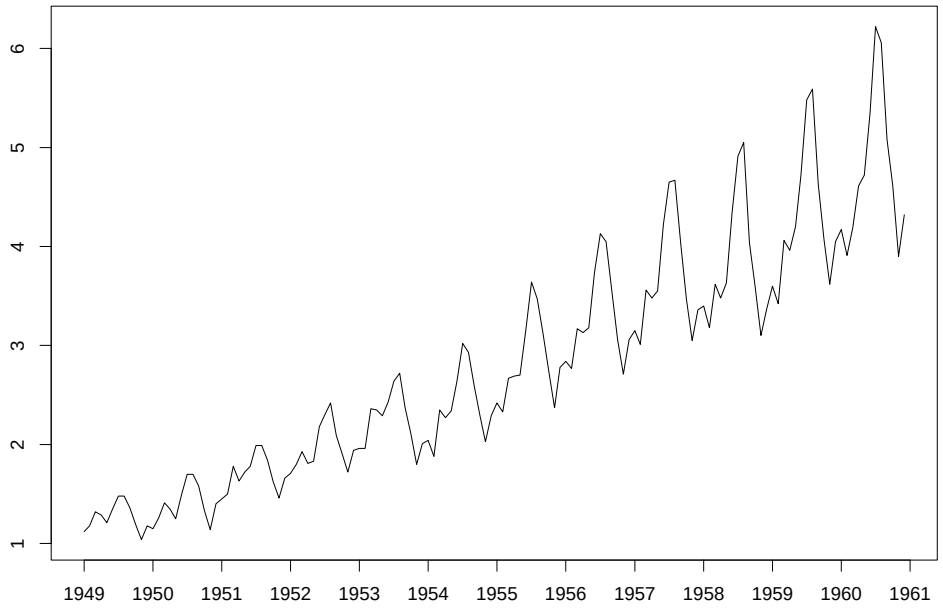


Figure 3: *The Airline Data*

Faraway & Chatfield (1998) applied the FFNN to the airline data listed. Their main conclusion was that it is dangerous to apply FFNN models blindly in the so called black box mode and that FFNN models did not offer a substantial improvement. In our review we find some problems with the methodology used in the study of Faraway & Chatfield (1998) and we show that there is a definite improvement in

the out-of-sample forecasts.

We redo the computations of Faraway & Chatfield (1998) and we find several flaws in their presentation and conclusions. All the calculations in this section were done by using the statistical software S-plus function `nnet` (Ripley, 1999) which was adapted by Faraway & Chatfield (1998) to time series prediction. We also compare the Splus neural net package on this airline data set with the *Neural Connection* package produced by SPSS Inc (1998).

2.2 Faraway & Chatfield's Paper

Faraway & Chatfield (1998) divided airline data into two parts. The first 132 values were used as training data and last 12 as test data for out-of-sample forecasting. Then they compared varying 13 FFNN models with the seasonal autoregressive integrated moving average model, SARIMA, of order $(0,1,1)\times(0,1,1)_{12}$, which was introduced by Box & Jenkins (1976). They are compared by using the following statistics.

- S, the sum of squared residuals up to time 132. Note the residuals are the within-sample one-step ahead forecast errors.
- $\hat{\sigma} = \sqrt{S/(n-p)}$, the estimate of residual standard deviation, where n denotes the number of effective observations used in fitting the model and p denotes the number of weights function required in the model.
- The Akaike information criterion, $\text{AIC} = n\ln(S/n) + 2p$.
- The Bayesian information criterion $\text{BIC} = n\ln(S/n) + p + p\ln(n)$.
- The sum of squares of multistep-ahead forecast errors made from origin time 132 up to maximum lead time 144. This is denoted as SS_{MS} in Table **??** of the observations from time $132+1$ to the end of the series.
- The sum of squares of one-step-ahead forecast errors of the observations from time $132+1$ to the end of the series. It is denoted as SS_{1S} in Table **??**.

The main results of Faraway & Chatfield (1998) are reproduced in Table **??**. Usually when we compare forecasts, Mean Square Error, denoted by MSE, or Root Mean Square error, RMSE, is used. The RMSE for Table **??** is given in Table **??**.

In the next two subsections we point out some major reservations that we have with the methodology of Faraway & Chatfield (1998).

2.2.1 Multistep Ahead Forecast Comparison

Faraway & Chatfield (1998) obtained the multistep-ahead forecasts in the following way:

- Construct a model with a training data set.
- Use the one-step-ahead forecast to replace the lag 1 value as one of the input variables. The same architecture could then be used to construct the two-step-ahead forecast and so on.

Their multistep-ahead forecasts are constructed by analogy with the known results for linear ARIMA models and are not really appropriate for nonlinear FFNN models. Rather we should train the FFNN model for each forecast. Multistep-ahead forecast calculated by the model trained for each forecast is given in Table **??**. We can see that the model NN$(1, 12, 13; 2)$ produced the best forecasts and is different from the results by one-step ahead forecast. We cannot see big differences between this method and Faraway and Chatfield's method, but it's apparent that if the steps become bigger, then this method will does much better than their multistep-ahead forecast.

The third way to get the multistep-ahead forecast is to train the NN model by setting several outputs for one FFNN model. For example, if we would like to get 12-steps-ahead-forecast, we should set 12 outputs for one FFNN model. This model indicates that we get the 12-steps ahead forecast by using last few observation. It is important to increase the number of hidden nodes correspondingly for each additional

Lags	**Number of hidden neurons**	**Number of weights**	**Measure of Fit**				**Forecast**	
			S	$\hat{\sigma}$	AIC	BIC	SS_{MS}	SS_{1S}
1,2,3,4	2	13	7.74	0.245	-333	-283	58.52	1.03
1-13	2	31	0.73	0.091	-545	-428	1.08	0.71
1-13	4	61	0.26	0.067	-605	-375	4.12	1.12
1,12	2	9	2.30	0.144	-456	-422	0.35	0.34
1,12	4	17	2.16	0.145	-448	-383	0.38	0.44
1,12	10	41	1.77	0.150	-424	-268	0.51	0.59
1,2,12	2	11	2.17	0.141	-459	-418	0.34	0.29
1,2,12	4	21	1.91	0.139	-455	-375	6.82	1.03
1,2,12,13	2	13	0.99	0.097	-543	-494	0.37	0.52
1,2,12,13	4	25	0.81	0.093	-543	-449	0.34	0.52
1,12,13	1	6	1.18	0.102	-537	-514	0.33	0.50
1,12,13	2	11	1.03	0.098	-543	-501	0.33	0.50
1,12,13	4	21	0.84	0.093	-547	-467	0.54	0.62
SARIMA model		2	1.08	0.095	-556	-546	0.39	0.43

Table 2.1: *Faraway & Chatfield (1998)Results. Comparison of various FFNN models together with the corresponding values for the SARIMA airline model by Chatfield. SS_{MS} denotes sum of squares of residuals for multi-step ahead forecasts for the test data and SS_{1S} denotes sum of squares of residuals for one-step ahead forecast for the test data. The number of training data is 132 and the number of test data is 12.*

Lags	**Number of hidden neurons**	**Number of weights**	**RMSE for SS_{MS}**	**RMSE for SS_{1S}**
1,2,3,4	2	13	2.208	0.293
1-13	2	31	0.734	0.243
1-13	4	61	0.586	0.306
1,12	2	9	0.171	0.168
1,12	4	17	0.178	0.191
1,12	10	41	0.206	0.222
1,2,12	2	11	0.168	0.155
1,2,12	4	21	0.754	0.166
1,2,12,13	2	13	0.176	0.296
1,2,12,13	4	25	0.168	0.208
1,12,13	1	6	0.166	0.206
1,12,13	2	11	0.166	0.200
1,12,13	4	21	0.212	0.297
SARIMA model		2	0.180	0.189

Table 2.2: *Root Mean Square Error (RMSE) Comparsions. Comparison of various FFNN models together with the corresponding values for the SARIMA airline model for 50 times iteration SS_{MS} denotes sum of squares of residuals for multi-step ahead forecasts for the test data and SS_{1S} denotes sum of squares of residuals for one-step ahead forecast for the test data. The number of training data is 132 and the number of test data is 12.*

Lags	**Number of hidden neurons**	**Number of weights**	**RMSE**
1,12	2	9	0.175
1,12	4	17	0.201
1,2,12	2	11	0.171
1,2,12	4	21	0.211
1,2,12,13	2	13	0.180
1,12,13	2	11	0.159
SARIMA model			0.159

Table 2.3: *Comparison of multistep-ahead forecasts trained for each forecast RMSE is given for last 12 forecasts out of 144 data points.*

output forecast. The result by this method is given in Table **??**. However it does not produce as good the result as the one produced by the other methods.

In addition there is another defect to be criticized for Faraway and Chatfield's multistep-ahead forecasts. Let $e_t(k)$ denote the forecast error of the k-step ahead forecast from forecast origin t. Then multistep ahead forecasts error variance can be written $\mathrm{Var}\,(e_t(l)) = \sigma^2 V(k)$, where σ^2 is innovation variance, ψ_j is coefficient at lag j in the infinite moving-average expansion of the SARIMA model and $V(k)$ is the variance inflation factor, given by

$$V(k) = \sum_{j=0}^{l-1} {\psi_j}^2. \tag{2.1}$$

Since the variance is completely different at each lag, it is not sensible to look at an average of the multistep-ahead forecasts at lag 1,2, $\cdots$, 12 as was done by Faraway & Chatfield (1998). Figure **??** shows that for the fitted SARIMA model the variance inflation factor, $V(k)$ increases with k and for $k = 12$ implies a nearly 5-fold increase. Figure **??** shows the difference of 60-steps ahead forecasts between the NN$(1, 12; 2)$ model and the SARIMA model.

Lags	**Number of hidden neurons**	**Number of weights**	**RMSE**
1-12	2		0.622
1-12	4		1.488
1-12	24		0.690
1-12	36		0.986
SARIMA model		2	0.159

Table 2.4: *Comparison of multistep-ahead forecasts obtained by setting 12 outputs. RMSE is given for last 12 forecasts out of 144 data points. NN(1-12;24) and NN(1-12;36) are calibrated by Neural Connection.*

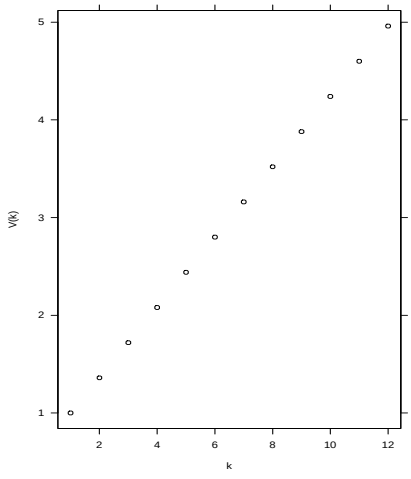


Figure 4: *The variance inflation factor, fitted SARIMA model.*

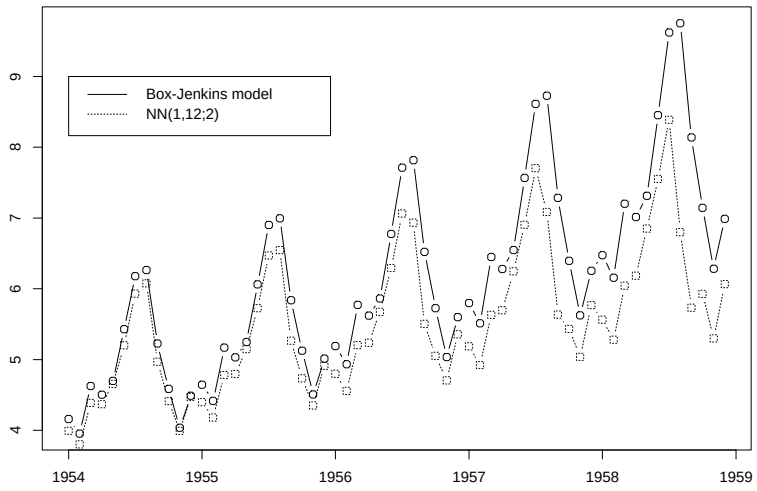


Figure 5: *Multistep-ahead forecasts for the airline data using the SARIMA model and the NN*$(1, 12; 2)$ *model. Forecasts are computed at lags* $k = 1, ..., 60$ *starting with a forecast origin of* $t = 72$ *which corresponds to January 1954.*

2.2.2 Misuse of AIC and BIC

One of Faraway and Chatfield's main conclusions about AIC is that if the model is chosen on the basis of minimizing the AIC or $\hat{\sigma}$, then the NN$(1 - 13; 4)$ model will be selected, see Table **??**. But as Faraway & Chatfield (1998) point out, it leads to poor forecast. Often it if there are a large number of parameters in the model, the result is a poor forecasts. This result can be established theoretically in the sense that if a parametric model is overfit, then the forecast is degraded.

Since AIC is expressed as $n \ln(S/n) + 2p$ for the neural network, the reason why NN$(1 - 13 : 4)$ model gets the large number of AIC is that the $n \ln(S/n)$ terms dominates the $2p$ term. In other words, AIC doesn't penalize enough for the extra parameters.

It is true that there is a trade-off between bias and variance on the number of parameters but the only justification of this is by analogy with parametric statistical models. The AIC is not defined for the FFNN model because there is no likelihood.

The *energy function* minimized is only analogous to a sum of squares in least squares. It is not sensible to assume a probability function for the errors.

Another point to note is that if we were to fit a ridiculous model such as a polynomial regression of high-order, say p, then by taking p large enough we can drive S to zero and so the AIC would select this absurd model. For this reason, Akaike has recommended the we never allow $p > n/4$ when using model selection with the AIC. The NN$(1-13;4)$ has 60 adjustable weight parameters and only 132 data values, so this principle has also been ignored by Faraway & Chatfield (1998).

Faraway & Chatfield (1998) use of the AIC and BIC is entirely on an ad hoc basis. No other respected researcher in the neural nets has ever recommended the use of AIC or BIC to select neural nets architecture. We believe the assumptions behind the development of these information criterion are so different from the assumptions in the FFNN and that consequently the AIC and BIC should not be used to select neural net architecture. Instead in the neural net literature it is recommended that we split the data into three parts: training, validation and test. We can make the final model choice by comparing the performance on the test data.

2.3 Local Minima

Faraway & Chatfield (1998) were concerned about the fact that there are many local minima for the *energy function* and these local minima typically produce different values for the weights. Their results are reproduced in the Table **??**. As a conclusion, they mention that there is no guarantee that the model in the Table **??** with the smallest S-value gives the global minima. They picked and examined only the model NN$(1, 12, 2)$, which has the smallest number of weights but what happens with models with even more weights?

Faraway & Chatfield (1998) reported that the models reported are the result of refitting the model at least 50 times from different random starting points and taking the best of the resulting minima. However there is no mention about the number of iterations. Table **??** shows the same experiment as Table **??** except the number of iteration times is fixed to 50. It is not difficult to notice that those two tables are not consistent with each other, especially in the cases where the number of parameters is large. The same thing happens to Table **??** with 100 times iteration. This tells us that the number of iterations is not enough for models with the large parameter number. Note also in **??** that there is a big difference in the fit when the number of hidden nodes increases from 2 to 4.

Table **??** shows four different local minima of the NN$(1 - 13; 2)$ models after 500 times iteration. This model has a large number of parameters. Although there are differences in forecast values, S-values are quite consistent. You could also see that the model from the Table **??** produces a better forecast than other models even though this model has the worst S-value.

Fit S	SS_{1S}
2.30	0.34
2.32	0.38
2.38	0.31
2.41	0.31
2.49	0.33
2.49	0.35
2.51	0.40

Table 2.5: *Fit and forecast accuracy for seven local minima for the NN*$(1, 12; 2)$ *model with 132 training data in ascending order of fit accuracy by Faraway & Chatfield (1998). S denotes sum of squares of one-step ahead forecast for 132 training data.* SS_{1S} *denotes sum of squares of one-step ahead forecast for the last 12 airline test data.*

2.4 Graphical comparison and statistical test for the SARIMA model and FFNN models

We saw that NN$(1, 2, 12; 2)$ produced the least sum of square value for the test data. In this section we compare NN$(1, 2, 12; 2)$ with the SARIMA$(0, 1, 1)(0, 1, 1)_{12}$ model graphically and we use Pitman's test to see if the MSE produced by both models are significantly different.

Graphical comparison gives more information than just comparing RMSEs. Fitted values and Observed values for the airline data by SARIMA model and NN$(1, 2, 12; 2)$ is given in Figure **??**. Both fitted values for out-of-sample forecast appear to be quite close to the observed values but this graph is not very good at comparing the differences in forecast errors.

Figure **??** and **??** show that the prediction errors for the NN$(1, 2, 12; 2)$ are slightly smaller. The difference is highly statistically significant, Pitman's test $r = 0.99$, $\hat{t} = 26.41$, achieved significance level of $2\Pr(t > \hat{t}) = 1.4 \times 10^{-10}$. In terms of the Pitman statistical test, the FFNN does significantly outperform the SARIMA in out-of-sample forecasting for the air data. There remains the question of practical

Lags	**Number of hidden neurons**	**Number of weights**	**S**	**SS_{1S}**
1,2,3,4	2	13	5.87	1.68
1-13	2	31	0.66	1.43
1-13	4	61	0.18	2.45
1,12	2	9	2.30	0.32
1,12	4	17	2.08	0.30
1,12	10	41	1.59	15.45
1,2,12	2	11	2.17	0.29
1,2,12	4	21	1.73	0.33
1,2,12,13	2	13	0.72	1.05
1,2,12,13	4	25	0.81	0.52
1,12,13	1	6	1.18	0.51
1,12,13	2	11	0.99	0.48
1,12,13	4	21	0.80	1.06
SARIMA model		2	1.18	0.35

Table 2.6: *Comparison of various FFNN models together with the corresponding values for the SARIMA airline model for 50 times iteration. S denotes sum of squares of one-step ahead forecast for 132 training data. SS_{1S} denotes sum of squares of one-step ahead forecast for the last 12 airline test data.*

Lags	**Number of hidden neurons**	**Number of weights**	**S**	**SS_{1S}**
1,2,3,4	2	13	5.86	1.67
1-13	2	31	0.67	1.31
1-13	4	61	0.21	1.67
1,12	2	9	2.30	0.32
1,12	4	17	2.07	0.30
1,12	10	41	1.44	1.00
1,2,12	2	11	2.17	0.29
1,2,12	4	21	1.76	0.36
1,2,12,13	2	13	0.98	0.50
1,2,12,13	4	25	0.72	0.37
1,12,13	1	6	1.18	0.51
1,12,13	2	11	0.99	0.48
1,12,13	4	21	0.78	1.01
SARIMA model		2	1.18	0.35

Table 2.7: *Comparison of various FFNN models together with the corresponding values for the SARIMA airline model for 100 times iterations. S denotes sum of squares of one-step ahead forecast for 132 training data. SS_{1S} denotes sum of squares of one-step ahead forecast for the last 12 airline test data.*

Fit S	$\mathbf{SS_{1S}}$
0.642	0.78
0.649	0.78
0.657	0.73
0.671	1.26
0.73	0.71
	(Table **??**)

Table 2.8: *Fit and forecast accuracy for four local minima for the* $NN(1-13;2)$ *model with* 132 *training data in ascending order of fit accuracy with 500 times iteration. S denotes sum of squares of one-step ahead forecast for 132 training data.* SS_{1S} *denotes sum of squares of one-step ahead forecast for the last 12 airline test data.*

significance. This is partially answered by the RF-spread and Tukey mean-difference plots which visually demonstrate the improved forecasts from the FFNN approach.

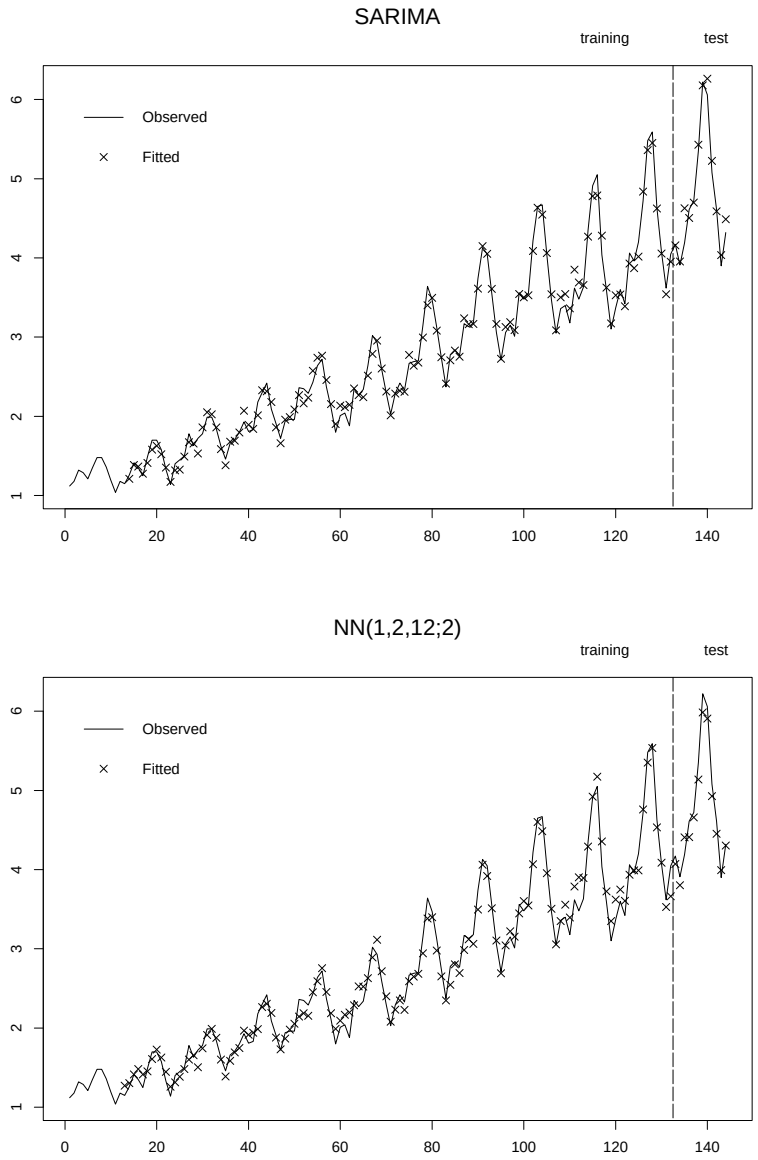


Figure 6: *Fitted values for the airline data by the SARIMA model and* $NN(1, 2, 12; 2)$. *Observations are plotted with a solid line. The number of training data is 132 and the number of test data is 20.*

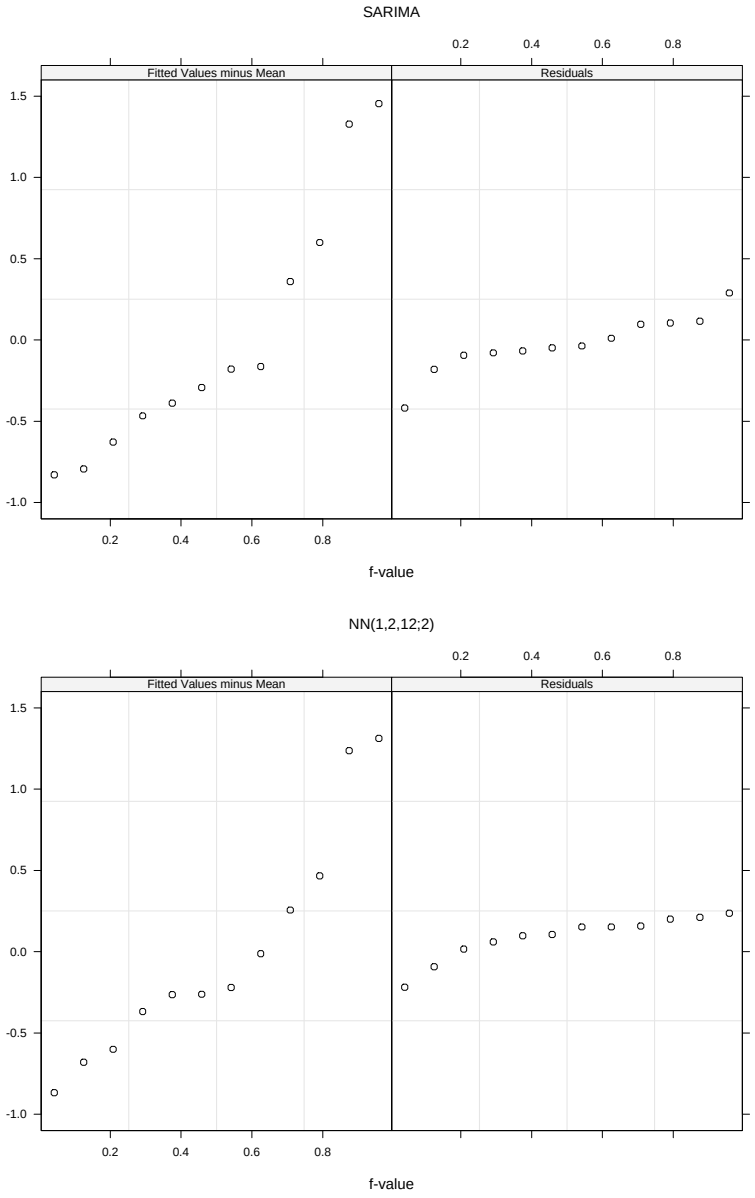


Figure 7: *Residual-fit spread plots of the 12 airline test data by SARIMA model and* $NN(1, 2, 12; 2)$*. The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.*

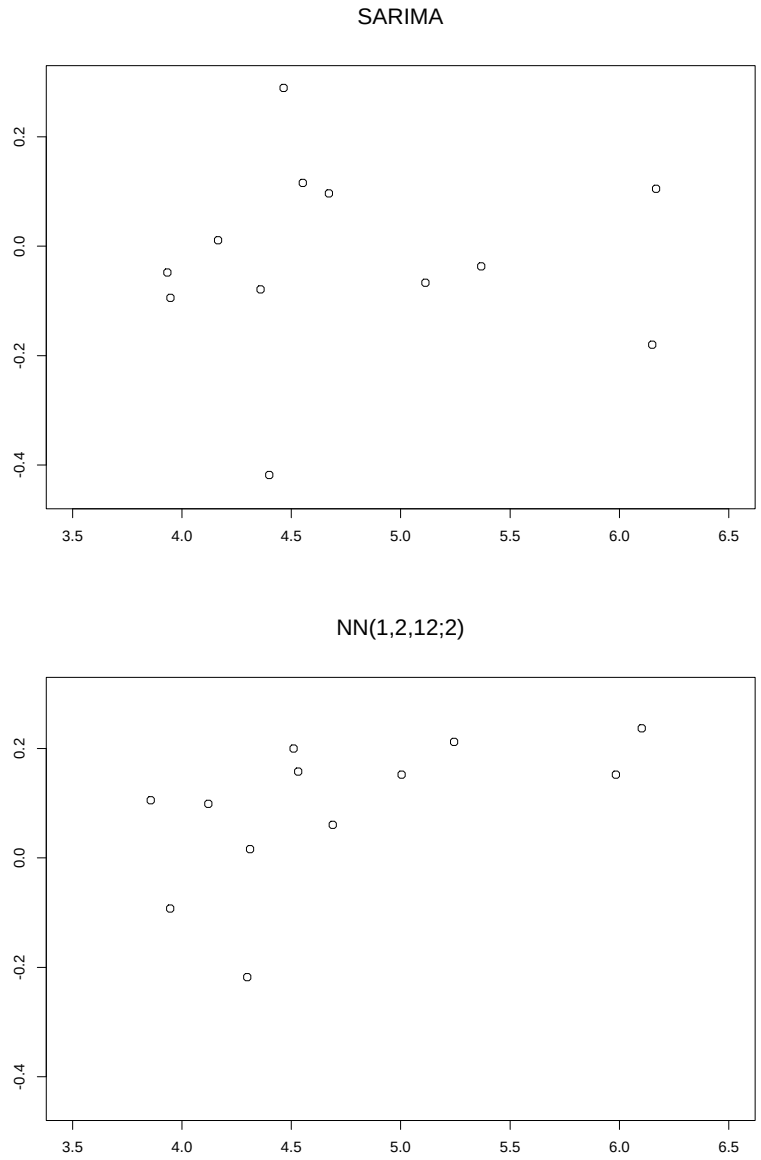


Figure 8: *Tukey mean difference plots of the 12 airline test data by SARIMA model and* $NN(1, 2, 12; 2)$. *The horizontal axis in each plot shows the* $(X + Y)/2$ *and the vertical axis* $Y - X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

2.5 Transformation of the Data

For statisticians, it is natural to use logarithms for the airline data to get the stable variance and make the fit easier. The *Application Guide* with *Neural Connection* SPSS Inc. (1998) recommends data preprocessing or filtering to remove if possible outliers, lack of symmetry, normality or variance changes and to make the region where the model is fit resemble as close as possible the region where data forecasts are to be made.

For these reasons we used the FFNN model for the transformed data

$$w_t = \nabla\nabla_{12}\ln(z_t),$$

where ∇ is the backward difference operator and ∇_{12} is the backward seasonal difference operator. A plot of the w_t shown if Figure **??** shows that the resulting series looks stationary and reasonably Gaussian.

The NN forecasts for fitting to the transformed series and the back-transforming the forecasts to the original domain are shown in Table **??**. It appears that the NN models does not give better forecasts than the SARIMA$(0,1,1)(0,1,1)_{12}$ model. This is confirmed with the Pitman's test. The attained significance level in comparing the SARIMA$(0,1,1)(0,1,1)_{12}$ model and the NN$(1,2,12;2)$ fit to w_t model yielded $r = 0.031$, $\hat{t} = 1.830$ and a significance level 0.070. This indicates the variances within samples from both models are significantly different. In short, contrary to the case without transformation, SARIMA$(0,1,1)(0,1,1)_{12}$ does significantly outperform NN$(1,2,12;2)$ in out-of-sample forecasting for the air data. The RF-spread and Tukey mean difference plots shown in Figures **??** and **??** also confirm that there is a little difference between the SARIMA and FFNN model.

Thus data transformation by logging, differencing and seasonal differencing do not produce a better model in this case. A possible reason is that the FFNN models is not suitable for the MA component that this modeled by the SARIMA. Using more hidden nodes would not help model this sort of dependence.

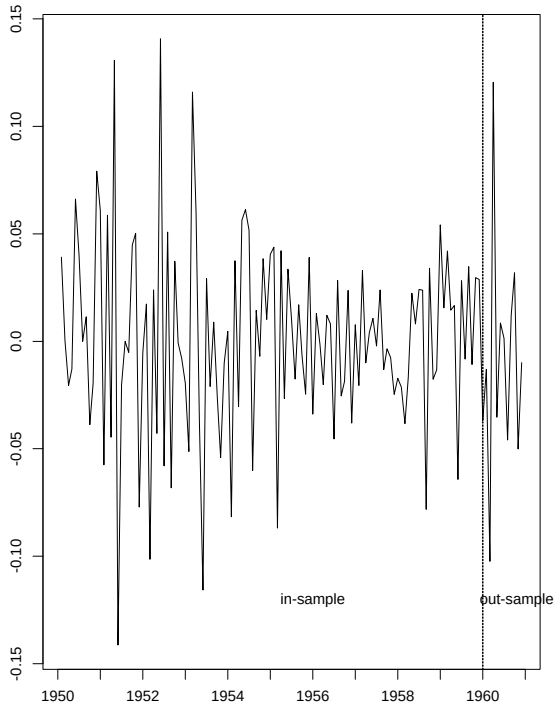


Figure 9: *Time series plot of w_t, the transformed air data.*

FFNN models	**S**	$\mathbf{SS_{1S}}$
NN(1; 1)	1.33	0.57
NN(1; 2)	1.30	0.55
NN(1, 2; 1)	1.33	0.57
NN(1, 2; 2)	1.30	0.56
NN(1, 12; 1)	1.14	0.49
NN(1, 12; 2)	1.02	0.52
NN(1, 12; 1)	1.14	0.50
NN(1, 12; 2)	1.02	0.64
NN(1, 12, 13; 1)	1.09	0.47
NN(1, 12, 13; 2)	0.97	0.50
NN(1, 2, 12, 13; 1)	1.08	0.57
NN(1, 2, 12, 13; 2)	0.94	0.64
SARIMA model	1.18	0.35

Table 2.9: *Results for FFNN models fitted to the differenced and seasonal differenced logged data and corresponding Box & Jenkins model. Sum of squares are given after back transformation S denotes sum of squares of one-step ahead forecast for 132 training data. SS_{1S} denotes sum of squares of one-step ahead forecast for the last 12 airline test data.*

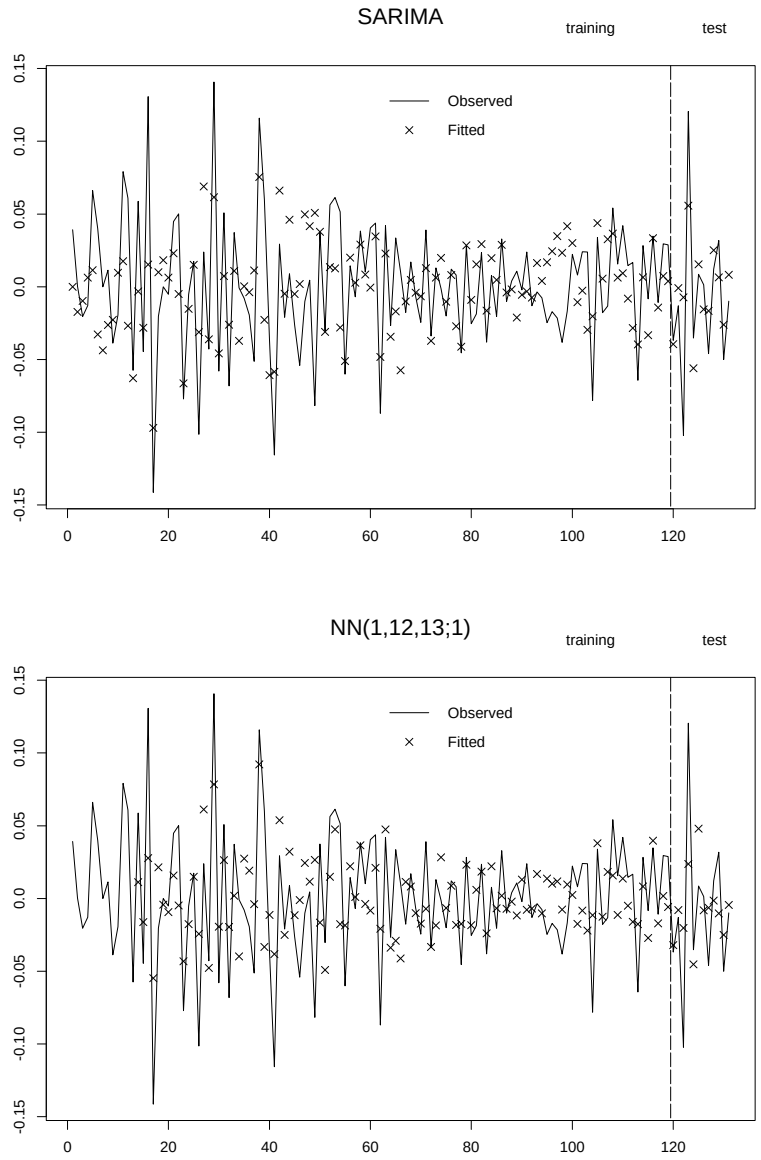


Figure 10: *Fitted values for the differenced and seasonally differenced logged airline data by the SARIMA model and* $NN(1, 12, 13; 1)$. *Observations are plotted with a solid line. The number of training data is 132 and the number of test data is 20.*

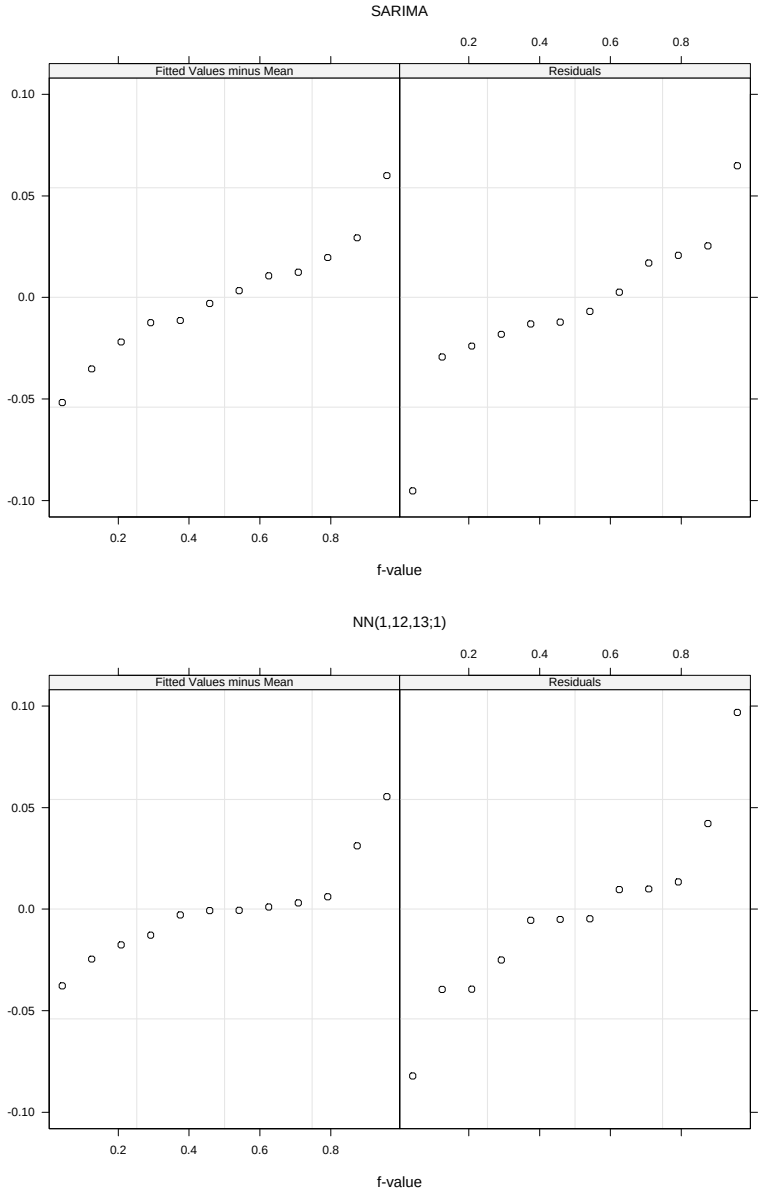


Figure 11: *Residual-fit spread plots of the last 12 differenced and seasonally differenced logged airline test data by SARIMA model and* $NN(1, 12, 13; 1)$. *The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.*

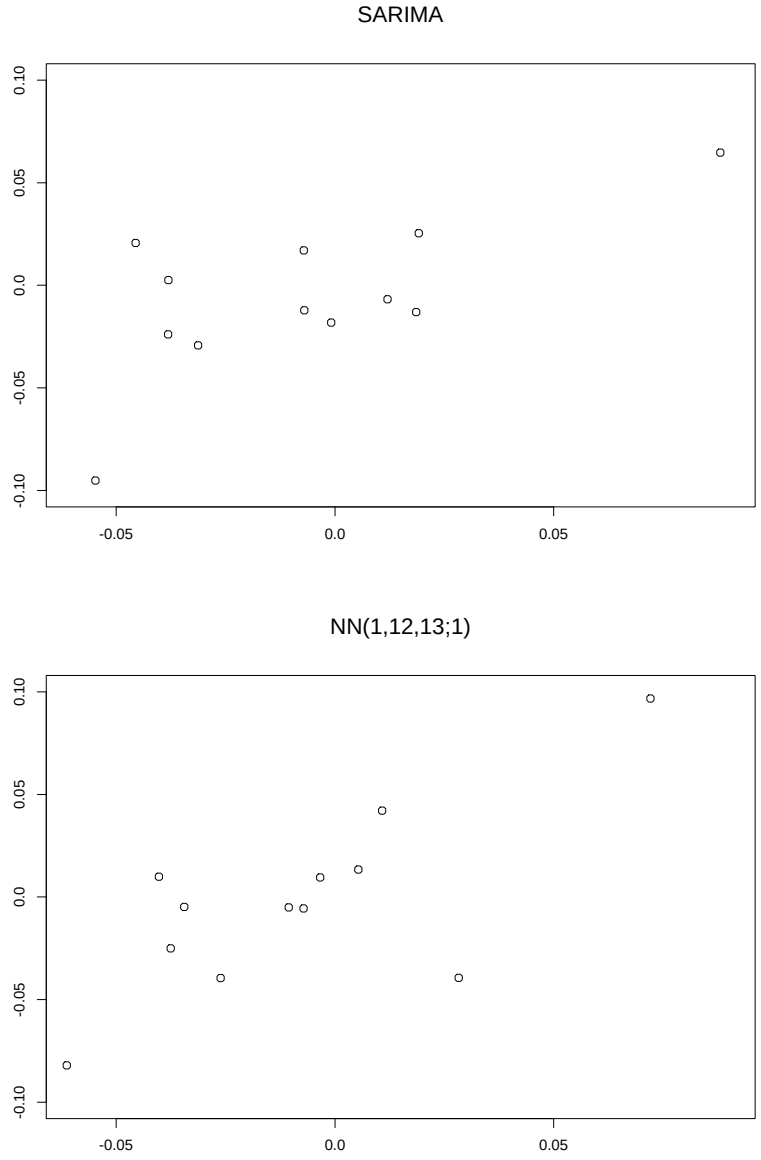


Figure 12: *Tukey mean difference plots of the last 12 differenced and seasonally differenced logged airline test data by SARIMA model and NN*$(1, 12, 13; 1)$*. The horizontal axis in each plot shows the* $(X + Y)/2$ *and the vertical axis* $Y - X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

2.6 Combining Forecasts

We investigate two methods of improving the forecast from the FFNN by combining its forecast with the forecast from the SARIMA$(0,1,1)(0,1,1)_{12}$. As suggested in McLeod (1993) and in Hipel and McLeod (1994), combination of forecasts makes most sense when the forecasts that are being combined make use of different information sets or when the forecasts are based on models which taken account of different types information in the data. Since the neural net contains both the linear and nonlinear information about the series, it is not clear from this principle whether combination will help.

The standard method of combining forecasts (Winkler and Makridakis, 1983; McLeod, Noakes, Hipel and Thompstone, 1987) consists of taking a weighted sum of the two different forecast algorithms. If there are k forecasts available, the combined forecast f_c would be

$$f_c = \sum_{i=1}^{k} w_i f_i, \tag{2.2}$$

where f_i is the forecast produced by the ith model; w_i is the weighting factor for the ith forecast. The optimal weights are given by

$$w_i = \frac{\sum_{j=1}^{k} \alpha_{ij}}{\sum_{h=1}^{k}\sum_{j=1}^{k} \alpha_{hj}}, \tag{2.3}$$

where the α_{ij} the (i,j)-entry in the inverse covariance matrix of the forecast errors from the k methods. In practice the covariance matrix is estimated using the sample covariance matrix of the forecast errors in the training sample. As can be seen from Table **??** there is an improvement in the FFNN method by combining with the SARIMA model.

Another approach is to utilize the forecast from the SARIMA$(0,1,1)(0,1,1)_{12}$ model directly in the FFNN,

$$y = f(\mathbf{x}, \hat{z}_t), , \tag{2.4}$$

where y is the output for target value $z(t)$, f is a FFNN function, $\mathbf{x}$ is a vector of the lagged observed values and $\hat{z}_t$ is the one-step forecast from SARIMA$(0,1,1)(0,1,1)_{12}$. For example, if the value at lag 1 and 12 is the input values then $\mathbf{x} = (z_{t-1}, z_{t-12})$,

Model	RMSE from FFNN models	Combined RMSE
NN$(1, 12; 2)$	0.163	0.158
NN$(1, 12; 4)$	0.158	0.157
NN$(1, 2, 12; 2)$	0.155	0.150
NN$(1, 2, 12, 13; 2)$	0.166	0.158

Table 2.10: *Combination: Standard Method. Combining SARIMA forecasts and one-step-ahead forecasts for 12 airline test data by FFNN models. The data in Table* **??** *is used. RMSE for the SARIMA model is 0.170.*

Model	Combined RMSE
NN$(1, 12, \hat{z}_t; 2)$	0.197
NN$(1, 12, \hat{z}_t; 3)$	0.200
NN$(1, 12, \hat{z}_t; 4)$	0.253
NN$(1, 2, 12, \hat{z}_t; 2)$	0.219
SARIMA model	0.170

Table 2.11: *Combination: New Method. Combining SARIMA forecasts and one-step-ahead forecasts of Neural Network by using the equation* **??**

We use the $\hat{z}_t$ forecasts from the SARIMA model and then fit the FFNN model using $\mathbf{x} = (z_{t-1}, z_{t-12})$. From Table **??** we see that the result is slightly worse than the forecasts only by the SARIMA models. So this method does not appear promising.

2.7 Comparison of *Neural Connection* and `nnet`

The software which was used previously uses a popular algorithm called back-propagation for computing the first derivatives of the objective function. Here, sum of squares of the within-sample one-step-ahead forecast errors, $E = \sum_t (\hat{x}_t - x_t)^2$ would be the most likely objective function. The derivatives, $\partial E/\partial w$, where w denotes a weight, may be computed by the backpropogation algorithm. There are many ways to use these derivatives for optimization and the fitting method relied on the Broyden-Fletcher-Goldfarb-Shanno algorithm (Fletcher, 1987) which is a quasi-Newton method.

Neural Connection includes several tools and neural network architecture for modeling and forecasting. For the FFNN architecture *Neural Connection* provides the following options:

1. number of hidden layers
2. number of nodes in each layer
3. transfer, or activation, function used by nodes
4. learning algorithm used by the FFNN
5. initial values of the weights between nodes

The first two options are the standard ways to improve the results and as a matter of course, the S-PLUS software also can handle these. The last three options are special to *Neural Connection* and could make this software more powerful. Although the initial weight values might be crucial to the modeling, they must be inputted by hand and this causes the iterations more troublesome for the user.

Table **??** and Table **??** show the differences between `nnet` and *Neural Connection* for the airline data. We have used RMSE instead of sum of squares of one-step-ahead forecasts. Mainly the default settings are used in Table **??** and Table **??** except the distribution of weight function is used in two different ways. The other settings of options are as follows:

- The activation function used by the nodes. The tanh function is one among three functions.
- The weights range is set between -0.1 and 0.1 and its random seed is set to 50.
- As a learning algorithm, the conjugate gradient method is selected in two methods.

The column *Total correct* indicates the total number of correct which results from the *Cross Tabulation Matrix*. This matrix, produced by *Neural Connection*, classifies the predicted values and tells us if the predicted values are close enough to the actual values automatically.

From Table **??** it can be seen that RMSEs are consistent for the smaller RMSE range between `nnet` and *Neural Connection*, but `nnet` does much better than *Neural Connection* for the other range. In other words, *Neural Connection* is as good as `nnet` only if the model is good as far as the airline data is concerned. In a sense *Neural Connection* is good at telling us the difference between a good model and a bad model compared to `nnet` .

There are few differences between uniform distribution, Table **??**, function and Gaussian distribution function, Table **??** for the weight.

As it is mentioned before, there are several ways to improve the model. Table **??** shows four local minimas for different starting weight values. All the values are almost same, which means that different starting weight values do not influence on the results for $\mathrm{NN}(1, 12; 2)$.

Table **??** shows four different local minimas for four different weight ranges. This option does not improve the result either, but rather make the result worse as the range becomes larger.

Table **??** shows 4 different local minima by using the steepest descent method as a learning algorithm. There are two ways to update the weight. `Epoch` is the way updating weights after an entire pass of patterns has been presented to the network and `Pattern` is the way after each pattern has been presented to the network. The number of patterns is same as the number of input sets. Though we can't see any

		Neural Connection		`nnet`	
$\mathcal{L}$	h	σ_F	σ_P	σ_F	σ_P
1,2,3,4	2	0.231	0.456	0.245	0.293
1-13	2	0.091	0.415	0.091	0.243
1-13	4	0.086	0.351	0.067	0.305
1,12	2	0.144	0.166	0.144	0.168
1,12	4	0.178	0.190	0.145	0.191
1,12	10	0.137	0.183	0.150	0.221
1,2,12	2	0.142	0.166	0.141	0.155
1,2,12	4	0.128	0.430	0.139	0.293
1,2,12,13	2	0.100	0.212	0.097	0.208
1,2,12,13	4	0.091	0.190	0.093	0.208
1,12,13	1	0.293	0.521	0.102	0.204
1,12,13	2	0.275	0.549	0.098	0.204
1,12,13	4	0.235	0.647	0.093	0.227

Table 2.12: *Uniform Weight Initialization. Comparison of various FFNN models. The initial weights are the uniform function between -0.1 and 0.1,* $\mathcal{L}$ *is the set of lags,* σ_F *is RMSE for training data and* σ_P *is RMSE for test data. For the* $SARIMA(0,1,1)(0,1,1)_{12}$, $\sigma_F = 0.095$ *and* $\sigma_P = 0.189$.

		Neural Connection		`nnet`	
$\mathcal{L}$	h	σ_F	σ_P	σ_F	σ_P
1,2,3,4	2	0.268	0.558	0.245	0.293
1-13	2	0.091	0.407	0.091	0.243
1-13	4	0.074	0.399	0.067	0.305
1,12	2	0.144	0.165	0.144	0.168
1,12	4	0.138	0.183	0.145	0.191
1,12	10	0.134	0.293	0.150	0.221
1,2,12	2	0.142	0.166	0.141	0.155
1,2,12	4	0.128	0.163	0.139	0.293
1,2,12,13	2	0.099	0.208	0.097	0.208
1,2,12,13	4	0.092	0.252	0.093	0.208
1,12,13	1	0.293	0.521	0.102	0.204
1,12,13	2	0.436	0.553	0.098	0.204
1,12,13	4	0.235	0.649	0.093	0.227

Table 2.13: *Gaussian Weight Initialization. Comparison of various FFNN models The initial weights are the Gaussian, $\mathcal{L}$ is the set of lags, σ_F is RMSE for training data and σ_P is RMSE for test data. For the SARIMA$(0,1,1)(0,1,1)_{12}$, $\sigma_F = 0.095$ and $\sigma_P = 0.189$.*

Uniform		**Gaussian**	
σ_F	σ_P	σ_F	σ_P
0.144	0.166	0.144	0.166
0.144	0.167	0.144	0.169
0.144	0.167	0.144	0.166
0.144	0.166	0.144	0.165

Table 2.14: *Local Minima Found With Different Starting Weights. Four different local minimas for the* NN$(1, 12; 2)$ *model by changing starting weights between -0.1 and 0.1. σ_F is RMSE for training data and σ_P is RMSE for test data.*

Weight range	**Uniform**		**Gaussian**	
	σ_F	σ_P	σ_F	σ_P
+-0.2	0.144	0.167	0.144	0.167
+-0.3	0.144	0.166	0.146	0.193
+-0.5	0.144	0.166	0.145	0.191
+-1	0.144	0.204	0.148	0.204

Table 2.15: *Local Minima Found by Varying the Weight Range. Four different local minimas for the* $\mathrm{NN}(1, 12; 2)$ *model by changing starting weights range.* σ_F **is RMSE for training data and** σ_P **is RMSE for test data.**

Weight update strategy	**Uniform**		**Gaussian**	
	σ_F	σ_P	σ_F	σ_P
Epoch	0.147	0.220	0.145	0.183
Epoch	0.145	0.183	0.145	0.182
Pattern	0.145	0.203	0.146	0.199
Pattern	0.145	0.194	0.145	0.198

Table 2.16: *Local Minima by Different Training Methods. Four different local minimas for the* $\mathrm{NN}(1, 12; 2)$ *model by using the steepest descent method* σ_F *is RMSE for training data and* σ_P *is RMSE for test data.*

Number of validation data	$\sigma_{\mathbf{P}}$	
	Uniform	Gaussian
12	0.279	0.283
18	0.269	0.204
24	0.319	0.311

Table 2.17: *Three different local minimas for the* $\mathrm{NN}(1, 12; 2)$ *model by setting the validation data* σ_P *is RMSE for test data.*

improvement of the results. This option might do better because there are still available options called *momentum coefficient* and *learning coefficient*, which can change the algorithm.

Now we try to improve the forecasting results by using validation data sets. It is given by dividing the training data set and allocating last few data. Using the validation dataset, the model which has less error for the validation dataset is selected. Table **??** shows the result by changing the number of validation data. In this case, we cannot see any improvement again.

In summary, for the airline data, we could not see any better results than the results by default setting for the airline data.

3 Sales of Company X

3.1 Introduction

In this section we treat a shorter time series and compare with the results for the air data. This data consists of 77 observations of the monthly sales of an engineered product over the period the January 1965–May 1971. A time series plot of this data is shown in Figure **??**.

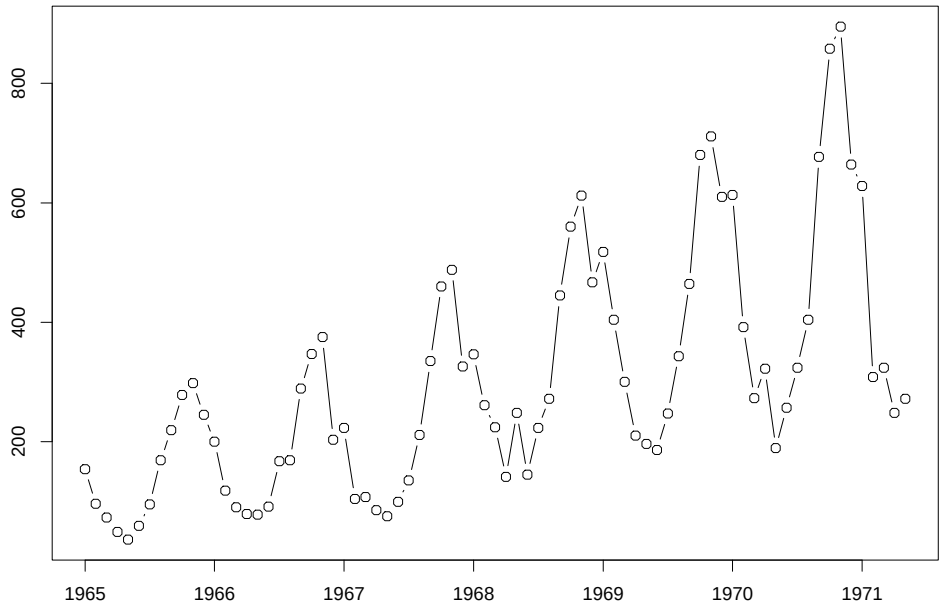


Figure 13: *The Sales Data*

Chatfield and Prothero (1973) misfit a SARIMA model to this data and wondered why their fit was not adequate. This point was solved by Box & Jenkins (1973) and Wilson (1973) who pointed out they had fit the wrong SARIMA model.

We briefly examine four methods of fitting a SARIMA model to this short time series. The SARIMA model we fit is the one recommended by Wilson (1973) and

Box and Jenkins (1973) but we refit with newer time series algorithms. Since the series is short, we might expect slightly different results from different ARIMA fitting algorithms. The fitted models were:

- *Method 1:* Wilson's model
- *Method 2:* McLeod approximate maximum likelihood with the optimal Box-Cox transformation given by the MHTS package This method has the merit that it reduces the calculation time.
- *Method 3:* McLeod approximate maximum likelihood with the optimal Box-Cox transformation 0.34 as is given by Wilson
- *Method 4:* Ansley exact maximum likelihood

Although the SARIMA parameter coefficients are quite different, all the four methods produce almost the same amount of RMSE and make no difference. We will use method 2. The model in the method 2 is expressed as

$$(1 + 0.556B)\nabla\nabla_{12}(X_t)^{.281} = (1 - 0.693B^{12})Z_t$$

and which produces 52.63 of RMSE value.

To compare the difference of variability of the airline data and the sales data, first we take a transformed and differenced data to make both data sets stationary time series. Then we can meaningfully compare how much of the variability in these two datasets are explained by the model using the RF-spread plot. The transformed and differenced Sales data is given by $(1-B)(1-B^{12}){X_t}^{0.347}$ and is shown in Figure **??**.

Then we apply an appropriate ARMA model to those two data sets and make R-F spread plots. The residual and fitted value spread plots of the airline data and the sales data are demonstrated in Figure **??**. Both plots show the variability of fitted value and residuals are almost same and hence the signal to noise ratios must be about same.

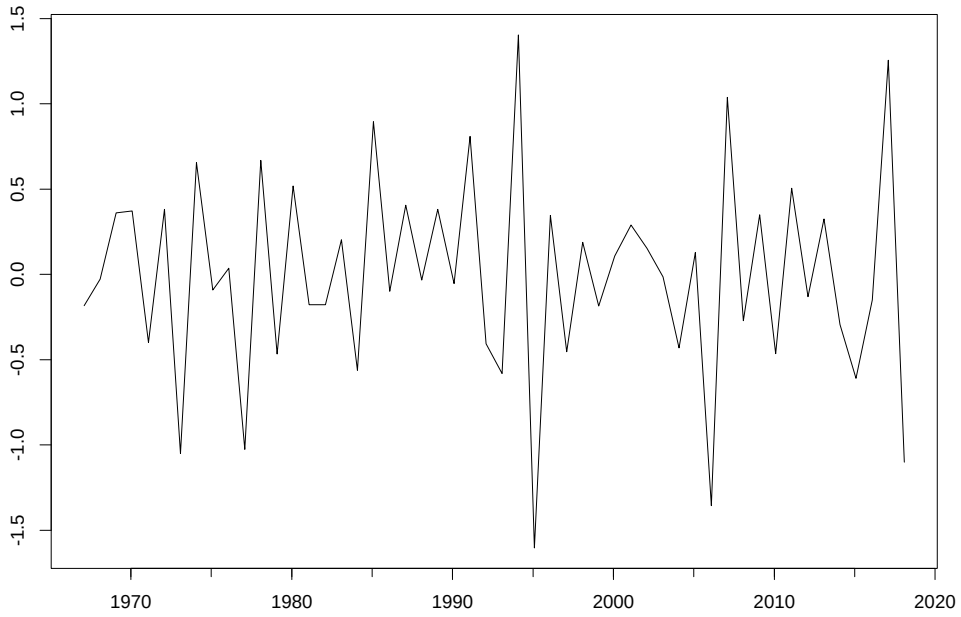


Figure 14: *Logged Differenced, Seasonally-Differenced Sales Data*

The Tukey mean-difference plot comparing the observed and fits for the best fitting SARIMA models for the air data and sales data are shown in Figure **??**. From Figure **??** it appears that there is still some systematic error left in the SARIMA model for the sales data since the five smallest data values on the left side correspond to large positive errors. In the air data, there is an apparent slight upward trend which also suggests lack of fit.

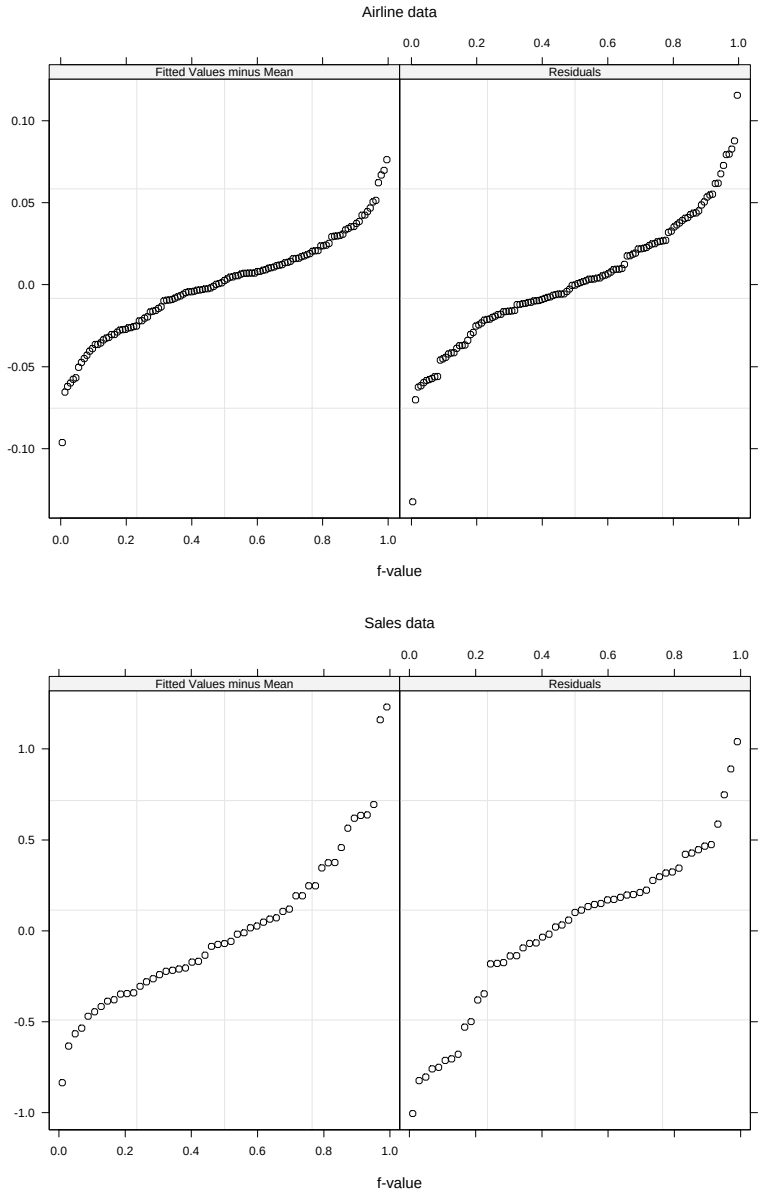


Figure 15: *Residual-fit spread plots of the airline data and the sales data. Both data are differenced and transformed. The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.*

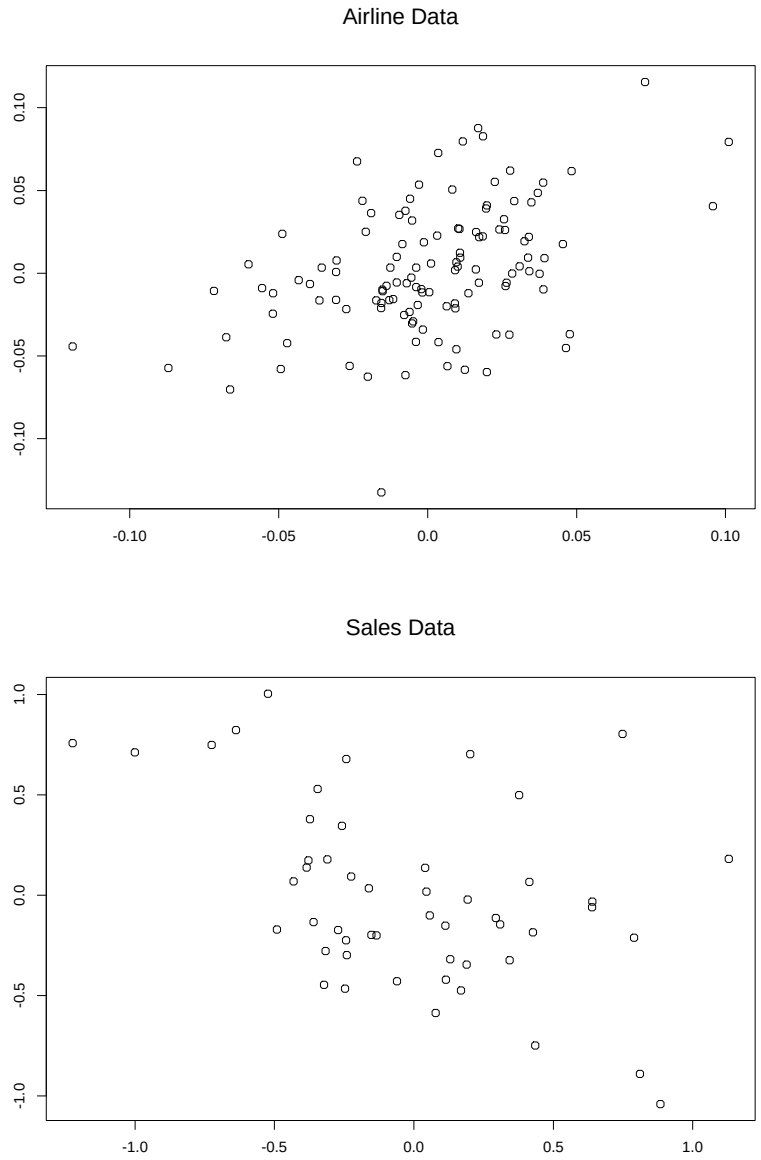


Figure 16: *Tukey mean difference plots of the airline data and the sales data. The horizontal axis in each plot shows the* $(X + Y)/2$ *and the vertical axis* $Y - X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

3.2 Comparison of forecasts between SARIMA model and FFNN model

We use $N_1 = 65$ as the length of training series and $N_2 = 12$ for the test data which correspond to the first 65 and last 12 values in the series. We fit the SARIMA$(1,1,0)(0,1,1)_{12}$ to the training data and obtained,

$$(1 + 0.594B)\nabla\nabla_{12}(X_t)^{.347} = (1 - 0.558B^{12})Z_t,$$

In this model, RMSE = 49.96 within-sample and RMSE = 68.20 out-of-sample. As is usually the case of the out-sample-performance is not as good.

3.2.1 FFNN with `nnet`

We use `nnet` for the same training and test data. The results appear in Table **??**. The σ_P produced by `nnet` is by far worse than those produced by the SARIMA model. Even the best model, NN$(1,2,12,13;2)$, gives only 88 of the RMSE value, which is worse than 68 of the RMSE value from the SARIMA model.

Plotting the fitted values by the SARIMA model and those by NN$(1,2,12,13;2)$ model is given in Figure **??**. At first sight, we can tell from the Figure **??** that the forecasts by the NN$(1,2,12,13;2)$ model does not trace the test data set as well as those by the SARIMA model does. Residual-Fit spread plots for the test data are given in Figure **??**. This plot shows that more variation is accounted by SARIMA model than by the NN model. Tukey mean difference plots for the test data are given in Figure **??**. Residuals by FFNN models are a little more spread than those by the SARIMA model.

The SARIMA model and NN$(1,2,12;2)$ model produce $r = 0.70$, $\hat{t} = 5.85$ and achieved significance level $2\Pr(t > \hat{t}) = 0.00067$ for the Pitman's test. This indicates the variances within samples from both models are significantly different.

For the out-of-sample forecasts, the SARIMA model and NN$(1,2,12;2)$ model produce PMC value $\hat{p} = 0.25$ with estimated standard deviation of 0.25. This also indicates that the SARIMA model has better performance than the NN$(1,2,12;2)$ model.

Lags	**Number of hidden neurons**	**Number of weights**	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
1,2	2	9	168	318
1,2,3	2	11	168	316
1,2,3	4	21	49	135
1,12	2	9	170	303
1,12	4	17	48	97
1,12	10	41	46	90
1,2,12	2	11	158	293
1,2,12	4	21	45	98
1,2,12,13	2	13	171	302
1,2,12,13	4	25	47	88
1,12,13	2	11	171	302
1,12,13	4	21	49	91
SARIMA model			50	68

Table 3.18: *Comparison of various* NN *models together with the SARIMA model. 64 training data and 12 test data.* σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

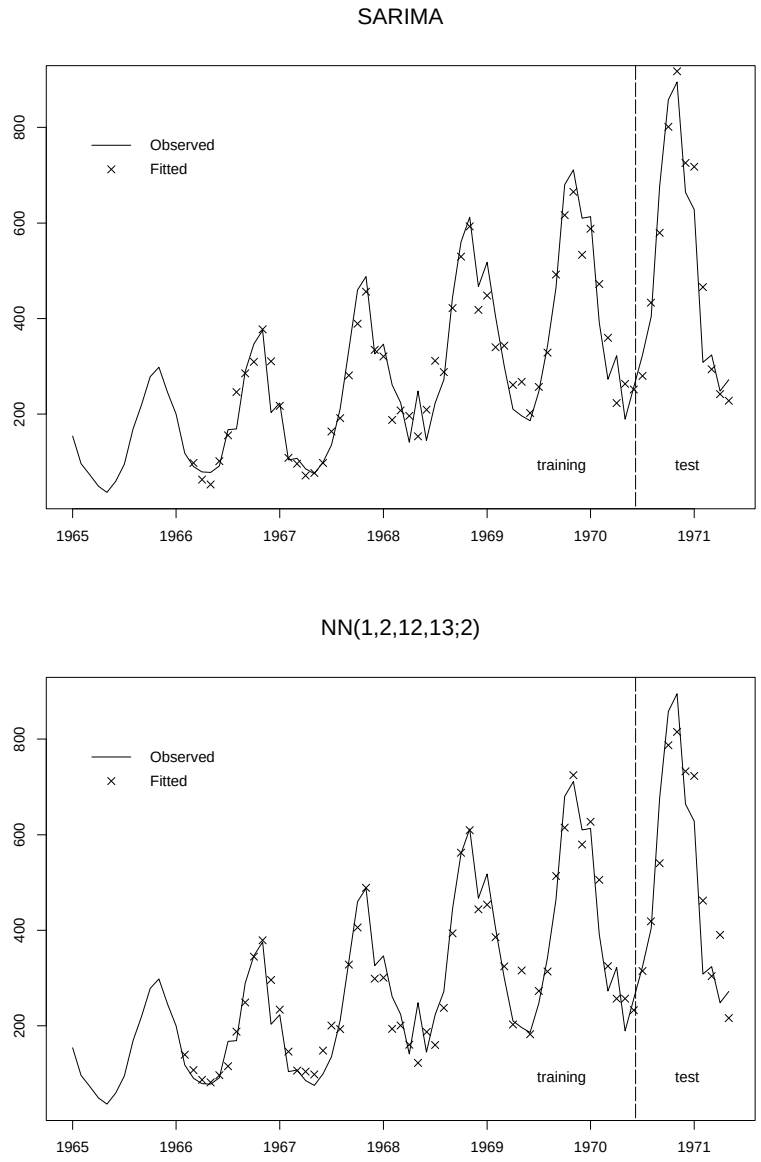


Figure 17: *Fitted values for the 12 sales test data by using SARIMA model and* $NN(1, 2, 12, 13; 2)$. *Observations are plotted with a solid line. The number of training data is 64 and the number of test data is 12.*

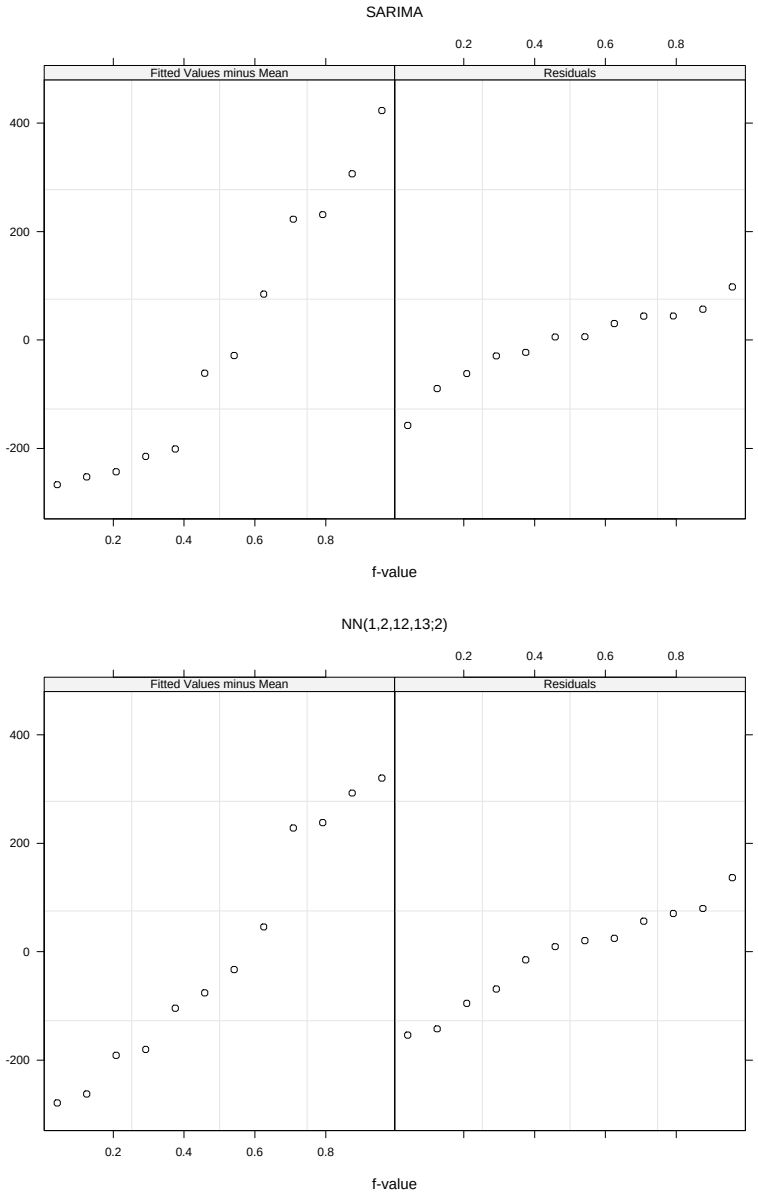


Figure 18: *Residual-fit values spread plot for the 12 sales test data of the SARIMA model and* $NN(1, 2, 12, 13; 2)$ *The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.*

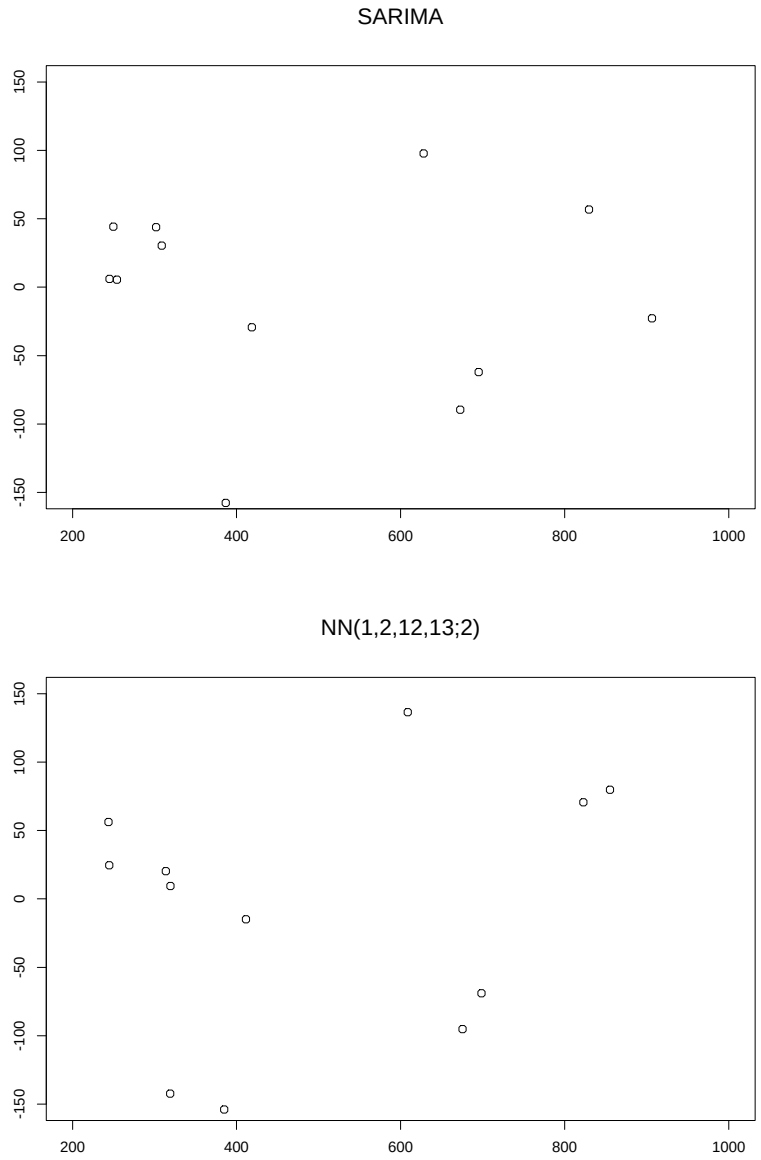


Figure 19: *Tukey mean difference plots for the 12 sales test data by SARIMA model and* $NN(1, 2, 12, 13; 2)$. *The horizontal axis in each plot shows the* $(X + Y)/2$ *and the vertical axis* $Y - X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

3.2.2 FFNN with *Neural Connection*

Now we use the *Neural Connection* instead of `nnet`. Only the default setting for the weight function and the way to update the weight is used. The result is given in Table **??**. There appears are some interesting differences between *Neural Connection* and `nnet`. *Neural Connection* does not require as many hidden nodes to get the better forecasts as NNET does. However the SARIMA model does much better than the FFNN models again.

It is curious that the validation data set might be useful for the data which has much variability. For the validation data set works as it prevents the model from being over-trained. However, Table **??** shows us that it cannot beat the SARIMA model. One possible reason that FFNN models failed is that the data length is too short and they can not build a appropriate model, and it is not because the signal to noise ratio is small since the sale data has almost same amount of signal to noise ratio for the SARIMA model.

Lags	**Number of hidden neurons**	**Number of weights**	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
1,2	2	9	67	119
1,2,3	2	11	66	121
1,2,3	4	21	55	128
1,12	2	9	47	95
1,12	4	17	36	104
1,12	10	41	23	146
1,2,12	1	5	49	97
1,2,12	2	11	47	93
1,2,12	4	21	38	125
1,2,12,13	2	13	44	94
1,2,12,13	4	25	33	100
1,12,13	2	11	45	89
1,12,13	4	21	39	164
SARIMA model			50	68

Table 3.19: *Comparison of various* NN *models by using Neural Connection together with the SARIMA model. 64 training data and 12 test data. σ_F is RMSE of the training data and σ_P is RMSE of the test data.*

Lags	**Number of hidden neurons**	**Number of validation data**	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
1,12	2	6	62	150
1,12	4	6	57	150
1,2,12	2	12	41	121
1,2,12	4	12	39	103
SARIMA model			50	68

Table 3.20: *Comparison of various* NN *models with a validation data set by using Neural Connection with SARIMA model 64 training data and 12 test data. σ_F is RMSE of the training data and σ_P is RMSE of the test data.*

3.3 Transformation of the sales data

The usual practice in time series modeling is first to make any necessary transformations to make the data approximately stationary and symmetrically distributed or Gaussian. So the Splus `nnet` is used for the transformed and differenced data,

$$w_t = (1 - B)(1 - B^{12})X_t^{0.347}.$$

The RMSE for the forecasts when back-transformed to the original untransformed domain are given in Table **??**. Though σ_F of SARIMA model is still smaller than that of FFNN model, there are visible improvement in the FFNN model by using the transformation, $\sigma_P = 0.52$ of NN$(1; 2)$ corresponds to $\sigma_P = 81$ for the model before transformation and it is the smallest value so far.

FFNN models	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN$(1; 1)$	0.48	0.53
NN$(1 : 2)$	0.45	0.52
NN$(1 : 3)$	0.42	0.57
NN$(1, 2; 1)$	0.48	0.54
NN$(1, 2; 2)$	0.41	0.75
NN$(1, 12; 1)$	0.47	0.53
NN$(1, 12; 2)$	0.41	1.79
NN$(1, 2, 12; 1)$	0.46	0.54
NN$(1, 2, 12; 2)$	0.37	0.85
NN$(1, 2, 12, 13 : 1)$	0.45	0.65
NN$(1, 2, 12, 13 : 2)$	0.35	0.65
SARIMA model	0.45	0.44

Table 3.21: *Comparison of various* NN *models with SARIMA model for the Sales data by using data transformation difference technique. 52 training data and 12 test data.* σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

The fitted values for the ARMA model and NN(1; 2) are plotted in Figure **??**. This plots shows clearly that observed values for the test data does not follow the same trend as observed values for the training data have, which causes the difficulty of forecasting. Considering the fact that SARIMA produces better forecasts than NN models, SARIMA model is in a sense more flexible than FFNN models.

Residual-fit value spread plots for the test data are given in figure **??**. It is shown that more variety is accounted by ARMA model than by the FFNN model. Tukey mean difference plots for the test data are given in figure **??**. There is an upward trend for the plot by NN(1; 2). Totally, ARMA model produced less forecast errors.

The SARIMA model and NN(1; 2) model produce $r = 0.27$, $\hat{t} = 1.93$, and achieved significance level $2\Pr(t > \hat{t}) = 0.081$ for the Pitman's test . This indicates the variances for within-sample forecasts from both models are significantly different.

Comparison of forecasts by three different methods is given in Figure **??**. We see that the ARMA model does better than the FFNN models, but at least data transformation and differencing did improve the performance in this case. As a conclusion, in keeping with standard statistical practice, the possibility of using a suitable data transformation should always be examined.

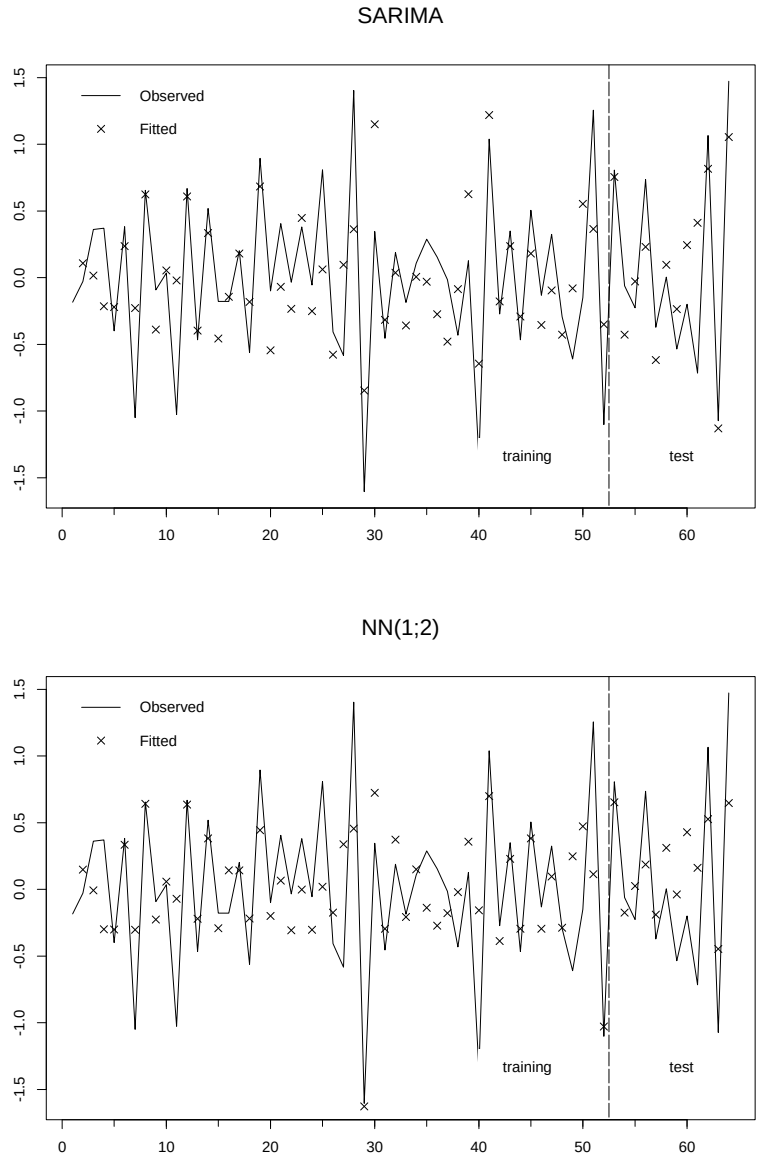


Figure 20: *The fitted values for the differenced and seasonally differenced power transformed sales data by the ARMA model and NN*$(1;2)$*. The number of the training data is 65 and the number of training data is 12. The sales data is transformed and differenced. Observations are given with the solid line. observed values for the test data does not follow the same trend as observed values for the training data have.*

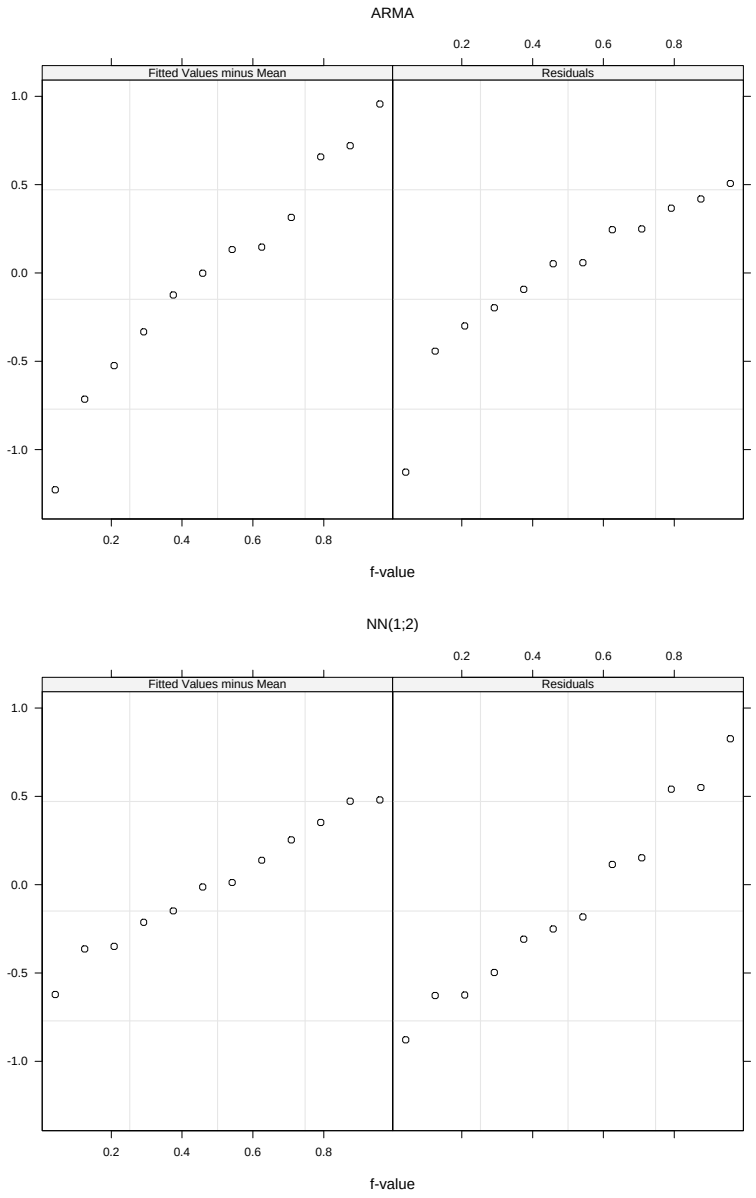


Figure 21: *Residual-fit value spread plot of the 12 test sales data for the ARMA model and* $NN(1;2)$ *The sales data is transformed, differenced and seasonally differenced. The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.*

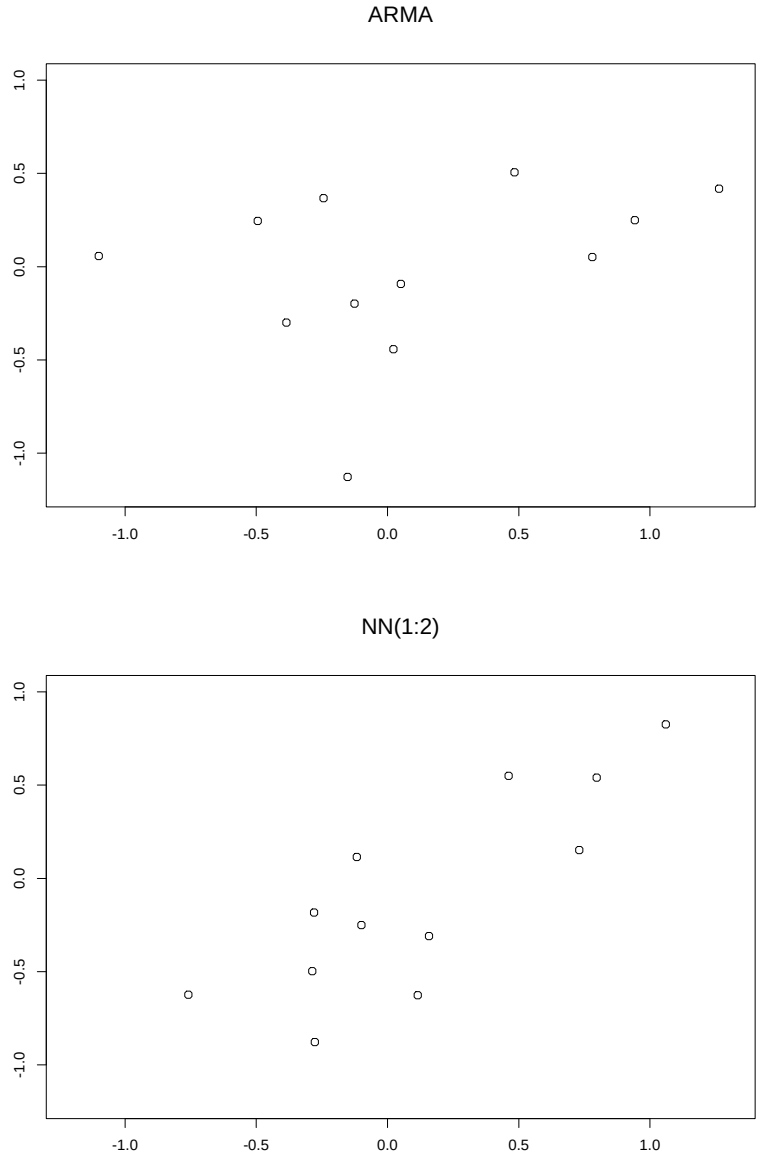


Figure 22: *Tukey mean difference plots of 12 sales test data for the ARMA model and $NN(1;2)$. The sales data is transformed, differenced and seasonally differenced. The horizontal axis in each plot shows the $(X + Y)/2$ and the vertical axis $Y - X$ where $X =$ one-step forecast and $Y =$ observed.*

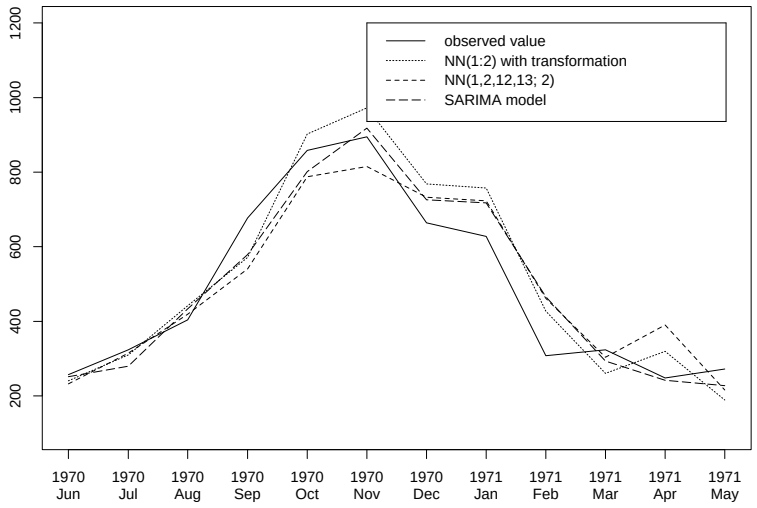


Figure 23: *Comparison of fitted values for the 12 sales test data by three different models,* $NN(1;2)$ *with data transformation and difference,* $NN(1,2,12,13;2)$ *and the ARMA model. Observed values are given with a solid line.*

4 The average monthly water usage

4.1 Introduction

We now turn to the average monthly consumption series in millions of liters per day in London from 1966 to 1988, which has longer data length than previous two examples. The time series plot in Figure **??** shows that the series is highly seasonal and nonstationary as was the case with the airline and sales data. It is somewhat longer than the airline and sales series.

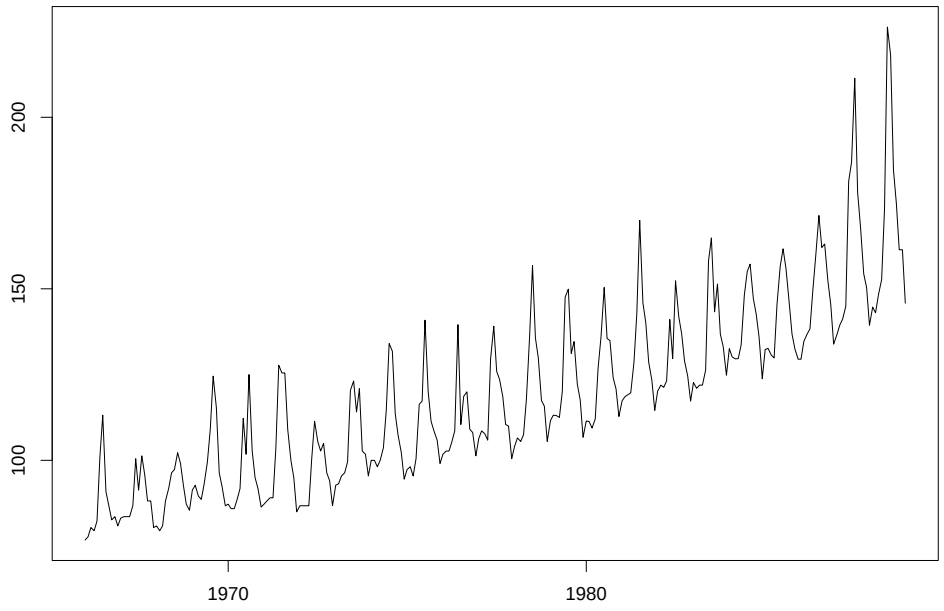


Figure 24: *The Average Monthly Water Usage.*

Hipel & McLeod (1994) fit the SARIMA $(1,0,1)(0,1,1)_{12}$ model with -0.75 power transformation to this data set. The residual-fit spread plot for the water usage data are illustrated in Figure **??**. As in the previous chapter, this plot is applied to the stationary series obtained after transformation and difference. To give us better understanding, the residual-fit spread plot for the SARIMA models fitted to the water usage and sales data are compared.

These plots suggest that signal to noise ratios of two data are almost same, but more proportion of the data of Water usage data is located around 0 than that of the data of Sales data is.

Figure **??** shows Tukey mean difference plots. The SARIMA model for the water usage seems fine but as we pointed out previously there is an indication of a problem with the SARIMA model for the sales data.

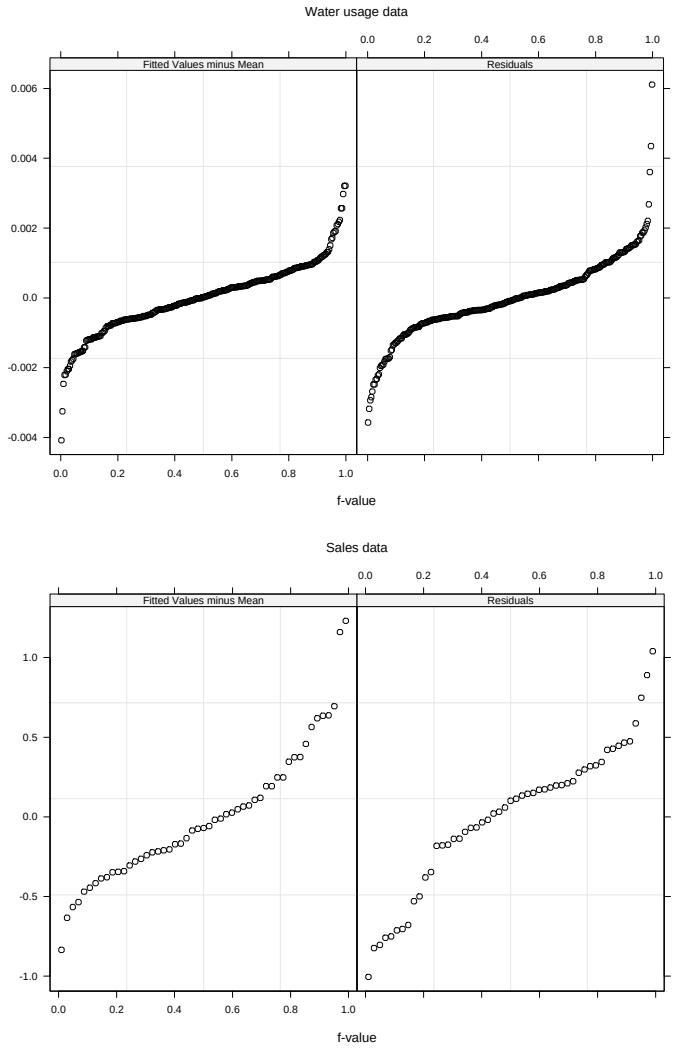


Figure 25: *Residual-fit spread plots of the water usage data and the sales data. The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.*

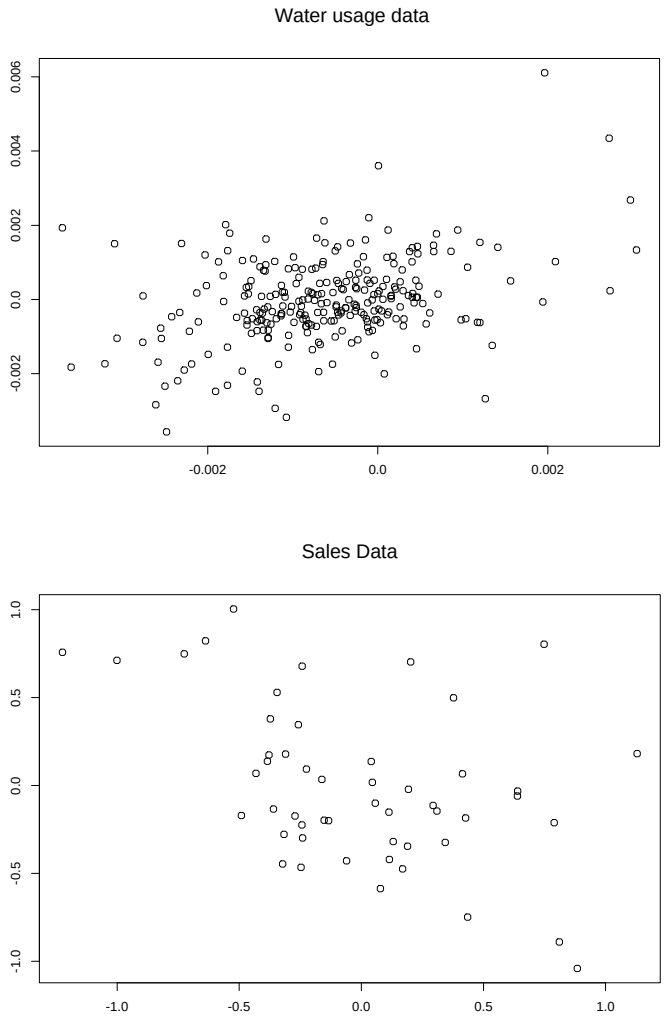


Figure 26: *Tukey mean difference plots of the water usage data and the sales data. The horizontal axis in each plot shows the* $(X + Y)/2$ *and the vertical axis* $Y - X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

4.2 Comparison of forecasts between SARIMA model and FFNN models

Now we apply SARIMA models and FFNN models to the water usage time series data. We divide the data set into two parts, training data set and test data set. Two schemes are examined. One scheme is 264 training data and 12 test data and the other is 200 training data and 76 test data. Because the last twenty four observations are quite different from previous observations, it would be difficult to predict those even though the data is less spread than the sales data. The point of these two dividing schemes is to check the robustness of our conclusions.

We also examine the effect of transformation and differencing. In this case the appropriate transformation is

$$w_t = (1 - B^{12}){X_t}^{-0.75}.$$

The modeling is carried out in the transformed domain of the w_t but the forecasts are back-transformed to the original domain, as was done in the last two chapters.

4.2.1 Original Domain, $N_1 = 264, N_2 = 12$

The results appear in Table **??**. NN$(1, 12, 13; 4)$ gives RMSE value of 11.80 and this is slightly better than 12.05 produced by SARIMA model.

Plotting the fitted values by the SARIMA model and those by NN$(1, 12, 13; 4)$ model is given in Figure **??**. We see that both the NN$(1, 12, 13; 2)$ model and the SARIMA model produce quite close fitted values. This means the performances of the FFNN and SARIMA are same as far as this data dividing scheme is concerned.

Residual-fit spread plots for the test data are given in Figure **??**. Residual-fit spread plots shows there is one positive outlier for both models.

Tukey mean difference plots, Figure **??**, show that NN$(1, 12, 13; 2)$ has upward trend, while the SARIMA model a little declining trend.

The SARIMA model and NN$(1, 12, 13; 4)$ model produce $r = 0.0021$, $\hat{t} = 0.14$, achieved significance level $2\Pr(t > \hat{t}) = 0.89$ for the Pitman's test. According to this

test the SARIMA and FFNN produce equal variances are in fact much closer might be expected.

FFNN models	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN$(1, 2; 2)$	23.70	59
NN$(1, 2 : 4)$	9.47	47.87
NN$(1, 2, 3 : 2)$	9.77	20.41
NN$(1, 2, 3; 4)$	9.20	23.83
NN$(1, 12; 2)$	23.38	58.34
NN$(1, 12; 4)$	7.10	15.37
NN$(1, 2, 12; 2)$	7.47	12.43
NN$(1, 2, 12 : 4)$	6.69	15.91
NN$(1, 2, 12, 13; 2)$	7.45	12.65
NN$(1, 2, 12, 13 : 4)$	7.41	12.78
NN$(1, 12, 13; 2)$	7.44	12.75
NN$(1, 12, 13; 4)$	7.18	11.80
Box-Jenkins model	6.75	12.05

Table 4.22: *Comparison of various* NN *models and Box-Jenkins model. The number of training data is 264 and that of test data is 12.* σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

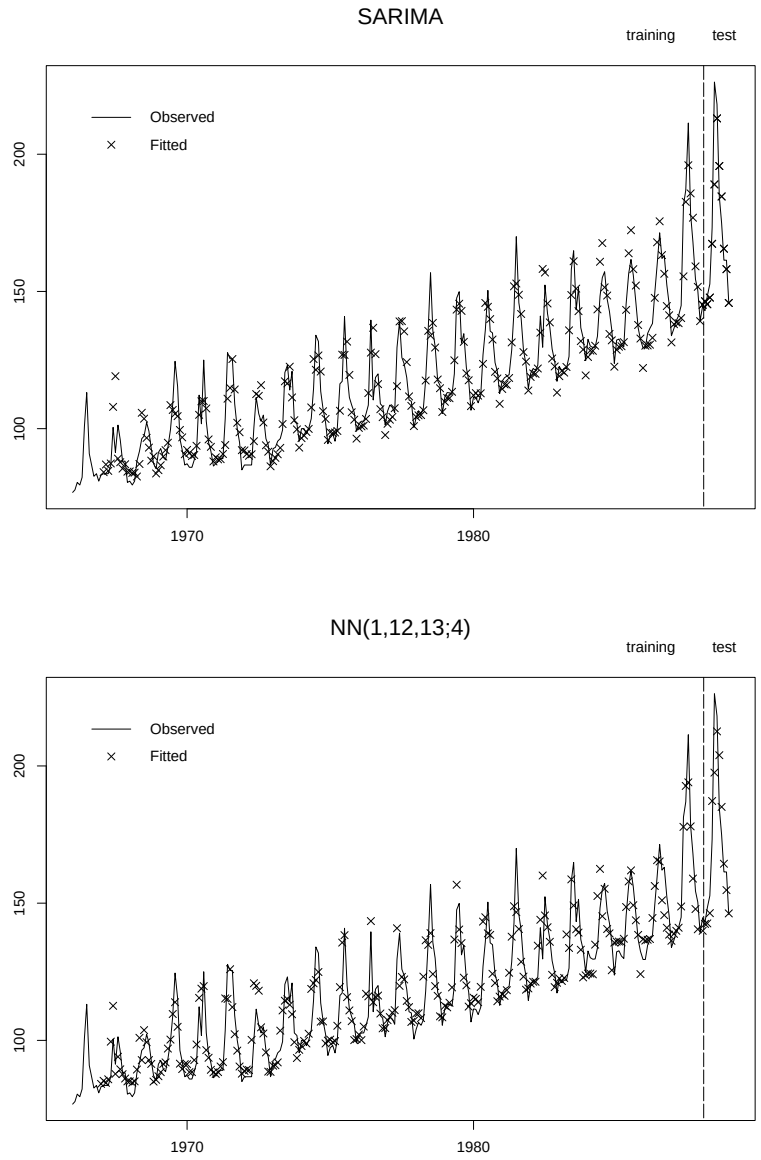


Figure 27: *Fitted values of the water usage by SARIMA model and by* $NN(1, 12, 13; 4)$. *Observations are plotted with a solid line. The number of training data is 264 and the number of test data is 12.*

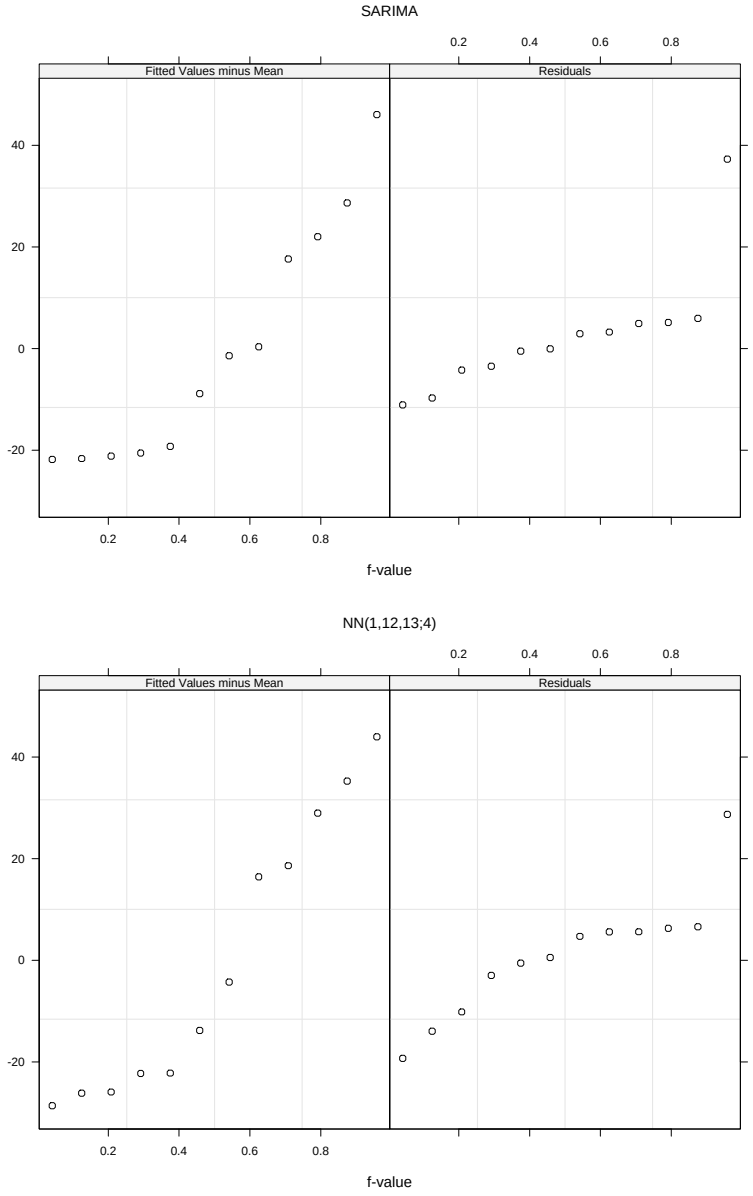


Figure 28: *Residual-fit value spread plots of the 12 water usage test data by SARIMA model and by* $NN(1, 12, 13; 4)$. *The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals. We see that SARIMA model shows one positive outlier more clearly. For the bulk of remaining data, the SARIMA model produced smaller residuals.*

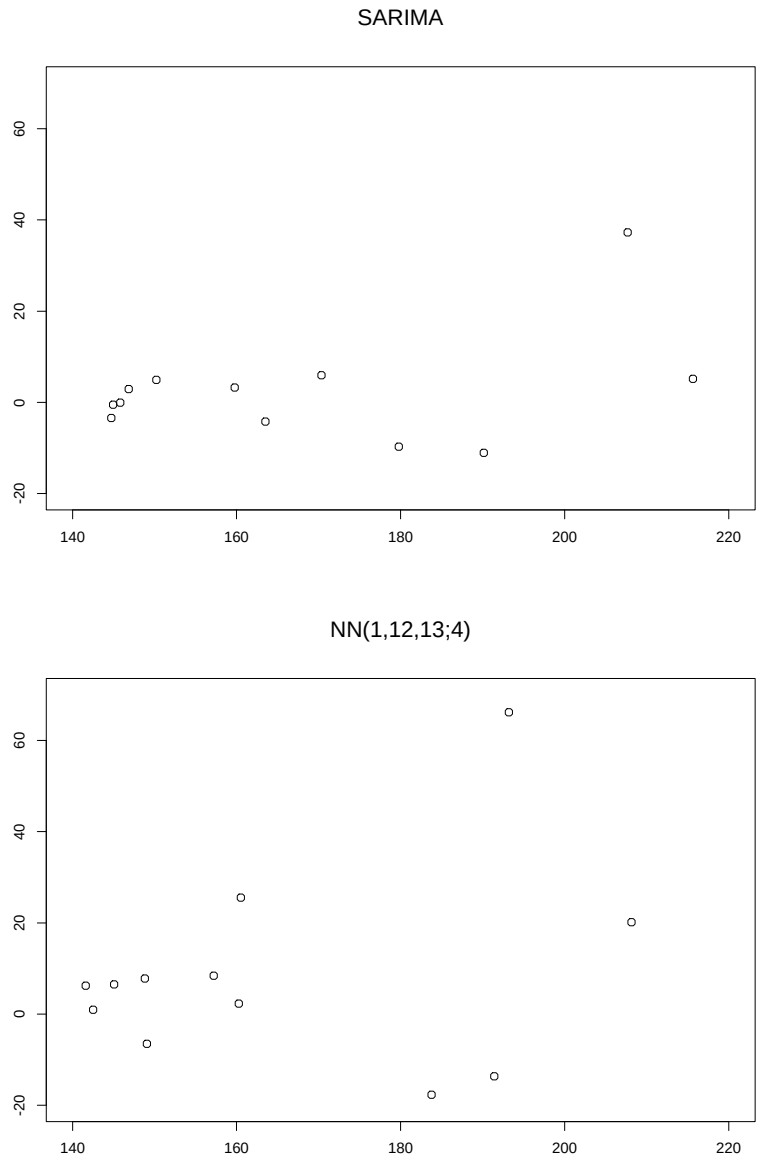


Figure 29: *Tukey mean difference plots of the 12 water usage test data by the SARIMA model and by NN*$(1, 12, 13; 4)$*. The horizontal axis in each plot shows the* $(X+Y)/2$ *and the vertical axis* $Y-X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

4.2.2 Original Domain, $N_1 = 200, N_2 = 76$

The results are given in Table **??**. The best FFNN model, NN$(1, 12, 13; 2)$ produces 13.93 of RMSE value, which is much worse than 7.67 of that produced by the SARIMA model.

Plots of the fitted values by the SARIMA model and those by NN$(1, 12, 13; 2)$ model are set out in Figure **??**.

Figure **??** clearly shows that NN$(1, 12, 13; 2)$ cannot trace the last 24 observations while SARIMA model traces the observation quite well. From these two data dividing schemes and experiments, the following conclusion can be led. The first data dividing scheme makes the training data set include the half of last 24 observations which looks a little extraordinary compared to the observations before. On the contrary to that, the second data dividing scheme does not include the last 24 different trend data for the training data set. Therefore, what FFNN models can do is only reproducing the old trend and cannot treat with the trend which is different from the former trend.

Residual-Fit spread plots for the test data are given in figure **??**. It is shown that more variety is accounted by SARIMA model than by the FFNN model. Tukey mean difference plots for the test data are given in figure **??**. Plot for the NN$(1, 12, 13; 2)$ has more systematic upward trend than that for the SARIMA model, which tells NN$(1, 12, 13; 2)$ fit is not enough for representing the test data.

Now we apply Pitman's test to out-of-sample forecasts for seeing the significance of difference for variances produced by both model. The SARIMA model and NN$(1, 12, 13; 2)$ model produce $r = 0.50$, $\hat{t} = 8.59$ achieved significance level $2\Pr(t > \hat{t}) = 9.87 \times 10^{-13}$. Therefore, the variances for out-of-sample forecast errors. from both models are significantly different. These model produce PMC $\hat{p} = 0.38$ for out-of-sample forecasts with the achieved significance level $\alpha = 0.033$ under the null hypothesis that $p = 0.5$, which tells us that the SARIMA model outperforms the NN$(1, 12, 13; 2)$ model because SARIMA is better $100 - 38 = 62\%$ of the time.

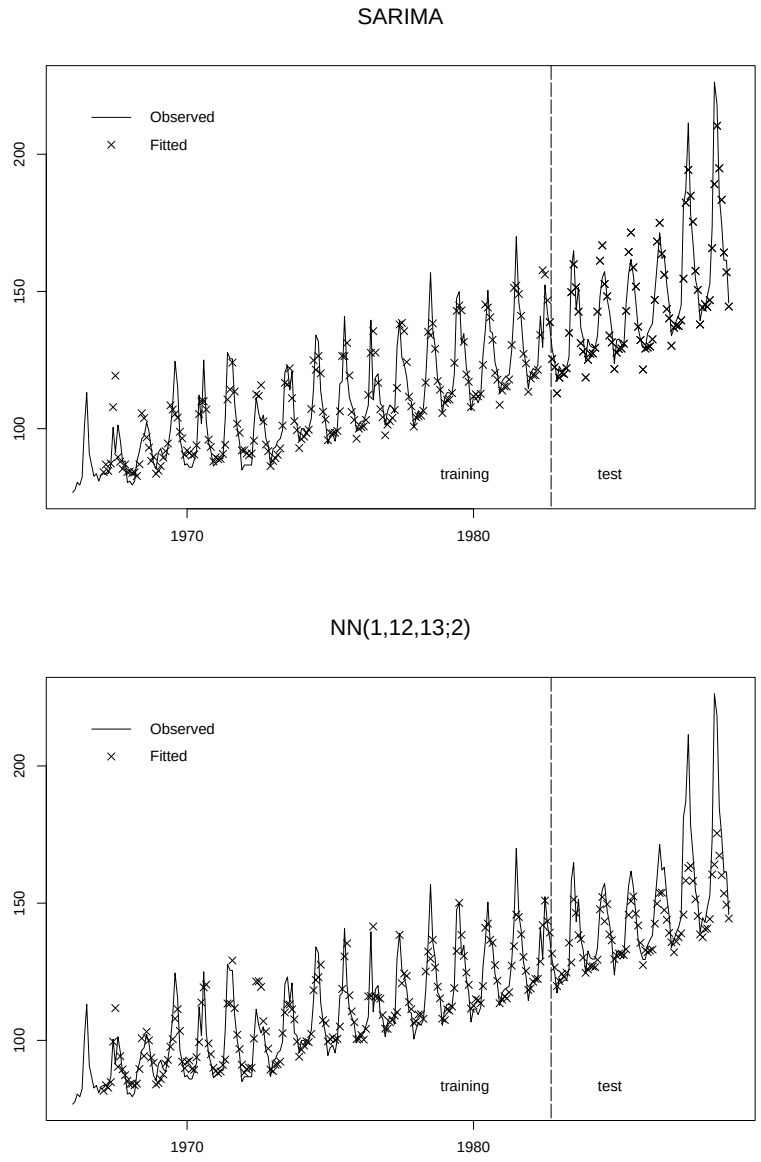


Figure 30: *Fitted values of the water usage by SARIMA model and by* $NN(1, 12, 13; 2)$. *Observations are plotted with a solid line. The number of training data is 200 and the number of test data is 76.*

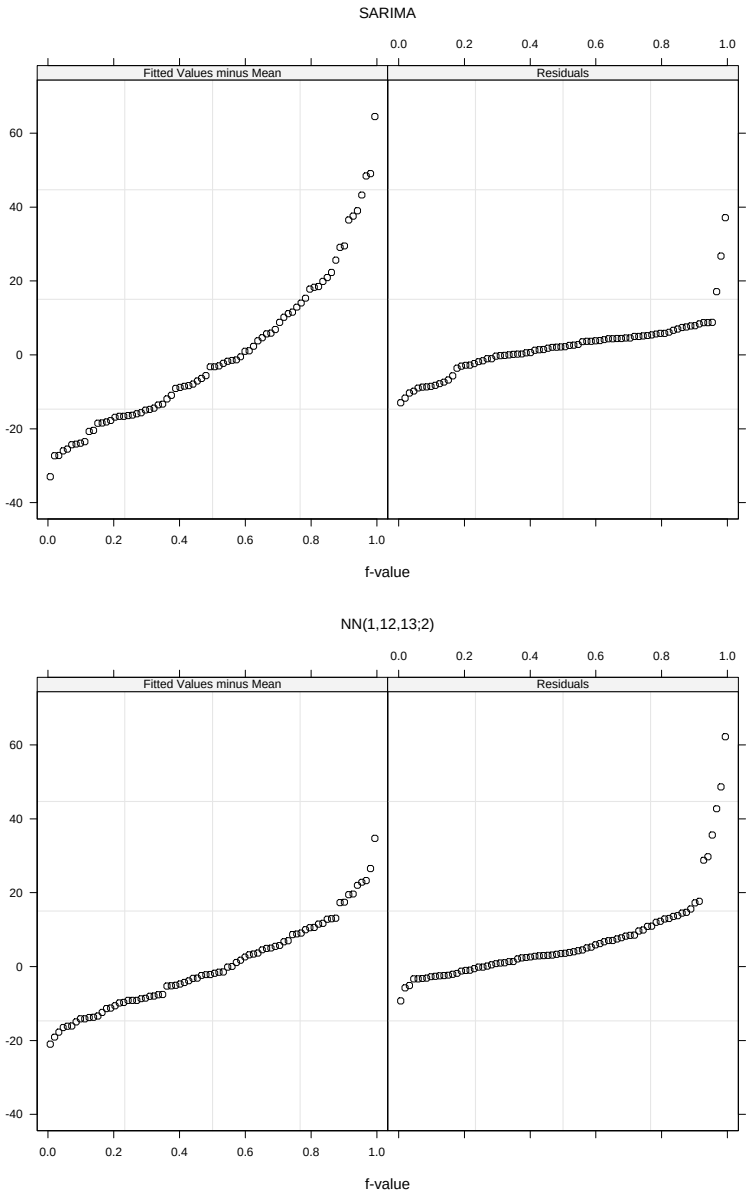


Figure 31: *Residual-fit value spread plots of the 76 water usage test data by SARIMA model and by* $NN(1, 12, 13; 2)$. *The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.*

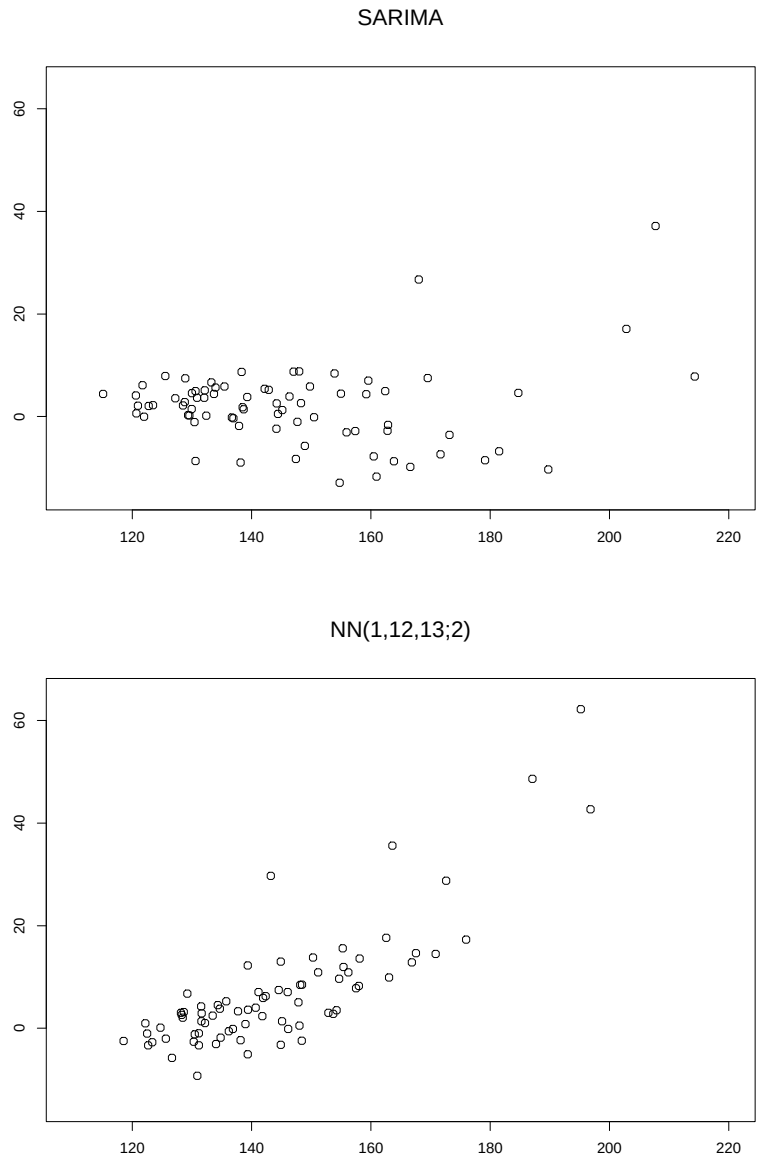


Figure 32: *Tukey mean difference plots of the 76 water usage test data by the SARIMA model and by NN*$(1, 12, 13; 2)$*. The horizontal axis in each plot shows the* $(X+Y)/2$ *and the vertical axis* $Y-X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

FFNN models	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN$(1, 2; 2)$	9.29	20.70
NN$(1, 2 : 4)$	9.34	20.85
NN$(1, 2, 3 : 2)$	9.15	21.40
NN$(1, 2, 3; 4)$	9.15	21.40
NN$(1, 12; 2)$	7.05	13.68
NN$(1, 12; 4)$	7.07	13.60
NN$(1, 2, 12; 2)$	7.03	14.87
NN$(1, 2, 12 : 4)$	7.06	14.00
NN$(1, 2, 12, 13; 2)$	7.04	14.11
NN$(1, 2, 12, 13 : 4)$	7.04	14.07
NN$(1, 12, 13; 2)$	7.04	13.93
NN$(1, 12, 13; 4)$	6.68	16.28
Box-Jenkins model	6.81	7.67

Table 4.23: *Comparison of various* NN *models and Box-Jenkins model. The number of training data is 200 and that of test data is 76.* σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

4.2.3 Transformed Domain, $N_1 = 252, N_2 = 12$

The results are given in Table **??**. In this time the Pitman test for variances declares no statistically significant difference.

$r = 0.0022$, $\hat{t} = 0.15$ and the achieved p-value was 0.88, which indicates once again the agreement is unusually strong.

FFNN models	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN$(1;1)$	1.3×10^{-3}	9.0×10^{-4}
NN$(1:2)$	1.3×10^{-3}	9.0×10^{-4}
NN$(1,2;1)$	1.3×10^{-3}	8.3×10^{-4}
NN$(1,2;2)$	1.3×10^{-3}	8.5×10^{-4}
NN$(1,12;1)$	1.4×10^{-3}	7.42×10^{-4}
NN$(1,12;2)$	1.4×10^{-3}	9.8×10^{-4}
NN$(1,2,12;1)$	1.1×10^{-3}	9.5×10^{-4}
NN$(1,12,13:1)$	1.4×10^{-3}	7.44×10^{-4}
NN$(1,2,12,13:1)$	1.2×10^{-3}	9.3×10^{-4}
Box-Jenkins model	1.2×10^{-3}	8.6×10^{-4}

Table 4.24: *Comparison of various* NN *models with the corresponding values for the water usage data. The data is seasonally differenced and power transformed. 252 training data and 12 test data are used.* σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.* $\sigma_P = 7.42 \times 10^{-4}$ *corresponds to 10.58 after back-transformation, which is less than the value 11.80 obtained by* $NN(1,12,13;4)$ *without transformation and difference.*

4.2.4 Transformed Domain, $N_1 = 188, N_2 = 76$

Some of the FFNN models perform better than ARMA model in Table **??**. The model which produces the least RMSE value 7.42×10^{-3} is NN$(1, 12; 1)$. This RMSE value corresponds to 10.58 when the back-transformation is exerted, and this is less than the value 11.80 without transformation and difference of data. Fitted values of the SARIMA model and the NN$(1, 12; 1)$ are plotted in Figure **??**, but the fitted values are not consistent with observed values, which indicates that NN$(1, 2; 2)$ should rather be used. Fitted values of NN$(1, 2; 2)$ are plotted in Figure **??**.

Residual-fit spread plots for the test data are given in figure **??**. It is shown that more variability is accounted by ARMA model than by the FFNN model. Tukey mean difference plots in **??** shows that there is upward trend of residuals for NN$(1, 2; 2)$ model.

The SARIMA model and NN$(1, 2; 2)$ model produce $r = 0.20$, $\hat{t} = 4.3$ and achieved significance level $2\Pr(t > \hat{t}) = 5.1 \times 10^{-5}$ for the Pitman's test. This indicates the variances within samples from both models are significantly different. The SARIMA model and NN$(1, 2; 2)$ model produce PMC value $\hat{p} = 0.58$ for out-of-sample forecasts with a achieved significance level $2\Pr(z > \hat{z}) = 0.16$, which indicates that SARIMA model and the NN$(1; 2)$ model has the equally close residuals to the observed values. This means, due to the transformation and difference, forecasting accuracy of the FFNN model is improved.

In Table **??**, ARMA model produces better RMSE value than any other FFNN model does, even though transformation and difference of the data is exerted. However if 8.7×10^{-3} of RMSE, which is produced by NN$(1, 2, 12; 1)$, is back transformed, this valued corresponds to 8.61. This is much better than 13.93 without transformation and difference of data. Considering this remarkable enhancement, it can be concluded that the transformation and difference should be examined in any cases.

Residual-fit value spread plots for the test data are given in figure **??**. It is shown that more variety is accounted by ARMA model than by the FFNN model. Tukey mean difference plots for the test data are given in figure **??**. The plot of

NN$(1, 2, 12; 1)$ has a systematic trend again.

The SARIMA model and NN$(1, 2; 2)$ model produce $r = 0.0021$, $\hat{t} = 0.15$ and achieved significance level $2\Pr(t > \hat{t}) = 0.89$ for the Pitman's test. This indicates the variances within samples from both models are not significantly different. The SARIMA model and NN$(1, 2; 2)$ model produce PMC value $\hat{p} = 0.58$ for out-of-sample forecasts with a achieved significance level $2\Pr(z > \hat{z}) = 0.16$, which indicates that the NN$(1; 2)$ model has as performance as the SARIMA model. This result also confirms the requirement of transformation.

FFNN models	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN$(1; 1)$	1.4×10^{-3}	9.4×10^{-4}
NN$(1 : 2)$	1.4×10^{-3}	9.4×10^{-4}
NN$(1, 2; 1)$	1.4×10^{-3}	9.4×10^{-4}
NN$(1, 2; 2)$	1.5×10^{-3}	9.7×10^{-4}
NN$(1, 12; 1)$	1.5×10^{-3}	9.7×10^{-4}
NN$(1, 2, 12; 1)$	1.3×10^{-3}	8.7×10^{-4}
NN$(1, 12, 13 : 1)$	1.5×10^{-3}	9.7×10^{-4}
NN$(1, 2, 12, 13 : 1)$	1.3×10^{-3}	9.0×10^{-4}
Box-Jenkins model	1.3×10^{-3}	7.8×10^{-4}

Table 4.25: *Comparison of various* NN *models with the corresponding values for the water usage data. The data is seasonally differenced and power transformed. 188 training data and 76 test data.* σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.* σ_P=8.7×10^{-3} *obtained by NN*$(1, 2, 12; 1)$ *corresponds to 8.61 and this is better than 13.60 of least RMSE value by NN*$(1, 12; 4)$ *without difference and transformation.*

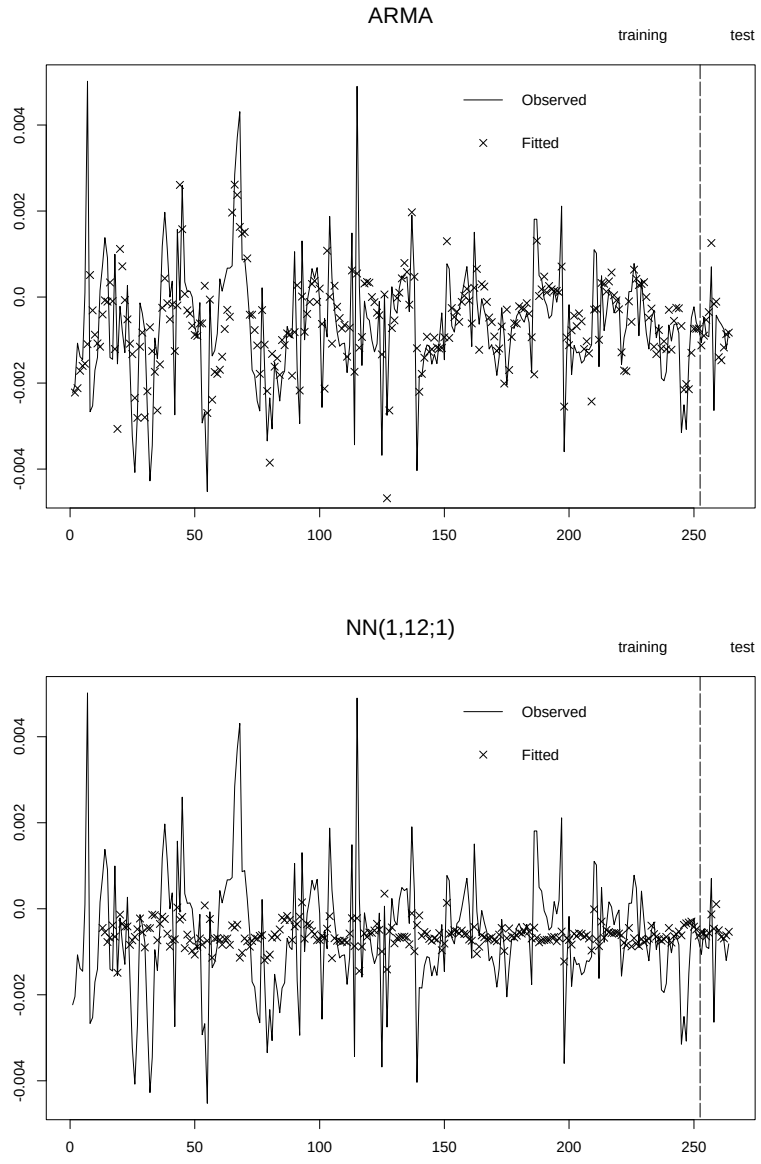


Figure 33: *Fitted values for the water usage data by the ARMA model and by* $NN(1, 12; 2)$. *The data is seasonally differenced and power transformed. Observations are plotted with a solid line. The number of training data is 252 and the number of test data is 12.*

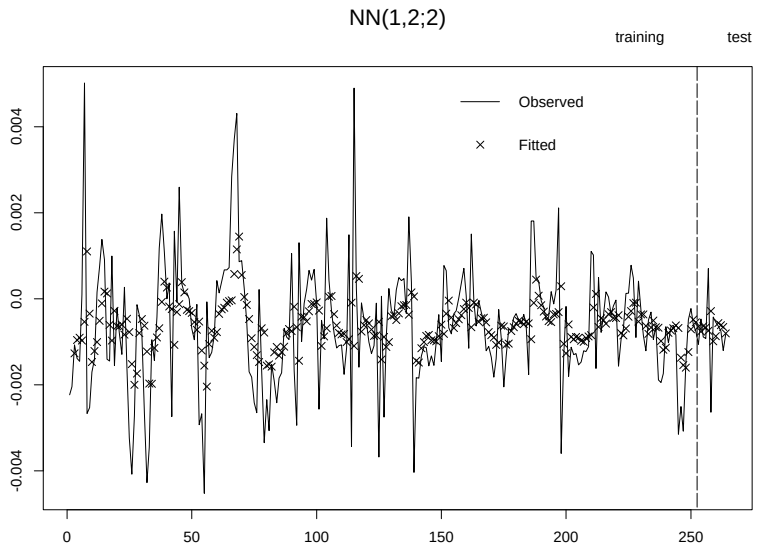


Figure 34: *Fitted values for the water usage data by* $NN(1, 2; 2)$*. The data is seasonally differenced and power transformed. Observations are plotted with a solid line. The number of training data is 252 and the number of test data is 12.*

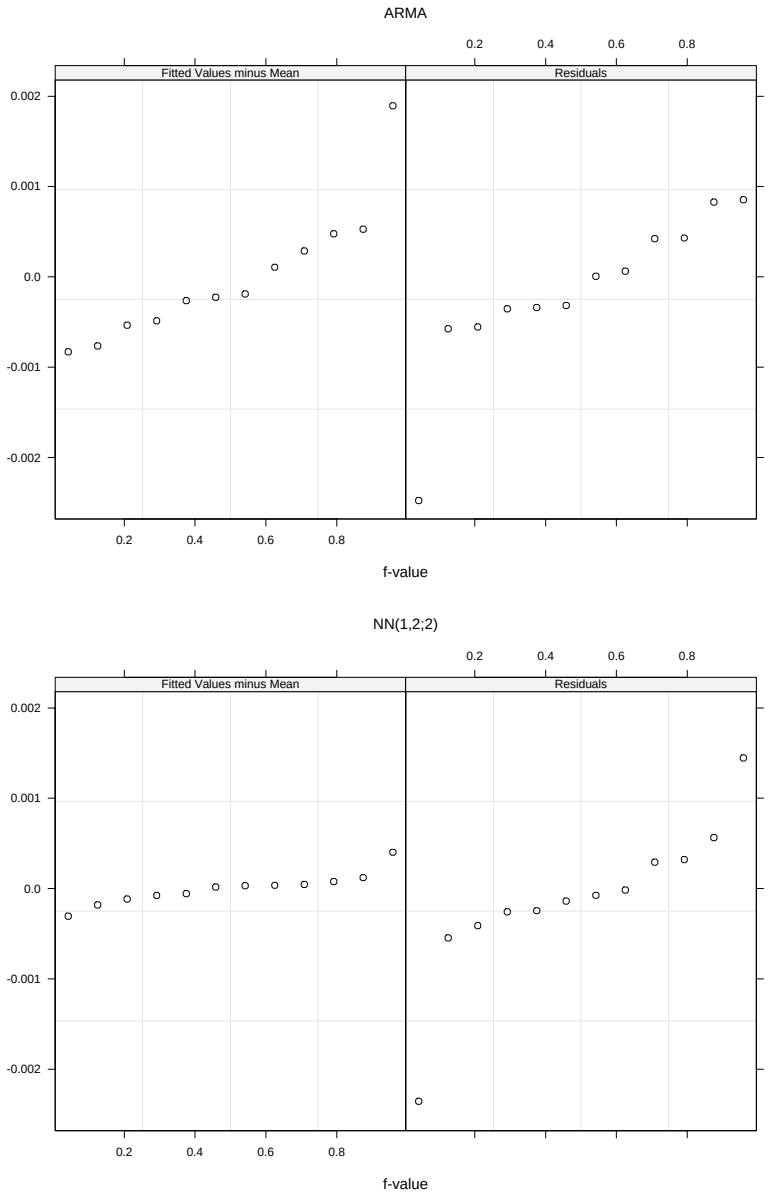


Figure 35: *Residual-fit spread plots of the transformed and seasonally differenced 12 water usage test data by ARMA and by NN*$(1, 2; 2)$*. The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.*

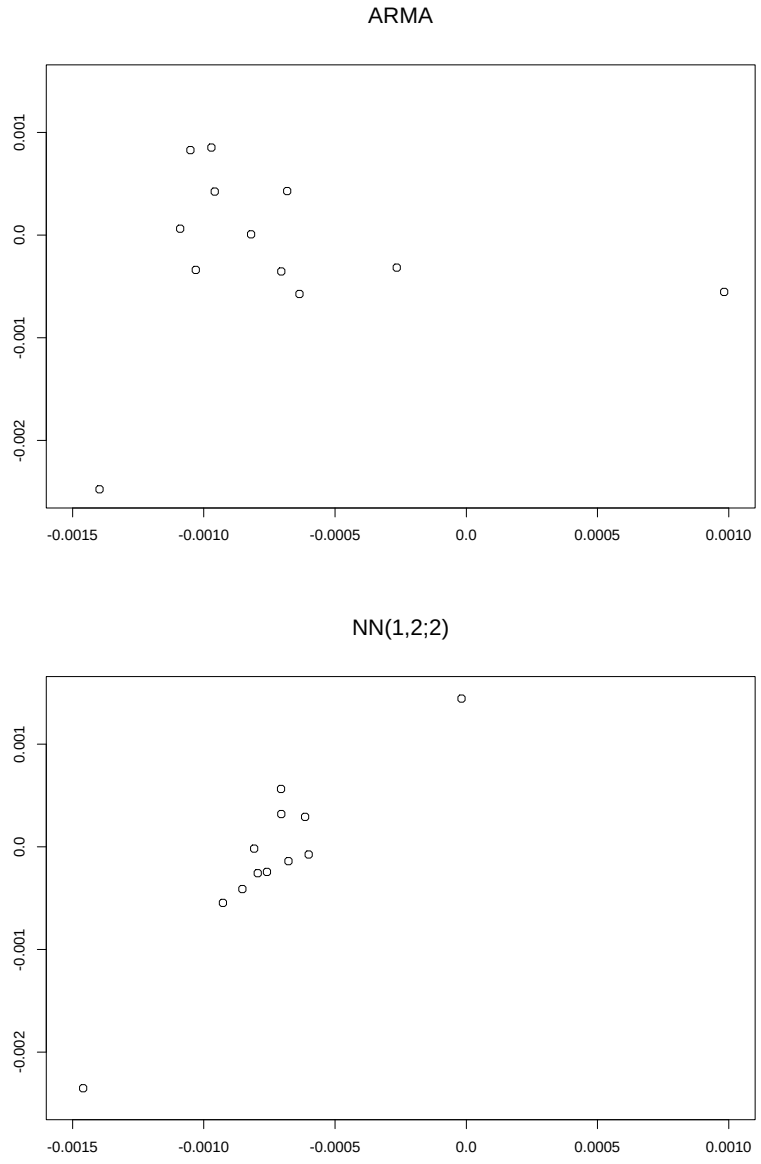


Figure 36: *Tukey mean difference plots of the transformed and differenced 12 water usage test data by ARMA model and by NN*$(1, 2; 2)$*. The horizontal axis in each plot shows the* $(X + Y)/2$ *and the vertical axis* $Y - X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

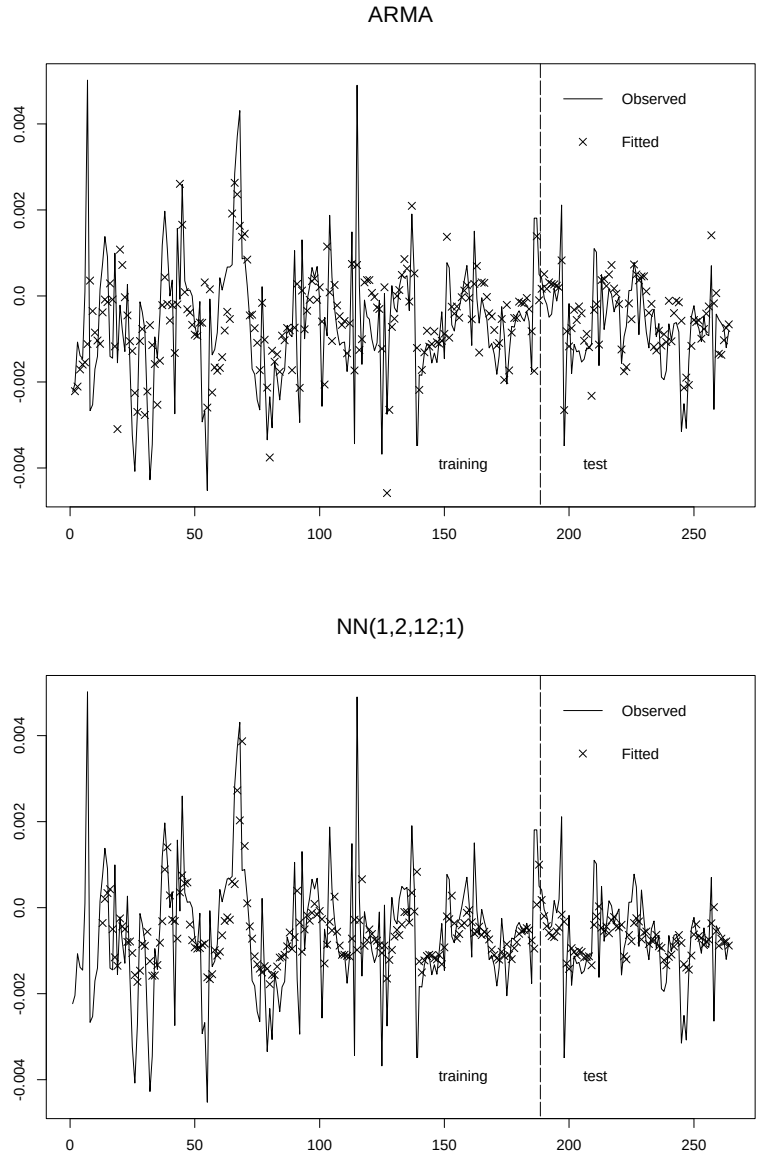


Figure 37: *Fitted values for the differenced and transformed water usage data by the ARMA model and by NN*$(1, 2, 12; 2)$*. The data is seasonally differenced and power transformed. Observations are plotted with a solid line. The number of training data is 188 and the number of test data is 76.*

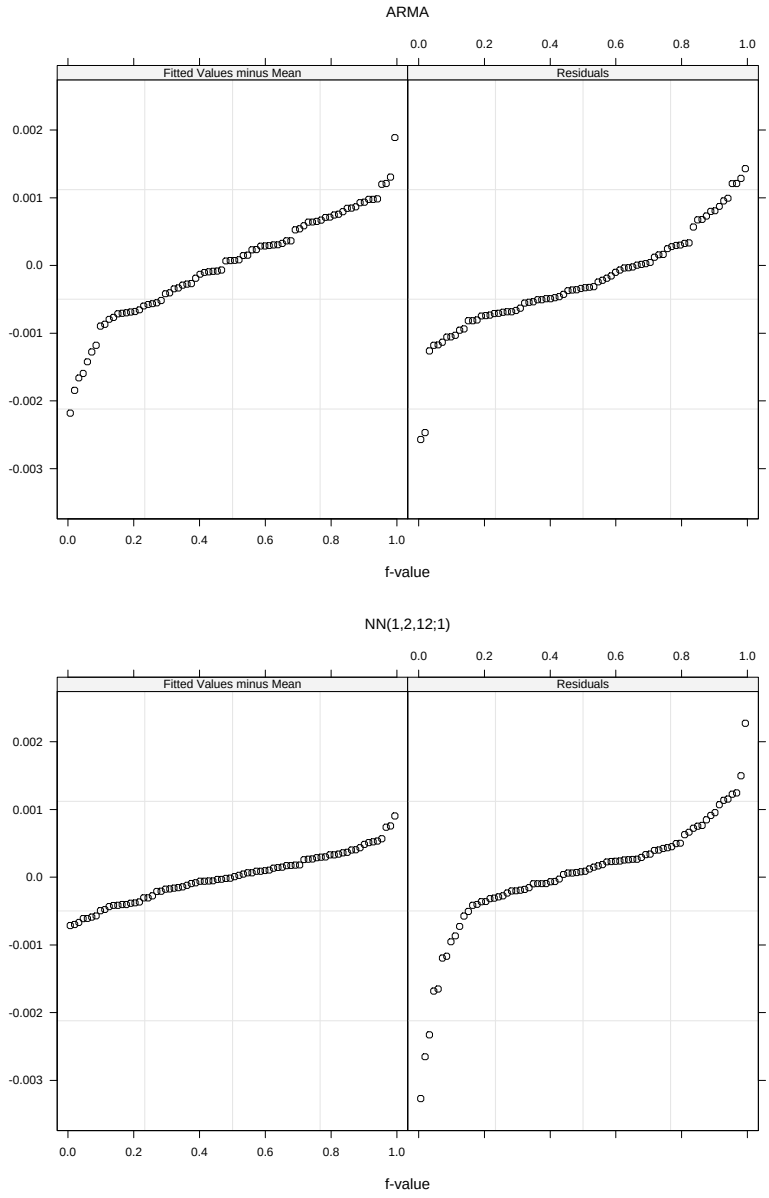


Figure 38: *Residual-fit value spread plots of the transformed and differenced 76 water usage test data by ARMA and by* $NN(1, 2, 12; 1)$. *The data is seasonally differenced and power transformed.*

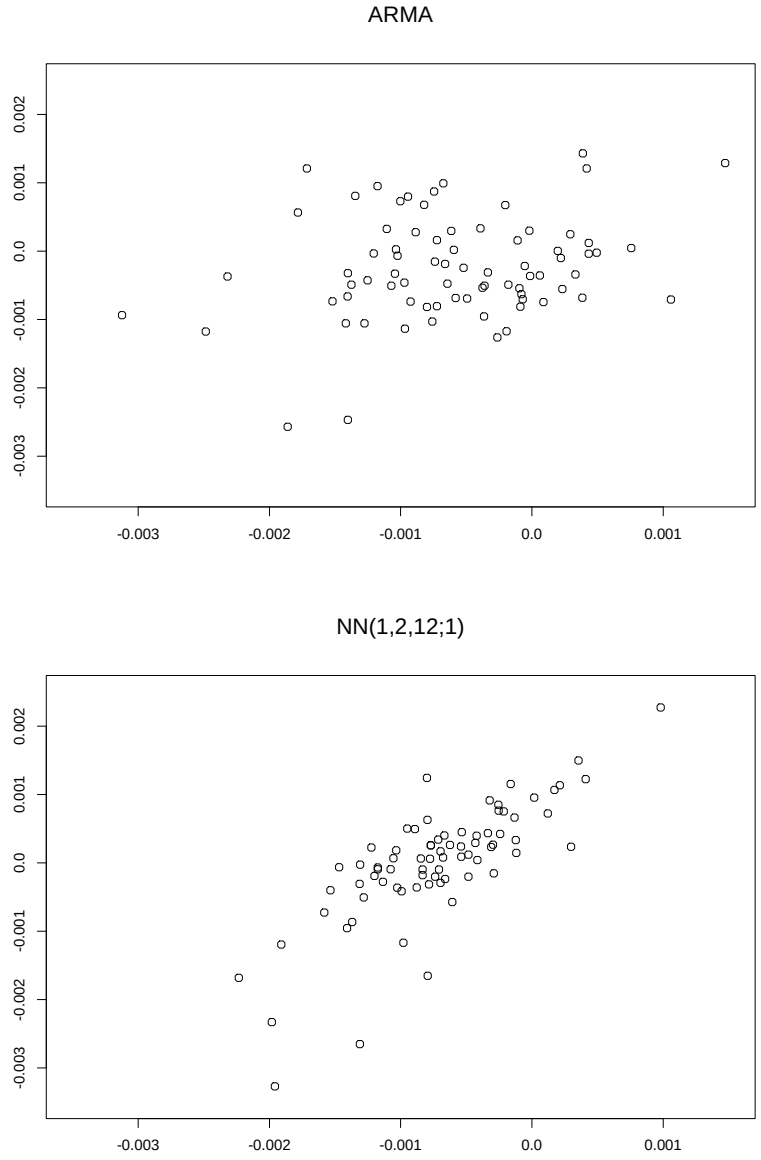


Figure 39: *Tukey mean difference plots of the transformed and differenced 76 water usage test data by ARMA model and by NN*$(1, 2, 12; 1)$*. The data is seasonally differenced and power transformed. The horizontal axis in each plot shows the* $(X + Y)/2$ *and the vertical axis* $Y - X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

5 Markovian Dependence

5.1 Introduction

Stern (1996) compared the forecasting abilities of ARMA models with those of FFNN. In particular Stern (1996) made comparisons between various types of second-order autoregressive models, AR(2) and the FFNN model. Stern (1996) found that FFNN models produced acceptable forecasts provided that the signal-to-noise ratio in the AR(2) model was high. In general for a stationary time series X_t with variance ${\sigma_x}^2$ and innovation variance ${\sigma_e}^2$, the signal-to-noise ratio is $\eta = \sigma_x^2/\sigma_e^2$. For the AR(2) it can be shown that,

$$\eta = \frac{1-\phi_2}{\left((\phi_2-1)^2 - {\phi_1}^2\right)\,(1+\phi_2)}. \tag{5.5}$$

For the data with low signal-to-noise ratio, so called over-training can happen easily. It should be noticed that the airline data and the sales data were non-stationary process and the idea of signal-to-noise ratio is only defined for stationary processes. Therefore, it is necessary to transform and difference the non-stationary data to make the series stationary.

In this chapter, we critically review Stern (1996) and show examples where the FFNN approach outperforms the AR model as well as a Markovian example where neither modeling approach works.

Stern (1996) used FFNN models with 5 inputs for AR(2) Data. However, in a natural sense it makes more sense to use NN$(1, 2; 1)$ or NN$(1, 2; 2)$ as an appropriate model instead of his FFNN model with lags at 1,2,3,4,5 because in an AR(2) model, the present value is determined only by previous two values. In practice, it may be useful to look at the autocorrelation function to assist in determining the number of lags to use in an FFNN model.

We perform the AR(2) simulation as in Stern (1996). In these simulations, σ_x is set to 1. Innovation variance is obtained after solving Equation **??**. The AR(2)

models used here and by Stern to generate AR(2) data sets are given in Table **??**.

Stern (1996) observed that models with small signal-to-noise ratio have smaller residual variances in the training set than their true innovation variance. This indicates that the over-training apparently happened and it is natural that those models produce poor forecasts. Table **??** shows the comparison between Stern's NN$(1-5;2)$ model and our own NN$(1,2;1)$, NN$(1,2,2)$, and NN$(1-5,2)$ model. By and large, we could see Stern's model and our NN$(1-5;2)$ produces worse forecasts from the table. For convenience, the predicted value is plotted in Figure **??** to see the difference of the performances clearly between the FFNN models with 2 inputs and FFNN models with 5 inputs. The variance of NN$(1,2;1)$ and NN$(1,2;2)$ are consistent with the variance of the true AR(2) model compared to the variance of NN$(1-5,2)$. Figure **??** shows the test data of AR(2) time series data and fitted values of NN$(1,2;1)$ model. They are quite consistent with each other and as a conclusion, NN model can do much better even though the signal-to-noise ratio is as low, as 1.69 in Figure **??**, provided that care is taken in selecting the lags used.

Model	ϕ (1)	ϕ (2)	$\sigma_{\mathbf{e}}{}^{\mathbf{2}}$	η
a	0.7	-0.49	0.7695	1.30
b	0.9	-0.81	0.5088	1.97
c	0.9	-0.97	0.2163	4.62
d	1.4	-0.99	0.1002	9.98

Table 5.26: *AR(2) models used here to generate data sets and corresponding η.*

σ	$\mathbf{NN}(1,2;1)$		$\mathbf{NN}(1,2;2)$		$\mathbf{NN}(1-5;2)$		**Stern**	
	σ_F	σ_P	σ_F	σ_P	σ_F	σ_P	σ_P	σ_F
0.7695	0.73	0.74	0.68	0.76	0.57	0.89	0.57	1.02
0.5088	0.54	0.47	0.51	0.52	0.45	0.57	0.42	0.63
0.2163	0.20	0.21	0.18	0.21	0.15	0.23	0.18	0.30
0.1002	0.09	0.10	0.09	0.10	0.08	0.11	0.84	0.13

Table 5.27: *RMSE of several FFNN models for the different AR(2) time series data. σ is a standard deviation of the noise. The column Stern show the results by NN$(1-5;2)$ given in Stern (1996). σ_F is RMSE of the training data and σ_P is RMSE of the test data.*

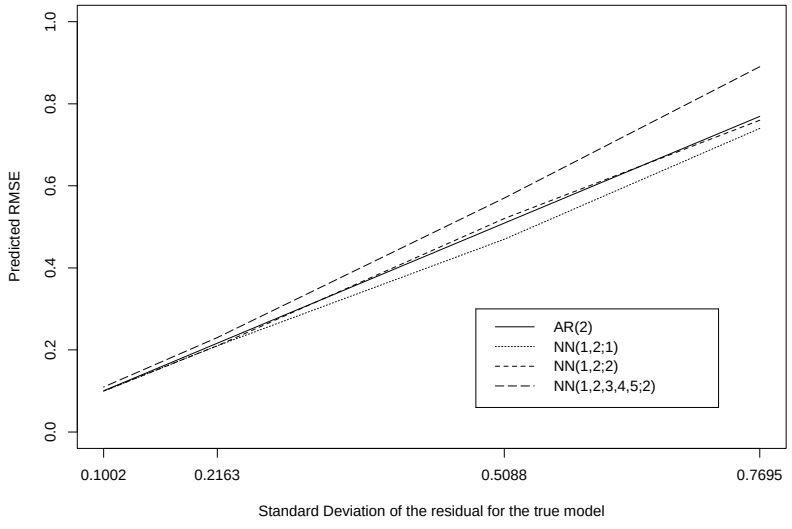
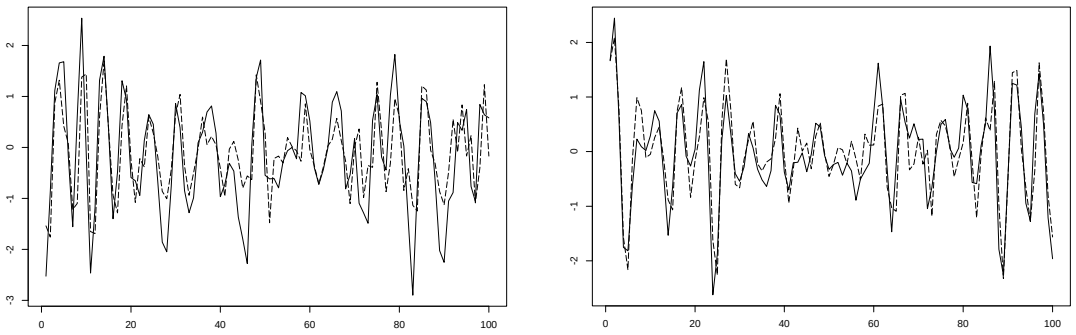


Figure 40: *RMSE of the fitted values of the FFNN model for the varying AR(2) time series data.*

a) $X_t = 0.7X_{t-1} + 0.49X_{t-2} + \mathrm{N}(0, 0.7695^2)$ b) $X_t = 0.9X_{t-1} - 0.81X_{t-2} + \mathrm{N}(0, 0.5088^2)$



c) $X_t = 0.9X_{t-1} - 0.97X_{t-2} + \mathrm{N}(0, 0.2163^2)$ d) $X_t = 1.4X_{t-1} - 0.99X_{t-2} + \mathrm{N}(0, 0.1002^2)$

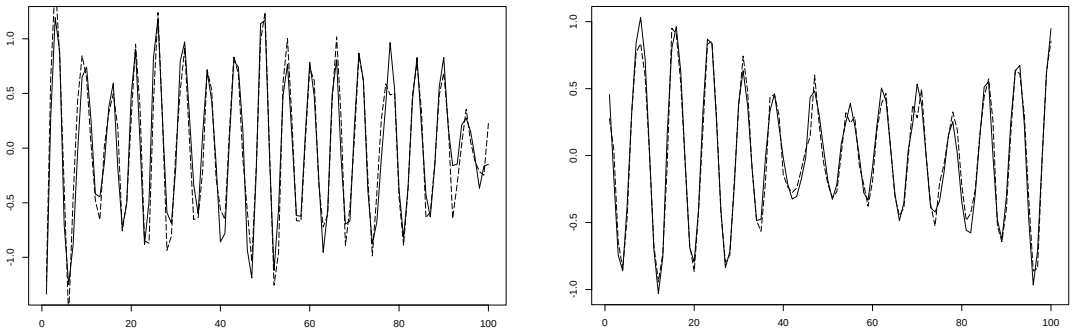


Figure 41: *AR(2) time series data to test the Neural Networks. Dividing scheme is E. Solid lines are time series; dashed lines are fitted values by the neural network.*

5.2 AR(1) Simulation Experiments

The AR(1) the model can be expressed by

$$X_t = \phi X_{t-1} + e_t,$$

where e_t are independent identically distributed normal deviates with variance ${\sigma_e}^2$. In this case, the variance of X_t can be written as

$${\sigma_x}^2 = \frac{\sigma_e^2}{1-\phi^2}. \tag{5.6}$$

where we assume $|\phi| < 1$. The signal-to-noise ratio is given by calculating σ_x^2/σ_e^2 is thus determined by ϕ. The AR(1) models which are examined here and corresponding are shown in Table **??**. The signal to the noise ratio ranges from 1.1 to 5.3 in those examples which is comparatively smaller than those in the previous section.

In this section we compare our conclusions with Stern (1996) when an AR(1) model is used instead of AR(2). The AR(1) model is a form of Markovian dependence. The behaviour of AR(1) series is less complex than AR(2). We concentrate on comparatively smaller signal to noise ratio than those in previous section. For, we already know NN model was valid for the data with smallest signal-to-noise ratio in the previous section as long as the model is appropriate. Further more, we examine the several lengths of test data sets to test how much length of data affects on the forecasting accuracy. Table **??** shows a data dividing scheme in each AR(1) model.

It can be easily found that RMSE of each FFNN model is varying for the small data set and they are quite consistent for the large data set. For small data sets, NN$(1;1)$ model produces better forecasts than other models. It is natural because the true model is AR(1) model and a present value only depends on the previous value. Therefore, we could say that it is important that we choose an appropriate model for the small data set.

Figure **??** illustrates the comparison of the test data of AR(1) models and forecasts by the NN$(1;1)$ model. Here, we use the scheme E which divides 200 data into 100 data as a training data set and 100 as a test data set similar to the scheme Stern (1996) used. Since the signal-to-noise ratios are lower in the AR(1) model than the

AR(2) and it looks like difficult that fitted values trace perfectly the test data as the NN$(1, 2; 1)$ model did for AR(2) data. Especially, models with negative coefficients look more chaotic, and fitted values are quite inconsistent with observed values.

Figure **??** shows residual fitted values spread plots for each AR(1) test data in Figure **??**. As the coefficient ϕ gets bigger, much variability gets to be accounted by the FFNN model. In other words, as the signal-to-noise ratio gets bigger, Fitted values by the FFNN model becomes better.

Figure **??** shows Tukey mean difference plots for each AR(1) test data in Figure **??**. As the coefficient ϕ gets bigger, the shape gets less systematic, which is more desirable.

Dividing Scheme	number of training data	number of test data
A	25	25
B	75	25
C	50	50
D	175	25
E	100	100
F	475	25
G	250	250

Table 5.28: *Data dividing scheme for each AR(1) model.*

ϕ	signal-to-noise ratio
-0.9	5.3
-0.6	1.6
-0.3	1.1
0.3	1.1
0.6	1.6
0.9	5.3

Table 5.29: *Coefficients of AR(1) model and corresponding signal-to-noise ratio.*

The results for each AR(1) time series data is given in from Table **??** to Table **??**.

Scheme	**NN**$(1;1)$		**NN**$(1;2)$		**NN**$(1;3)$		**NN**$(1,2;2)$		**NN**$(1,2,3;2)$	
	σ_F	σ_P	σ_F	σ_P	σ_F	σ_P	σ_P	σ_F	σ_P	σ_F
A	0.54	0.36	0.51	0.40	0.38	0.41	0.40	0.52	0.31	0.52
B	0.42	0.50	0.42	0.49	0.40	0.53	0.40	0.54	0.38	0.46
C	0.39	0.50	0.39	0.52	0.36	5.10	0.35	0.52	0.31	0.56
D	0.43	0.45	0.42	0.45	0.41	0.44	0.42	0.45	0.41	0.45
E	0.46	0.41	0.45	0.42	0.44	0.42	0.43	0.43	0.42	0.44
F	0.45	0.41	0.45	0.41	0.44	0.41	0.44	0.41	0.44	0.40
G	0.43	0.47	0.42	0.47	0.42	0.48	0.42	0.48	0.42	0.49

Table 5.30: *RMSE of various FFNN models for the AR(1) data generated by* $X_t = -0.9X_t + \mathrm{N}(0, 0.19)$. σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

Scheme	**NN**$(1;1)$		**NN**$(1;2)$		**NN**$(1;3)$		**NN**$(1,2;2)$		**NN**$(1,2,3;2)$	
	σ_F	σ_P	σ_F	σ_P	σ_F	σ_P	σ_P	σ_F	σ_P	σ_F
A	0.57	0.90	0.49	1.02	0.31	1.04	0.41	0.98	0.27	5.46
B	0.75	0.71	0.72	0.87	0.70	0.96	0.69	0.74	0.63	0.87
C	0.76	0.85	0.67	0.82	0.59	1.42	0.62	0.93	0.59	0.98
D	0.84	0.83	0.82	0.84	0.81	0.80	0.81	0.83	0.78	0.86
E	0.82	0.88	0.79	0.91	0.77	0.94	0.76	0.96	0.74	1.00
F	0.84	0.83	0.84	0.80	0.83	0.82	0.84	0.86	0.84	0.85
G	0.84	0.85	0.84	0.86	0.83	0.87	0.83	0.86	0.82	0.91

Table 5.31: *RMSE of various FFNN models for the AR(1) data generated by* $X_t = -0.6X_t + \mathrm{N}(0, 0.64)$. σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

Scheme	**NN**$(1;1)$		**NN**$(1;2)$		**NN**$(1;3)$		**NN**$(1,2;2)$		**NN**$(1,2,3;2)$	
	σ_F	σ_P	σ_F	σ_P	σ_F	σ_P	σ_P	σ_F	σ_P	σ_F
A	0.73	1.10	0.66	1.20	0.62	1.24	0.46	1.09	0.42	1.66
B	0.94	0.90	0.90	1.18	0.84	1.21	0.86	1.17	0.82	1.23
C	0.87	1.16	0.84	1.29	0.80	1.19	0.76	1.21	0.71	1.39
D	0.96	1.14	0.96	1.14	0.95	1.13	0.92	1.25	0.89	4.60
E	0.97	1.01	0.94	1.26	0.92	1.21	0.88	1.13	0.85	1.22
F	0.99	1.11	0.98	1.11	0.98	1.04	0.97	1.22	0.96	1.15
G	0.98	1.03	0.96	1.04	0.94	1.04	0.96	1.07	0.91	1.10

Table 5.32: *RMSE of various FFNN models for the AR(1) data generated by* $X_t = -0.3X_t + \mathrm{N}(0, 0.91)$. σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

Scheme	**NN**$(1;1)$		**NN**$(1;2)$		**NN**$(1;3)$		**NN**$(1,2;2)$		**NN**$(1,2,3;2)$	
	σ_F	σ_P	σ_F	σ_P	σ_F	σ_P	σ_P	σ_F	σ_P	σ_F
A	0.89	1.17	0.72	1.55	0.65	1.46	0.50	1.11	0.41	1.32
B	0.98	1.10	0.95	1.23	0.88	1.11	0.92	1.05	0.86	1.22
C	1.00	1.06	0.92	1.13	0.83	1.68	0.90	1.46	0.78	1.31
D	0.93	1.07	0.91	1.16	0.90	1.17	0.89	1.10	0.85	1.15
E	0.93	0.97	0.91	0.99	0.89	1.00	0.86	4.82	0.85	1.07
F	0.94	0.86	0.93	0.86	0.93	0.84	0.93	0.83	0.91	0.88
G	0.90	0.99	0.89	0.97	0.88	0.98	0.87	1.00	0.85	1.06

Table 5.33: *RMSE of various FFNN models for the AR(1) data generated by* $X_t = 0.3X_t + \mathrm{N}(0, 0.91)$. σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

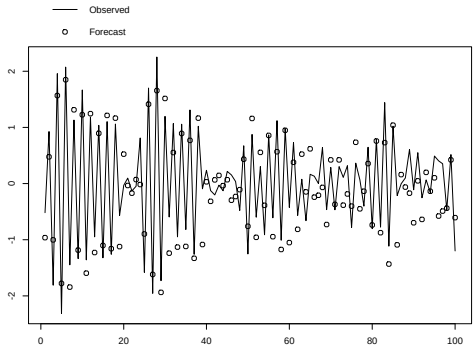
Scheme	**NN**$(1;1)$		**NN**$(1;2)$		**NN**$(1;3)$		**NN**$(1,2;2)$		**NN**$(1,2,3;2)$	
	σ_F	σ_P	σ_F	σ_P	σ_F	σ_P	σ_P	σ_F	σ_P	σ_F
A	0.66	0.89	0.60	0.91	0.55	1.81	0.49	1.42	0.46	1.15
B	0.84	0.80	0.82	0.73	0.79	0.77	0.75	0.79	0.73	0.83
C	0.76	0.94	0.70	1.03	0.65	1.00	0.66	1.01	0.64	1.04
D	0.80	0.77	0.77	0.80	0.76	0.80	0.76	0.76	0.75	0.89
E	0.75	0.89	0.73	0.87	0.71	10.30	0.70	0.87	0.69	0.95
F	0.79	0.77	0.78	0.78	0.78	0.80	0.77	0.77	0.77	0.78
G	0.76	0.81	0.75	0.83	0.73	0.86	0.73	0.86	0.73	0.86

Table 5.34: *RMSE of various FFNN models for the AR(1) data generated by* $X_t = 0.6X_t + \mathrm{N}(0, 0.64)$. σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

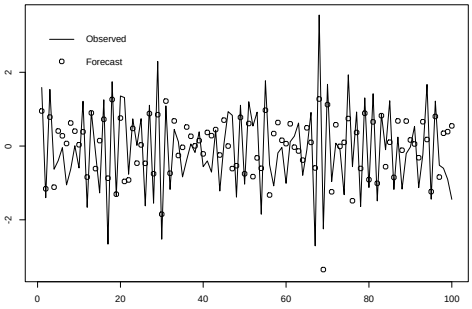
Scheme	**NN**$(1;1)$		**NN**$(1;2)$		**NN**$(1;3)$		**NN**$(1,2;2)$		**NN**$(1,2,3;2)$	
	σ_F	σ_P	σ_F	σ_P	σ_F	σ_P	σ_P	σ_F	σ_P	σ_F
A	0.34	0.46	0.25	0.61	0.23	0.63	0.19	1.52	0.20	0.75
B	0.43	0.35	0.42	0.39	0.39	0.43	0.40	0.47	0.35	0.41
C	0.42	0.40	0.40	0.48	0.36	0.56	0.35	0.62	0.27	0.61
D	0.36	0.38	0.35	0.37	0.35	0.37	0.36	0.37	0.35	0.36
E	0.35	0.38	0.33	0.41	0.32	0.43	0.34	0.41	0.33	0.39
F	0.44	0.39	0.44	0.39	0.43	0.39	0.43	0.40	0.43	0.40
G	0.44	0.44	0.43	0.44	0.43	0.46	0.43	0.45	0.42	0.46

Table 5.35: *RMSE of several FFNN models for the AR(1) model* $X_t = 0.5X_t + \mathrm{N}(0, 0.19)$. σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

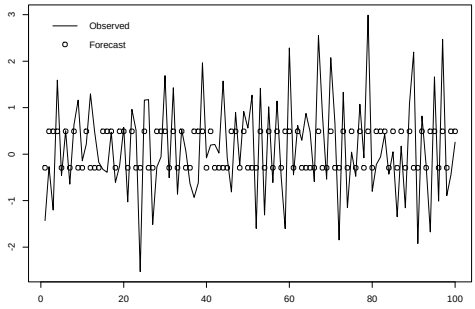
a) $\phi = -0.9$



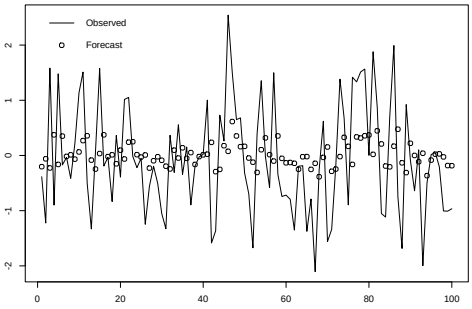
b) $\phi = -0.6$



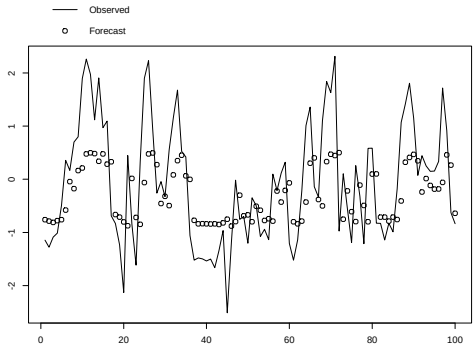
c) $\phi = -0.3$



d) $\phi = 0.3$



e) $\phi = 0.6$



f) $\phi = 0.9$

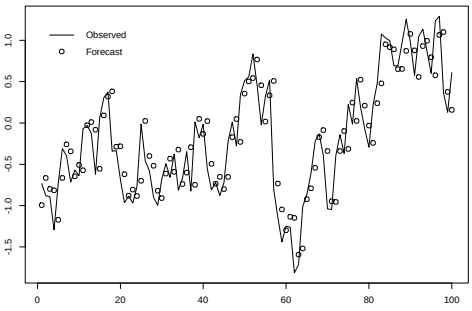
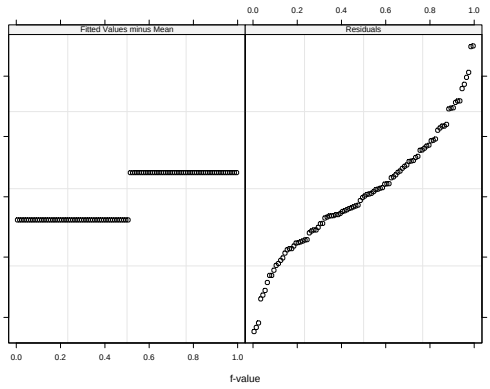
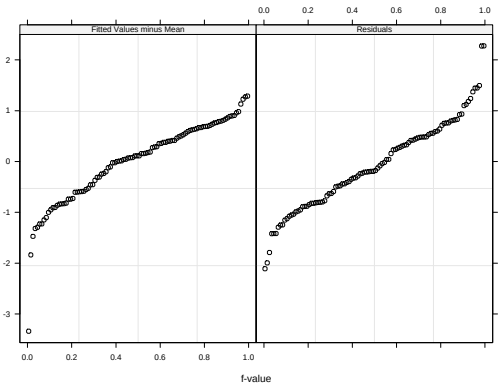


Figure 42: *AR(1) time series data. Dividing scheme is E. Solid line is observed data and points are fitted values by the FFNN models for the test data.*

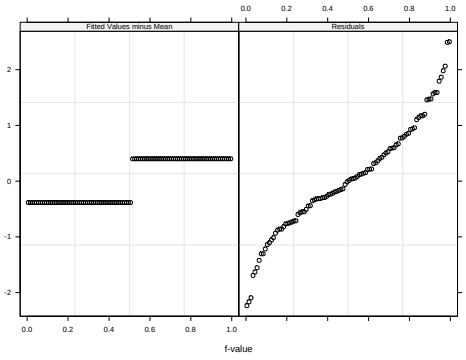
a) $\phi = -0.9$



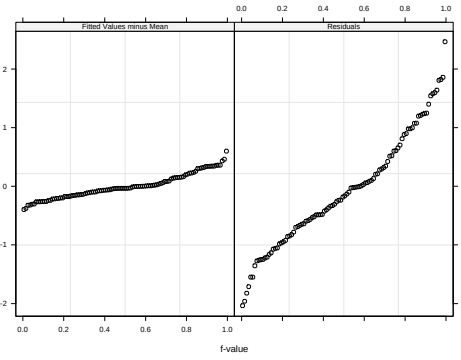
b) $\phi = -0.6$



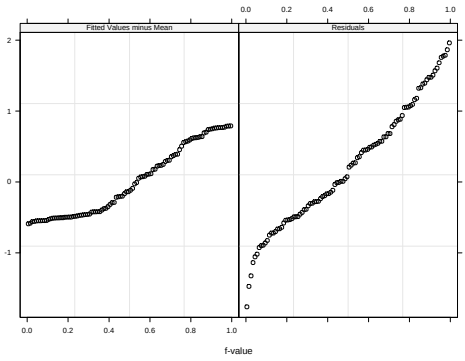
c) $\phi = -0.3$



d) $\phi = 0.3$



e) $\phi = 0.6$



f) $\phi = 0.9$

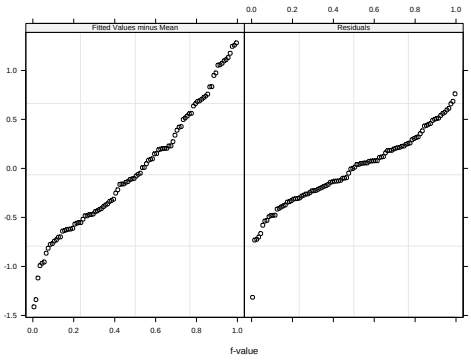
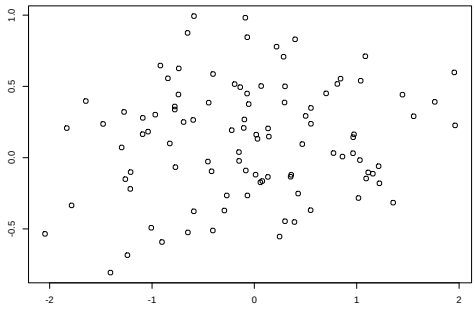
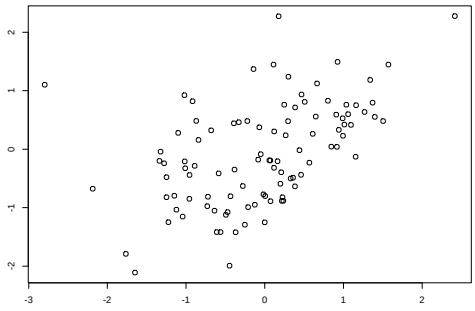


Figure 43: *Residual fitted value spread plots for test data of the AR(1) time series data sets.*

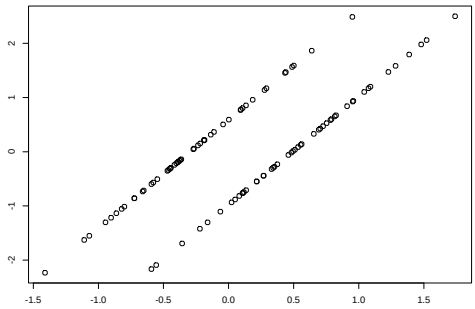
a) $\phi = -0.9$



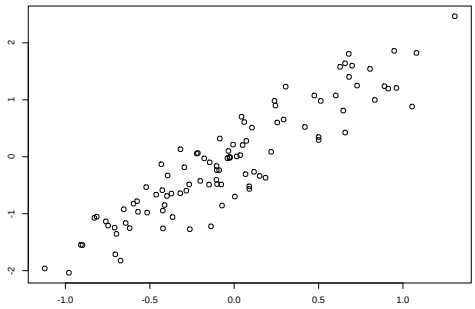
b) $\phi = -0.6$



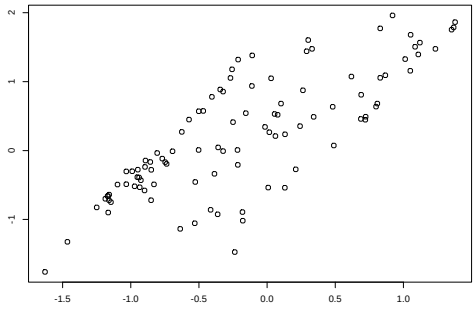
c) $\phi = -0.3$



d) $\phi = 0.3$



c) $\phi = 0.6$



d) $\phi = 0.9$

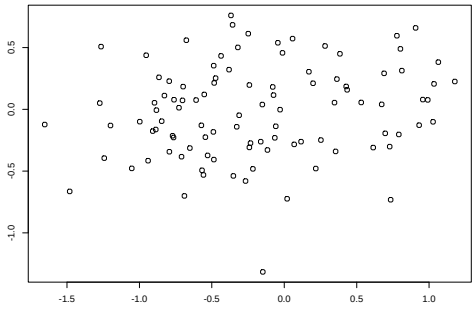


Figure 44: *Tukey mean difference plots for test data of the AR(1) time series data sets.*

5.3 Tent Map Series

Freedman, Navidi & Peters (1988) pointed out that the following map will generate a white noise like sequence, $z_t = f(z_{t-1})$, where

$$\begin{aligned} f(x) &= 2x \text{ if } x < 1/2 \\ &= 2(1-x) \text{ if } x \geq 1/2. \end{aligned}$$

This function when plotted looks like an upside-down V or a tent.

Tent map data is a chaotic series and consists of only signal. Therefore, if the formula for this series is given, it is possible to predict future values perfectly.

A series of length 501 was generated with *Mathematica* starting with an initial value of $z_0 = 0.177221$. If exact arithmetic is not used, the function degenerates quickly to 0 due to rounding error. A time series plot of the data is shown in Figure **??**.

Figure **??** shows autocorrelation function of the tent map data. It does not show any large correlation between any lags, which tells us that the series is quite chaotic. Figure **??** shows Bartlett's cumulative periodogram test of the tent map data. It shows that the whiteness of data is quite large. From the autocorrelation and cumulative periodogram, it is clear that linear time series methods will be of no avail in modelling this data. In this section, it will be examined if the FFNN model can handle this kind of chaotic data.

Before FFNN models are applied to this data, the number of the training data and the number of test data should be decided. The length of the training set greatly affects the forecast accuracy and the estimation of the weights. From the previous sections, it appears that when the training sets are small, the differences in the FFNN models is emphasized. Therefore we examine two scenarios using small and large training sets.

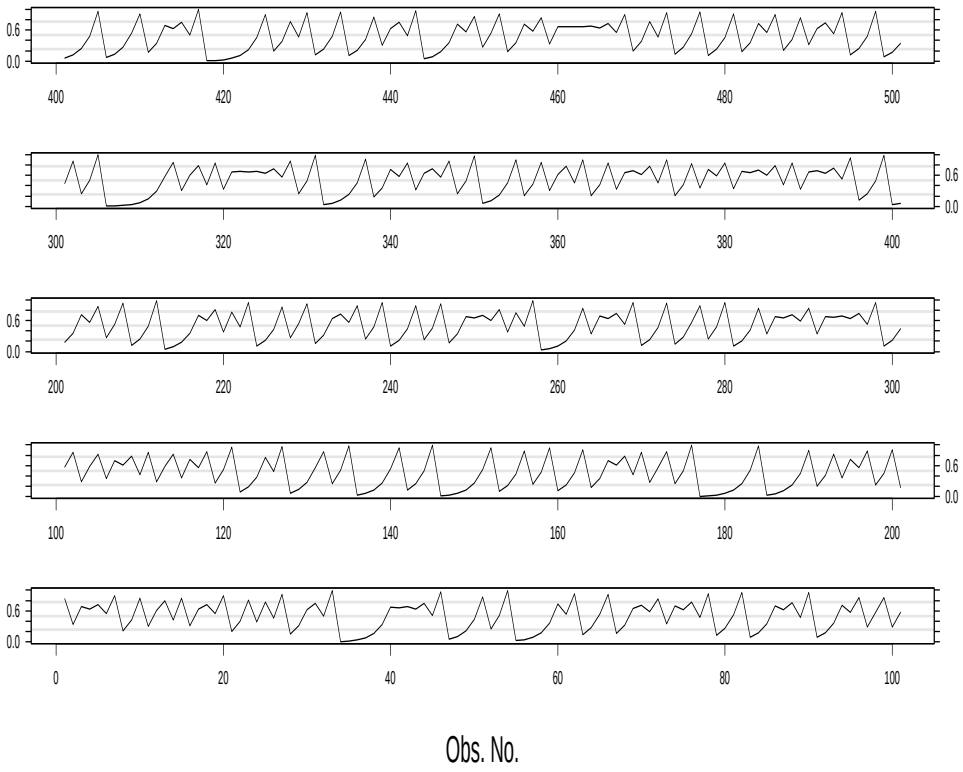


Figure 45: *Time series plot of the tent time series.*

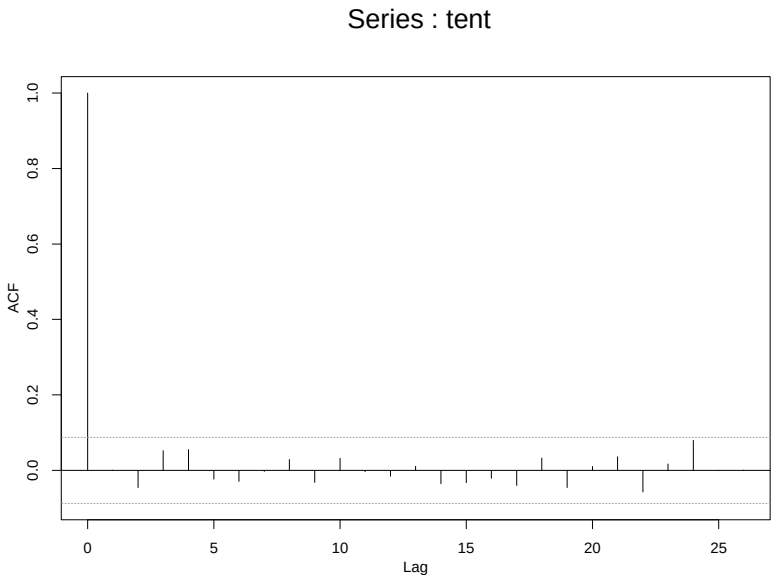


Figure 46: *Autocorrelation function of some data generated by the tent map.*

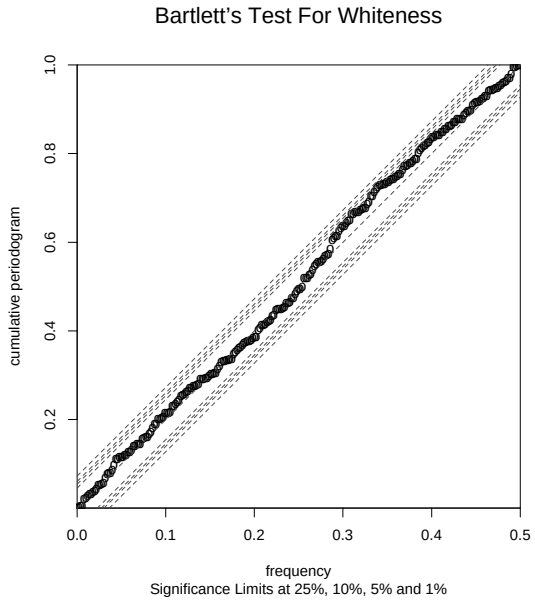


Figure 47: *Bartlett's test for Tent map data.*

5.3.1 $N_1 = 25, N_2 = 476$

The result when the combination of 25 training data and 476 training data is chosen, which holds comparatively smaller training data, is given in Table **??**.

From Table **??**, the best model which produces least RMSE value is NN(1; 3) and putting too many hidden nodes does not improve the results. Fitted values of NN(1; 3) model for the tent map data with 25 training data and first 100 test data are plotted in Figure **??**. It illustrates that the Fitted values for the test data are almost tracing the original data, but some points miss tracing the original data. Residual Fitted value spread plot for the test data is given in Figure **??** and Tukey mean difference plot for the test data is given in Figure **??**. It is shown that almost all the variability is accounted by NN(1; 3) and mean difference is spread around 0. However, Tukey mean difference plot shows that there is some pattern left by NN(1 : 3) model. This indicates that the tent map data is systematic data although it looks chaotic.

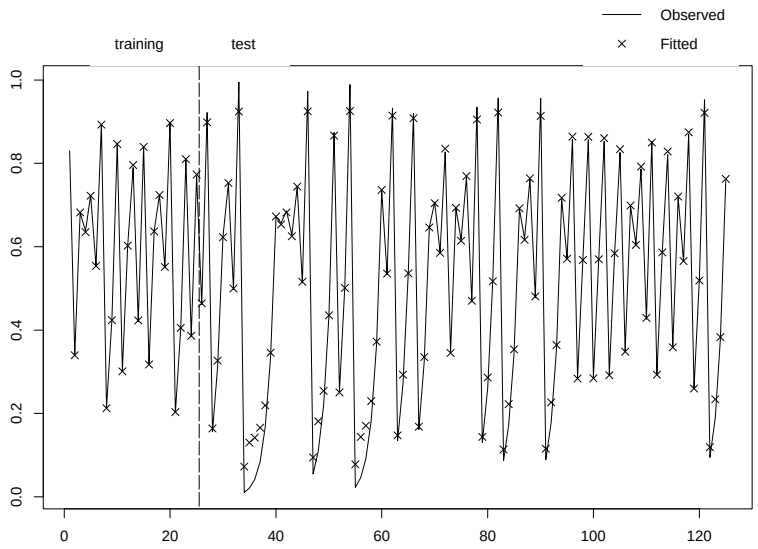


Figure 48: *Fitted values of the NN*(1; 3) *model for the tent map data. 25 training data and first 100 test data is shown with a solid line.*

Model	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
$\mathrm{NN}(1:1)$	1.2×10^{-1}	2.7×10^{-1}
$\mathrm{NN}(1,2;1)$	1.0×10^{-2}	2.7×10^{-1}
$\mathrm{NN}(1,2,3;1)$	1.2×10^{-1}	2.7×10^{-1}
$\mathrm{NN}(1;2)$	1.0×10^{-2}	8.3×10^{-2}
$\mathrm{NN}(1,2;2)$	9.3×10^{-3}	8.2×10^{-2}
$\mathrm{NN}(1,2,3;2)$	8.1×10^{-3}	6.1×10^{-2}
$\mathrm{NN}(1-4;2)$	8.3×10^{-3}	6.1×10^{-2}
$\mathrm{NN}(1-5;2)$	7.1×10^{-3}	5.6×10^{-2}
$\mathrm{NN}(1-6;2)$	7.1×10^{-3}	5.8×10^{-2}
$\mathrm{NN}(1-12;2)$	2.8×10^{-3}	4.3×10^{-1}
$\mathrm{NN}(1;3)$	2.1×10^{-3}	3.2×10^{-2}
$\mathrm{NN}(1,2;3)$	2.1×10^{-3}	3.7×10^{-2}
$\mathrm{NN}(1,2,3;3)$	2.8×10^{-3}	5.3×10^{-2}
$\mathrm{NN}(1,2,3,4;3)$	2.1×10^{-3}	2.0×10^{-1}
$\mathrm{NN}(1;10)$	2.0×10^{-3}	3.4×10^{-2}

Table 5.36: *RMSEs of 25 training data set and 476 test data set for the tent map data by several FFNN models.* σ_F *is RMSE for the training data and* σ_P *is RMSE for the test data.*

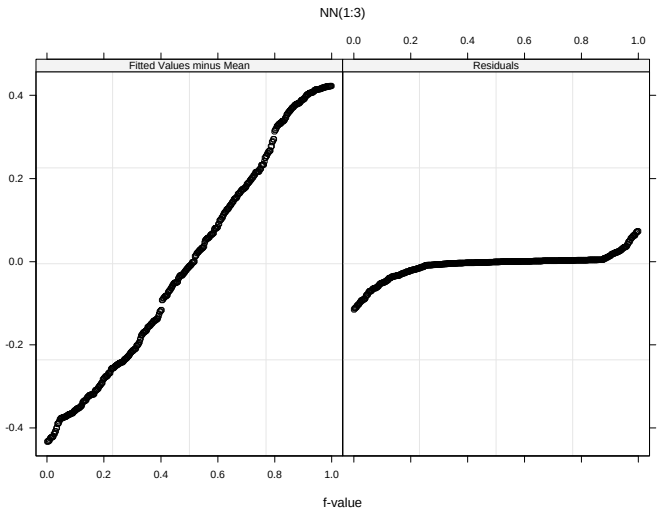


Figure 49: Residual fitted value spread plot of 100 tent map test data by $\mathrm{NN}(1;3)$. The $\mathrm{NN}(1;3)$ is trained with 25 data.

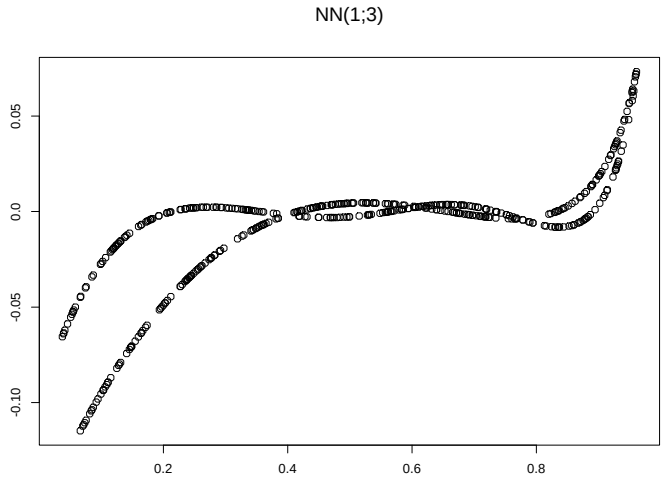


Figure 50: Residual fitted value spread plot of 100 tent map test data by NN(1; 3). The NN(1; 3) is trained with 25 data.

5.3.2 $N_1 = 100, N_2 = 401$

Next, we divide the data into 100 training data set and 401 test data and apply several NN$(1; x)$ models to it. The result is shown in Table **??**. As the number of hidden nodes gets larger, RMSE for the test data gets smaller. RMSE only for the test data is shown in Figure **??**. Amazingly, the predicted values trace perfectly the original data. This is because there is enough training data and the tent map data does not have any noise. Residual Fitted value spread plot and Tukey mean difference plot for the test data are given in figure **??**. Residual fitted values spread plot shows that almost all the variability is accounted by the fitted values. Tukey mean difference plot shows that there is still some pattern left by NN$(1 : 10)$ model.

We conclude that FFNN model does a quite good job for the data, which is chaotic but of which noise is 0 as long as enough training data length is given.

Model	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN$(1 : 1)$	2.3×10^{-1}	2.4×10^{-1}
NN$(1; 2)$	2.6×10^{-2}	2.5×10^{-2}
NN$(1; 3)$	1.4×10^{-2}	1.2×10^{-2}
NN$(1; 4)$	4.8×10^{-3}	5.0×10^{-3}
NN$(1; 5)$	3.1×10^{-3}	3.3×10^{-3}
NN$(1; 6)$	2.4×10^{-3}	2.5×10^{-3}
NN$(1; 10)$	1.5×10^{-3}	1.6×10^{-3}

Table 5.37: *RMSEs of 100 training data set and 401 test data set for the tent map data. σ_F is RMSE for the training data and σ_P is RMSE for the test data.*

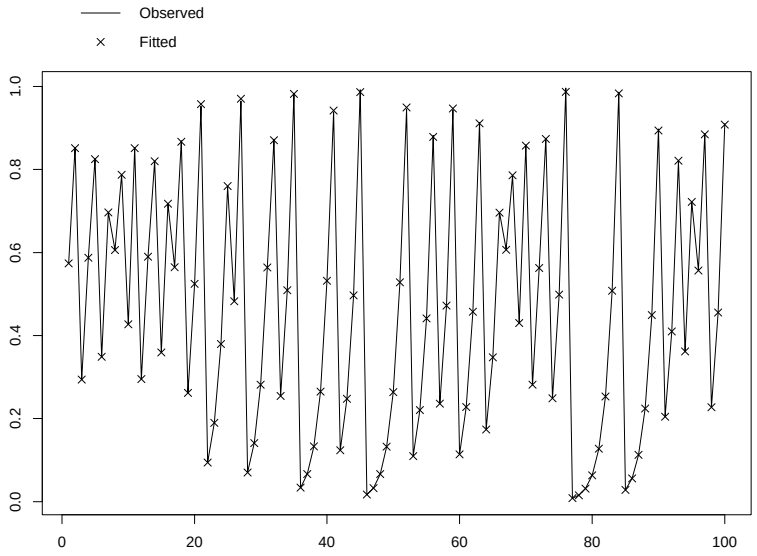


Figure 51: *Fitted values of the* $NN(1;3)$ *model for the tent map test data. First 100 test data is shown with a solid line. The* $NN(1;3)$ *is trained with 25 data.*

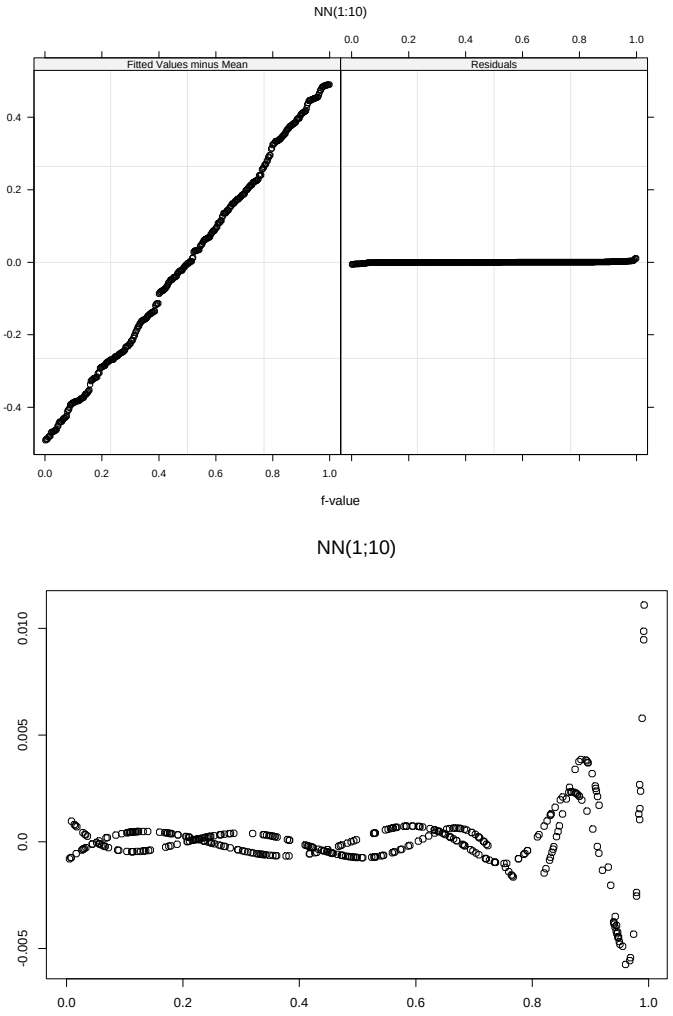


Figure 52: Residual fitted value spread plot and Tukey mean difference plot of 100 tent map test data by NN(1; 10). The NN(1; 10) is trained with 100 data.

5.4 Pseudo-Random Number Generator

An old favorite simple congruential pseudo-random number generator was developed by Lewis, Goodman & Miller (1969) based on the iteration,

$$X_i = 7^5 X_{i-1} \mod 2^{31} - 1. \tag{5.7}$$

The output is then rescaled, $U_i = X_i/2^{31}$ to produce a sequence on $(0, 1)$. Note that this generator exhibits Markov type dependence.

A sequence of 5000 rescaled data values was generated by eqn. **??** starting with $X_0 = 2^{29} + 1$. and will be referred to as the Lewis data. The first 300 values in the sequence of the Lewis data are plotted in Figure **??**.

Figure **??** shows autocorrelation function of Lewis data. It does not show any large correlation between any lags, which tells us that the series is quite chaotic. Figure **??** shows Bartlett's test of Lewis data. It shows that the whiteness of data is quite large. Because each data point has quite large value and NNET can give only some constant fitted values for the raw data, the data divided by 2^{31} to make the value smaller is rather used. 5000 data points are used here and first 500 is used as a training data. The results are given in Table **??**. All the model produce almost same amount of RMSE values.

Figure **??** illustrates last 50 fitted values of training data and first 50 fitted values of test data. As might be expected the FFNN model is unable to cope with adequately modeling this type of Markovian dependence.

Model	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN(1; 1)	0.29	0.29
NN(1; 2)	0.29	0.29
NN(1; 3)	0.28	0.29

Table 5.38: *RMSE of several FFNN models for the Lewis data divided by* 2^{31}*. The number of training data is 500 and the number of test data is 4500.* σ_F *is RMSE for the training data and* σ_P *is RMSE for the test data.*

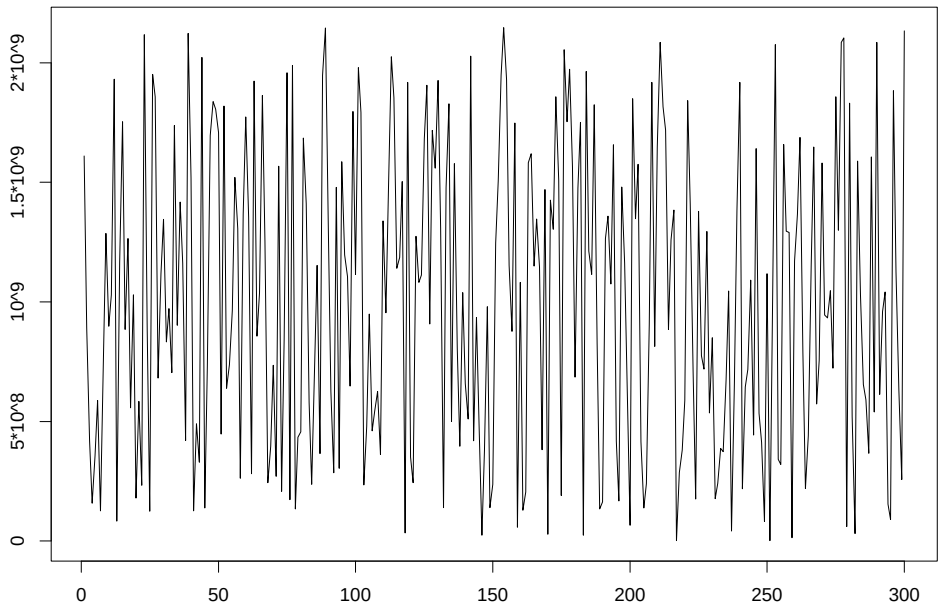


Figure 53: *The Lewis data.*

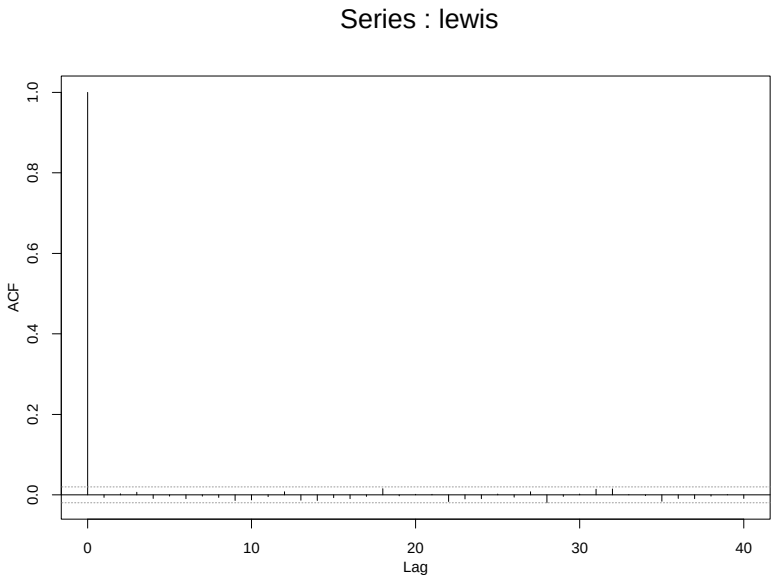


Figure 54: *Autocorrelation function of Lewis data.*

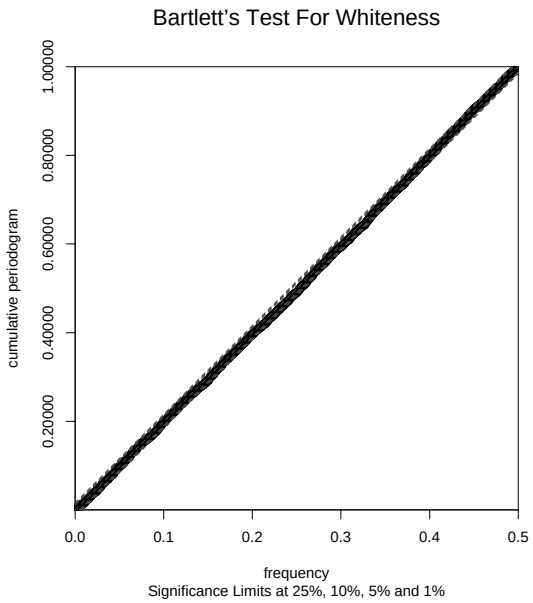


Figure 55: *Bartlett's test for Lewis data.*

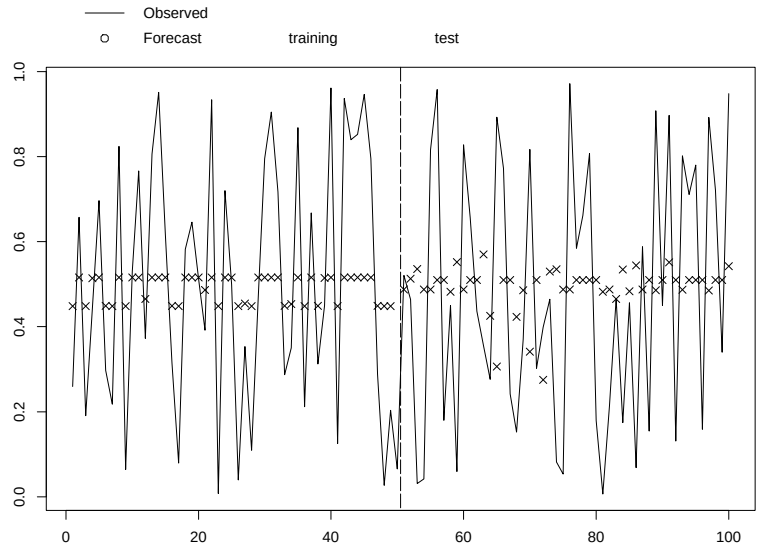


Figure 56: *Last 50 fitted values of training data and first 50 fitted values of test data by* $NN(1;1)$ *model.*

6 Lynx Data

6.1 Introduction

The data which is used this section give annual number of lynx trappings in the Mackenzie River District of North-West Canada for the period 1821 to 1934. It is assumed by ecologists that this data indicates the relative magnitude of the lynx population and is therefore of great interest to ecological researchers. This data is plotted in Figure **??**. Taking logarithms makes the data more symmetrically distributed as shown in Figures **??** and **??**. Ecologists believe that cycles in population such as observed with the lynx series occur when there is a strong predator-prey relationship. The predator causes the prey to decline and this is followed by a decline in the predator population. The prey population recovers and this causes an increase in the predator population and so the cycle continues.

This lynx data is one of the most frequently used time series. It is actually part of a much larger collection of time series derived originally from Hudson Bay Company archives and first published by Elton (1927) and analyzed by Elton and Nicholson (1942). The first modern time series analysis was carried out by Moran (1953) who fit an AR(2) to the logged data. Almost every family nonlinear time series models have been shown to produce a better fit than the AR(2). An extensive review by Tong (1990, §7.2) discusses the numerous nonlinear models that have been tried. Tong (1990, §7.2) favours the self-exciting threshold autoregression (SETAR) models. Tong (1983) used the SETAR model for some out-of-sample forecast comparisons with other nonlinear models.

In this chapter it is shown that the FFNN is just as good or better than the SETAR models of Tong (1983) for one-step out-of-sample forecasting. The FFNN is also compared with the AR(2).

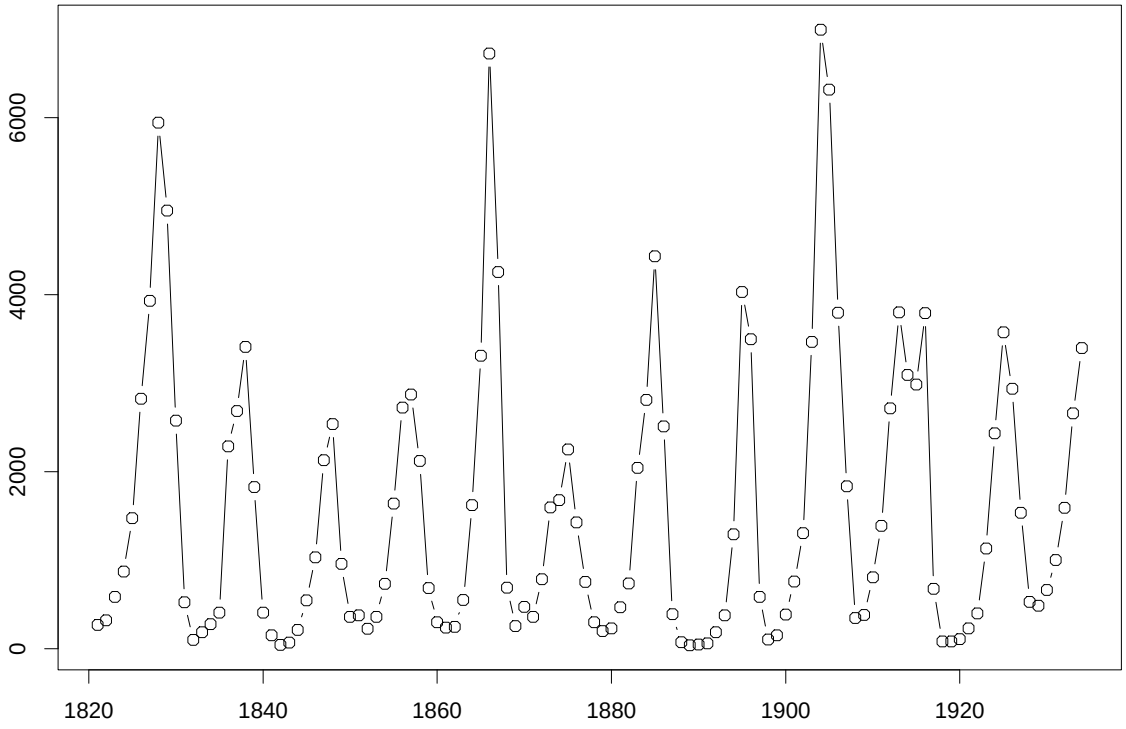


Figure 57: *Lynx data*

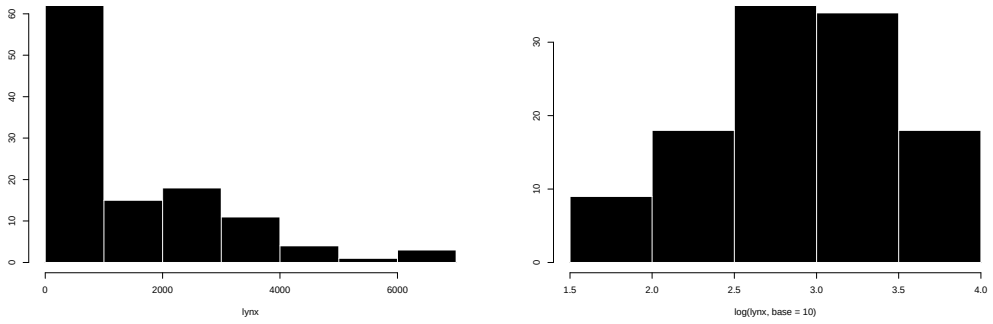


Figure 58: *histograms of the lynx data and the logged lynx data.*

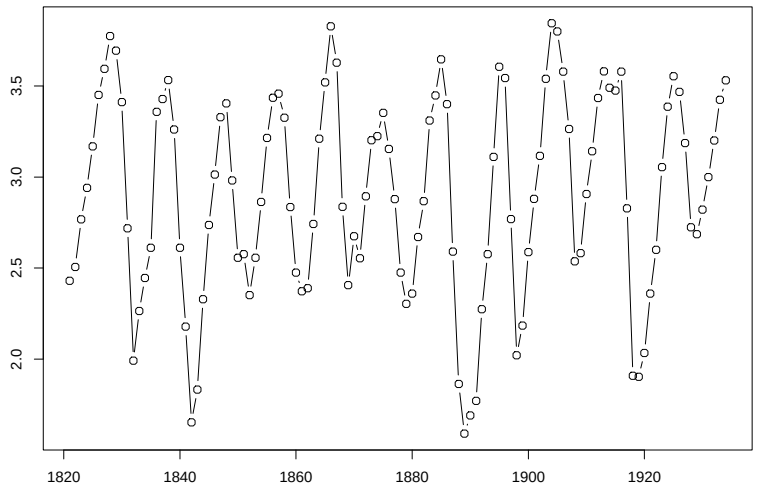


Figure 59: *Logged Lynx data*

6.2 Application of NN models to Lynx data

We divide lynx data into two data sets as before. One is a training data set and the other is a test data set. Because we know the data length affects the forecasting accuracy from the previous section, three different training and test data lengths will be examined. We use training lengths 74, 94 and 100 with respective test data lengths 40, 20 and 14. We compare the last case with the results reported by Tong (1983).

The lynx data is divided by 100 for convenience. Figure **??** compares histograms of the lynx data and the logged lynx data. The histograms of the lynx data is skewed to right and it is recommended to use the transformation $\ln(x + a)$, where a is a constant (SPSS, 1998). In fact, Tong (1983) used log with base = 10, which makes the lynx data more symmetric looking, see Figure **??**. Figure **??** shows the time series plot of the logged Lynx data. We will show the effect of the transformation for the non-linear time series data.

All the results are given in Table **??**, **??**, and **??**. Overall the best model would be NN$(1, 2; 2)$ with log transformation, which produces generally small RMSE values.

From these tables it is inferred that NN models produce better results than the AR(2) models, and even the SETAR model. SETAR model which produces the least RMSE value is singled out.

We have compared the out-of-sample forecasts with the observed data visually using time series plots of data and forecast, the residual-fit spread plot and Tukey mean difference plot. The plots offer more information that just reporting σ_P the root-mean-square error of the out-of-sample predictions. The residual-fit spread plots show that the NN$(1, 2; 2)$ model accounts for more variability than AR(2) does and draw our attention to outliers. In many applications, outlier detection is an important goal and here again the FFNN seems to work better than the AR(2). The Tukey-mean difference plots again show the FFNN produces generally smaller forecast errors.

SETAR model is explained in Tong (1983) and it shows taking a long time to find out the appropriate SETAR model. On the other hand, FFNN models do not take that long time to find the appropriate model. Considering these facts, we conclude that the lynx data is a case where FFNN models outperform statistical models.

It is also shown in this example that the log transformation is helpful.

Model	**Non-transformed**		**Transformed**	
	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN(1 : 1)	11.13	8.95	17.27	9.19
NN(1; 2)	10.59	10.52	17.53	9.19
NN(1, 2; 1)	7.83	9.59	8.84	7.71
NN(1, 2; 2)	6.96	8.12	7.67	7.77
NN(1, 2, 3; 1)	7.67	10.28	8.74	7.74
NN(1, 2, 3; 2)	6.22	9.63	7.81	7.70
NN(1, 10; 1)	10.68	7.95	10.83	7.57
NN(1, 10; 2)	9.72	7.92	10.91	7.24
NN(1, 2, 10; 1)	7.87	9.45	8.44	7.43
NN(1, 2, 10; 2)	6.63	6.99	7.46	7.60
NN(1, 2, 10, 20; 2)	6.04	13.77	7.40	8.75
AR(2)	10.22	9.53	9.49	8.24

Table 6.39: *Comparison of RMSEs between AR(2) model and several NN models for the lynx data and the logged data with base* $= 10$. *The lynx data is divided by 100 and RMSE for the logged data is given after back transforming. The number of training data is 94 and the number of test data is 20.* σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

Model	**Non-transformed**		**Transformed**	
	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN$(1:1)$	9.92	12.23	10.48	13.27
NN$(1;2)$	9.13	12.89	10.53	13.49
NN$(1,2;1)$	6.96	9.93	8.43	9.11
NN$(1,2;2)$	6.03	11.26	7.06	9.08
NN$(1,2,3;1)$	6.28	11.43	7.86	10.32
NN$(1,2,3;2)$	5.16	12.89	6.58	9.93
NN$(1,10;1)$	9.16	12.20	9.35	13.39
NN$(1,10;2)$	7.44	14.86	9.35	14.19
NN$(1,2,10;1)$	6.73	10.12	7.64	10.03
NN$(1,2,10;2)$	5.27	10.76	6.15	10.13
NN$(1,2,10,20;2)$	4.60	14.89	6.13	10.89
AR(2)	9.32	11.37	8.92	9.87

Table 6.40: *Comparison of RMSEs between AR(2) model and several NN models for the lynx data and the logged data with base* $= 10$. *The lynx data is divided by 100 and RMSE for the logged data is given after back transforming. The number of training data is 74 and the number of test data is 40.* σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

Model	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN(1 : 1)	0.35	0.24
NN(1; 2)	0.35	0.24
NN(1, 2; 1)	0.23	0.14
NN(1, 2; 2)	0.21	0.10
NN(1, 2, 3; 1)	0.23	0.15
NN(1, 2, 3; 2)	0.20	0.092
NN(1, 10; 1)	0.33	0.22
NN(1, 10; 2)	0.32	0.22
NN(1, 2, 10; 1)	0.19	0.21
NN(1, 2, 10; 2)	0.20	0.14
NN(1, 2, 10, 20; 2)	0.19	0.21
AR(2)	0.31	0.19
SETAR(2;6,3)		0.12

Table 6.41: *Comparison of RMSEs among AR(2) model, SETAR model and several NN models for the logged lynx data with base* = 10. *RMSE for the logged data is given without back transforming. The number of training data is 100 and the number of test data is 14.* σ_F *is RMSE of the training data and* σ_P *is RMSE of the test data.*

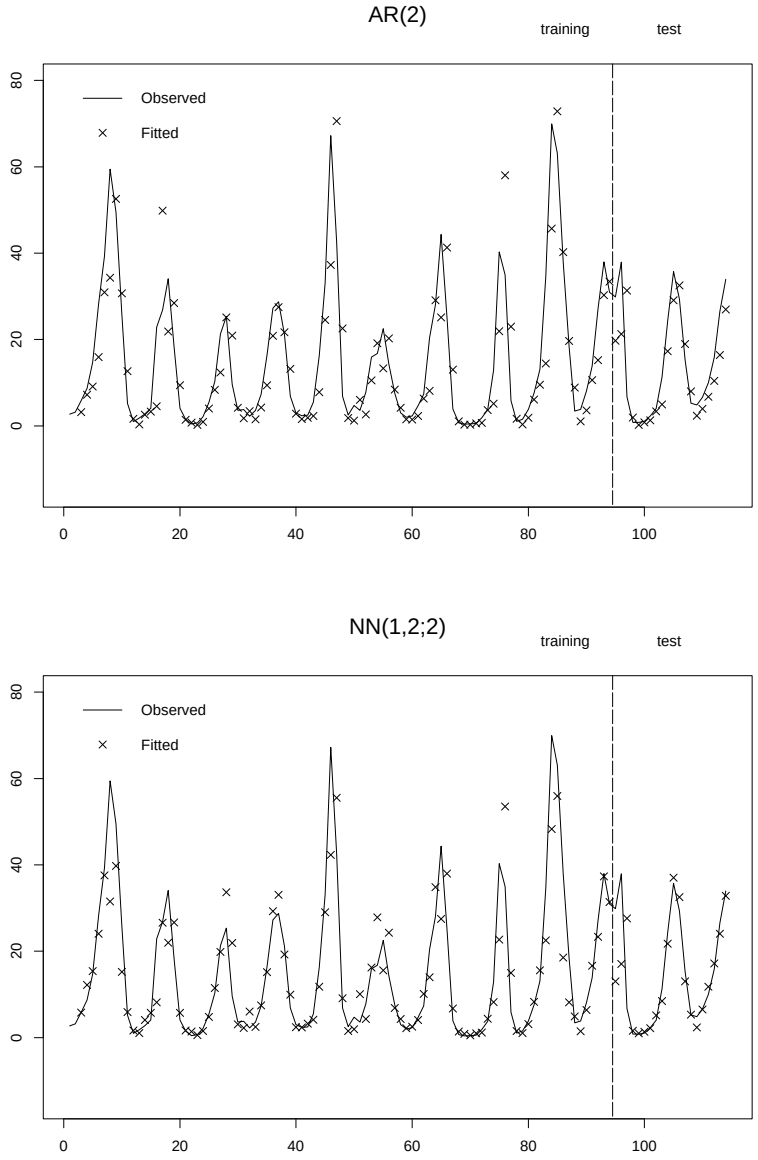


Figure 60: Fitted values for the log transformed lynx data by AR(2) model and by NN$(1, 2; 2)$. Lynx data is divided by 100 and fitted values are plotted after back transformation. Observations are plotted with a solid line. The number of training data is 94 and the number of test data is 20.

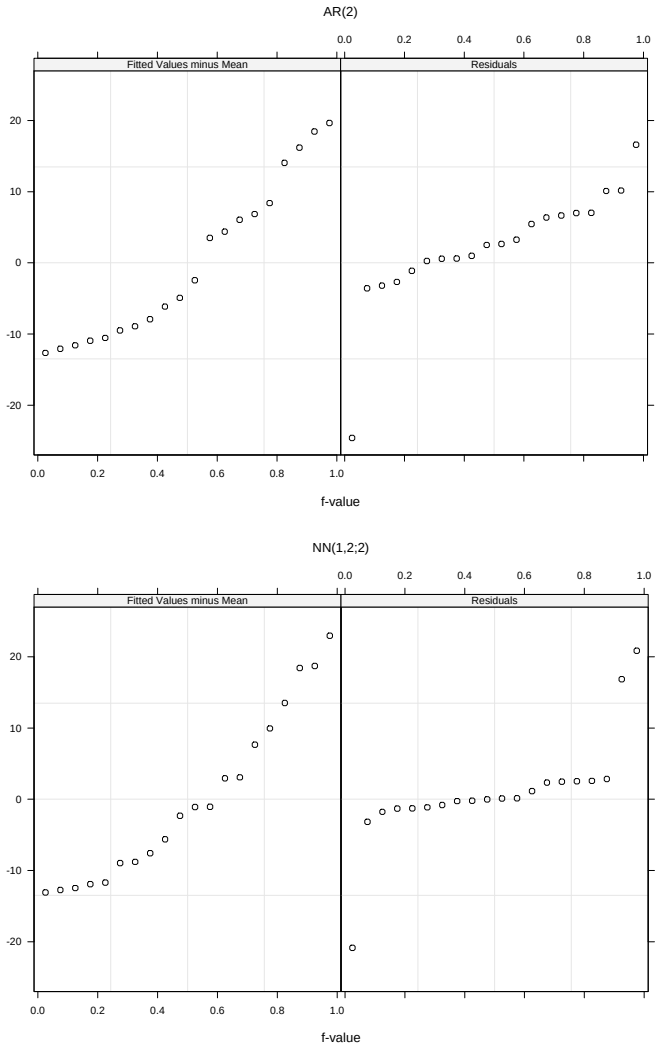


Figure 61: Residual fitted value spread plots of the 20 lynx test data by AR(2) and by NN$(1, 2; 2)$. The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals. We see that the AR(2) model produced one large negative residual outlier and the FFNN produced 2 positive and one negative outlier. For the bulk of the remaining data, the FFNN produced smaller residuals.

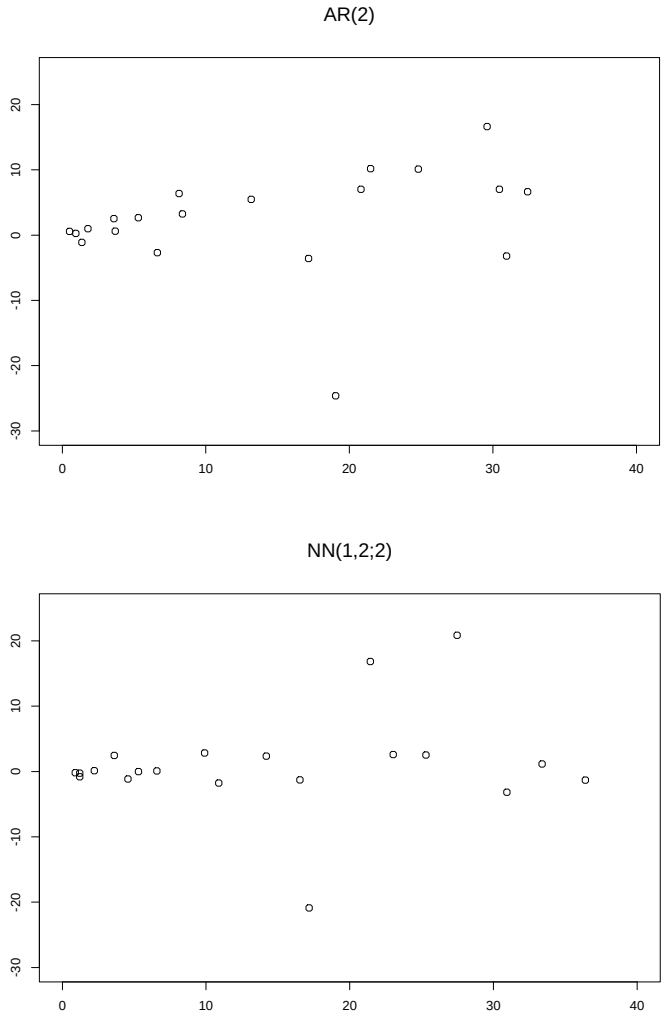


Figure 62: Tukey mean difference plots of the 20 lynx test data by AR(2) and by NN$(1, 2; 2)$. The horizontal axis in each plot shows the $(X + Y)/2$ and the vertical axis $Y - X$ where $X =$ observed and $Y =$ one-step forecast.

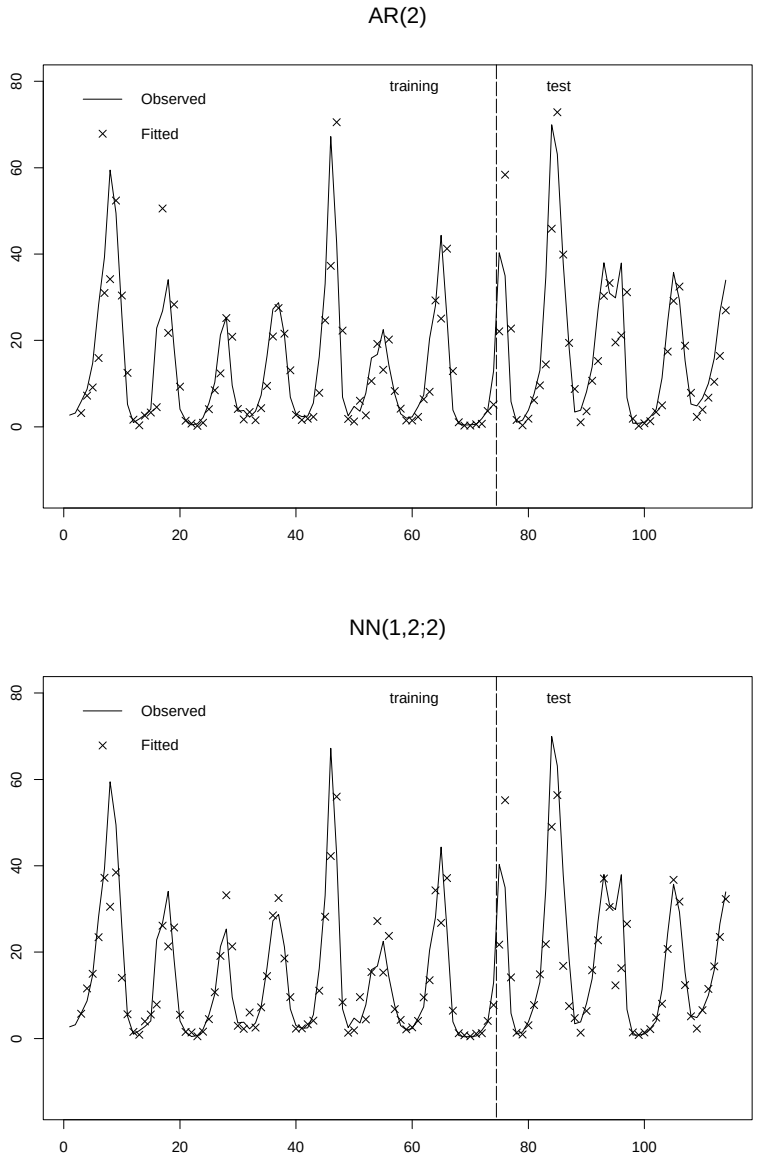


Figure 63: *Fitted values for the log transformed lynx data by AR(2) model and by* $NN(1, 2; 2)$. *Lynx data is divided by 100 and fitted values are plotted after back transformation. Observations are plotted with a solid line. The number of training data is 74 and the number of test data is 40*

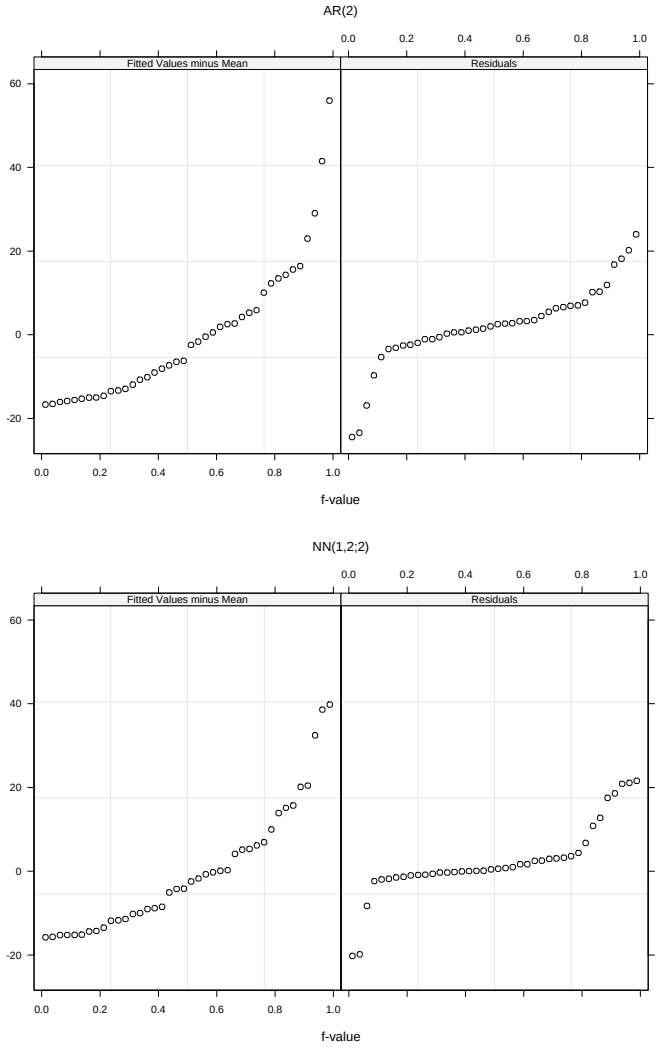


Figure 64: Residual fitted value spread plots of the 40 lynx test data by AR(2) and by NN$(1, 2; 2)$. The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.

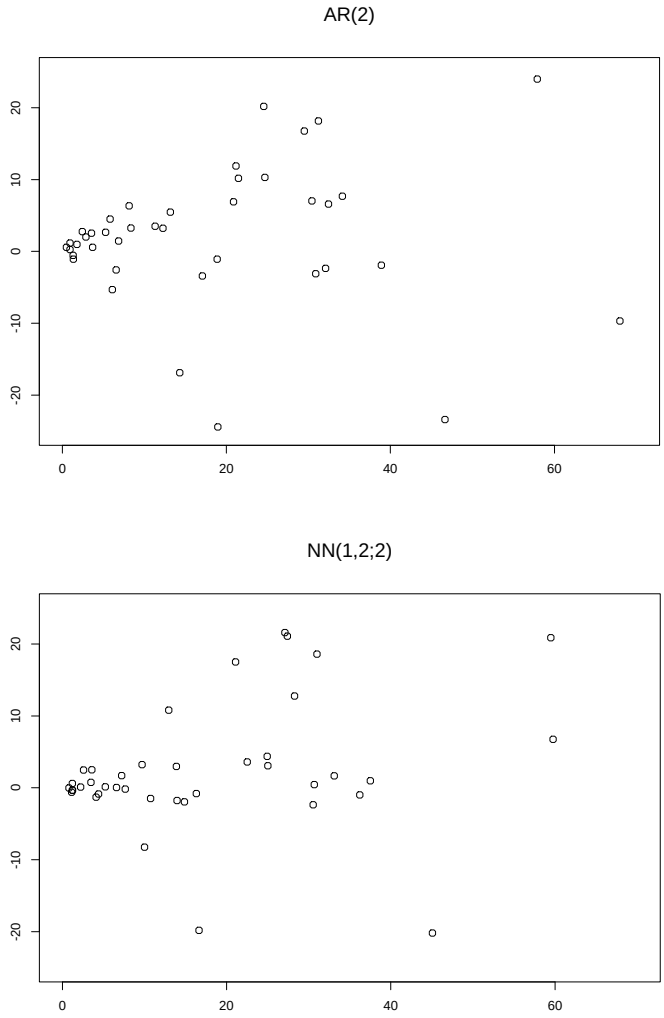


Figure 65: Tukey mean difference plots of the 40 lynx test data by AR(2) and by NN$(1, 2; 2)$. The horizontal axis in each plot shows the $(X + Y)/2$ and the vertical axis $Y - X$ where $X =$ observed and $Y =$ one-step forecast.

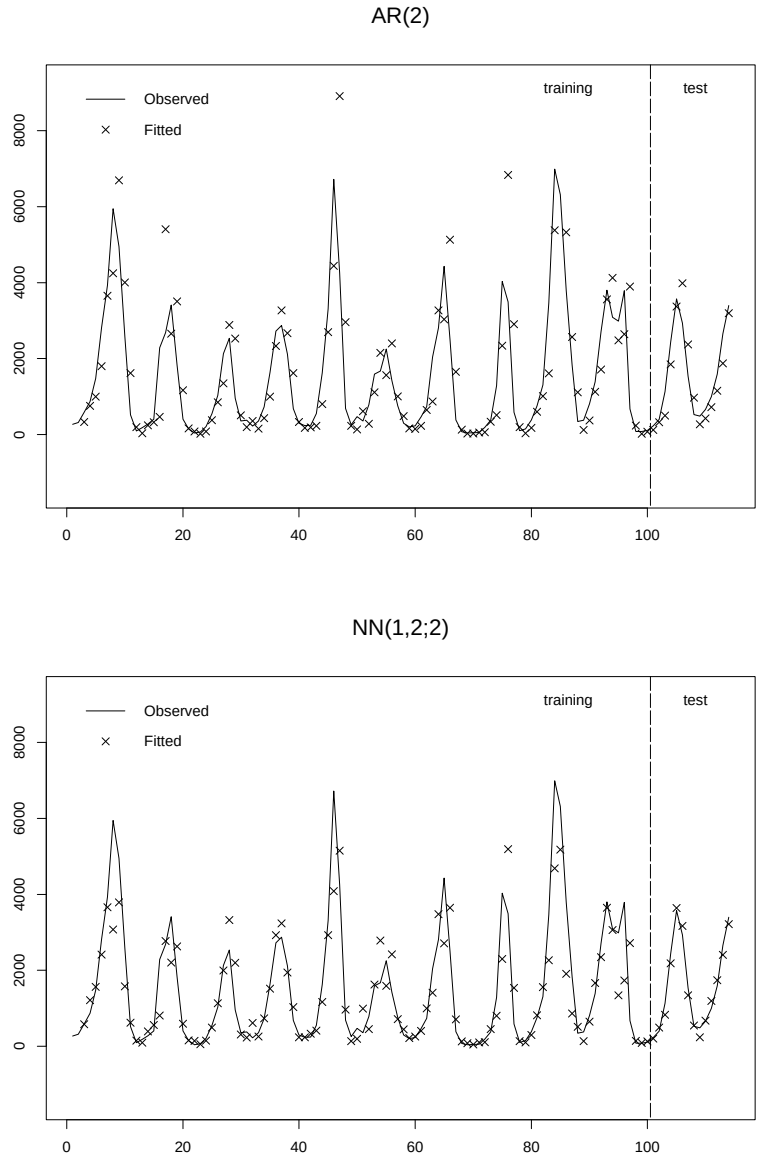


Figure 66: *Fitted values for the log transformed lynx data by AR(2) model and by* $NN(1, 2; 2)$. *Fitted values are plotted after back transformation. Observations are plotted with a solid line. The number of training data is 100 and the number of test data is 14*

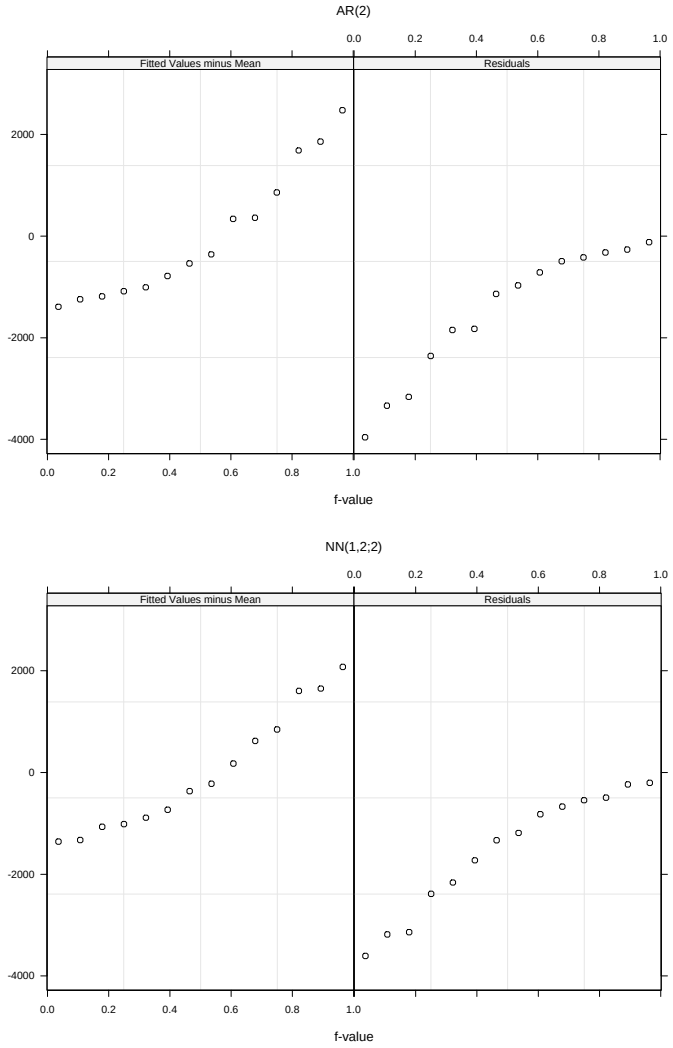


Figure 67: Residual fitted value spread plots of the 14 lynx test data by AR(2) and by NN$(1, 2; 2)$. The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.

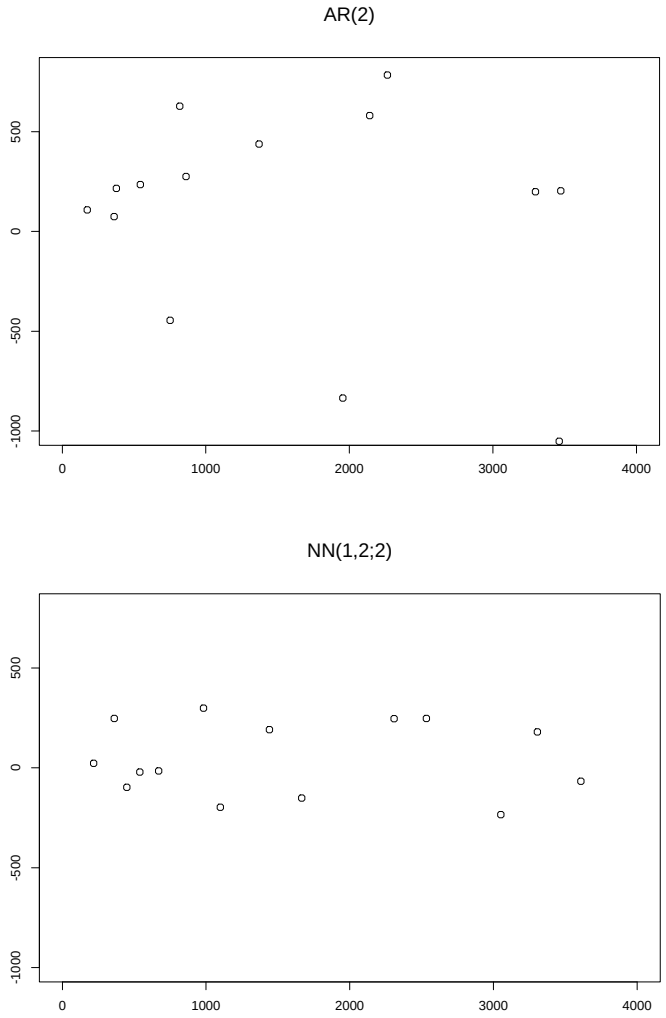


Figure 68: Tukey mean difference plots of the 14 lynx test data by AR(2) and by NN$(1, 2; 2)$. The horizontal axis in each plot shows the $(X + Y)/2$ and the vertical axis $Y - X$ where $X =$ observed and $Y =$ one-step forecast.

7 Half-hourly Streamflow Forecasting

7.1 Introduction

The data set used in this chapter is discrete time streamflow data at half-hourly interval if River Hirnant, Wales, U.K. from November to December, 1972. This data consists of 2928 rain data and flow data which cover an area of 33.9 km^2. Although it is quite important for the flood control and river regulation to predict short term run-off models, there has not been satisfactory model for this data set so far. It is mentioned by Weiss, G.(1984) that fitting of models has so far been based mainly on least squares techniques, which may not suitable to the point process structure of the rain process. The half-hourly streamflow data is plotted in Figure **??**. Considering the difficulty of building the rain-flow relation model, we will use just streamflow time series data and compare FFNN with AR model.

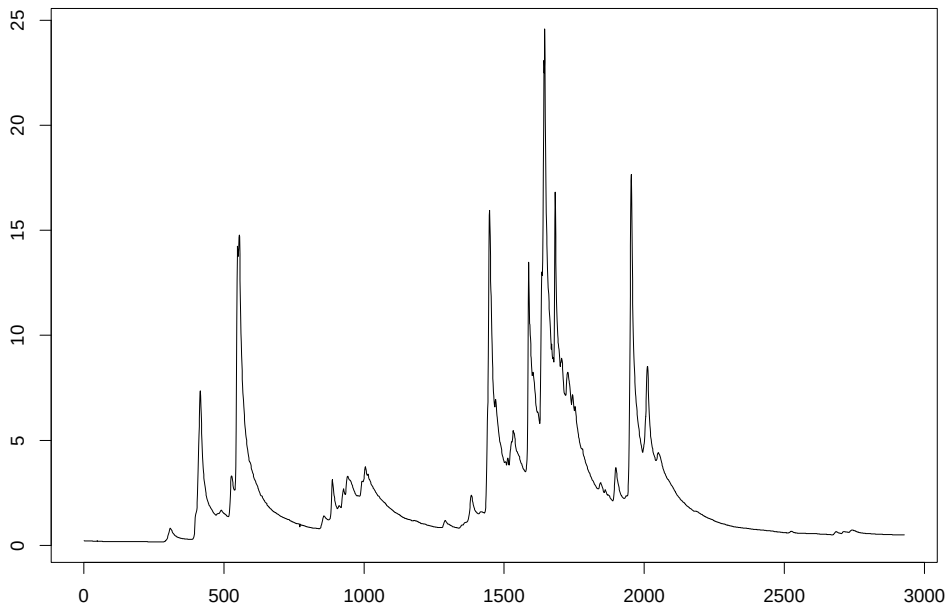


Figure 69: *The half hourly streamflow data from November to December, 1972.*

7.2 Forecasting by NN models

Considering this non-linear looking data in Figure **??**, it would be a good try to use NN models this data. However, as Figure **??** shows, the data is quite skewed to the right. Therefore, some transformation should be applied as is the case in the previous chapter. In this case, we use simply natural logarithm and the histogram of the flow data after log transformation is given in Figure **??**. We see that this histogram is more uniform looking than that in Figure **??**. Then We apply NN models to the logged flow data with 1500 training data and 1428 test data.

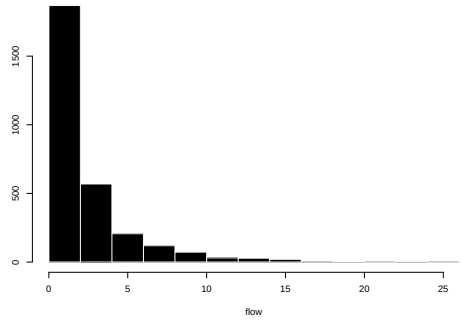


Figure 70: *Histograms of the flow data.*

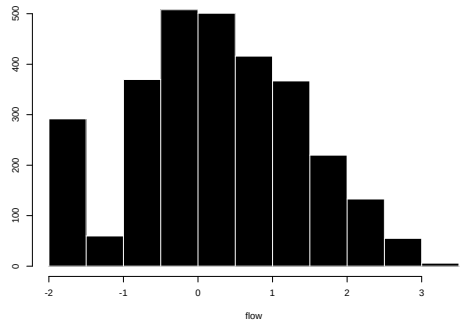


Figure 71: *Histograms of the log transformed flow data.*

Before applying NN models, we have to choose an appropriate lags, PAC (Partial Autocorrelation) of the logged flow data in Figure **??** suggests that AR(5) should be used. The large partial autocorrelation at lag one suggests that perhaps first differ-

encing could be used. However since it is meaningful to think of average daily and monthly flows, a model which is tied to a mean level seems suitable. If differencing is taken then this would induce a wandering type of behaviour, found in the stock market but not with riverflow series. Moreover the estimation procedure will consistently estimate an AR(5) model with a root lying on the unit circle. Since the data length is long, the precise form of the best fitting ARIMA model will not matter very much so long as the model is not statistically inadequate due to autocorrelation in the residuals. The AR(5) model seems to fit adequately in this respect. Hence, we compare NN(1-5;a), where a is a number of hidden nodes, with the AR(5) model.

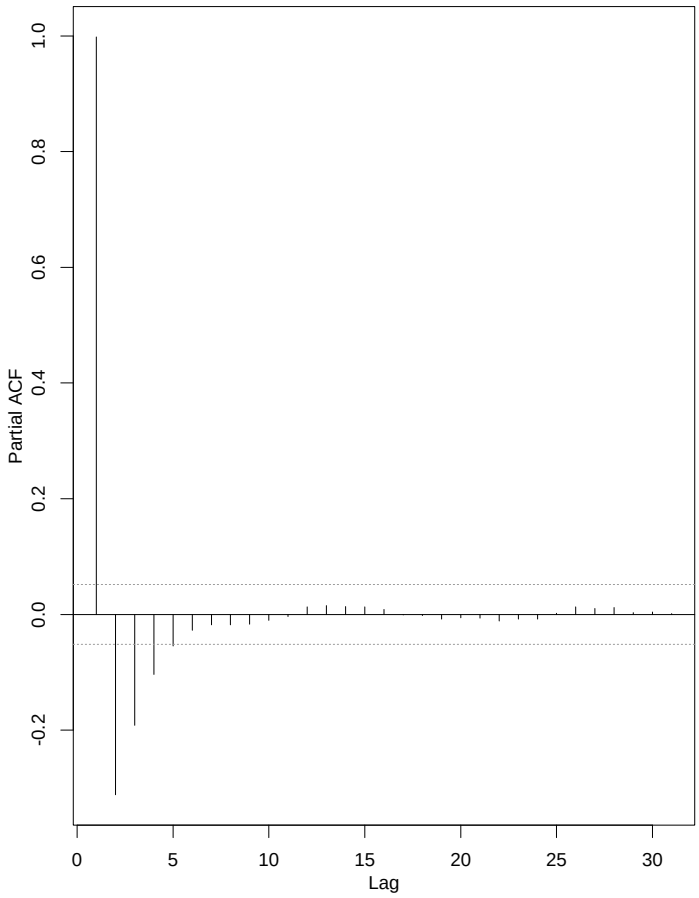


Figure 72: *Partial Autocorrelation of the 1500 logged test streamflow data.*

Model	**Transformation**	$\sigma_{\mathbf{F}}$	$\sigma_{\mathbf{P}}$
NN(1-5;1)	Yes	0.901	0.202
NN(1-5;2)	Yes	0.0780	0.206
NN(1-5;3)	Yes	0.0755	0.214
NN(1-5;4)	Yes	0.0754	0.233
NN(1-5;1)	No	0.0797	1.345
NN(1-5;2)	No	0.0705	1.352
AR(5)	Yes	0.141	0.253

Table 7.42: *Comparison of various* NN *models with AR(5) model for the half-hourly streamflow data with log transformation and without log transformation. The number of training data is 1500 and that of test data is 1428.* σ_F *is RMSE for training data and* σ_P *is RMSE for test data.*

Table **??** shows various NN models chosen to fit the data and their RMSEs for the fitted values.

Figure **??** and **??** show that the prediction errors for the NN(1-5;2) is slightly smaller. The difference is highly statistically significant, Pitman's test $r = 0.30$, $\hat{t} = 24.74$ and achieved significance level of $2Pr(t > \hat{t}) = 0$. However, PMC = 0.508 with achieved significance level of 0.56. In terms of PMC, there is hardly any difference.

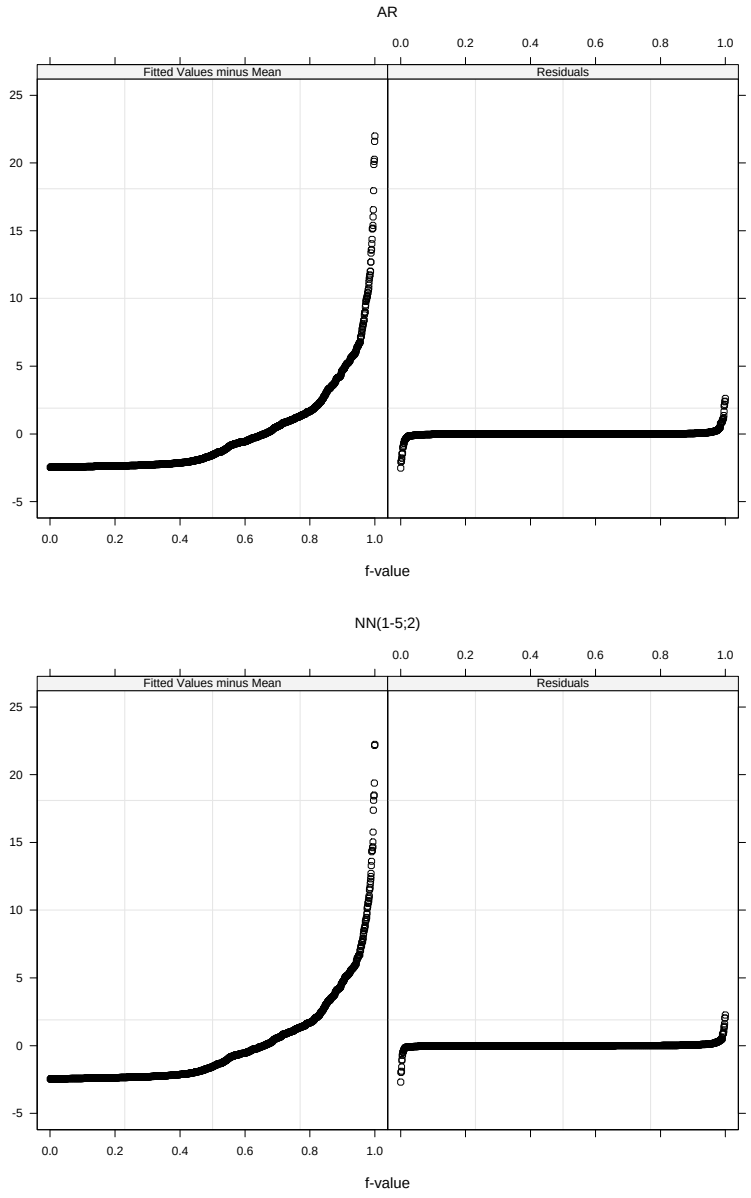


Figure 73: *Residual-fit spread plots of the 1428 streamflow test data by AR(5) model and* $NN(1-5;2)$. *The left panel in each display shows the plot of quantile plot of the fits minus the mean. The right panels are the quantile plots of the residuals.*

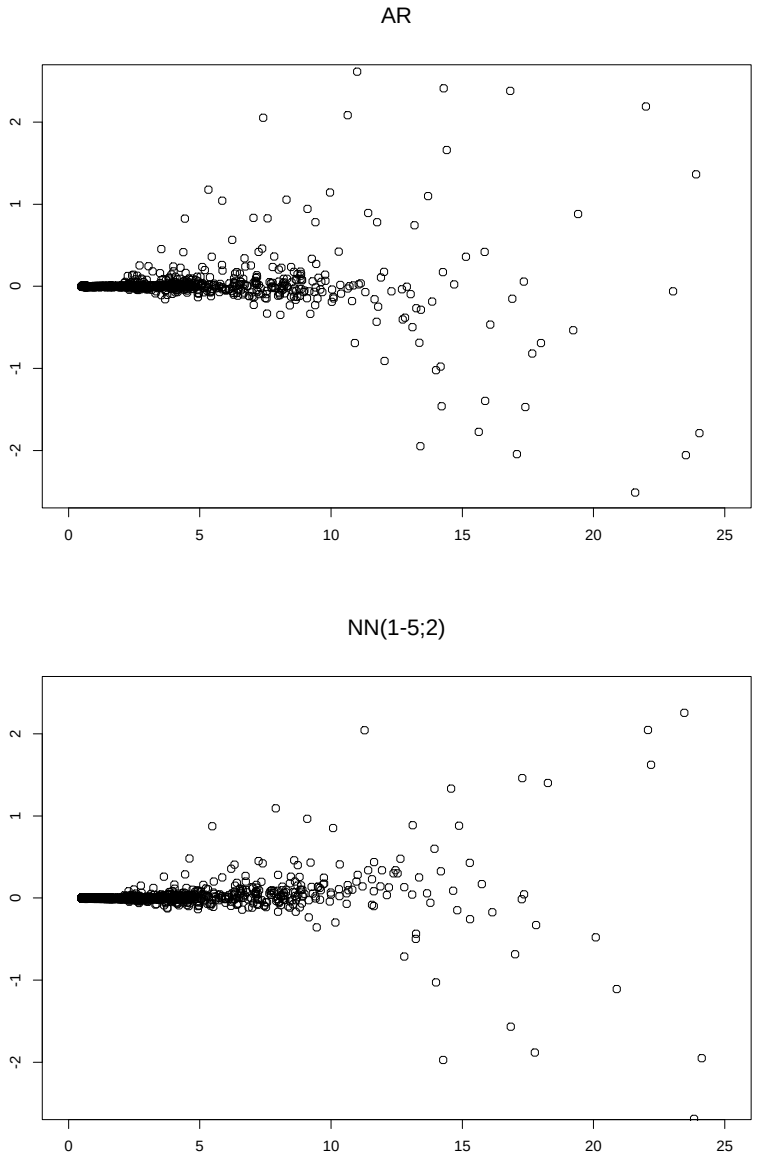


Figure 74: *Tukey mean difference plots of the 1428 streamflow test data by AR(5) model and NN(1-5;2). The horizontal axis in each plot shows the* $(X+Y)/2$ *and the vertical axis* $Y - X$ *where* $X =$ *one-step forecast and* $Y =$ *observed.*

8 Summary & Conclusion

We have examined possibility of FFNN and tried to find a appropriate way of applying them into time series data. Time series data sets with seasonality are analyzed from Chapter 2 to Chapter 4 and nonlinear time series data sets are analyzed in Chapter 6 and Chapter 7. In these chapters, performance of FFNN are compared with corresponding statistical models not only by evaluating RMSE (root mean square errors), but also by visualization techniques and statistical tests. We could show that those methods are quite successful in this thesis.

In Chapter 2, we first criticized introducing AIC and BIC as a criterion for choosing NN models, which is recommended by Faraway & Chatfield (1998). Instead, we suggest in the whole thesis that choosing an appropriate NN models, or in other words choosing a appropriate input lags should be done as if identifying the SARIMA model. Even if the FFNN works in a black-box way, we have to be careful about lags to input and the number of hidden nodes. Faraway & Chatfield (1998) concluded that transformation and differencing of data does not improve the results. It is partly true only for this airline data, but we pointed out their chosen input lags are inappropriate and FFNN can do better than they thought if we choose an appropriate lags to input. In other seasonal data sets, transformation and differencing helps improving the accuracy of out-of-sample forecasts better.

Since the long-term forecast of a stationary time series is just the mean, we have concentrated in this thesis on comparing the one-step forecasts. This may perhaps be expected to give a more sensitive comparison between models.

The visual and statistical comparison of FFNN and SARIMA for Airline data, showed FFNN outperforms the SARIMA.

On the other hand, in Chapter 3, SARIMA model outperforms FFNN for Sales data, which is analyzed in this chapter. The possible reason comes out after plotting the differenced and seasonally differenced power transformed data. The trend of test data is apparently different from the trend of training data and FFNN could not handle the data which is quite different from the old trend as well as SARIMA

model can do. However, we could see the transformation and difference helped the forecast accuracy of FFNN very much.

In Chapter 4, to see the difference of performances between the FFNN and SARIMA model more clearly, we used the water usage data which has 276 data and a jump in the last 24 data set. In short, the trend of last 24 data set is different from the trend of the first 252 data. In the first data dividing scheme which includes half of last 24 data, FFNN could do at least as well as SARIMA model do. On the other hand, in the second data dividing scheme which does not include any of last 24 data, FFNN performed worse than SARIMA model did.

In Chapter 5, we analyzed Markovian data sets. First we generate AR(1) model with various signal-to-noise ratio and then applied FFNN to them. Stern(1996) exerted the same simulation with AR(2) models, but all of his models suffered from over-training. We pointed out that NN(1,2;h), with $h = 2$ is more suitable. Our NN(1,2;h) worked much better than his models for AR(2) data. In the AR(1) simulation, we showed that the length of training data and the signal-to-noise ratio affect the forecast accuracy. Basically FFNN trained with shorter length of training data can not perform as well as than FFNN with longer length of training data. Also smaller signal-to-noise ratio cannot give a good out-of-sample forecast, which is consistent with the AR(2) simulation in Stern (1996). Those results are consistent with our intuition.

In Chapter 6, the Lynx data, is analyzed by FFNN. We showed that for the Lynx Data it is the case that FFNN outperforms the best linear and nonlinear statistical models. This is clearly shown with the help of RF-Spread plot.

In Chapter 7, half-hourly flow data is modelled. We compared FFNN with an AR(5) model and Pitman's test showed that the FFNN produces significantly better forecast errors than AR(5) model does. To identify the lags to in the FFNN model, PACF is used. However, different input lags and transformation should also be tried and it would be interesting to build the relation model between rainfall and streamflow by FFNN.

In summary, we demonstrated that the FFNN model can often do as well and

sometimes better than statistical models. This is especially true when the time series is generated by a nonlinear and nonGaussian model and there is enough data.

Bibliography

ANDERSON, J.A. AND ROSENFELD, E. (1998), *Talking Nets: An Oral History of Neural Networks*, Cambridge: MIT Press.

BOX, G.E.P. & JENKINS, G.M. (1973). Comments on "Box-Jenkins seasonal forecasting: Problems in a case-study" by C. Chatfield and D.L. Prothero, *Journal of the Royal Statistical Society* A 136, 295–308.

BOX, G.E.P. & JENKINS, G.M. (1976), *Time Series Analysis: Forecasting and Control*, (2nd edn), San Francisco: Holden-Day.

CHEN, B. & TITTERINGTON, D.M. (1994), Neural networks: A review from a statistical perspective, *Statistical Science* 9, 2–54.

CASDAGLI, M. (1989), Nonlinear prediction of chaotic time series, *Physica* D35, 335–356.

CHATFIELD, C. & PROTHERO, D.L. (1973), Box-Jenkins seasonal forecasting: Problems in a case-study, *Journal of the Royal Statistical Society* A 136, 295–308.

DEBOECK, G. & KOHONEN, T. (1998), *Visual Explorations in Finance with Self-Organizing Maps*, New York: Springer-Verlag.

DE VEUX, R.D., SCHUMI, J., SCHWEINSBERG, J. & UNGAR, L.H. (1998), Prediction intervals for neural networks via nonlinear regression, *Technometrics* 40, 273–282.

DYSON, G. (1997), *Darwin Among the Machines: The Evolution of Global Intelligence*, Reading: Addison-Wesley

ELTON, C (1927), *Animal Ecology*, New York: MacMillan.

ELTON, C & NICHOLSON, M. (1942), The ten-year cycle in numbers of the lynx in Canada, *Journal of Animal Ecology* 11, 215–244.

FARAWAY, J. & CHATFIELD, C. (1998), Time series forecasting with neural networks: a comparative study using the airline data, *Applied Statistics* 47, 231–250.

FARAWAY, J. (1998), http:
www.stat.lsa.umich.edu/ faraway.

FREEDMAN, D.A., NAVIDI, W. & PETERS, S.C. (1988), On the impact of variable selection in fitting regression equations. In *On Model Uncertainty and its Statistical Implications: Proceedings of a Workshop Held in Groningen*, ed. T.K. Dijkstra, New York: Springer-Verlag

FREEMAN, J.A. (1994), *Simulating Neural Networks with Mathematica*, Reading: Addison-Wesley

JOHNSON, R.C. & BROWN, C. (1988), *Cognizers: Machines that Think*, New York: Wiley.

HERTZ, J., KROGH, A. & PALMER, R.G. (1991), *Introduction to the Theory of Neural Computation*, Reading: Addison-Wesley.

HIPEL, K.W. & MCLEOD, A.I. (1994). *Time Series Modelling of Water Resources and Environmental Systems*. Elsevier: Amsterdam.

HOWELL, T. (1983), *Threshold Models in Non-linear Time Series Analysis*, Lecture Notes in Statistics, vol 21

HUTCHINSON, J.M. (1994), *A Radial Basis Function Approach to Financial Time Series Analysis*, Ph.D. Disseration, Massachusetts Institute of Technology.

KASABOV, N.K. (1998), *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*, Cambridge: MIT Press.

KURZWEIL, R. (1999), When machines think, *Maclean's*, March 1, pages 54–57.

LEWIS, P.A.W., GOODMAN, A.L. & MILLER, J.M. (1969), A pseudo-random number generator for the System 360. *IBM Systems Journal* 8, 136–146.

LISI, F. & Schiavo, R.A. (1999), A comparison between neural networks and chaotic models for exchange rate prediction, *Computational Statistics and Data Analysis* 30, 87–102.

MOZER, M.C. (1993), Neural net architectures for temporal sequence processing. In *Forecasting the Future and Understanding the Past*, eds. Andreas S. Weigend & Neil A. Gershenfeld, Reading: Addison-Wesley.

MCLEOD, A.I., (1993), Parsimony, Model Adequacy and Periodic Correlation in Forecasting Time Series, *International Statistical Review* 61, 387–393.

MCLEOD, A.I. & HIPEL, K.W. (1999), *MHTS PC Package*, http://www.stats.uwo.ca/mcleod/epub/mhts.

MENROTRA, K., MOHAN, C.K. & RANKA, S. (1997) *Elements of Artifical Neural Nets*, Cambridge: MIT Press.

MORAN, P.A.P. (1953), The statistical analysis of the Canadian lynx cycle, *Australian Journal of Zoology* 1, 163-173.

MURTAGN, F. (1999), Neural networks and related massively parallel method for statistics, *International Statistical Review* 62, 275–288.

PARK, Y.R., CHEN, C. & MURRAY, T.J. (1992), A study of neural network design and its application to statistical forecasting, in the *Proceedsings of the 24th Symposium on the Interface of Computer Science and Statistics*, ed. H.J. Newton.

SPSS Inc. (1998), *Neural Connection, Version 2.0*, SPSS Inc./ Recognition System Inc., Chicago.

STERN, H.S. (1996), Neural networks in applied statistics, *Technometrics* 38, 205–220.

RIPLEY, B.D. (1996), *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University Press.

RIPLEY, B.D. (1999), http://www.stats.ox.ac.uk/pub/MASS2/.

TONG, H. (1983), *Threshold Models in Nonlinear Time Series Analysis* New York: Springer-Verlag.

TONG, H. (1990), *Nonlinear Time Series* Oxford: Oxford University Press.

WARNER, B. AND MISRA, M. (1996), Understanding neural networks as statistical tools, *The American Statistician* 50, 284–293.

VENABLES, W.N. & RIPLEY, B. (1997), *Modern Applied Statistics with S-plus* (2nd ed), New York: Springer-Verlag

VON NEUMANN, J. (1956), The General and logical theory of automata, *The World of Mathematics, Volume 4*, ed. James R. Newman, New York: Simon and Schuster

WAN, E. A. (1993), Time series prediction using a connectionist network with internal delay lines. In *Forecasting the Future and Understanding the Past*, eds. Andreas S. Weigend & Neil A. Gershenfeld, Reading: Addison-Wesley.

WEISS, G. (1984), Half-hourly precipitation and streamflow, River Hirant, Wales, U.K., November and December, 1972. In *Data: A Collection of Problems from Many Fields for the Student and Research Worker* by D.F. Andrews & A. Hertzberg, New York: Springer-Verlag

WILSON, G.T. (1973). Comments on "Box-Jenkins seasonal forecasting: Problems in a case-study" by C. Chatfield and D.L. Prothero, *Journal of the Royal Statistical Society* A 136, 295–308.

WINKLER, R. L. & MAKRIDAKIS, S. (1983). The combination of forecasts. *J. R. Statist. Soc.* A 146, 150–157.

PERSONAL INFORMATION

Address: Department of Statistical and Actuarial Sciences,
University of Western Ontario,
London, Ontario N6A 5B7, CANADA
Date of Birth: July 12, 1974
Place of Birth: Japan

EDUCATION

B.Sc. (*Civil Engineering*), Department of Civil Engineering, Kyoto University, Kyoto, Japan, 1997.

THESIS

Graduation Thesis: Developement of the Criterion for Redundancy Feature of the Urban Highway by Topological Index; The Department of Civil Engineering, Kyoto University: March 1997.

CONFERENCE PRESENTATION

Visualization of Effects of Effluents; Statistical Society of Canada Annual Meeting; University of Sherbrooke, Sherbrooke; June 1998.

CONFERENCE PRESENTATION

Graduate Teaching Assistant: The Department of Statistical and Actuarial Sciences, University of Western Ontario: January 1998 - April 1999.