Here are some links to my public work samples:

Economics research project:
https://academicworks.cuny.edu/bb_etds/75/

Tableau sample visualization:
https://public.tableau.com/profile/helen4743#!/vizhome/Lesson2Workbook/CreditScoreMap

And two more attached:
Feature analysis (unsupervised learning) using PCA on public mortgage datasets in R
SAS modeling of public agriculture dataset

Some Datacamp certificates:

Intermediate PostgreSQL course
https://www.datacamp.com/statement-of-accomplishment/course/28ff58cb152eb5c6db447c53c854c69ce1e1de69

Git course
https://www.datacamp.com/statement-of-accomplishment/course/9cc52b5ccc62ec2afb03d16b46cffb36cc1fba01

Python supervised learning in scikit-learn
https://www.datacamp.com/statement-of-accomplishment/course/7e627d3f2e1d1c128eb94bf4fc38ab22ff03d608

Unsupervised learning in Python
https://www.datacamp.com/statement-of-accomplishment/course/c00acc844f8fcc488d2e990594a58bb2727ffb78

Data Analyst with Python certificate
https://www.datacamp.com/statement-of-accomplishment/track/f8f755fd38795f39ecfadffbdcc78c8cad186a75

Unsupervised Learning: PCA Analysis of Fannie Mae Third Quarter 2016 Mortgage Acquisition Data

Hanxiao Helen Yue

December 22, 2017

Time Series Analysis of Monthly Total Number of Pigs Slaughtered in Victoria, Australia, January 1980 - August 1995

Hanxiao Helen Yue

December 29, 2017

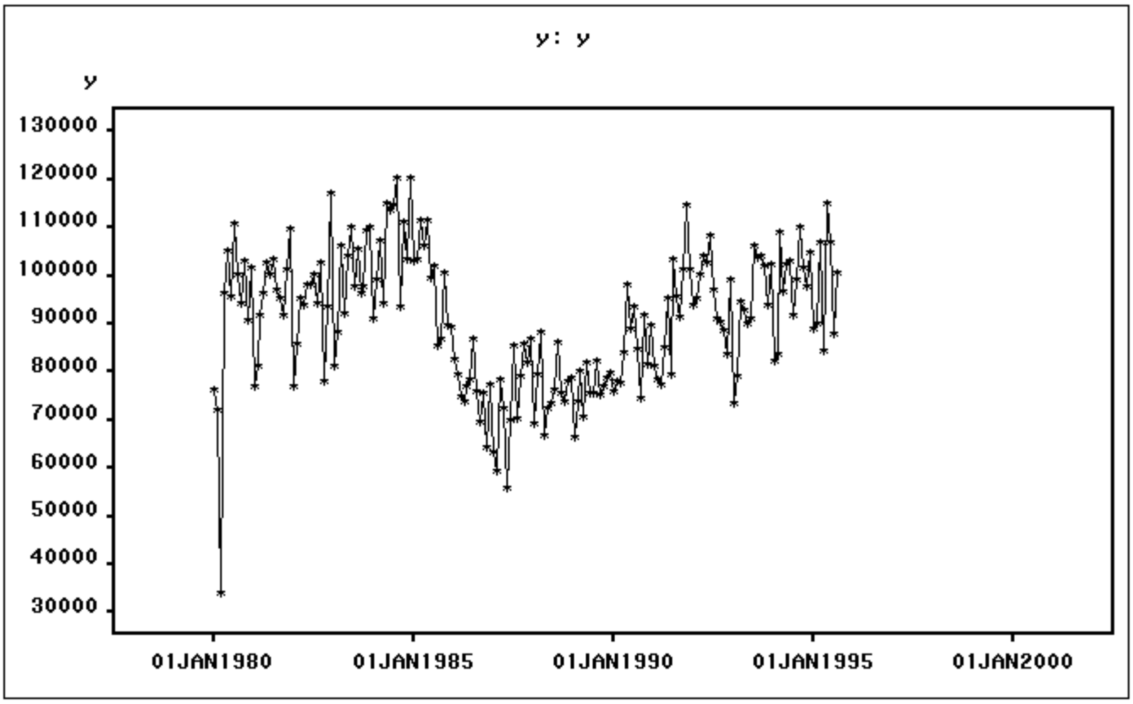
Contents

1 Setup & Description of Data

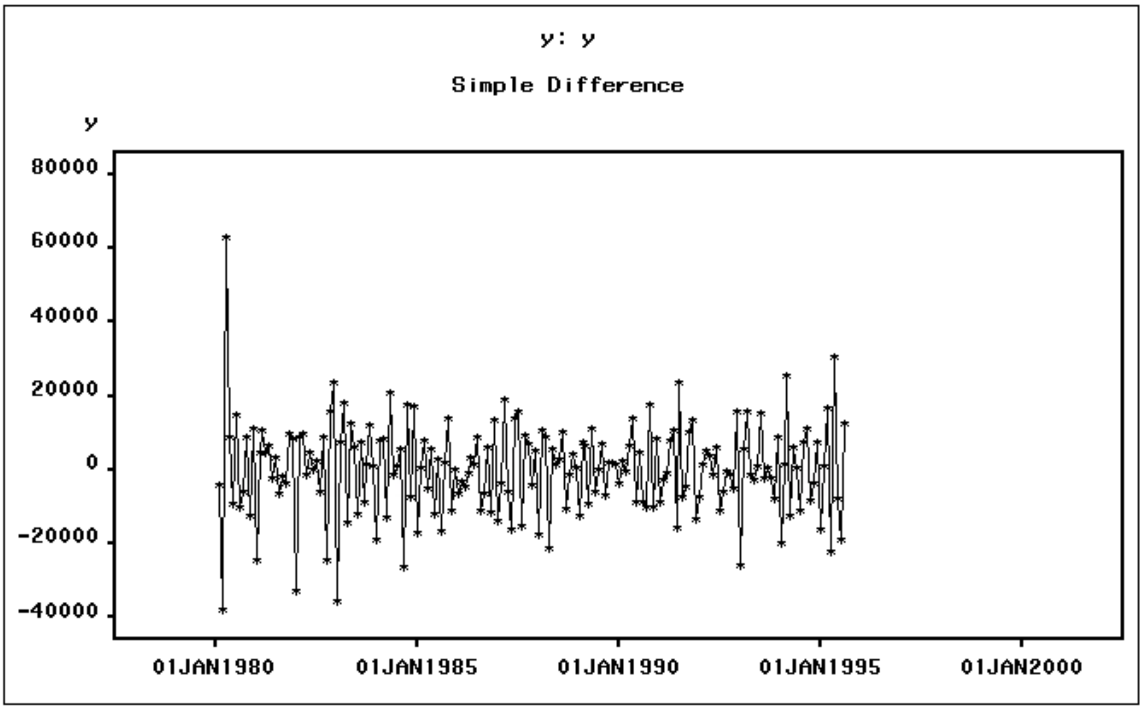
The dataset I received is a record of the total number of pigs that were slaughtered every month in Victoria, Australia, during the period from January 1980 to August 1995. There are 188 observations in total, consistent with the 188 months in this period. I exported the data to an xlsx file from the link: `https://datamarket.com/data/set/22ky/monthly-total-number-of-pigs-slaughtered\-in-victoria\-jan-1980-august-1995#ds=22ky&display=line`.
This data file came with a lot of extra headings and information in the top, so I had to reorganize the dataset to just contain a series of observations. I then tried to use the SAS import wizard to load the data in SAS, however, there were some errors. So I tried the following:

```
proc import out = work.pigs
        datafile="C:\Users\hanxiao.yue73\DOCUMENTS\My SAS Files\9.4\pigs.xlsx"
        DBMS=XLSX;
run;
```

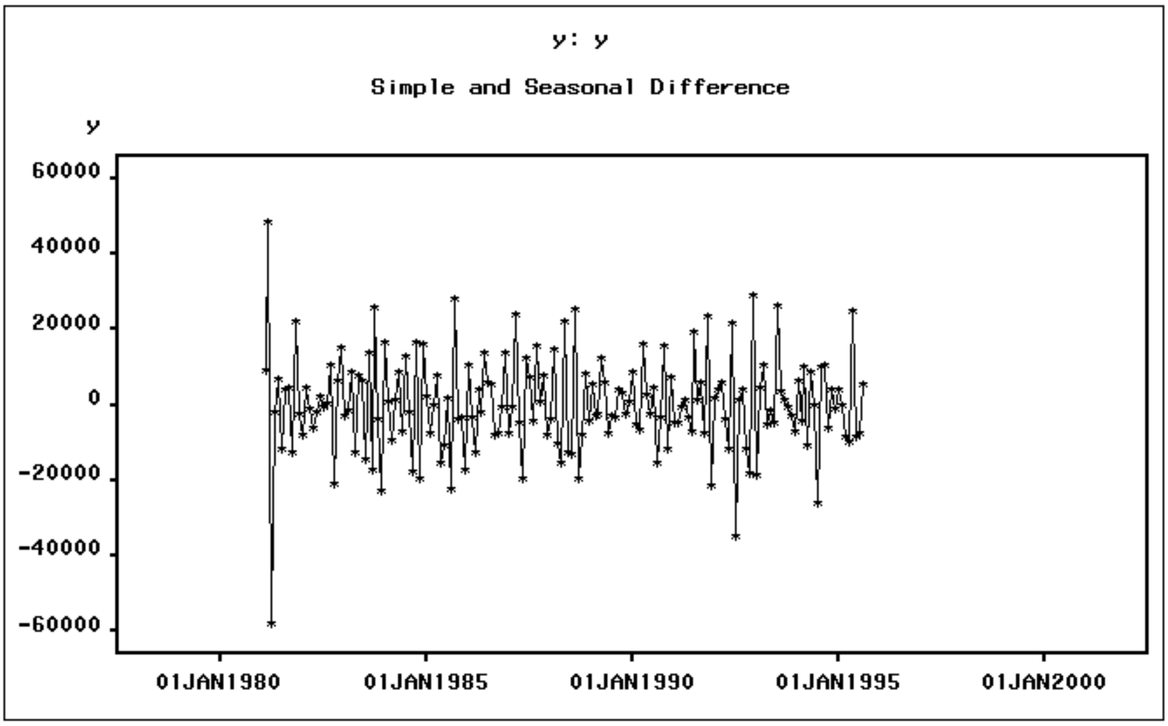
Using the Time Series Viewer, we create observation variables starting from the date 1980/01/01. Below is a visualization of the original data.



The time series viewer showed that the data stops at 1995/08/01 just as expected.
We note that the original data does not seem to be stationary (the overall mean and variance of the data does not remain constant over time). In fact, we see some seasonal fluctuations at the beginning of the year. Also, a large drop is observed around March 1980.
We try the simple and seasonal differencing transformations to see if we can introduce stationarity to the data. A simple differencing transformation showed:



We see that the level (mean) is immediately stabilized around 0. This is an important sign that when we fit the model, we will need a differencing of some sort. We also explore the additional of seasonal differencing to the simple differencing. Both adjustments seems to have similar effects on the data. However, these adjustments do not get rid of the outlier observation in March 1980, so we will need to add an intervention adjustment there.

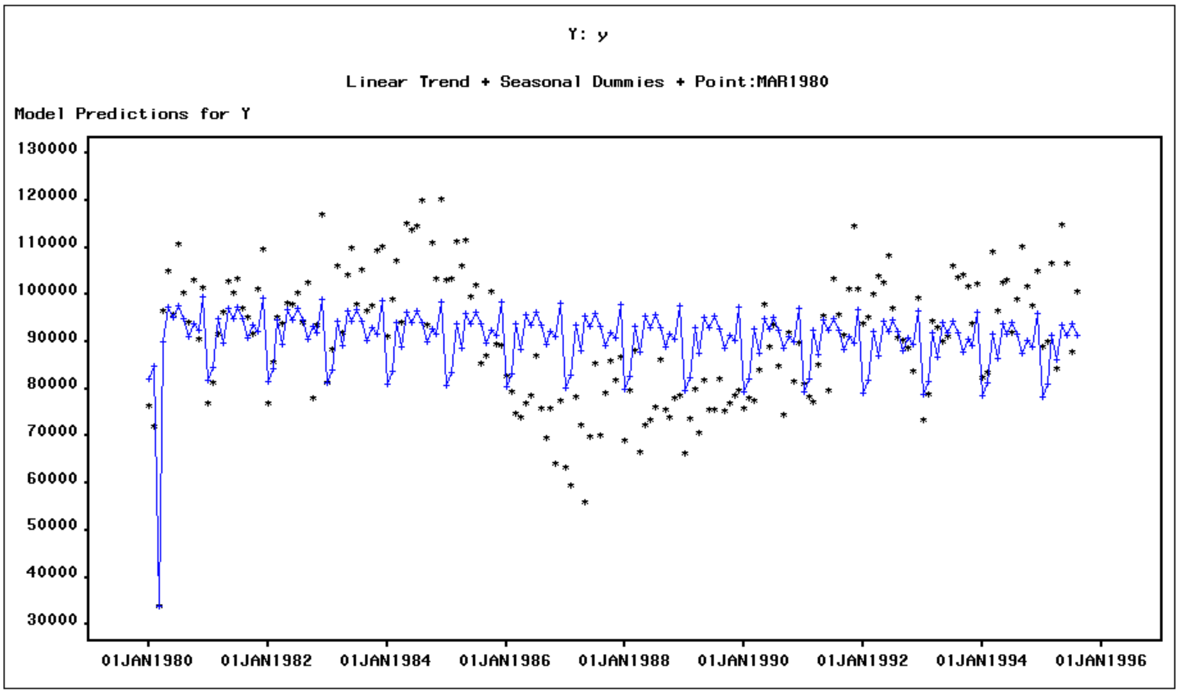


2 Fitting the Model

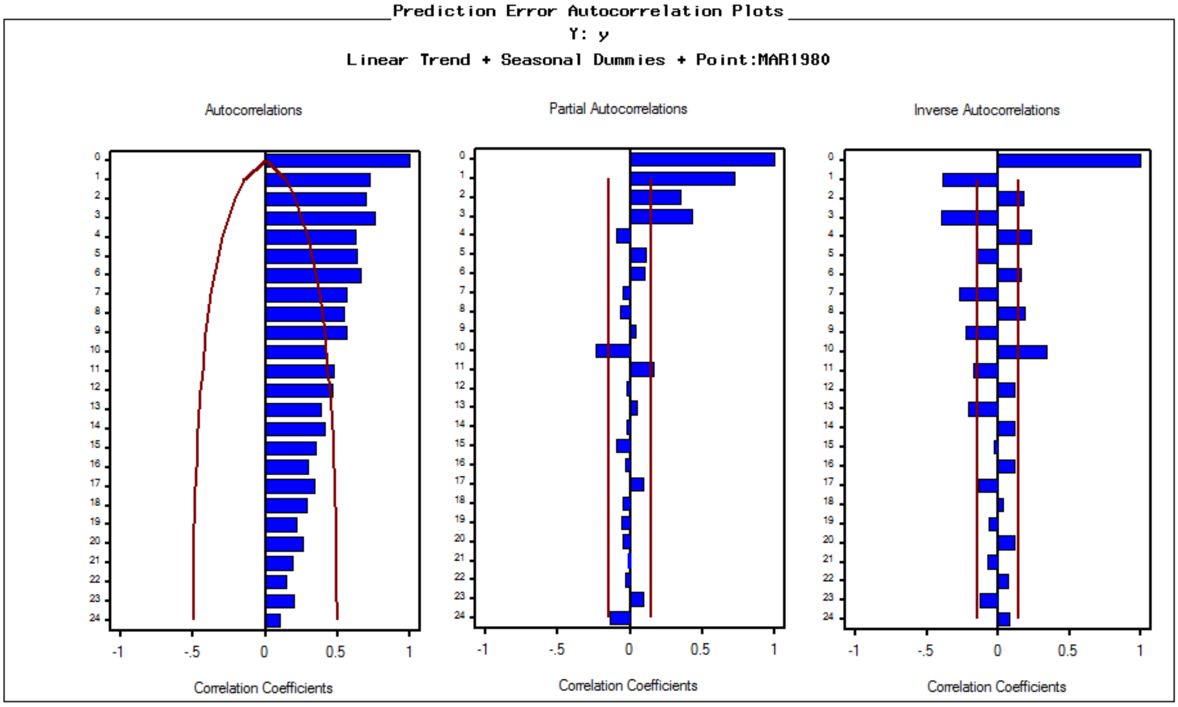
There are three main types of models we could choose from: Linear Regression, Exponential Smoothing, and Box Jenkins.

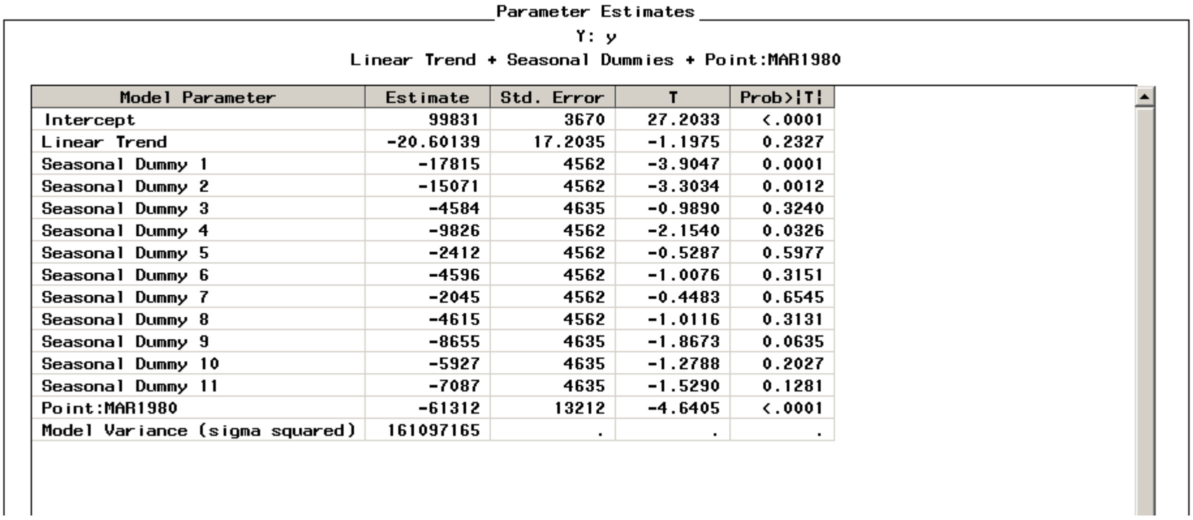
2.1 Linear Regression

We first try out the Linear Regression model with seasonal dummies, a linear trend, and an intervention point at March 1980. Here, we immediately see that the model does not adequately capture the observed variability of the data:



The root MSE comes out to 12210.7. Also, looking at the sample autocorrelation function, it doesn't cross below the bounds before lag 12, and also spectacularly failed the white noise test. This suggests that seasonal dummies were not enough to eliminate the non-stationarity of the data, simple differencing is also needed. Also, most of the seasonal dummy parameter estimates were insignificant, including the linear trend, so we can rule out a seasonal linear regression model.




Parameter Estimates

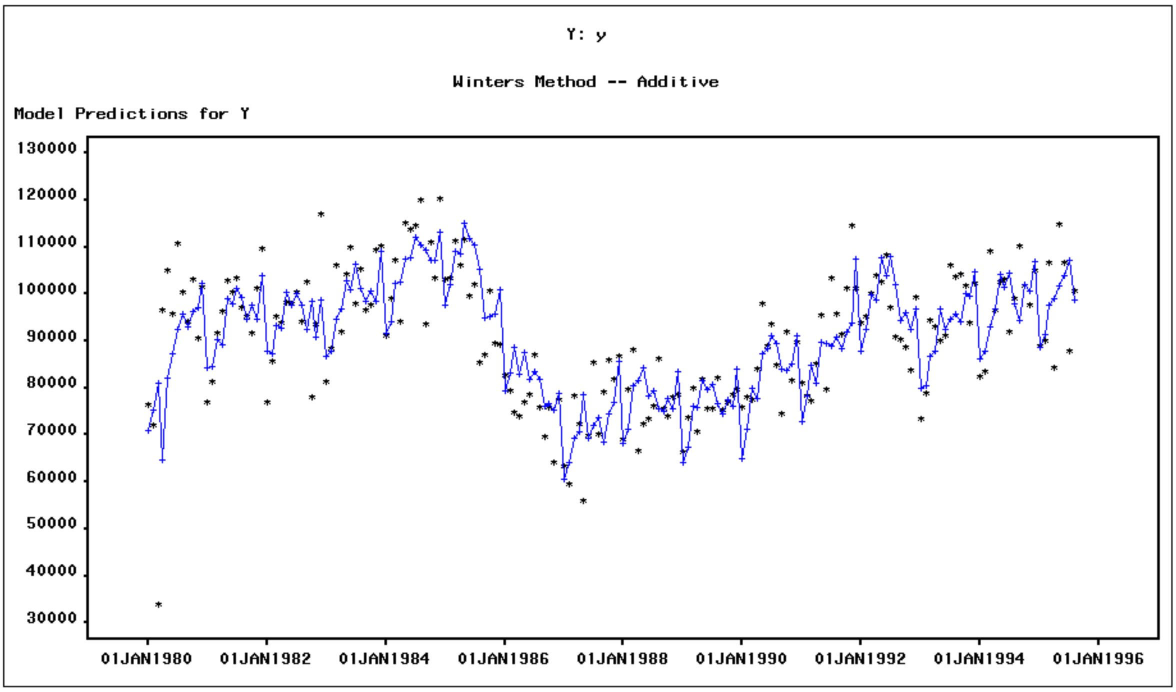
Y: y

Linear Trend + Seasonal Dummies + Point:MAR1980

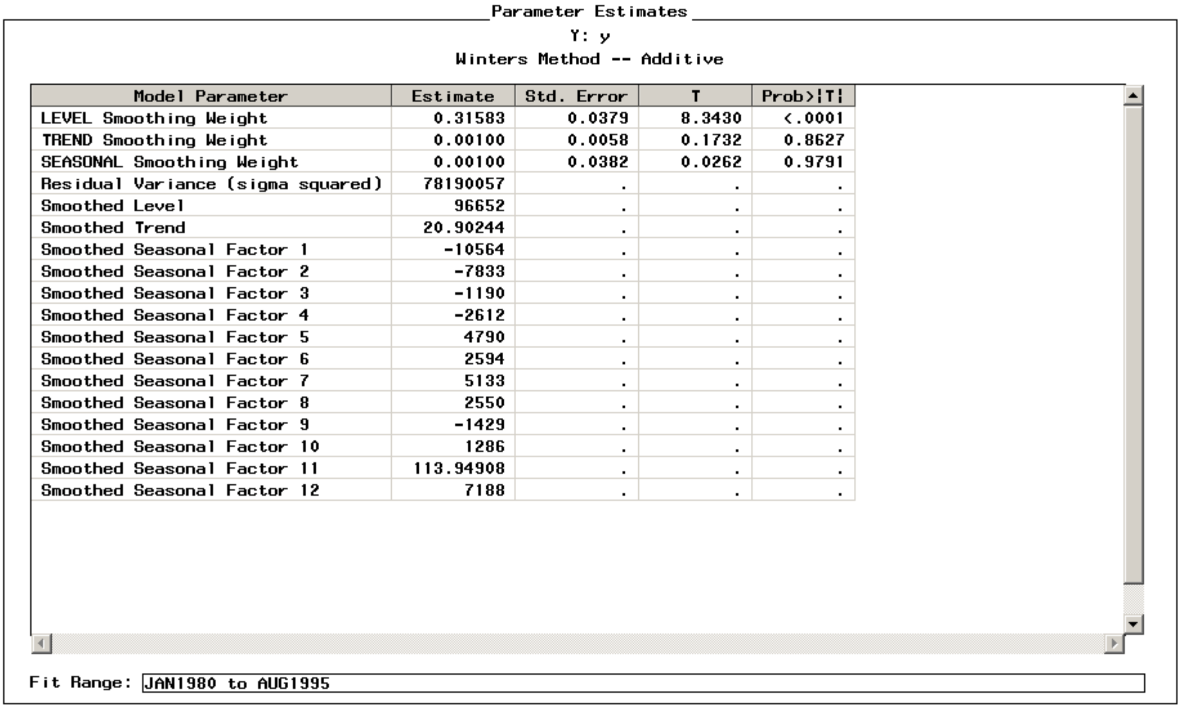
Model Parameter	Estimate	Std. Error	T	Prob>|T|
Intercept	99831	3670	27.2033	<.0001
Linear Trend	-20.60139	17.2035	-1.1975	0.2327
Seasonal Dummy 1	-17815	4562	-3.9047	0.0001
Seasonal Dummy 2	-15071	4562	-3.3034	0.0012
Seasonal Dummy 3	-4584	4635	-0.9890	0.3240
Seasonal Dummy 4	-9826	4562	-2.1540	0.0326
Seasonal Dummy 5	-2412	4562	-0.5287	0.5977
Seasonal Dummy 6	-4596	4562	-1.0076	0.3151
Seasonal Dummy 7	-2045	4562	-0.4483	0.6545
Seasonal Dummy 8	-4615	4562	-1.0116	0.3131
Seasonal Dummy 9	-8655	4635	-1.8673	0.0635
Seasonal Dummy 10	-5927	4635	-1.2788	0.2027
Seasonal Dummy 11	-7087	4635	-1.5290	0.1281
Point:MAR1980	-61312	13212	-4.6405	<.0001
Model Variance (sigma squared)	161097165	.	.	.

2.2 Exponential Smoothing

We next try to fit an exponential smoothing model. We use the Winter's smoothing model, which includes the level, growth, and seasonal factors. We specify that we want additive, because the seasonal variation seems to be constant. I was not able to add the intervention point because there was no option for it. The root MSE still decreased to 8771.7, so it was an improvement from the linear regression model.



Looking at the parameters, we see that only the level smoothing weight is significant in this situation, while the trend and seasonal weights are not significant. This supports with the earlier conclusions on the data, after being adjusted for seasonality, as having no significant trend, and that most of the monthly seasonal coefficients are insignificant.



Parameter Estimates

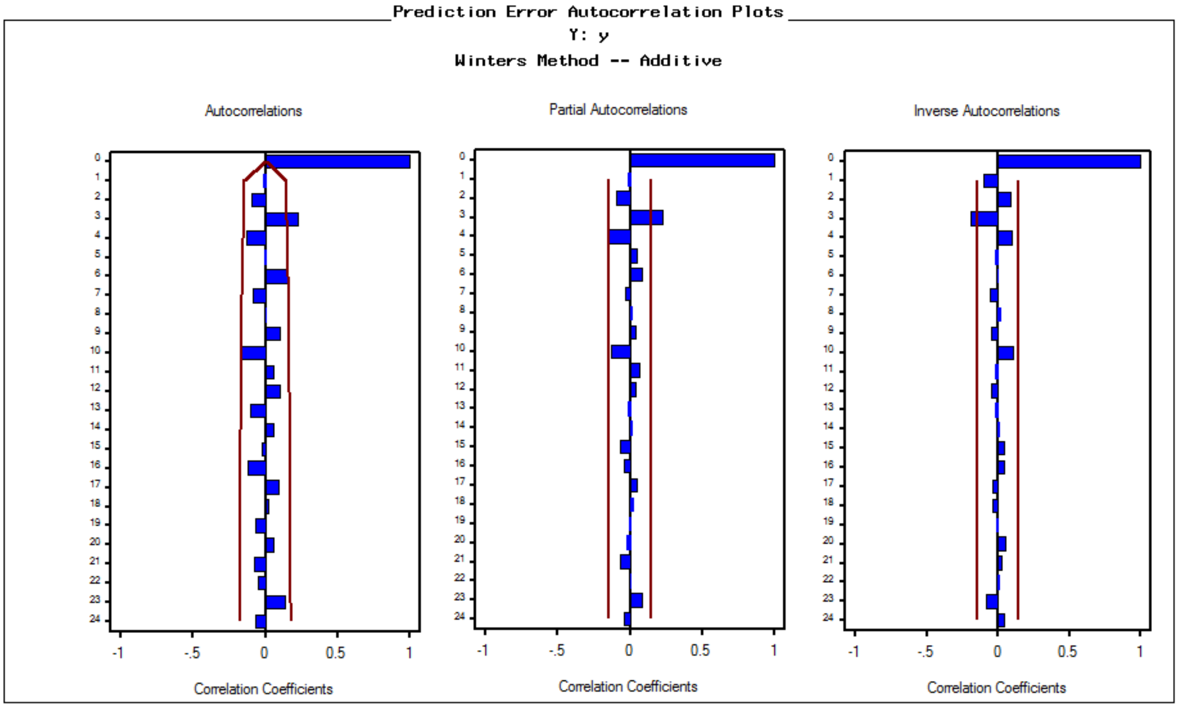
Y: y

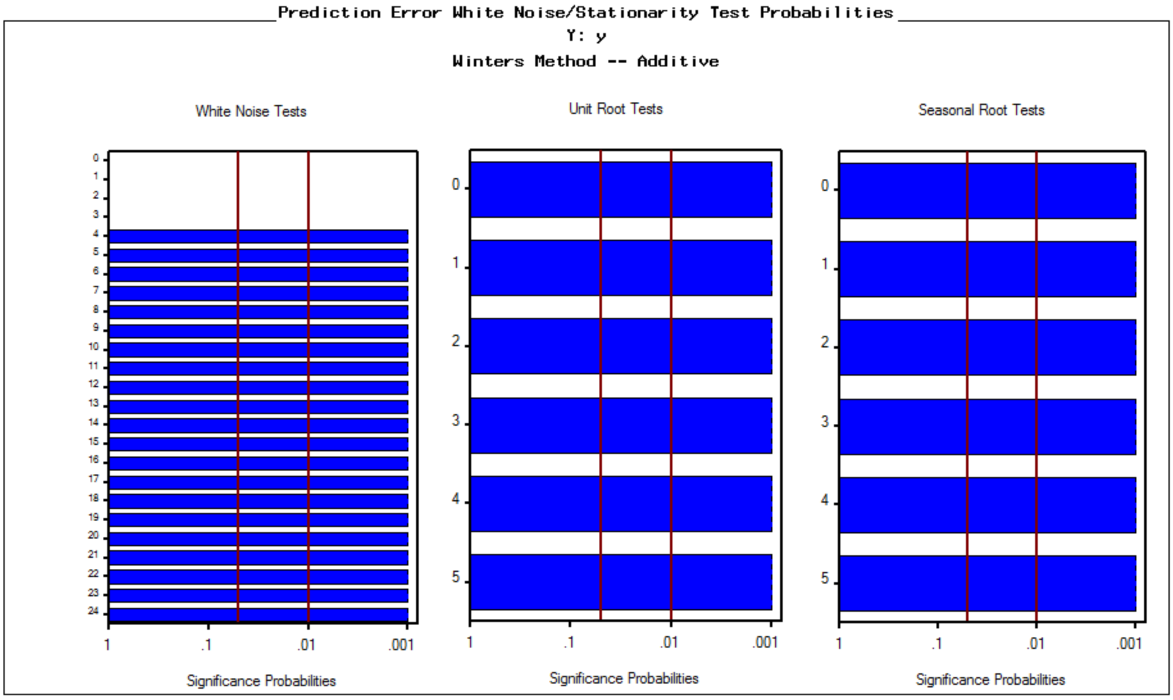
Winters Method -- Additive

Model Parameter	Estimate	Std. Error	T	Prob>¦T¦
LEVEL Smoothing Weight	0.31583	0.0379	8.3430	<.0001
TREND Smoothing Weight	0.00100	0.0058	0.1732	0.8627
SEASONAL Smoothing Weight	0.00100	0.0382	0.0262	0.9791
Residual Variance (sigma squared)	78190057	.	.	.
Smoothed Level	96652	.	.	.
Smoothed Trend	20.90244	.	.	.
Smoothed Seasonal Factor 1	-10564	.	.	.
Smoothed Seasonal Factor 2	-7833	.	.	.
Smoothed Seasonal Factor 3	-1190	.	.	.
Smoothed Seasonal Factor 4	-2612	.	.	.
Smoothed Seasonal Factor 5	4790	.	.	.
Smoothed Seasonal Factor 6	2594	.	.	.
Smoothed Seasonal Factor 7	5133	.	.	.
Smoothed Seasonal Factor 8	2550	.	.	.
Smoothed Seasonal Factor 9	-1429	.	.	.
Smoothed Seasonal Factor 10	1286	.	.	.
Smoothed Seasonal Factor 11	113.94908	.	.	.
Smoothed Seasonal Factor 12	7188	.	.	.

Fit Range: JAN1980 to AUG1995

An inspection of the sample autocorrelation function and partial autocorrelation functions shows significant autocorrelations at lag 3. It also clearly fails the white noise test.



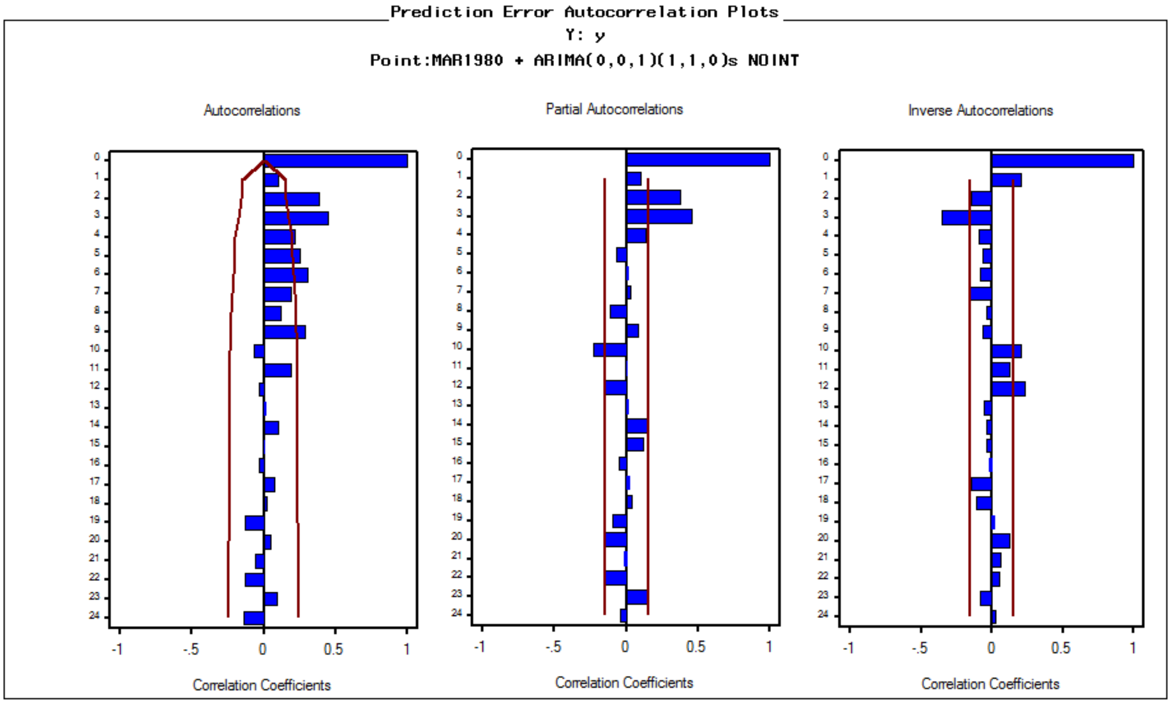


Based on these observations, we can clearly see that the Winters Additive model is insufficient.

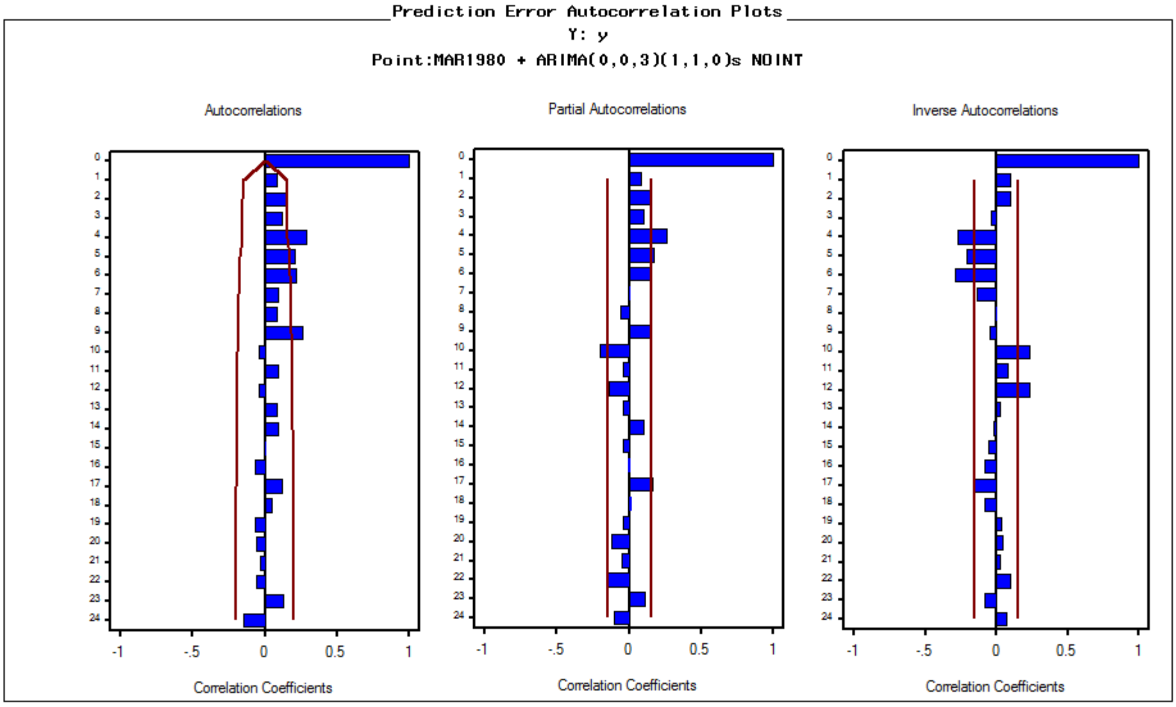
2.3 Box-Jenkins Methods

We now try out the Box-Jenkins methods of AR, MA and ARIMA combinations. The autocorrelation and sample autocorrelation functions of the previous methods all hinted at this approach. In the linear trend, we noticed that lags 1 to 12 in the sample autocorrelation function were significant, and in the partial sample autocorrelations, lags 1 to 3 were significant. We start off simple, hoping that having small lags would do away with the autocorrelation. The first model I had a regular nonseasonal MA(1), a seasonal AR(1), and seasonal difference of lag 1. The root MSE starts out at 10643.2, so it is clearly worse than the exponential smoothing model. However, this method allows us room for improvement.

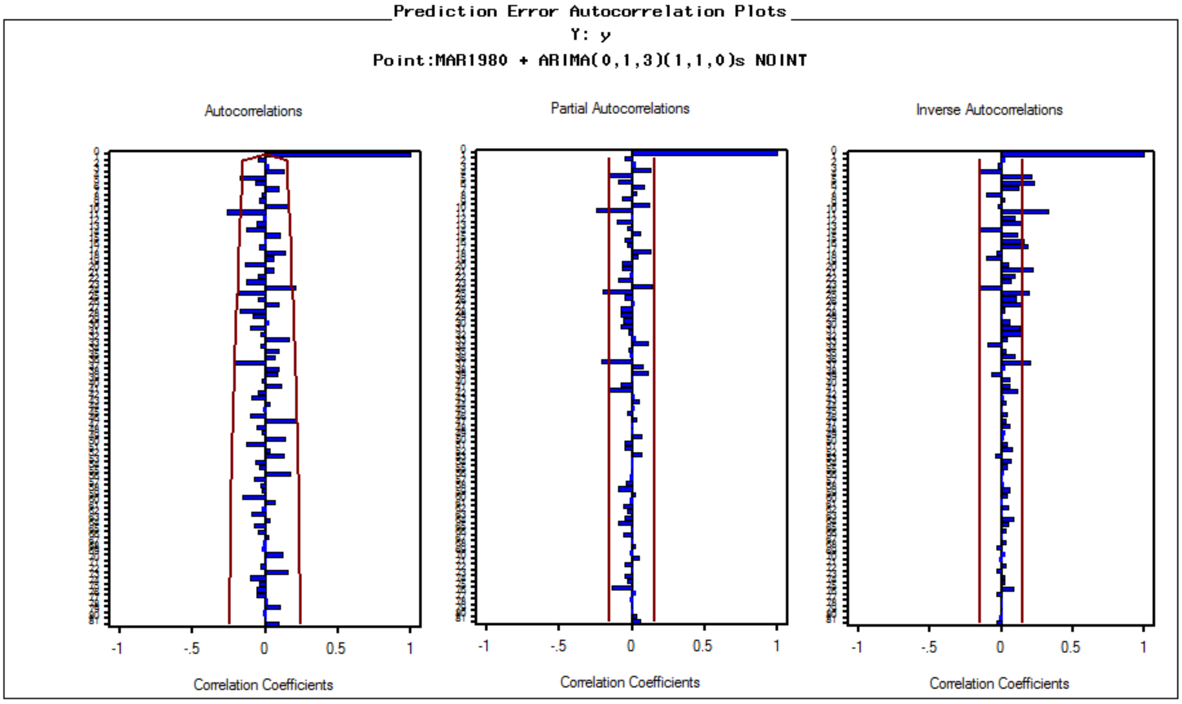
An inspection of the autocorrelation functions shows that lags 2 and 3 are still significant in both the sample and partial autocorrelations.



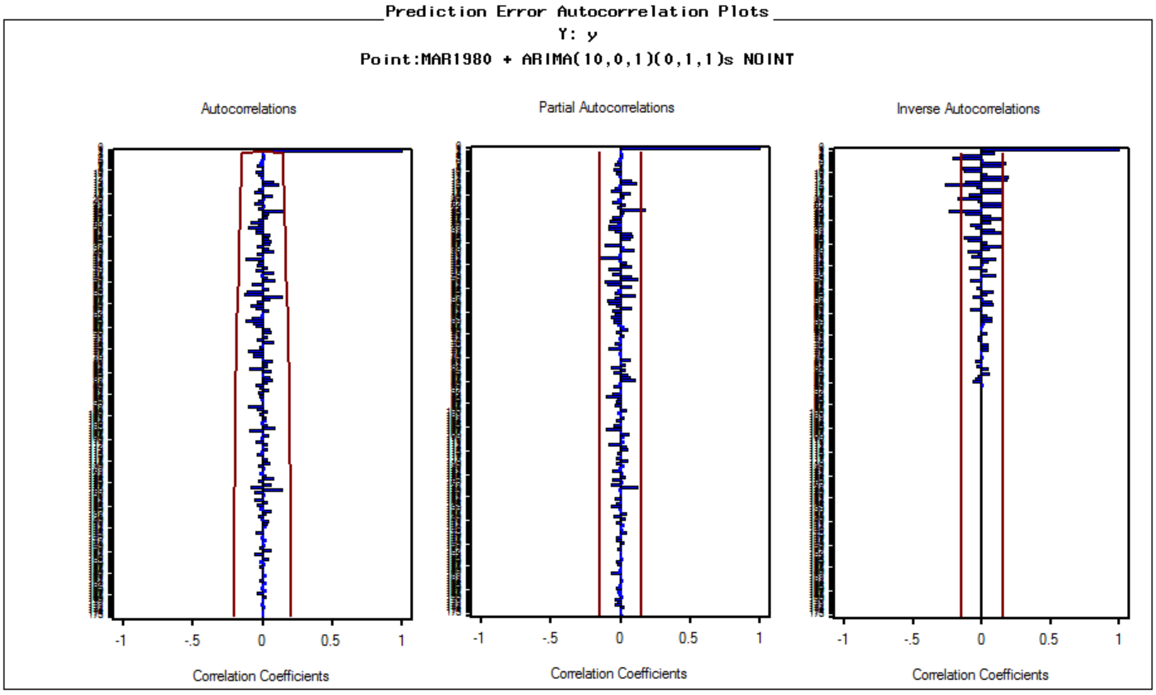
Based on these observations, we decide to modify the MA to include up to 3 to the model. This gives us the new autocorrelations charts.

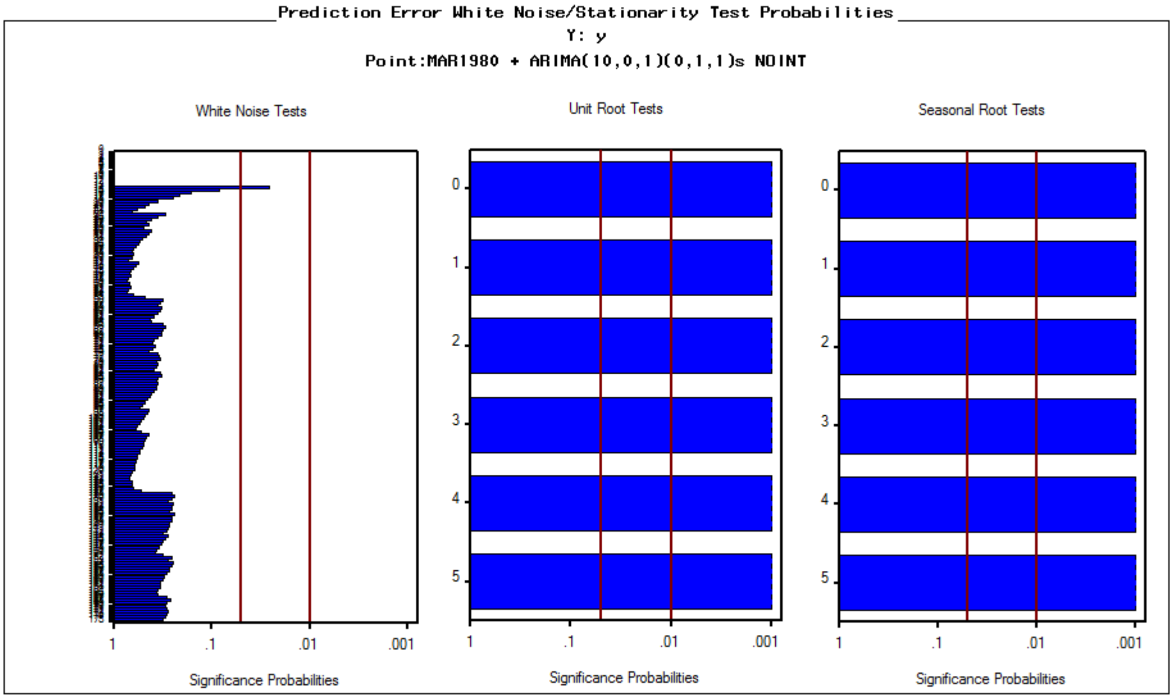


However, I noticed that the simple differencing component was gone, and remembered that earlier, in time series viewer, simple differencing significantly transformed the data to be more stationary. So I decided to add a d=1 to the nonseasonal ARIMA model. The autocorrelation functions turned out to be like the below.

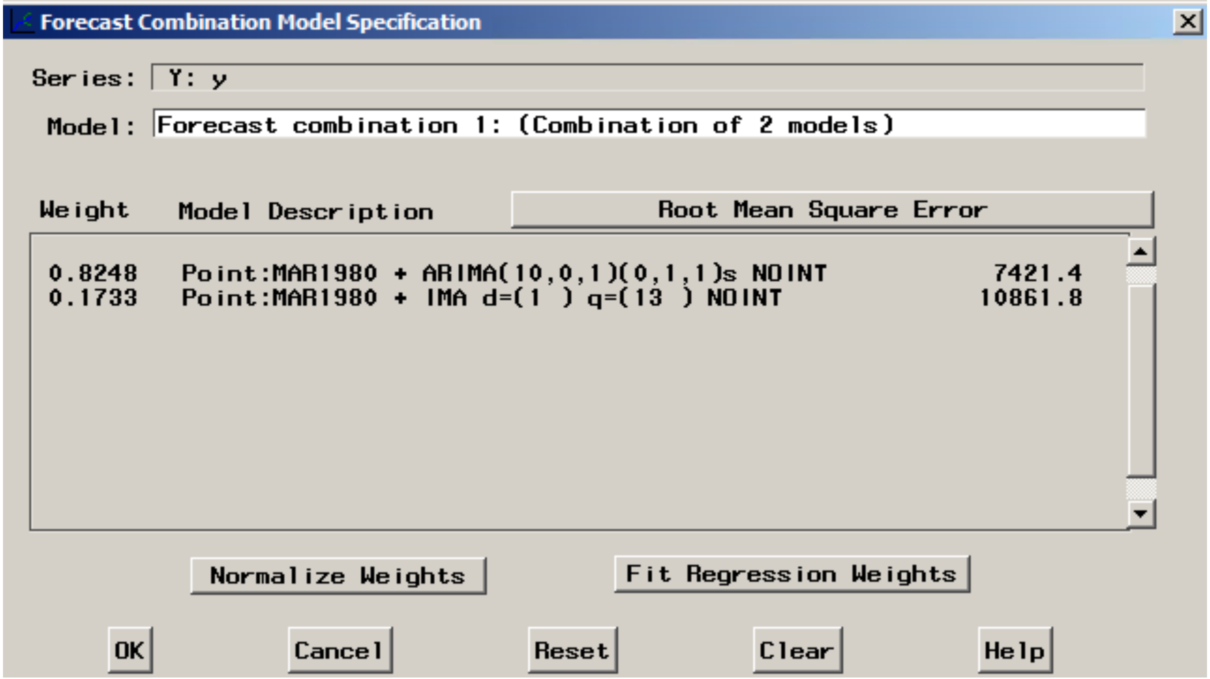


Lag 10 still turned out to be significant, in both the sample autocorrelation function and partial autocorrelation function, so I tried to modify the model just a little to have the only change be an AR(10) in the nonseasonal section. However, this turned out to be an invalid model, so I needed to play around with the settings a bit more. I took out the non-seasonal differencing component. Also, I couldn't keep the 3 either, and since the non-seasonal component of AR already accounted for 10 lags, I took out the seasonal AR lag 1. However, leaving the seasonal ARIMA like (0,1,0) also invalidated the model, so I added a seasonal MA(1) to it. The model ended up being an ARIMA(10,0,1)(0,1,1)s NOINT with an intervention, and root MSE = 7421.4. At this point, the autocorrelation function was looking good, and almost passed the white noise test, albeit there was one significant lag at 13.





I wanted to just include a parameter of t=13 in a moving average, nothing before that, and add back the simple differencing which was shown to transform the data well in the time series viewer, so I created a factored MA model with just an intervention and simple differencing d = 1. I was looking for a way to combine the seasonal differencing option with the specific factored MA model, so I used the option of Combination of Forecasts. This gave me the ability to directly target the significant autocorrelation at t=13 and include the simple differencing transformation. To create this model, I selected the ”Fit Regression Weights” option which automatically scaled their weights.

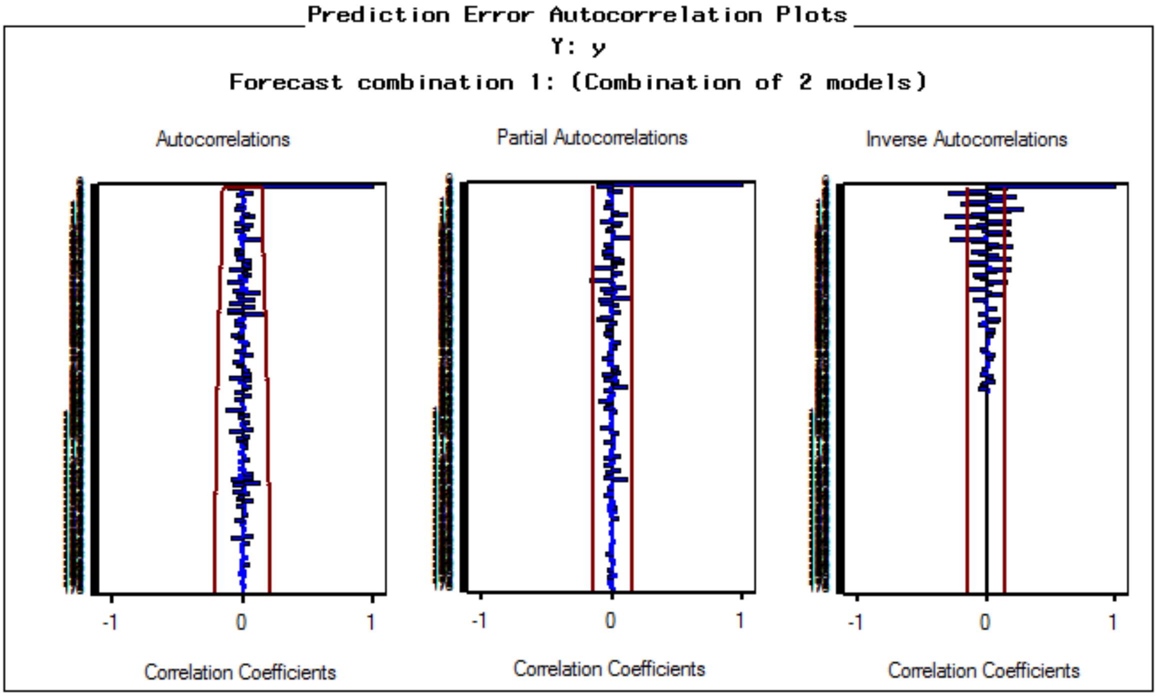


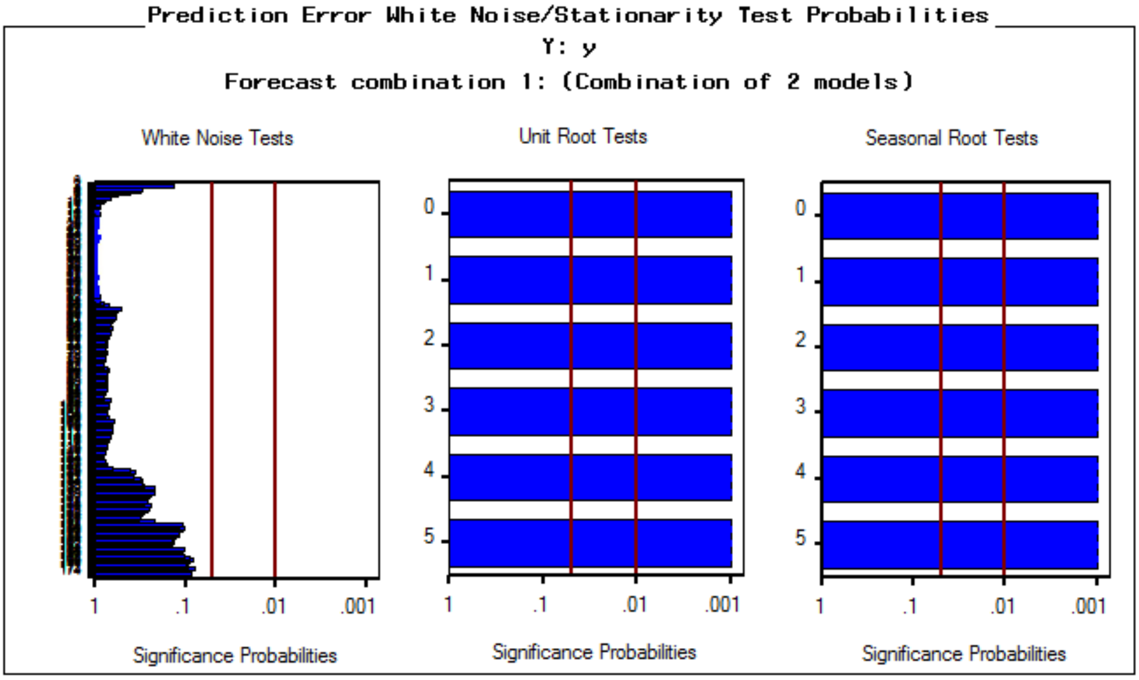
Series: Y: y

Model: Forecast combination 1: (Combination of 2 models)

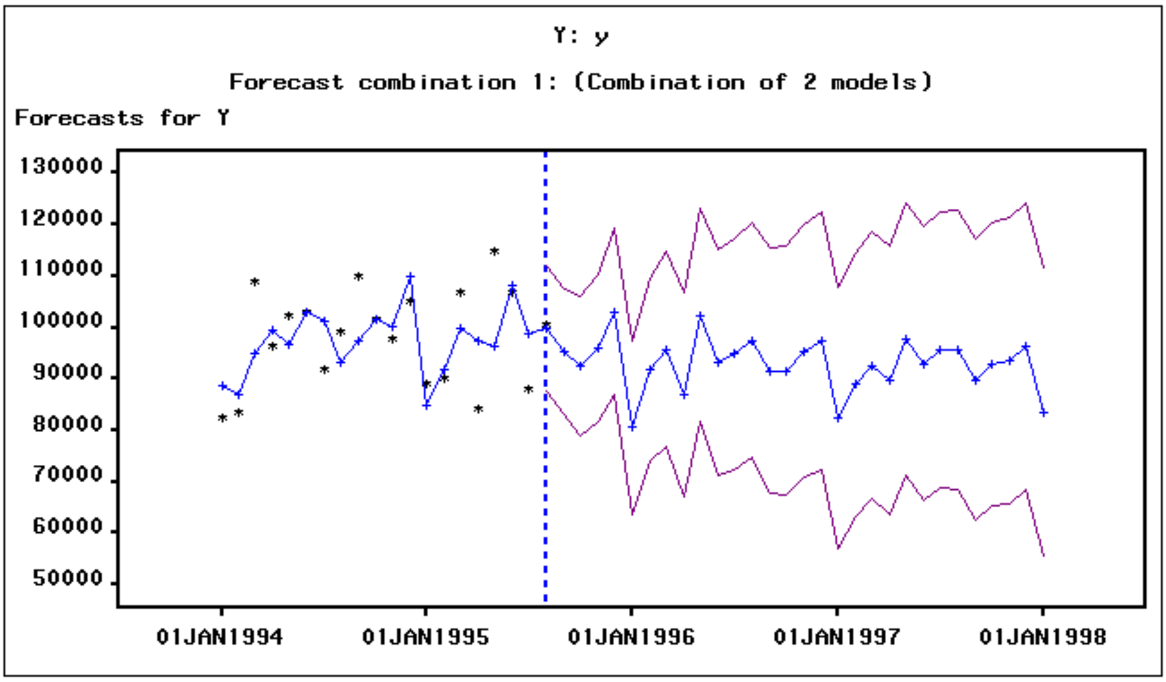
```
Weight    Model Description                          Root Mean Square Error

0.8248    Point:MAR1980 + ARIMA(10,0,1)(0,1,1)s NOINT          7421.4
0.1733    Point:MAR1980 + IMA d=(1 ) q=(13 ) NOINT            10861.8
```

The white noise test of the combination model finally passed the white noise test, and did well on the autocorrelations charts too.

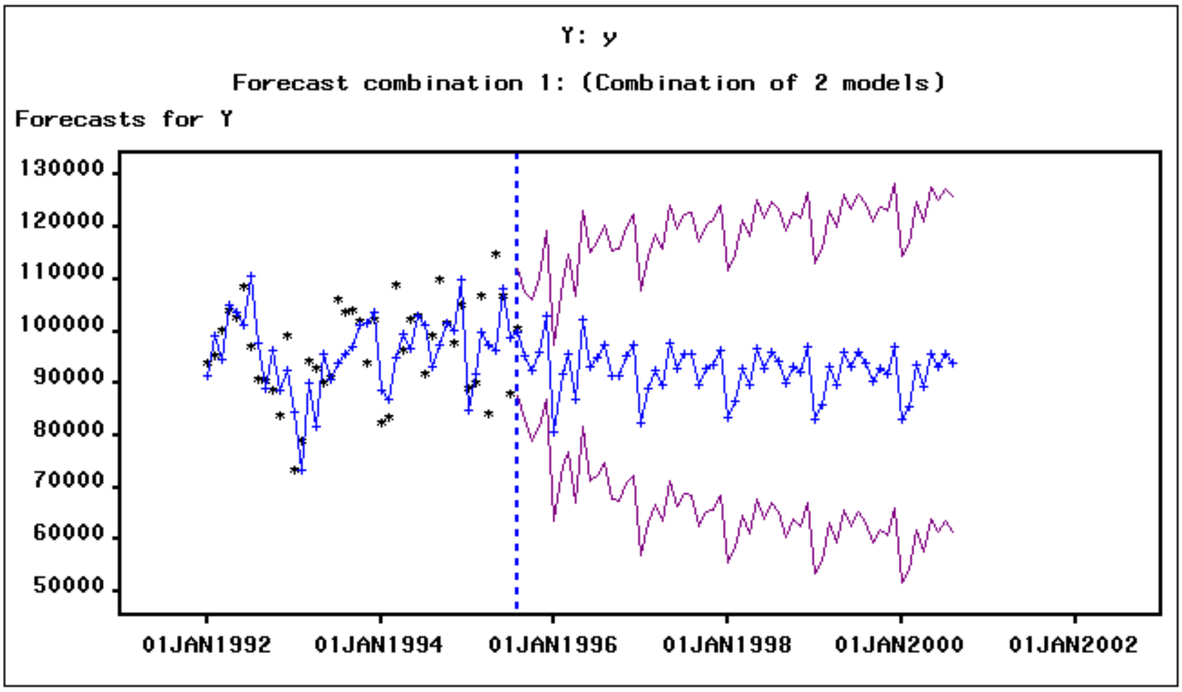




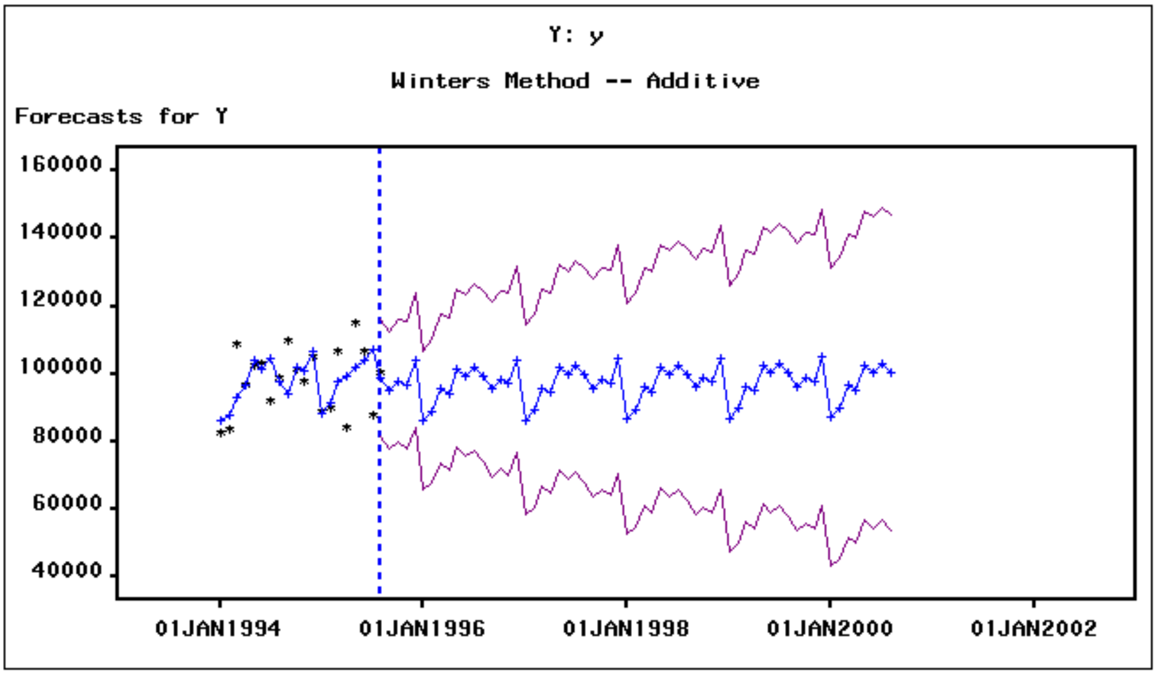
The only qualm I have about this is that the prediction plot tends to fan out, over a longer period, such as five years. However it does a good job of forecasting just one or two years ahead.



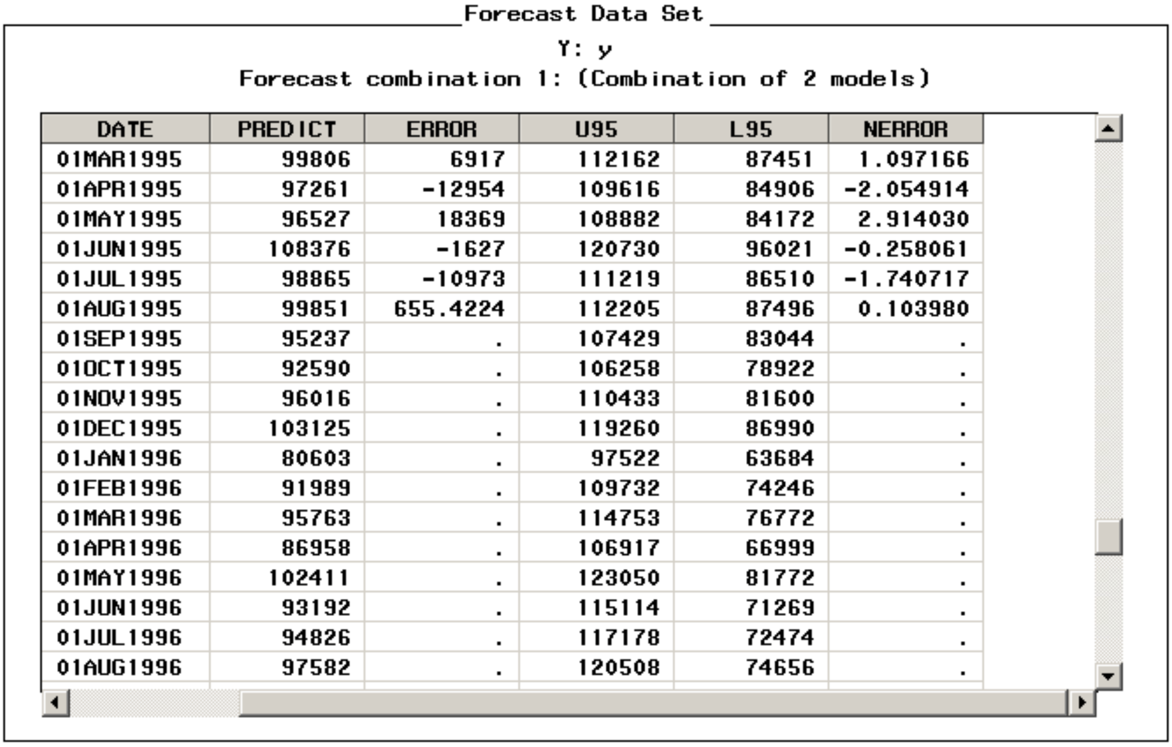
We do observe a bit of widening confidence intervals as the forecasting time range increases, in this case, shown to around five years into the future.



However, even this slight fanning out is still better than the more pronounced fanning out of the alternative winter's method seen below, which was suggested automatically by SAS.



The confidence intervals for forecasts given one year into the future are

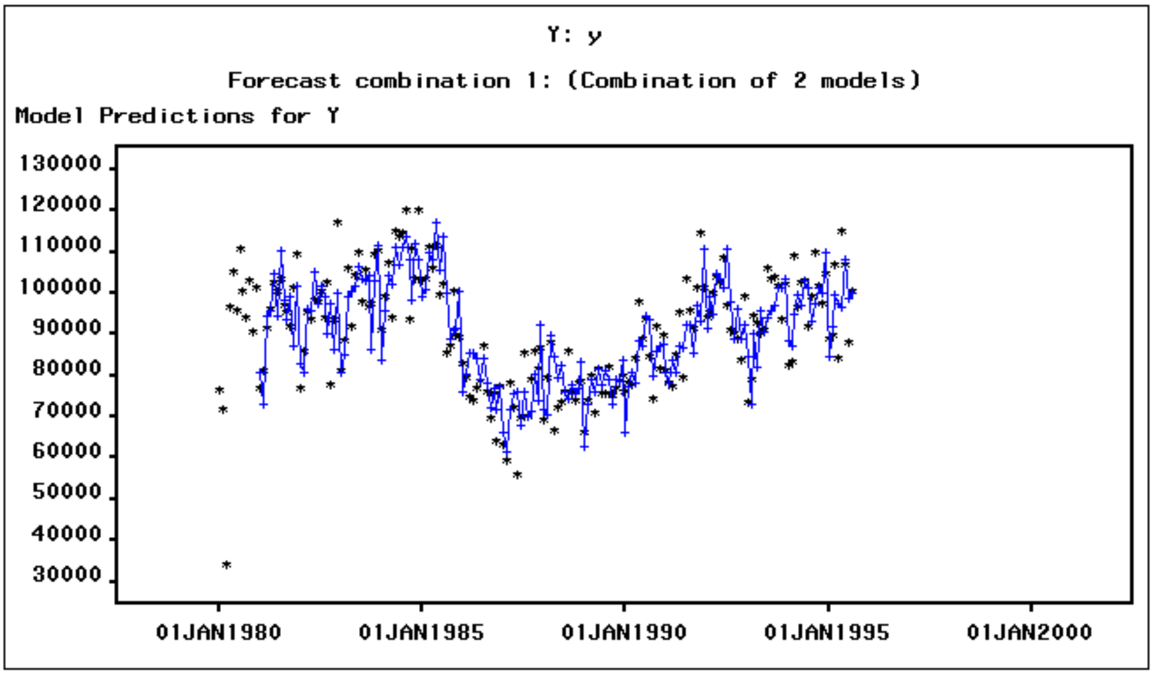

Forecast Data Set

Y: y

Forecast combination 1: (Combination of 2 models)

DATE	PREDICT	ERROR	U95	L95	NERROR
01MAR1995	99806	6917	112162	87451	1.097166
01APR1995	97261	-12954	109616	84906	-2.054914
01MAY1995	96527	18369	108882	84172	2.914030
01JUN1995	108376	-1627	120730	96021	-0.258061
01JUL1995	98865	-10973	111219	86510	-1.740717
01AUG1995	99851	655.4224	112205	87496	0.103980
01SEP1995	95237	.	107429	83044	.
01OCT1995	92590	.	106258	78922	.
01NOV1995	96016	.	110433	81600	.
01DEC1995	103125	.	119260	86990	.
01JAN1996	80603	.	97522	63684	.
01FEB1996	91989	.	109732	74246	.
01MAR1996	95763	.	114753	76772	.
01APR1996	86958	.	106917	66999	.
01MAY1996	102411	.	123050	81772	.
01JUN1996	93192	.	115114	71269	.
01JUL1996	94826	.	117178	72474	.
01AUG1996	97582	.	120508	74656	.

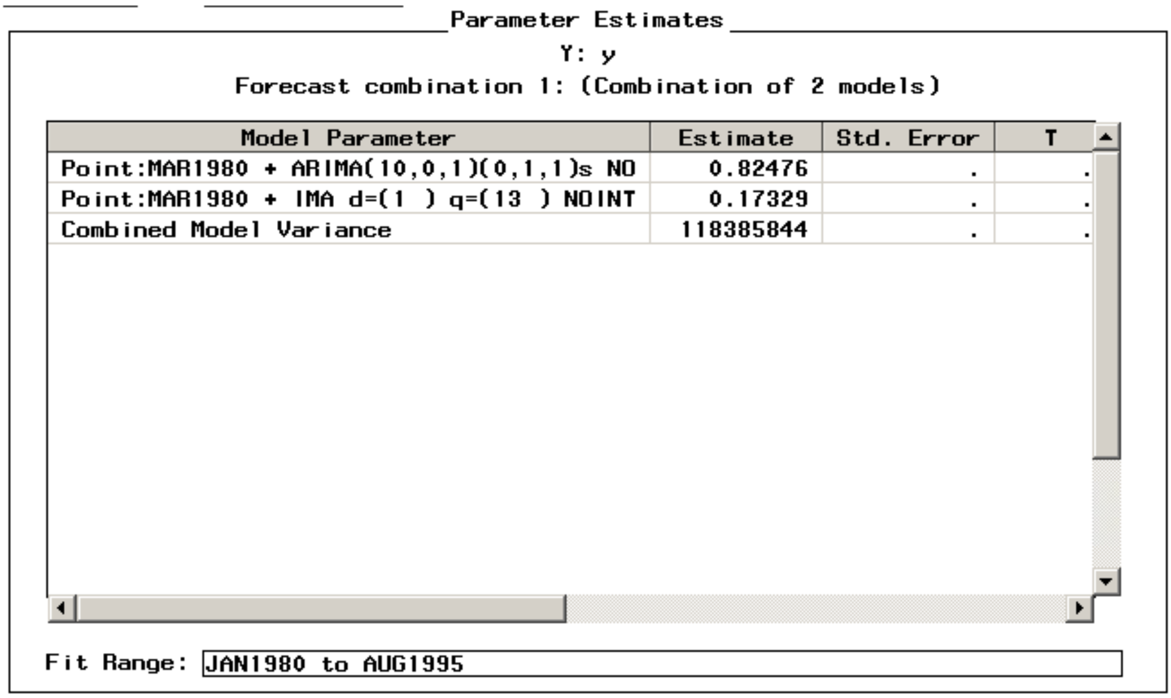
The final model is shown below. It does a good job of fitting most of the datapoints.



3 Conclusion & Practical Implications

From this analysis, we learned that the the monthly slaughter of hogs in Victoria, Australia, is affected by seasonal cycles, and the average levels and fluctuations in number change over time, hence, it is nonstationary. There seemed to be a sharp decrease in number of hog slaughtered in March 1980. However, through applying seasonal differencing and simple differencing transformations to the data, we were able to transform it to a stationary dataset that can be fit with a Box Jenkins ARIMA model. Based on the model, we predict, for the upcoming August 1996, that around 97582 hogs will be slaughtered that month.

4 Appendix



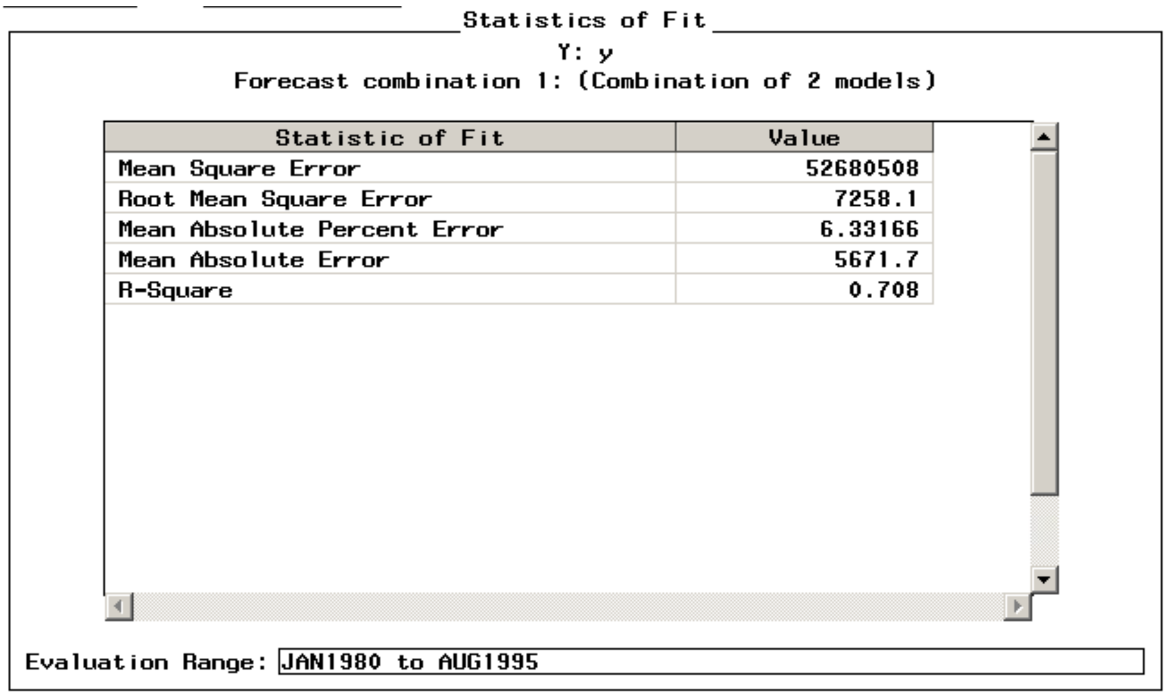
Parameter Estimates

Y: y

Forecast combination 1: (Combination of 2 models)

Model Parameter	Estimate	Std. Error	T
Point:MAR1980 + ARIMA(10,0,1)(0,1,1)s NO	0.82476	.	.
Point:MAR1980 + IMA d=(1) q=(13) NOINT	0.17329	.	.
Combined Model Variance	118385844	.	.

Fit Range: JAN1980 to AUG1995



Statistics of Fit

Y: y

Forecast combination 1: (Combination of 2 models)

Statistic of Fit	Value
Mean Square Error	52680508
Root Mean Square Error	7258.1
Mean Absolute Percent Error	6.33166
Mean Absolute Error	5671.7
R-Square	0.708

Evaluation Range: JAN1980 to AUG1995

Contents

1 Motivation

1.1 Supervised vs. Unsupervised Learning

Throught the semester, we have only worked with supervised methods, which aim to get the relationship between independent variables and a dependent output variable, whether in a classification (qualitative) or regression (quantitative) context. However, unsupervised methods have a completely different objective. According to a Stanford lecture on unsupervised methods, the aim of these methods is to understand and get meaningful relationships between the independent variables in a dataset, no dependent variables are considered. Oftentimes, insights from unsupervised methods can be used to simplify datasets that have too many categories, as which is usually the case in practical settings. The problems of datasets with too many categories is discussed in the section on Curse of Dimensionality. Both unsupervised and supervised learning methods can be used in the data analysis process-unsupervised methods help us summarize the meaningful relationships and independent variables in a dataset, as a pre-processing measure, and later, supervised methods can be adopted on the simplified dataset to derive any relationships between the independent and dependent variables.

1.2 Principal Components Analysis

Much of what I have learned here comes from the ISLR book, in the section on Principal Components Analysis in the chapter on Unsupervised Learning (Chapter 10), unless cited otherwise. Principal Components Analysis (PCA) is an unsupervised learning method that computes the principal components of a dataset. Here, we think of a dataset as an $n \times p$ dimension data matrix. n indicates the number of observations, while p indicates the number of categories (predictors). So each observation can be visualized as a $1 \times p$ row vector with p columns. The principal components indicate the underlying structure of the data matrix, and act as the directions where the most variation is observed (Dallas, 2013). We want the direction of most variation because variation is a measure of information-the more variation along that direction, the more information we get from this direction, and perhaps, we won't need some other directions if they capture the same information(exhibit redundancies). Although by default, PCA will compute p components, one for each of the original predictors, we will be able to assess them based on their order of importance, and hopefully eliminate some. The principal components computed from PCA will give us a set of orthogonal vectors (a set of axes) that is an alternative projection of the data. So we will be able to reorient the data from the the set of axis determined by its original predictors to a new set of axis determined by its principal components. As hinted above in the note about variability, not all components are equally important, and PCA gives us a basis for eliminating the non-important vectors to reduce the dimensions of a dataset to a set of simpler underlying relationships. Intuitively, the principal components represent a new set of axes to view a high dimensional dataset based on the directions the data varies the most in, and the objective is to be able to pick a smaller set of principal components to represent the data without compromising much of the variability.

1.3 Technical Explanation of Principal Components Analysis

Principal Components Analysis is mathematically based in linear algebra as an optimization problem. The principal components $(Z_1, Z_2, ..., Z_p)$ are linear combinations of the original predictors with coefficients that are constrained to have sum of squares = 1. In practice, this is called scaling the variance, and we do this because otherwise a variable that just happens to have large absolute value would have a large variance, even though other variables with smaller absolute values have larger relative variances (ie. distance values would have larger absolute values variances than human height data). We don't want to be left to the mercy of the original scaling of the different categorical variables, hence we scale to convert to relative variance. Mathematically, the jth principal component would be represented as:

$$Z_j = \phi_{1j}X_1 + \phi_{2j}X_2 + ... + \phi_{pj}X_p$$

and the coefficient constraints of the jth principal component(the loading of X_p onto the Z_jth principal component) is represented as

$$\Sigma_{i=1}^{p}\phi_{ij}^2 = 1$$

The first principal component in a dataset, which we are most interested in, would be the linear combination Z_1 that maximizes the variance. To get the first principal component, we need to solve the optimization problem that maximizes the sample variance of the scores (explanation to follow) via maximizing the loadings onto the first component $\phi_{11}, ..., \phi_{p1}$

$$\underset{\phi_{11},...,\phi_{p1}}{\text{maximize}} \left(\frac{1}{n} \Sigma_{i=1}^{n} z_{i1}^2 \right)$$

subject to the constraint

$$\Sigma_{j=1}^{p} \phi_{j1}^2 = 1$$

where

$$z_{i1} = \phi_{11} x_{i1} + \phi_{21} x_{i2} + ... + \phi_{p1} x_{ip}$$

Note, the $z_{i1}..z_{n1}$ are different from Z_1! The Z_p's are vectors, because the $X_1, X_2, ..., X_p$ are vectors. $X_1, X_2, ..., X_p$ are a set of column vectors of dimension $n \times 1$, so Z_p is also dimension $n \times 1$. For example X_1 is a vector that consists of all the entries of category 1 in the data, ie: all the whisker lengths of a population of cats. However, z_{i1} is not a vector because the $x_{i1}, x_{i2}, ..x_{ip}$ are numbers, that indicate the numerical values of the p predictors along each observation i of the n total observations. So the z_{ij} are the individual "scores" of the Z_p (the individual row entries of the Z_p that is dimension $n \times 1$). They can be thought of as scores because they measure how well each observation vector fits along the principal component. To summarize, the objective is to maximize the sample variance of the scores of the first component, via manipulating the loadings onto the first component. There is an alternative approach I read by Chris Nicholson from Skymind that you can use to find the principal components of the data, if you have the covariance matrix of the data handy. To do this, we need to understand the eigenvector and eigenvalue concepts. Eigenvectors are the directions of a matrix transformation (the covariance matrix which describes the spread of the data). Mathematically, it is a vector $\mathbf{v}$ that responds to the matrix transformation just as if the matrix was a scalar coefficient. So you have

$$\mathbf{Av} = \lambda \mathbf{v}$$

All $n \times n$ square matrices have n linearly independent, orthogonal eigenvectors. Eigenvalues are simply the magnitudes corresponding to each eigenvector. We can find the eigenvalues analytically by solving for the λ_n that satisfy the characteristic equation

$$det(A - \lambda I) = 0$$

where A is the covariance matrix and I is the identity matrix. The eigenvectors for each eigenvalue are found by solving for $\mathbf{X}$ in the equation

$$(A - \lambda I)X = 0$$

The eigenvectors and eigenvalues of the covariance matrix are direction and magnitudes of the principal components, and the eigenvalues λ can be used to rank the eigenvectors by order of importance.

2 Curse of Dimensionality & Bias Variance Tradeoff

As we have discussed in the lecture on December 9th, as the number of dimensions in a dataset increases, the volume of the n-dimensional space also increases exponentially, which isolates the datapoints along the edges of the space. It is often hard to find patterns based on grouping data, since the dissimilarities of each datapoint is highlighted by differences in each dimension. Tools we have used this semester, such as the K Nearest Neighbors algorithm would have trouble finding suitable neighbors to conduct classifications on in this high dimensional space. This dilemma is known as the curse of dimensionality. PCA is a way to bypass this route, by identifying the directions where the data is most spread out, and gives us justification to reduce the number of dimensions.A dataset with a large number of dimensions also leads to problems with the bias variance tradeoff: as the amount of dimensions increase, the bias decreases, but the variance increases. PCA allows us to find the optimal number of dimensions that achieves a model with a reasonable bias variance tradeoff.

3 Objective

There is a wealth of publicly released mortgage data that have a lot of different information categories. I will be taking a look at the Fannie Mae quarterly released loan acquisitions for the third quarter of

2016. This dataset records certain characteristics of all the loans acquired or originated by Fannie Mae in the third quarter. However I suspect not all of the characteristics are equally important. This would be a great dataset to conduct exploratory analysis with PCA to discover the minimal set of principal components (PCs) to summarize the variability in the dataset and find interpretations of these PCs.

4 Description of Data and Processing

I retrieved the 2016 third quarter mortgage acquisitions data from the Fannie Mae website. I then loaded and processed the data in R. Because the data came without any headings, I had to manually format the data according to the headers I found in the layout file. I reviewed what each of the categories meant through looking them up in the glossary file, and added the column names. I saved the named data in a file called `titled.csv`, and the column names are given below.

```
> acquisition <- read.csv("~/Downloads/2016Q3/titled.csv")
> names(acquisition)
 [1] "X"                   "LoanId"
  "OrigChannel"         "SellerName"           "OrigIntR"
 [6] "OrigUPB"             "OrigLoanTerm"
 "OrigDate"            "FirstPmtDate"         "OrigLTV"
[11] "OrigCLTV"            "NoBorrowers"          "DTI"                 "
   BorrowerCreditScore" "FTHBI"
[16] "LoanPurp"            "PropType"
 "NoUnits"             "OccpStat"             "PropSt"
[21] "ZIP3"                "MortIntPerc"
"ProductType"         "CoBoCred"
 "MortInsType"
[26] "RelocMortInd"
```

Since PCA only works on numerical variables, I've omitted others from the analysis. The ones I will prospectively be using are `OrigIntR, OrigUPB, OrigLoanTerm, OrigLTV, OrigCLTV, NoBorrowers, DTI, BorrowerCreditScore,MortIntPerc, CoBoCred`. I called this new dataset `new`. I decided to use `prcomp` instead of `princomp` because of its relatively higher accuracy. Aside from specifying the data, the additional arguments I specified were `na.action = na.omit`, that is, deleting rows where any NA's exist, and `scale.=TRUE` to scale the data so that the variances measured are relative. One issue I encountered was that there were issues with `na.action = na.omit`, it did not work properly to omit the rows with NA values from consideration, so an initial run of `prcomp` was not able to return valid results. After digging around to see which of the variables had NA values, I found that some DTI (Debt To Income Ratio) entries were blank. I think perhaps by being blank, instead of `NA`, confused the `na.omit` function. So I removed those from consideration.

```
> new <- filter(acquisition, acquisition$DTI!=is.na(acquisition$DTI))
```

I also checked whether any NA values existed in the other sections.

```
> sum(is.na(new$BorrowerCreditScore))
[1] 221
> sum(is.na(new$OrigIntR))
[1] 0
> sum(is.na(new$OrigUPB))
[1] 0
> sum(is.na(new$OrigLoanTerm))
[1] 0
> sum(is.na(new$OrigLTV))
[1] 0
> sum(is.na(new$OrigCLTV))
[1] 0
> sum(is.na(new$NoBorrowers))
[1] 0
> sum(is.na(new$DTI))
[1] 0
```

```
> sum(is.na(new$MortIntPerc))
[1] 465373
> sum(is.na(new$CoBoCred))
[1] 308122
```

I decided to remove the 221 missing values of Borrower Credit Score, because the number of missing values is still relatively small, but I decided to omit the Mortgage Insurance Percentage and the Co Borrower Credit Score from consideration. That's because they aren't really crucial to each mortgage: not everyone decides to have insurance on their mortgage, and not every mortgage has a co-borrower. So now, the `new` dataset has removed those two variables and all remaining ones do not have `NA`'s.

```
> new <- filter(new, BorrowerCreditScore != is.na(BorrowerCreditScore))
> sum(is.na(new$BorrowerCreditScore))
[1] 0
```

5 Analysis and Results

I then run the `prcomp` on the newly cleaned dataset.

```
> acquisition.pca <- prcomp(select(new, OrigIntR, OrigUPB, OrigLoanTerm,
   OrigLTV, OrigCLTV, NoBorrowers, DTI, BorrowerCreditScore), scale.=TRUE,
   na.action = na.omit)
Warning message:
In prcomp.default(select(new, OrigIntR, OrigUPB, OrigLoanTerm, OrigLTV,
    OrigCLTV, NoBorrowers, DTI, BorrowerCreditScore), scale. = TRUE,
    na.action = na.omit) :
 extra argument   na.action   will be disregarded
```

The coefficients of the linear combination of the original variables for each principal component column is given in `prcomp$rotation` which is summarized in the table below:

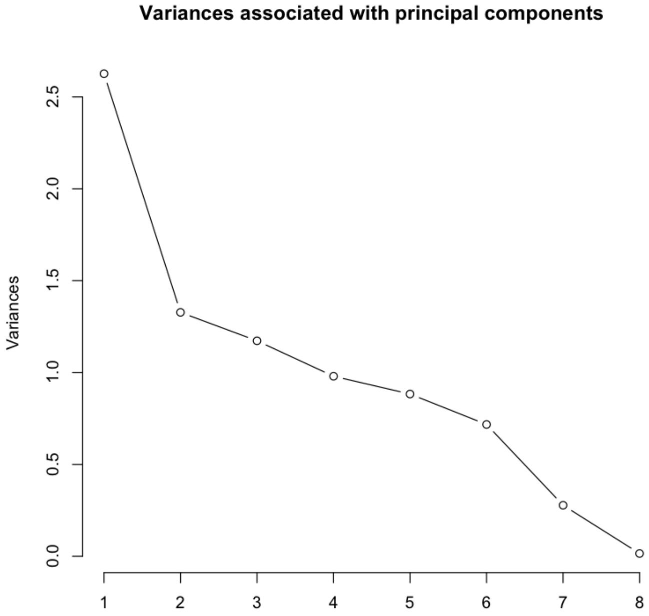
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
OrigIntR	-0.4304	0.3719	-0.2252	0.3614	0.07834	0.04969	-0.6975	0.0046
OrigUPB	-0.0492	-0.2632	-0.6575	-0.3898	-0.2567	-0.5010	-0.1641	0.0123
OrigLoanTerm	-0.4184	0.1463	-0.3983	0.4025	-0.2241	0.0745	0.6535	-0.0026
OrigLTV	-0.5118	-0.4105	0.2480	-0.0876	0.0308	0.0165	-0.0191	0.7063
OrigCLTV	-0.5139	-0.4105	0.2359	-0.0951	0.0302	0.0069	-0.0278	-0.7078
NoBorrowers	0.1036	-0.3498	-0.4436	0.0936	0.7634	0.2765	0.0468	0.0007
DTI	-0.2073	0.3311	-0.1554	-0.6844	-0.0435	0.5929	0.0375	0.0011
BorrowerCredit Score	0.2399	-0.4494	-0.1421	0.2480	-0.5397	0.5592	-0.2340	-0.0025

The sum of squares of the loadings (vertical sum) are the eigenvalues (the variance) of each principal component(eigenvector), which rank their explanatory strength. Loadings = principal components $\times \sqrt{\text{eigenvalues}}$. In addition, the summary of the variation explained by each principal component can be found in `summary(acquisition.pca)` as:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Standard deviation	1.6206	1.1522	1.0829	0.9899	0.9397	0.84697	0.52709	0.1232
Proportion of Variance	0.3283	0.1659	0.1466	0.1225	0.1104	0.08967	0.03473	0.0019
Cumulative Proportion	0.3283	0.4943	0.6408	0.7633	0.8737	0.96338	0.9981	1.00

We first look at the second table to determine the importance of the principal components. We can reasonably rule out PC8, since it accounts for less than 1% of the variance observed. I'm less comfortable with ruling out PC6 and PC7 even though they each explain less than 10% of the total variance. Below we see a graph showing the variances associated with each principal component.

```
> plot(new.pca, type = "l", main = "Variances associated with principal components")
```



6 Cross Validation and Model Selection with PCA

This brings us to the question-how many predictors can we use in a PCA analysis? We sort of determined, in a hand wavy fashion, that 7, not 8 components, may be sufficient. But is that really the best model? Is that really the most minimal set of predictors we can arrive at? We can't say, unless we apply model selection methods. Recalling from previous classes, cross validation aims to divide the data into several training, and one test subset to check the validity of a model. In K-fold cross validation, one of the most common approaches, the data is randomly separated into K folds of approximately equal size. A fold is simply a subsample. 1 fold is assigned to be a test and the rest K-1 is training. The classification is tested out on the training data. Then, the process is repeated K times, as each of the folds acts as the test data one time, and the classification results from the K procedures are averaged to make a single estimation. A common K fold size is 10, which means to divide the data into 10 subsamples and iterate the process 10 times. The most extreme, rigorous, and time intensive classification is to let K = n, the number of observations, called leave-one-out cross validation, but that is often too time intensive if n is too large. Based on what I have learned from the ISL text, there is still no universally accepted standard for cross validating with PCA, because it is an unsupervised approach, so no classification variable is specified. Without a classification variable, there is no standard definition of "accuracy". Based on Andrew Ng's discussion on PCA in his Machine Learning course on Coursera, I found that in practice, people find the minimal number of principal components that can explain a pre-determined level of variance. In a more rigorous definition, we will run this test starting from the first principal component, up to the kth principal component, such that the proportion of squared projection error as a percent of the total variation of the data falls below a prespecified percent α.

$$\frac{\frac{1}{n}\Sigma_{i=1}^{n}\|x^i - x^i_{approx}\|^2}{\frac{1}{n}\Sigma_{i=1}^{n}\|x^i\|^2} \leq \alpha$$

where x^i is one of the n datapoints with k components.

7 Interpretation of the Principal Components

Now comes the creative part, actually interpreting the new principal components. They no longer represent the original predictors, so it's important to interpret what they actually measure now. Since the dataset only contains mortgages that have been made, it is likely that these principal components represent the different profiles of successful borrowers, the basis on which mortgage issuance decisions have been made, and these profiles are totally different from each other. We look at the first table to analyze the loadings of the original predictors projected onto the principal components. The first principal component, for example, can be written as:

$$z_1 = -.4304\text{OrigIntR} - 0.0492\text{OrigUPB} - 0.4184\text{OrigLoanTerm} - 0.5118\text{OrigLTV} - 0.5139\text{OrigCLTV}$$

$$+0.1036\text{NoBorrowers} - 0.2073\text{DTI} + 0.2399\text{BorrowerCreditScore}$$

This is the line that is closest to all the datapoints, and also the direction of the highest variability in the data. According to the second table, it captures almost 33% of the total variability. What we see here is that all the original variables except for NoBorrowers and BorrowerCreditScore (number of borrowers, and their credit scores) have a negative projection onto the first component. In addition, we observe that the heaviest weights are the ones that take into account LTV, the interest rate, and the loan term. So I think that in general, the first principal component is measuring loans with the best characteristics-ie. Low LTV. and this determines around a third of the variability in the types of mortgages acquired/issued in this quarter. The first principal component likely represents the group of mortgages required as the highest quality, where issued mortgages of people with high credit scores and a larger number of borrowers tend to be observed with lower interest rates, shorter loan terms, and lower DTI and LTV ratios.

The second principal component is perpendicular to the first, and capture the variability of loans that is left out by the first, around 17%. These represent a different profile than the first. Here, we see interest rate, loan term, and DTI as observed with opposite directions from the last term. Since we observe higher interest rates along with lower loan to value ratios, and higher debt to income ratios, this represents a riskier population.

The third principal component is perpendicular to both the first and second, and captures the variability of loans that were still left out by the second, around 15%. Here, we observe another profile of loans made with higher LTV ratios and lower credit scores, but are offset by lower DTI (debt to income) ratios and very low beginning balances (Original UPB).

8 Additional Considerations

Additional considerations include running a principal components regression (PCR) and performing cross validation on the results. We can also use other unsupervised methods that allow us to re-include qualitative variables.

9 Citations

Chris Nicholson, & Adam Gibson. A Beginner's Guide to Eigenvectors, PCA, Covariance and Entropy. Deeplearning4j. Skymind. https://deeplearning4j.org/eigenvector. Accessed 20 December 2017

Dallas, G. M. (2013, October 30). Principal Component Analysis 4 Dummies: Eigenvectors, Eigenvalues and Dimension Reduction. George Dallas. https://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-dummies-eigenvectors-eigenvalues-and-dimension-\reduction/. Accessed 20 December 2017

First National Bank of Absecon. (2016). First National Bank of Absecon. https://www.fnbabsecon.com/wp-content/uploads/2016/06/mortgage-icon.png.Accessed 20 December 2017

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). An introduction to statistical learning: with applications in R. In An introduction to statistical learning: with applications in R (pp. 373385). essay, New York: Springer.

Lecture 2: Supervised vs. unsupervised learning, bias-variance tradeoff. (2017, December 20). STATS 202: Data mining and analysis. lecture, Palo Alto: CA, USA.

Curse of dimensionality. (2017, December 10). Wikipedia. Wikimedia Foundation. https://en.wikipedia.org/wiki/Curse_of_dimensionality. Accessed 20 December 2017 http://www.fanniemae.com/portal/funding-the-market/data/loan-performance-data.html

https://loanperformancedata.fanniemae.com/lppub-docs/FNMA_SF_Loan_Performance_File_layout.pdf

https://loanperformancedata.fanniemae.com/lppub-docs/FNMA_SF_Loan_Performance_Glossary.pdf