## Verifying the ARMA Model Calculations

This document shows how you could verify an ARMA model with manual calculations.

Where c is a constant for unknown error;

= -

ARMA Model = c + AR1\* + MA1\*

To help verify that the model does actually work, I assembled a small program to generate an ARMA(1,1) model to automate the prediction. The model coefficients are then used to manually calculate the prediction.

Note: To make this work I used a deprecated version of the ARIMA class which allowed me to set the constant, c, to zero.

model\_fit = model.fit(trend='nc', # 'nc' means no constant.

disp=False) # 'disp' used to show/hide documentation.

The output shows that the automated prediction matches the prediction that is manually calculated with the model coefficients.

History length: 3649

Residual\_t-1 = Actual\_t-1 - Prediction\_t-1

=[15.7] - [13.43443857] = [2.26556143]

Residual t-1: [2.26556143]

**Automated prediction**: [14.40106082]

**Manual prediction**:

Model: ARcoef\_t-1\*[15.7] + MAcoeff\_t-1\*[2.26556143]

Model: 0.9950082575886002\*[15.7] + [-0.53893932]\*[2.26556143]

= [14.4006295]

ARMA Model Results

==============================================================================

Dep. Variable: y No. Observations: 3649

Model: ARMA(1, 1) Log Likelihood -8679.471

Method: css-mle S.D. of innovations 2.610

Date: Fri, 15 Oct 2021 AIC 17364.942

Time: 13:54:11 BIC 17383.548

Sample: 0 HQIC 17371.568

==============================================================================

coef std err z P>|z| [0.025 0.975]

------------------------------------------------------------------------------

ar.L1.y 0.9950 0.002 549.162 0.000 0.991 0.999

ma.L1.y -0.5389 0.033 -16.202 0.000 -0.604 -0.474

Roots

=============================================================================

Real Imaginary Modulus Frequency

-----------------------------------------------------------------------------

AR.1 1.0050 +0.0000j 1.0050 0.0000

MA.1 1.8555 +0.0000j 1.8555 0.0000

-----------------------------------------------------------------------------

Test RMSE: 1.755

|  |
| --- |
| from pandas import read\_csv  import matplotlib.pyplot as plt  from statsmodels.tsa.arima\_model import ARIMA  from sklearn.metrics import mean\_squared\_error  from math import sqrt  import warnings  warnings.filterwarnings("ignore")  PATH = "/Users/pm/Desktop/DayDocs/data/"  series = read\_csv(PATH + 'daily-min-temperatures.csv', header=0, index\_col=0)  # Split the data set so the test set is 7.  NUM\_TEST\_DAYS = 7  X = series.values  size = len(X) - NUM\_TEST\_DAYS  train, test = X[0:size], X[size:]  # Create a list with the training array.  history = [x for x in train]  predictions = []  def showManualPredictionDetail(arCoeff, maCoeff, ar, diff, yhat):  # Actual values will be used as t-1, t-2 etc next iteration.  print("Model: ARcoef\_t-1\*" + str(ar) + " + MAcoeff\_t-1\*" + str(diff))  print("Model: " + str(arCoeff) + "\*" + str(ar) + " + " + str(maCoeff) +\  "\*" + str(diff))  print("= " + str(yhat))  def showResidual(actual\_t\_1, prediction\_t\_1):  residual\_t\_1 = actual\_t\_1 - prediction\_t\_1  print("Residual\_t-1 = Actual\_t-1 - Prediction\_t-1")  print("=" + str(actual\_t\_1) + " - " + str(prediction\_t\_1) + " = " + \  str(resid\_t\_1))  for t in range(len(test)):  print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  print("History length: " + str(len(history)))  #################################################################  # Model building and prediction section.  model = ARIMA(history, order=(1, 0, 1))  # 'nc' stands for no constant.  # Setting 'disp' to false allows cleaner output and is faster.  model\_fit = model.fit(trend='nc', # 'nc' means no constant.  disp=False) # 'disp' used to show/hide documentation.  resid\_t\_1 = 0  ma1 = 0  if(t>0):  actual = test[t - 1]  resid\_t\_1 = actual-prediction  ma1 = model\_fit.maparams  showResidual(actual, prediction)  print("Residual t-1: " + str(resid\_t\_1))  prediction = model\_fit.predict(start=len(history), end=len(history))  print("Automated prediction: " + str(prediction))    print("residuals")  # Get the ar\_modle parameters.  ar\_coef = model\_fit.arparams  ar1 = ar\_coef[0]  history\_t\_1 = history[len(history)-1]  # Make the prediction.  yhat = ar1\*history\_t\_1 + ma1\*resid\_t\_1  showManualPredictionDetail(ar1, ma1, history\_t\_1, resid\_t\_1, yhat)  #################################################################  print(model\_fit.summary())  predictions.append(yhat) # Store the prediction in a list.  obs = test[t] # Get the actual current value.  history.append(obs) # Append the actual current value to the history list.  rmse = sqrt(mean\_squared\_error(test, predictions))  print('Test RMSE: %.3f' % rmse)  plt.plot(test, label='Actual', marker='o', color='blue')  plt.plot(predictions, label='Predictions', marker='o', color='orange')  plt.legend()  plt.title("AR Model")  plt.show() |