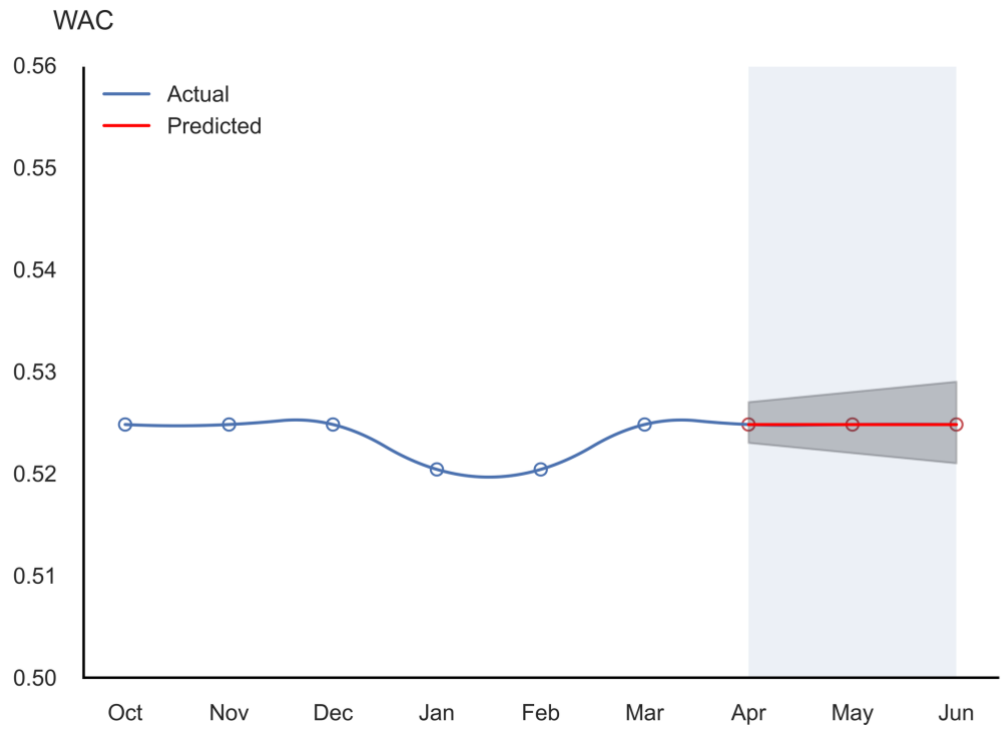
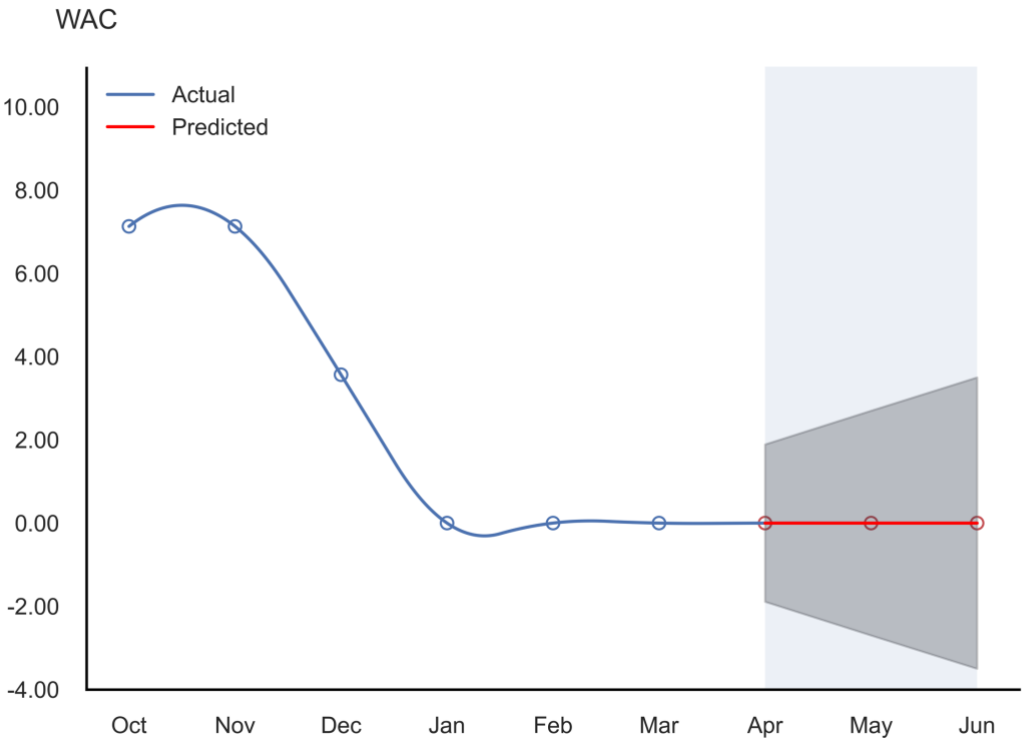
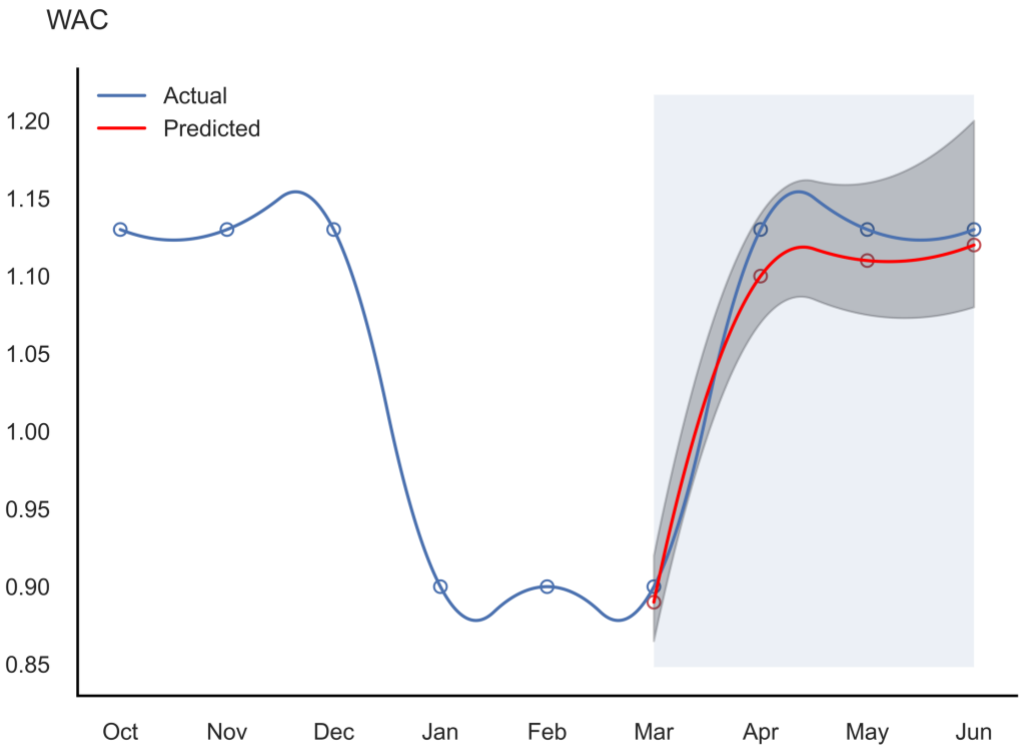
The codes on the next pages serve to build the following visualizations:

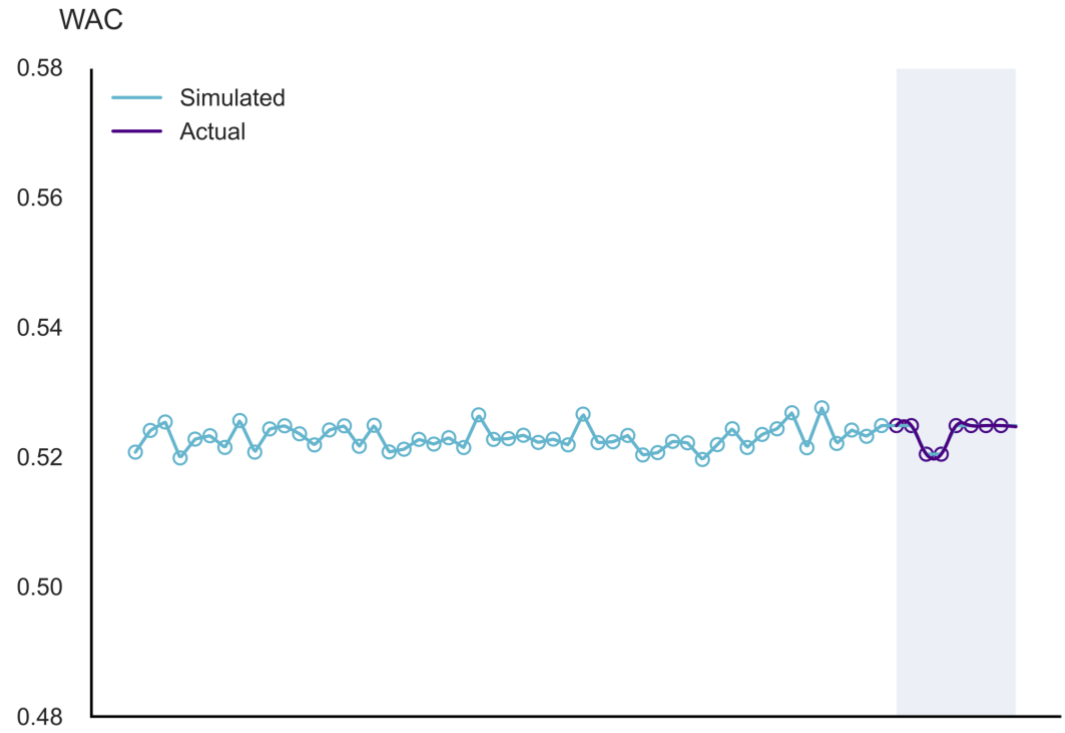
1) Depicting model performance of fitted ARIMA models:







2) Depicting data simulations



Data Visualizations

July 12, 2021

1 Visualising Model Performance

[ ]:
```
def optimal_arima_build_viz(row, top_5_best_results_with_ci_df):
    X = np.array(row[2:58], dtype='float')  #sim wac 1 to wac 56: used to train␣
↪model
    y = row[58:].reset_index(drop=True)  #wac 57 to wac 59: to be predicted by␣
↪model

    #optimal arima order is the one that gives the smallest ci width. try the␣
↪most optimal order onwards, bc the most optimal order may throw an error for␣
↪cases where WAC doesnt vary at all
    sorted_df = top_5_best_results_with_ci_df.sort_values(by='next months ci␣
↪width', ascending=True)

    for i in range(len(sorted_df)):
        optimal_arima_order = sorted_df.iloc[i]['arima order']

        try:
            model = ARIMA(X, order=optimal_arima_order)
            model_fit = model.fit()
            y_pred_df = model_fit.get_forecast(steps=3).summary_frame(alpha=0.
↪05)
            y_pred_df = y_pred_df[['mean', 'mean_ci_lower', 'mean_ci_upper',␣
↪'mean_se']]
            y_pred_df.fillna('NA', inplace=True)

            if y_pred_df['mean_ci_lower'] != 'NA':
                break
            else:
                continue

        except:
            continue

    y_results = pd.concat([y, y_pred_df, pd.Series(f'{optimal_arima_order}')],␣
↪axis=1)
```

```
    y_results.columns = ['actual', 'predicted', 'prediction_lower',␣
↪'prediction_upper', 'prediction_sd', 'optimal_arima_order']

    return y_results
```

1.1 1) WAC stays constant at some positive value: 395 cases (52%)

[ ]:
```
row = wac_df_for_model.iloc[452:453]
row
```

[ ]:
```
top_5_best_results_with_ci_df = try_combinations(wac_df_for_model.iloc[452])

top_5_best_results_with_ci_df
```

[ ]:
```
y_results_452 = optimal_arima_build_viz(wac_df_for_model.iloc[452],␣
↪top_5_best_results_with_ci_df)

y_results_452.index = [6, 7, 8]

y_results_452
```

[ ]:
```
actual_y_452 = wac_df_for_model.iloc[452][52:].reset_index(drop=True)
actual_y_452
```

[ ]:
```
import matplotlib.pyplot as plt
plt.style.use('seaborn')

from matplotlib.ticker import FormatStrFormatter
```

[ ]:
```
#create a smooth curve plot from the data
from scipy.interpolate import make_interp_spline
spline = make_interp_spline(actual_y_452.index, list(actual_y_452), k=2)
X = np.linspace(0, 8, 500)
Y = spline(X)
plt.plot(X, Y, label='Actual')

#plot the actual datapoints too as scatter points
actual_y_452_scatter = actual_y_452[:6]
plt.scatter(actual_y_452_scatter.index, actual_y_452_scatter,␣
↪facecolors='none', edgecolors='b')

#plot predicted WAC data and its scatter points
y_results_452['predicted'].plot(label='Predicted', color='red')
plt.scatter(y_results_452['predicted'].index, y_results_452['predicted'],␣
↪facecolors='none', edgecolors='r')
```

```
#shaded area for SD
ax = plt.gca()  #get current plot's axes
ax.fill_between([6, 7, 8],
                y_results_452['prediction_lower'],
                y_results_452['prediction_upper'],
                color='k', alpha=.25)

#format legend location
plt.legend(loc='upper left')

#y ticks
plt.ylim(1.649, 1.651)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.2f'))  #set y ticks to 4dp

#blue shaded area to signal prediction starting
ax.fill_betweenx(ax.get_ylim(), x1=6, x2=8, alpha=0.1)

#x ticks
plt.xticks([0,1,2,3,4,5,6,7,8], ['Oct', 'Nov', 'Dec', 'Jan', 'Feb', 'Mar',␣
 ↪'Apr', 'May', 'Jun'])

#format y axis title
plt.ylabel('WAC', rotation=0, fontsize=13)
ax.yaxis.set_label_coords(0,1.05)

#format y axis and x axis lines
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')

plt.savefig('1.png', dpi=1000)
```

1.2 2) WAC changes between different positive values: 269 cases (35%)

[ ]:
```
row = wac_df_for_model.iloc[0:1]
row
```

[ ]:
```
top_5_best_results_with_ci_df = try_combinations(wac_df_for_model.iloc[0])

top_5_best_results_with_ci_df
```

[ ]:
```
y_results_0 = optimal_arima_build_viz(wac_df_for_model.iloc[0],␣
 ↪top_5_best_results_with_ci_df)

y_results_0.index = [6, 7, 8]
```

[ ]:
```
y_results_0
```

[ ]:
```
actual_y_0 = wac_df_for_model.iloc[0][52:].reset_index(drop=True)
actual_y_0
```

[ ]:
```
#create a smooth curve plot from the data
from scipy.interpolate import make_interp_spline
spline = make_interp_spline(actual_y_0.index, list(actual_y_0), k=2)
X = np.linspace(0, 8, 500)
Y = spline(X)
plt.plot(X, Y, label='Actual')

#plot the actual datapoints too as scatter points
actual_y_0_scatter = actual_y_0[:6]
plt.scatter(actual_y_0_scatter.index, actual_y_0_scatter, facecolors='none',␣
 ↪edgecolors='b')

#plot predicted WAC data and its scatter points
y_results_0['predicted'].plot(label='Predicted', color='red')
plt.scatter(y_results_0['predicted'].index, y_results_0['predicted'],␣
 ↪facecolors='none', edgecolors='r')

#shaded area for SD
ax = plt.gca()  #get current plot's axes
ax.fill_between([6, 7, 8],
                y_results_0['prediction_lower'],
                y_results_0['prediction_upper'],
                color='k', alpha=.25)

#format legend location
plt.legend(loc='upper left')

#y ticks
plt.ylim(0.50, 0.56)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.2f'))  #set y ticks to 4dp

#blue shaded area to signal prediction starting
ax.fill_betweenx(ax.get_ylim(), x1=6, x2=8, alpha=0.1)

#x ticks
plt.xticks([0,1,2,3,4,5,6,7,8], ['Oct', 'Nov', 'Dec', 'Jan', 'Feb', 'Mar',␣
 ↪'Apr', 'May', 'Jun'])

#format y axis title
plt.ylabel('WAC', rotation=0, fontsize=13)
ax.yaxis.set_label_coords(0,1.05)

#format y axis and x axis lines
ax.spines['left'].set_color('black')
```

```
ax.spines['bottom'].set_color('black')

plt.savefig('2.png', dpi=1000)
```

1.3 3) WAC varies a LOT: 59 cases (8%)

[ ]:
```
row = wac_df_for_model.iloc[612:613]
row
```

[ ]:
```
top_5_best_results_with_ci_df = try_combinations(wac_df_for_model.iloc[612])

top_5_best_results_with_ci_df
```

[ ]:
```
y_results_612 = optimal_arima_build_viz(wac_df_for_model.iloc[612],␣
 ↪top_5_best_results_with_ci_df)

y_results_612.index = [6, 7, 8]
```

[ ]:
```
#create a smooth curve plot from the data
from scipy.interpolate import make_interp_spline
spline = make_interp_spline(actual_y_612.index, list(actual_y_612), k=2)
X = np.linspace(0, 8, 500)
Y = spline(X)
plt.plot(X, Y, label='Actual')

#plot the actual datapoints too as scatter points
actual_y_612_scatter = actual_y_612[:6]
plt.scatter(actual_y_612_scatter.index, actual_y_612_scatter,␣
 ↪facecolors='none', edgecolors='b')

#plot predicted WAC data and scatter points
y_results_612['predicted'].plot(label='Predicted', color='red')
plt.scatter(y_results_612['predicted'].index, y_results_612['predicted'],␣
 ↪facecolors='none', edgecolors='r')

#shaded area for SD
ax = plt.gca()  #get current plot's axes
ax.fill_between([6, 7, 8],
                y_results_612['prediction_lower'],
                y_results_612['prediction_upper'],
                color='k', alpha=.25)

#y ticks
plt.ylim(-4, 11)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.2f'))  #set y ticks to 4dp
```

```
#blue shaded area to signal prediction starting
ax.fill_betweenx(ax.get_ylim(), x1=6, x2=8, alpha=0.1)

#format legend location
plt.legend(loc='upper left')

#x ticks
plt.xticks([0,1,2,3,4,5,6,7,8], ['Oct', 'Nov', 'Dec', 'Jan', 'Feb', 'Mar',␣
 ↪'Apr', 'May', 'Jun'])

#format y axis title
plt.ylabel('WAC', rotation=0, fontsize=13)
ax.yaxis.set_label_coords(0,1.05)

#format y axis and x axis lines
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')

#save fig
plt.savefig('3.png', dpi=1000)
```

2 4) WAC changes between the prediction months

[ ]:
```
row = wac_df_for_model.loc[1075:1076]
row
```

[ ]:
```
top_5_best_results_with_ci_df = try_combinations(wac_df_for_model.loc[1075])

top_5_best_results_with_ci_df
```

[ ]:
```
y_results_1075 = optimal_arima_build_viz(wac_df_for_model.loc[1075],␣
 ↪top_5_best_results_with_ci_df)

y_results_1075.index = [5, 6, 7, 8]
```

[ ]:
```
#create a smooth curve plot from the data
from scipy.interpolate import make_interp_spline
spline = make_interp_spline(actual_y_1075.index, list(actual_y_1075), k=2)
X = np.linspace(0, 8, 500)
Y = spline(X)
plt.plot(X, Y, label='Actual')

#plot the actual datapoints too as scatter points
plt.scatter(actual_y_1075.index, actual_y_1075, facecolors='none',␣
 ↪edgecolors='b')
```

```
#plot predicted WAC data and scatter points
spline = make_interp_spline(y_results_1075.index,␣
↪list(y_results_1075['predicted']), k=2)
X = np.linspace(5, 8, 500)
Y = spline(X)
plt.plot(X, Y, label='Predicted', color='red')

plt.scatter(y_results_1075['predicted'].index, y_results_1075['predicted'],␣
↪facecolors='none', edgecolors='r')

#shaded area for SD
spline = make_interp_spline(y_results_1075.index,␣
↪y_results_1075['prediction_lower'], k=2)
X = np.linspace(5, 8, 500)
Y_pred_lower = spline(X)

spline = make_interp_spline(y_results_1075.index,␣
↪y_results_1075['prediction_upper'], k=2)
X = np.linspace(5, 8, 500)
Y_pred_upper = spline(X)

ax = plt.gca()  #get current plot's axes
ax.fill_between(X,
                Y_pred_lower,
                Y_pred_upper,
                color='k', alpha=.25)

#blue shaded area to signal prediction starting
ax.fill_betweenx(ax.get_ylim(), x1=5, x2=8, alpha=0.1)

#format legend location
plt.legend(loc='upper left')

#x ticks
plt.xticks([0,1,2,3,4,5,6,7,8], ['Oct', 'Nov', 'Dec', 'Jan', 'Feb', 'Mar',␣
↪'Apr', 'May', 'Jun'])

#y ticks
ax.yaxis.set_major_formatter(FormatStrFormatter('%.2f'))  #set y ticks to 4dp

#format y axis title
plt.ylabel('WAC', rotation=0, fontsize=13)
ax.yaxis.set_label_coords(0,1.05)

#format y axis and x axis lines
ax.spines['left'].set_color('black')
```

```
ax.spines['bottom'].set_color('black')

#save fig
plt.savefig('4.png', dpi=1000)
```

3 Visualising Data Simulation

[ ]:
```
data = wac_df_for_model.iloc[0][2:]
```

[ ]:
```
len(data)
```

[ ]:
```
#plot simulated data
spline = make_interp_spline(range(59), list(data), k=1)
X = np.linspace(0, 59, 500)
Y = spline(X)
plt.plot(X, Y, label='Simulated', color='c')

plt.scatter(range(59), data, facecolors='none', edgecolors='c')

#plot actual data
spline = make_interp_spline(range(51, 59), list(data[51:59]), k=2)
X = np.linspace(51, 59, 200)
Y = spline(X)
plt.plot(X, Y, label='Actual', color='indigo')

plt.scatter(range(51, 59), data[51:59], facecolors='none', edgecolors='indigo')


#change y axis
plt.ylim(0.48, 0.58)

#blue shaded area to signal actual value starting
ax = plt.gca()  #get current plot's axes
ax.fill_betweenx(ax.get_ylim(), x1=51, x2=59, alpha=0.1)

#format legend location
plt.legend(loc='upper left')

#x ticks: remove
plt.xticks([], [])

#format y axis title
plt.ylabel('WAC', rotation=0, fontsize=13)
ax.yaxis.set_label_coords(0,1.05)

#format y axis and x axis lines
```

```
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')

#save fig
plt.savefig('simulation.png', dpi=1000)
```