Project_2

Jake Peters

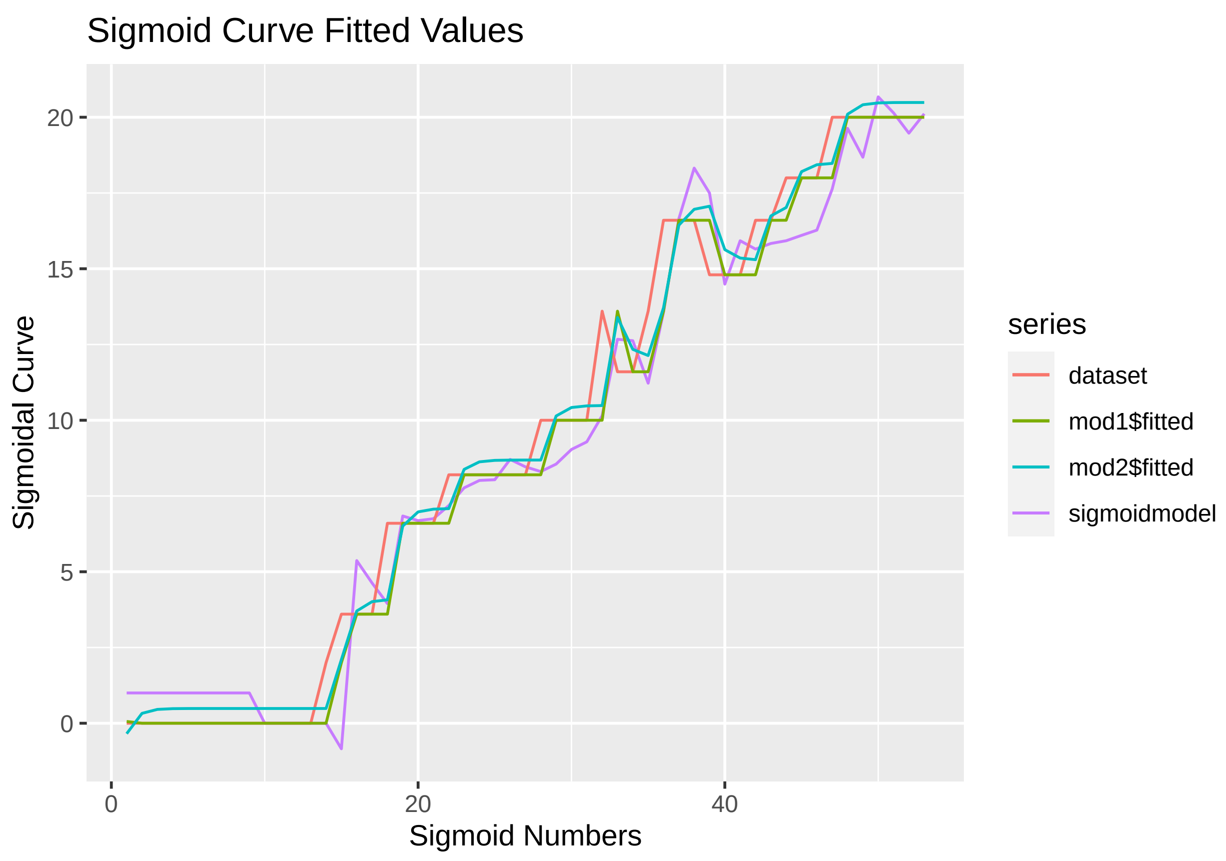
11/10/2021

sigmoid function with 1-step forecast and optimized parameters

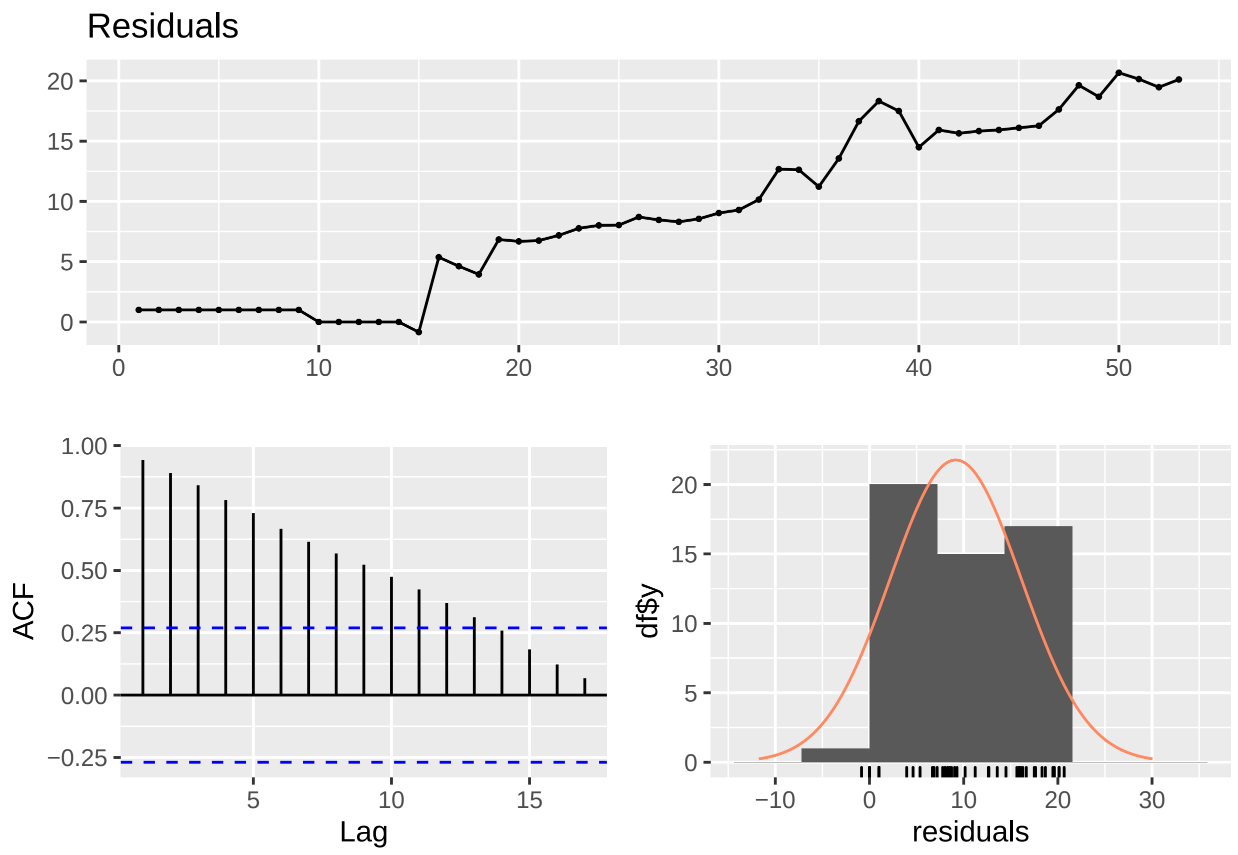
As a description of my project:

My forecasting method is a function uses a moving window of different time periods and a sigmoid function as a kind of logistic regression to fit values to the dataset for which it is trying to forecast. The sigmoid function itself has six different parameters. Each of these parameters modifies the equation in its own way to try to best fit the data points. A regualar sigmoid function would squash everything between one and zero, but the extra parameters enable the sigmoid function to map to virtually any set of values in the real space. The actual sigmoid function equation with its six parameters is: y_hat <- (form / (topshifter + xposmult * exp(-shape * (index - xshifter))) + yshifter), where form makes it stretchy in the middle, topshifter makes it taller, xposmult increases the x-direction tilt, shape changes the width of it, xshifter changes the starting x-value, and yshifter changes the starting y-value. The is in the sigmoidOpt function. These parameters are then optimized using the optim function, with a error function of sum squared error, with the actual value minus the expected value squared for each index. With the parameters now able to be optimized, the next piece of code, sigCyclic or sigNaive, is the function which obtains fitted values and forecast values. It does this by having a moving window that splits up the dataset by integers up to the mmMax value. This enables the dataset to be fitted by sigmoid functions at every iteration level from 3 up to whatever the user chooses for mmMax. After fitted values are created for every set of window values, the function uses RMSE of each fitted set minus the actual dataset and chooses the fitted set with the least RMSE. This window is used as the fitted set. Then, to account for the NA values at the start of the dataset, 1 is replacing each NA. Then, for sigNaive, the very last fitted value from fitted is used for forecasting arbitrary H values into the future. For sigCyclic, the next H values into the future are forecasted from the last H values of the fitted set.

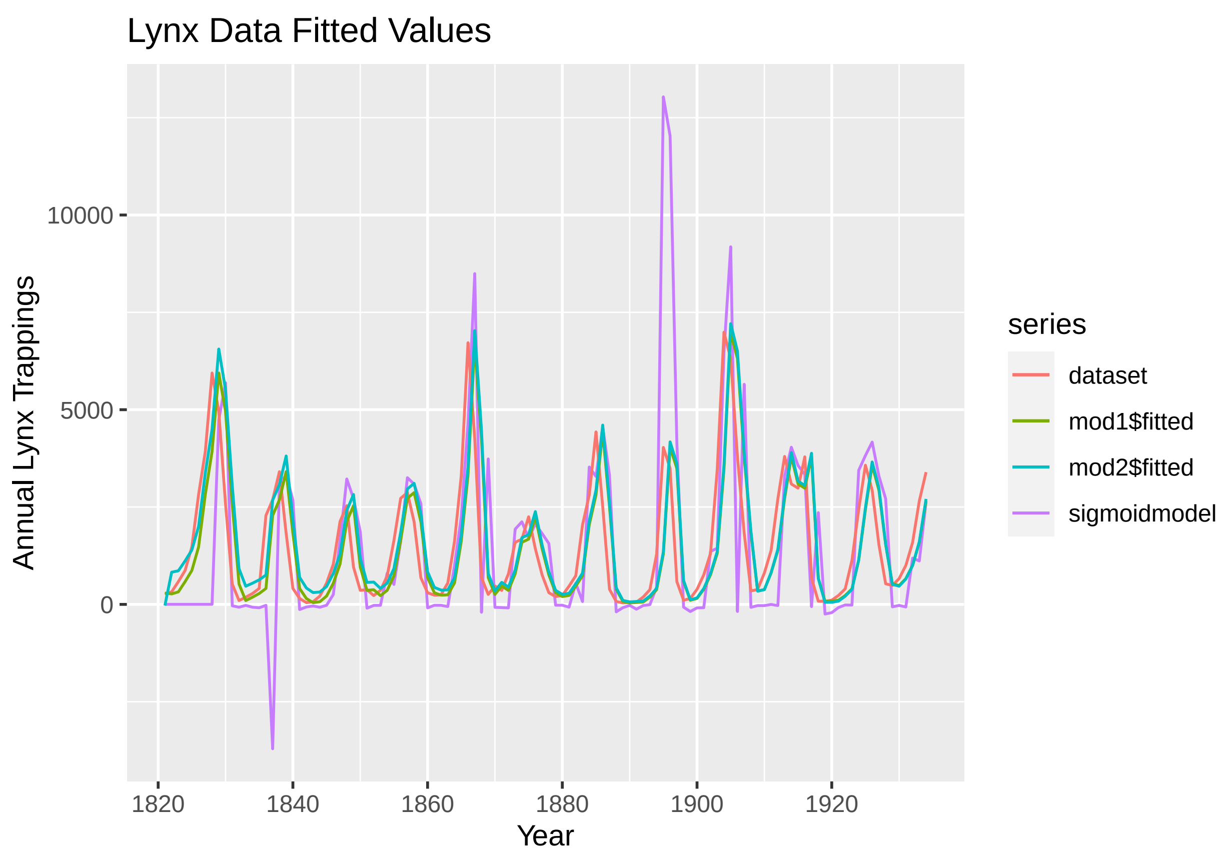
A discussion of how well my method fits data (by the way, both variations fit data the same way so I will only include Naive sigmoidal for this part: Lets start with the obvious: my function fits a sigmoidal data set very well! Here is an exmaple of a canned set of data points that mirror a sigmoidal curve. This is an example of trended data without cyclicity.



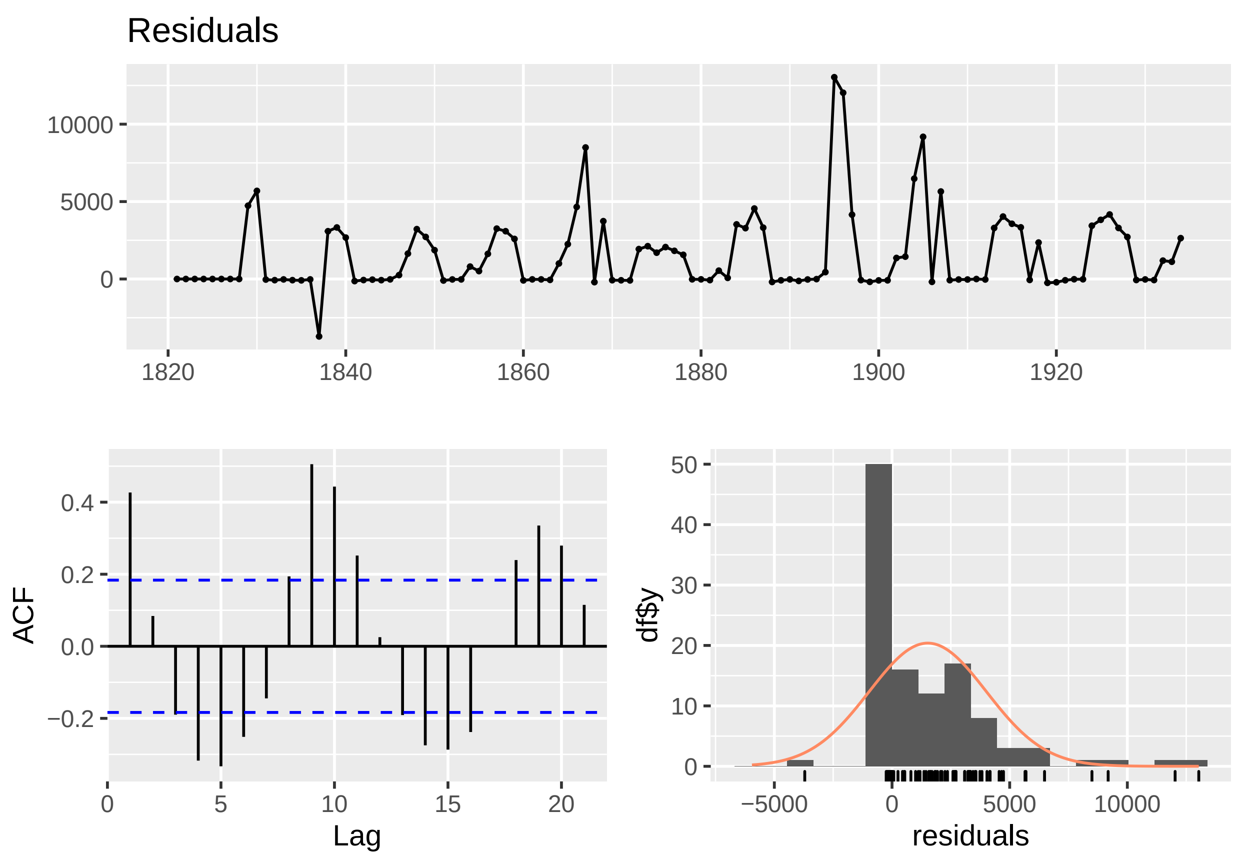
```
## [1] 1.46185

## [1] 1.096855

## [1] 1.019989

## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



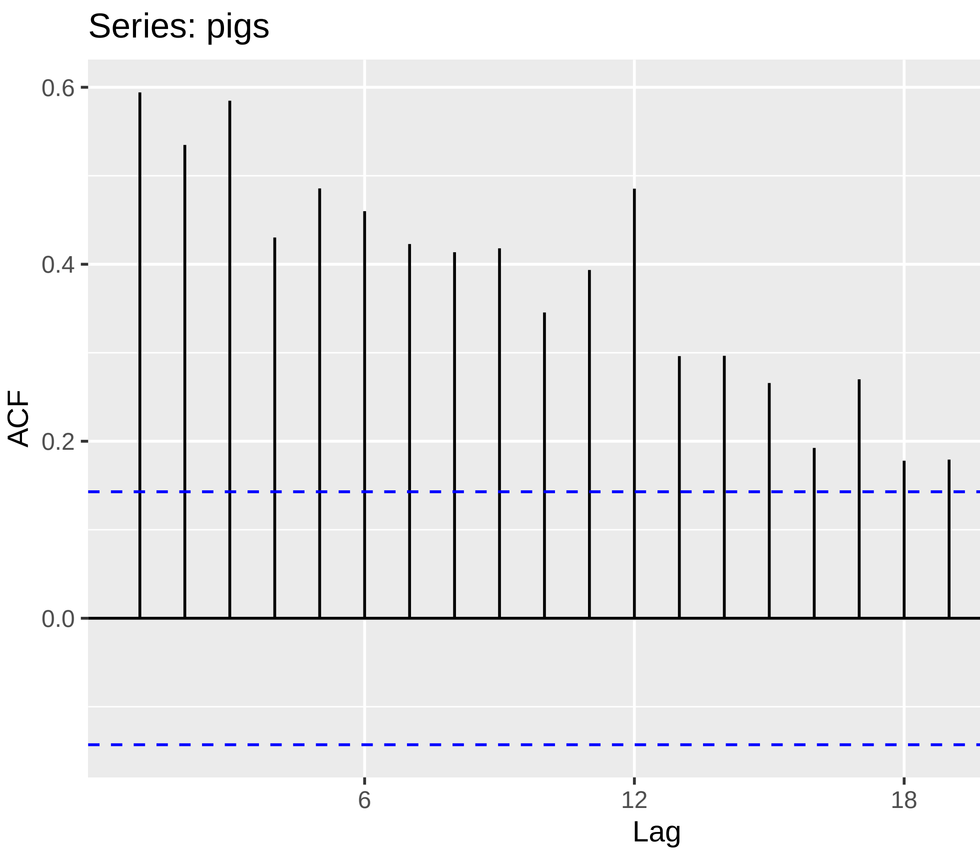
As we can see, the function here mirrors the dataset relatively well, although with the eye test it is hard to differentiate between the this function's performance and the SES and HOLT method. However, taking a closer look, it appears there is kind of a pattern in the sigmoid line where it undercuts a step in the data. This means that it undershoots some values where vertical steps are taken. I am not quite sure exactly why this is; it might have something to do with the sigmoid function taking longer to adjust to a new section with a window of data. This is borne out by the RMSE values, where the holt data maps it best, with ses next and sigmoid as the worst, with RMSE of 1.02, 1.10, and 1.46 respctively. This indicates the two built-in functions performaned better on this step-function-looking dataset. The residuals are not centered at zero and are autocorrelated, so this is kind of concerning, although I am not quite sure why they are so weird in this instance.



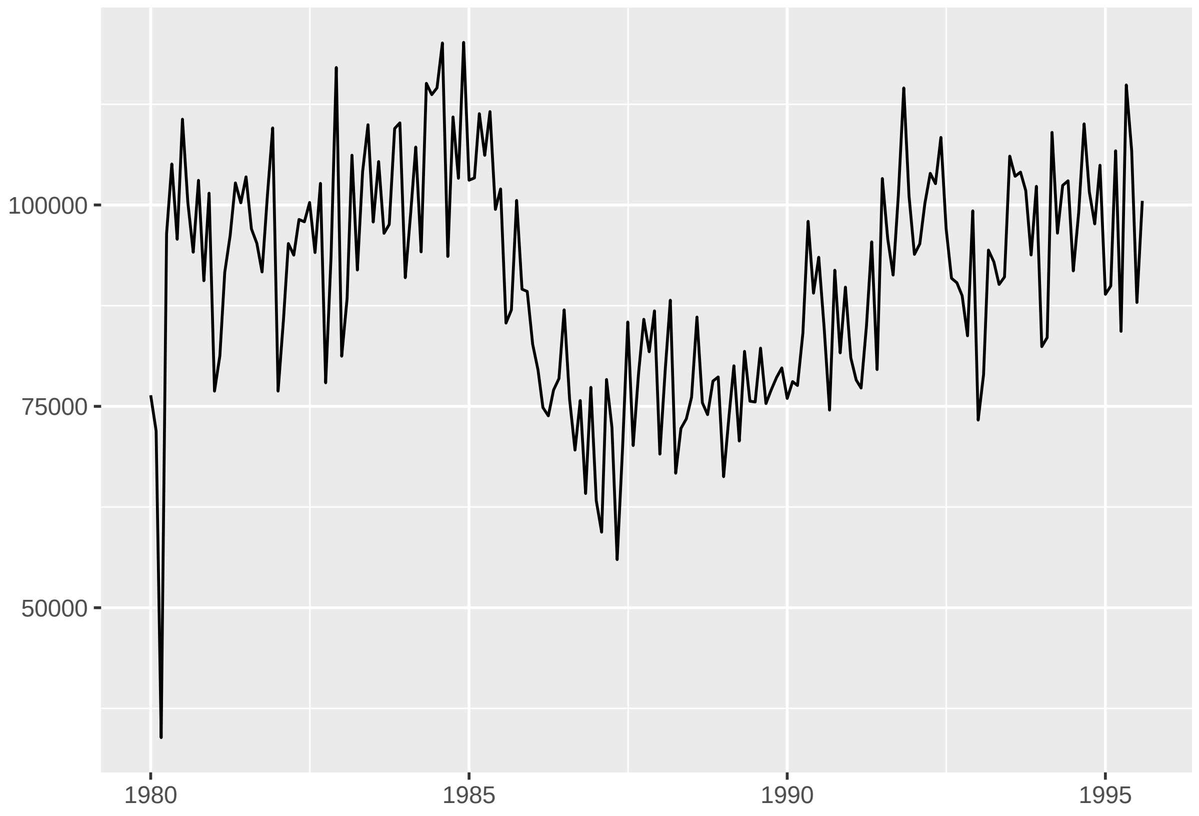
```
## [1] 1985.435

## [1] 1182.145

## [1] 1215.338

## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



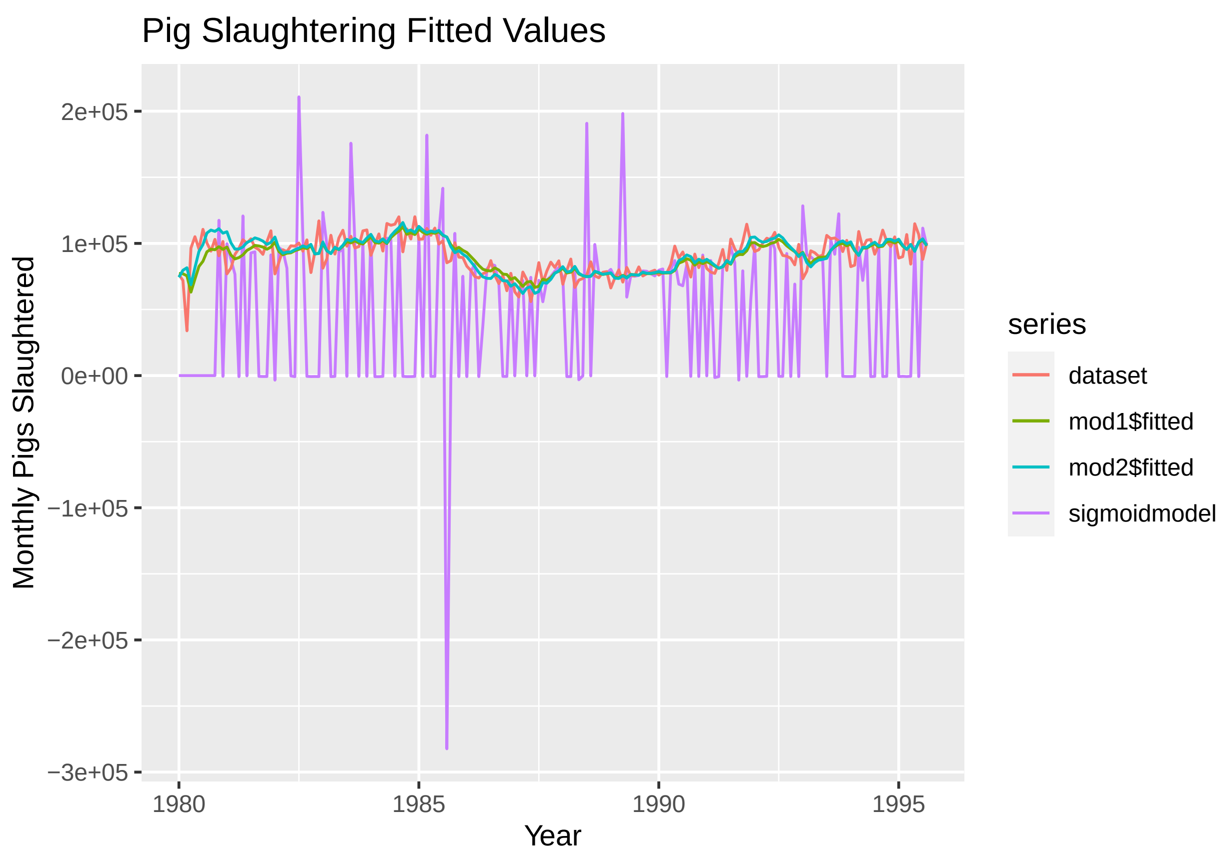
On the lynx data, which is annual lynx trappings in Canada from 1821-1934, the sigmoid function approximates the curves with fitted values fairly well here. There are a few dramatic spikes and dips in two or three of the waves, but overall the sigmoid function looks like it is doing a fairly decent job. This is probably doing a decent job because there is cyclicity in this data, but no trend, so it is easier for the function to cope with the rises and falls in the data. However, the sigmoid function trails the values in a way that it consistently goes lower than the two other models. This comes to fruition in the RMSE value, where it has a value of 1985, with the others being around 1200, so the others were significantly better in that way. The residuals are not ideal, as they are not centered at zero and are autocorrelated.



Lets look at a variable trended dataset.



100000
75000
50000
1980
1985
1990
1995

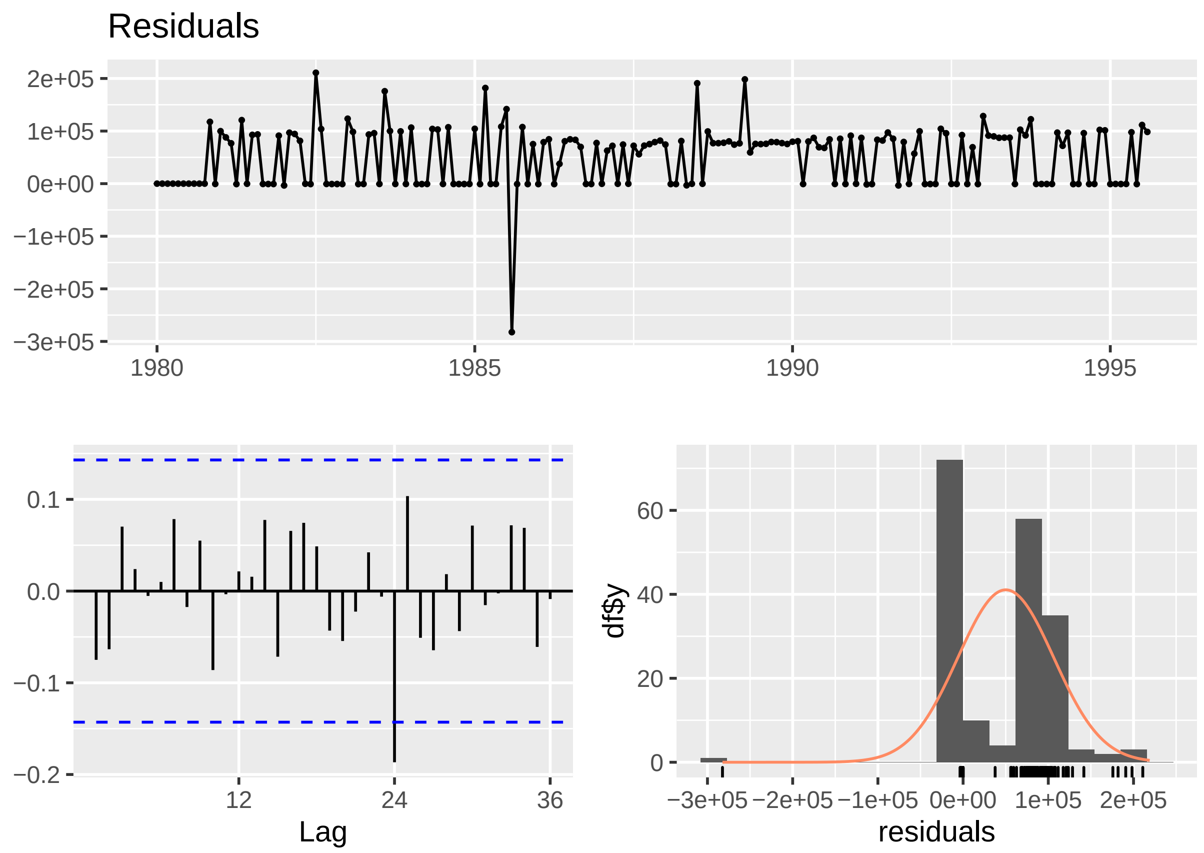


```
## [1] 70681.04
```

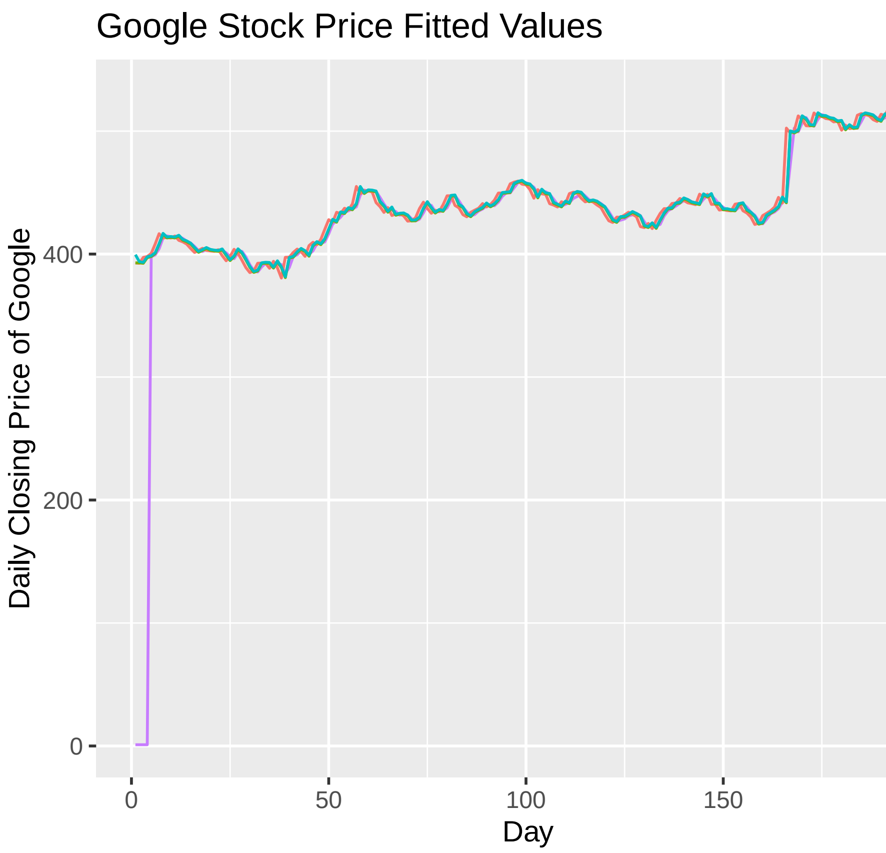
```
## [1] 10253.6
```

```
## [1] 10583.01
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



Here, the function performs terribly on the pigs data, which is the number of pigs slaughtered each month in Victoria, Australia. The reason I believe it performs so badly here is because of the sharp spikes and changes in direction for the pigs data. I think the window is not small enough for the sigmoid to be able to adjust to the changes in direction, so it always overshoots the reversals, sometimes by incredibly drastic amounts. This is shown in the RMSE values, with my function having a value of around 70,700, and the SES and HOLT being a little more than 10,000, so my function has almost seven times the amount of error the others do, which means it is not performing well at all. The residuals look good here, as they are centered at zero and not autocorrelated.



Now, lets look at how it does on a random walk!

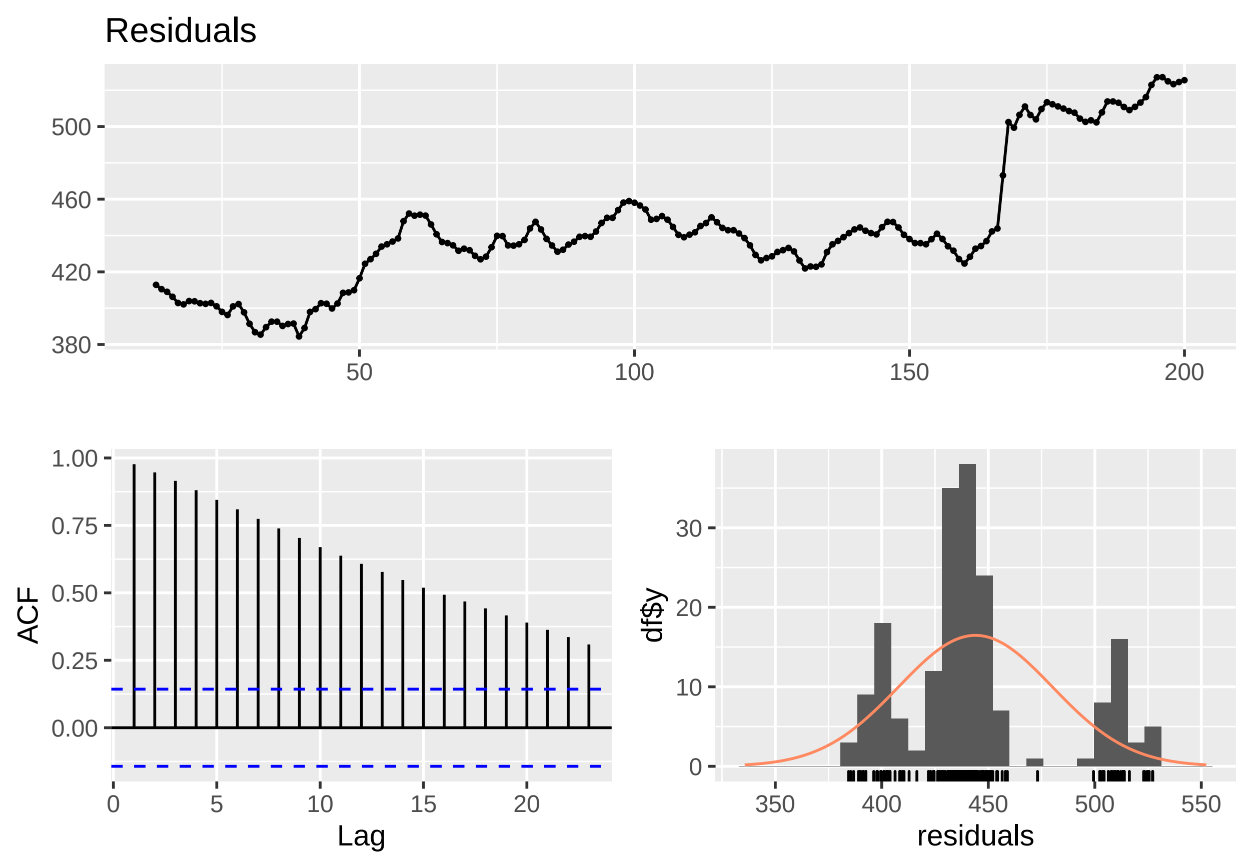
```
## [1] 56.14838
```

```
## [1] 6.186586
```

```
## [1] 6.162532
```

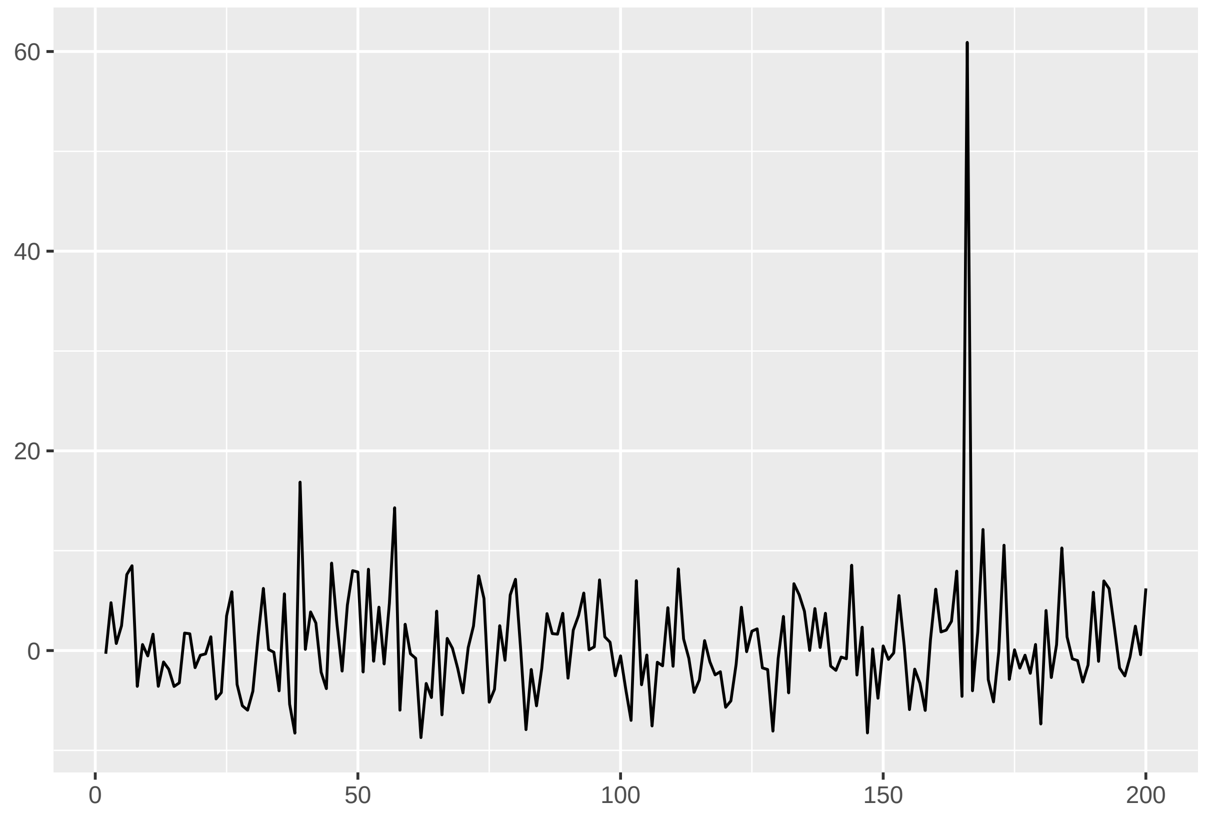
```
## [1] 6.838284
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```

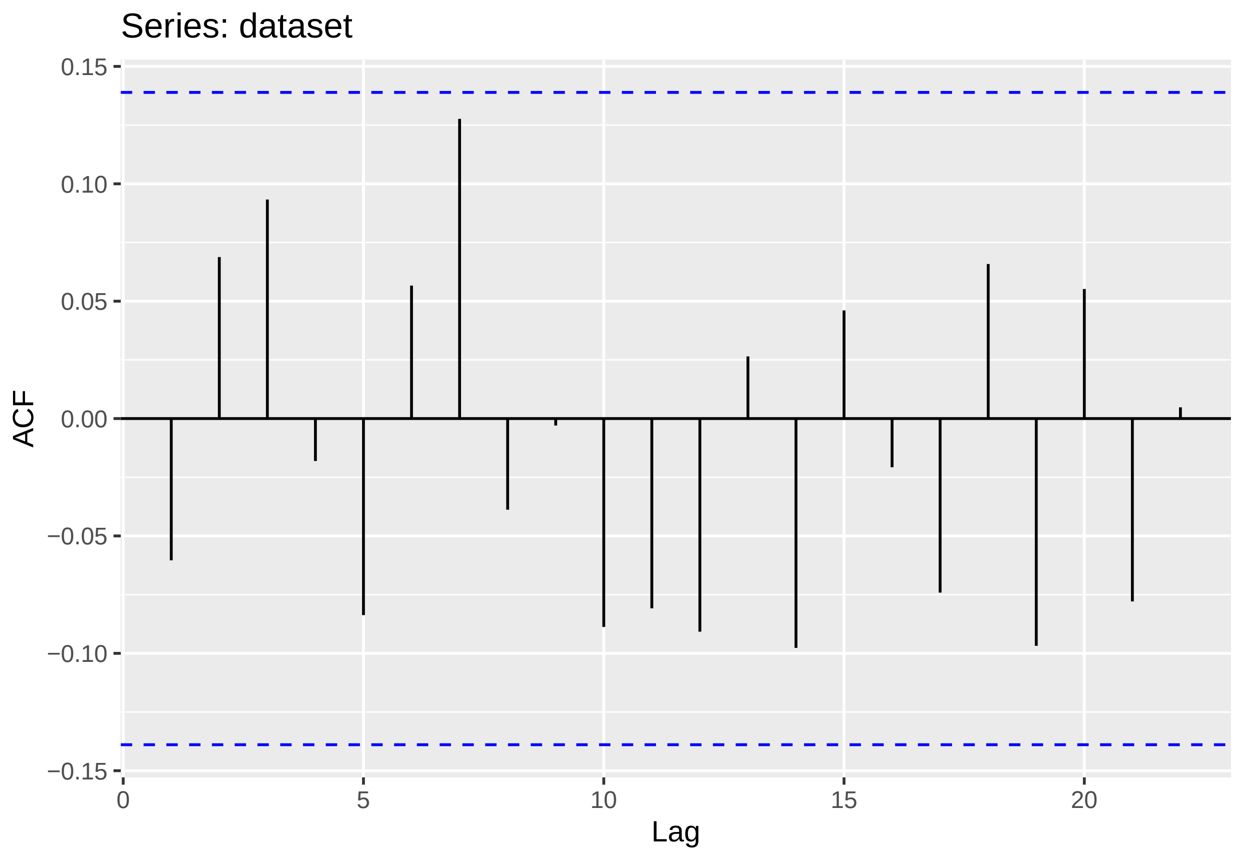


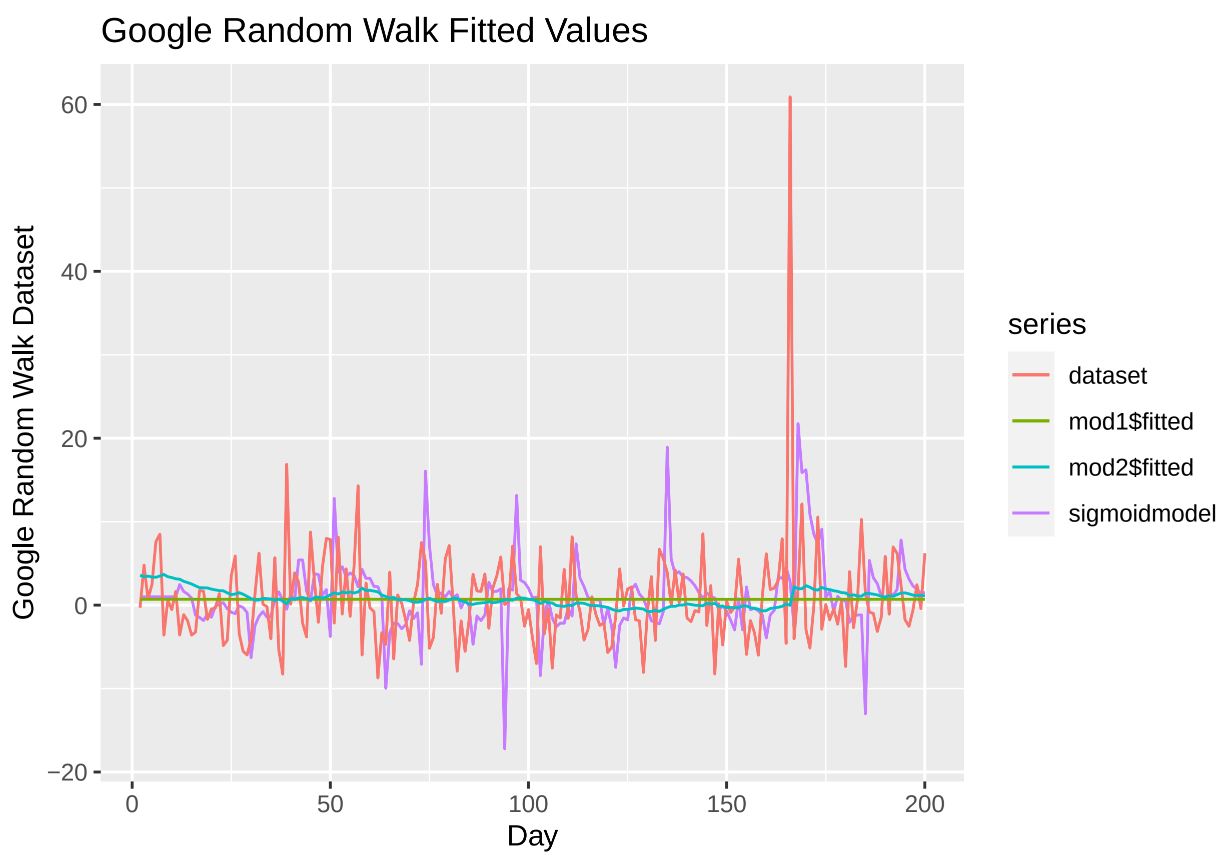
For the random walk google data of daily closing google stock prices from the NASDAQ, the The sigmoid function here does a suprisingly good job of tracking the walk all the way through to the end of the data. The only downfall of these fitted values are the inital 1 values at the beginning. When removing those from the equation, the difference in RMSE for the fitted values from the sigmoid function to the SES and HOLT functions is very small, at less than 0.7 units, which I think is a very good performance from this function. I think the reason why it is so good here is that there are a decent amount of values, and they do not change very much percent wise from day to day. This is conducive to the sigmoid function performing well because it does not have to navigate huge spikes, so it would not overshoot the changes it needs to make for the parameters most of the time. Here, the residuals are not looking great, they look right-skewed and they are autocorrelated; I'm not sure why.

Here is a difference of the google stock random walks from one day to the next, so basically a white noise

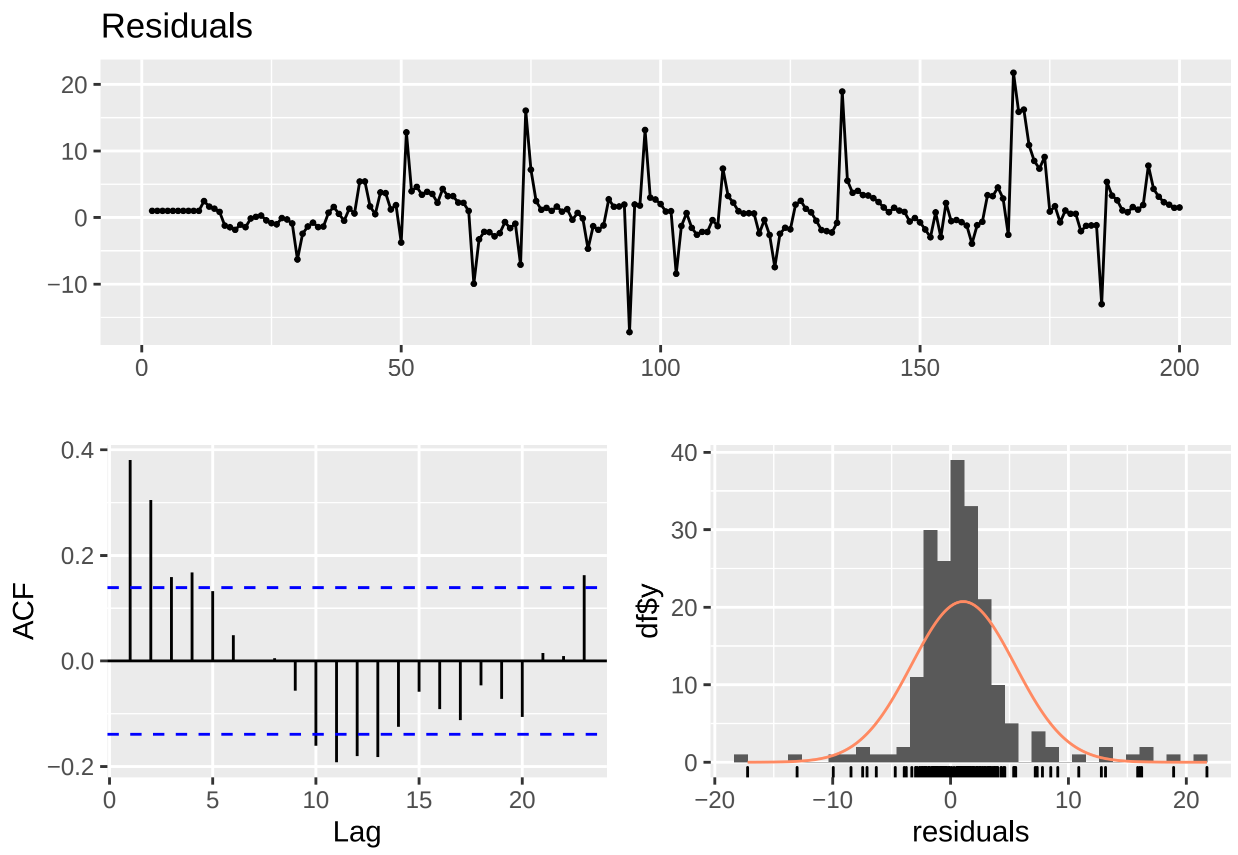


dataset.


Series: dataset
ACF
Lag
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
5
10
15
20

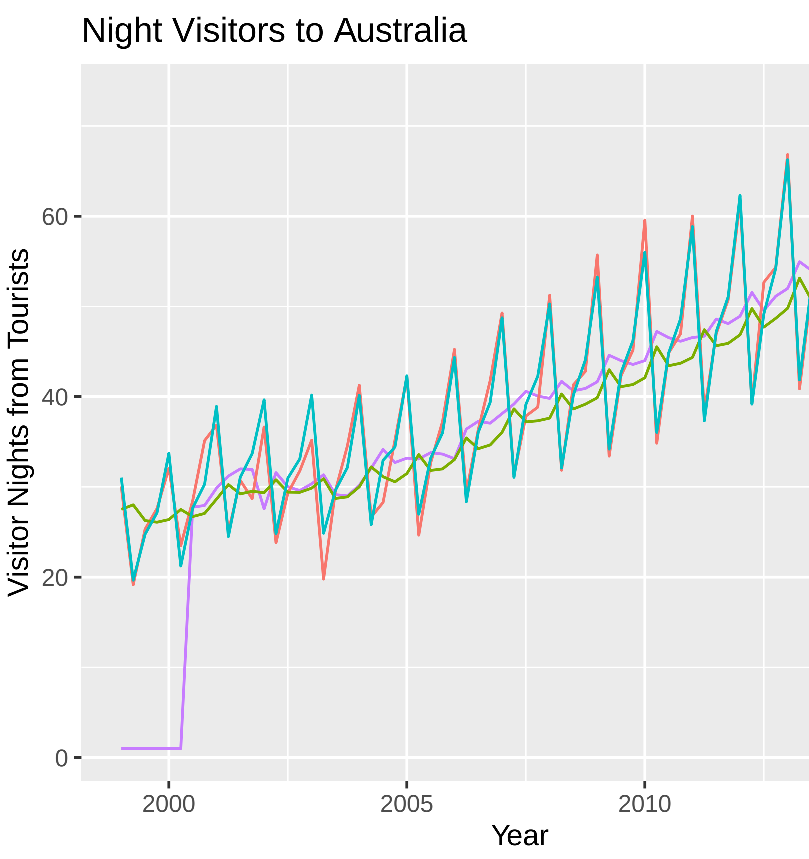


```
## [1] 7.405551

## [1] 6.169236

## [1] 6.29559

## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



The white noise here from the google stock differences is something that is tough for the models to follow. The SES and HOLT methods kind of tend towards the median of zero, not wavering very much. However, the sigmoid model does attempt to fit the spikes in the data. The issue is that it attempts to do this, but gets a negative of the spike, which greatly increases the RMSE of the function. This leads it to have a larger RMSE than the other SES and HOLT functions, even though it is capturing the spirit of the data more than the others. Also, the residuals here are normally distributed and are not autocorrelated mostly, which is good. However, I think it is more of an indicator of white noise of the original data than anything else.

Here is a look at a seasonal dataset with trend; or the number of quarterly total visitor nights from interna-



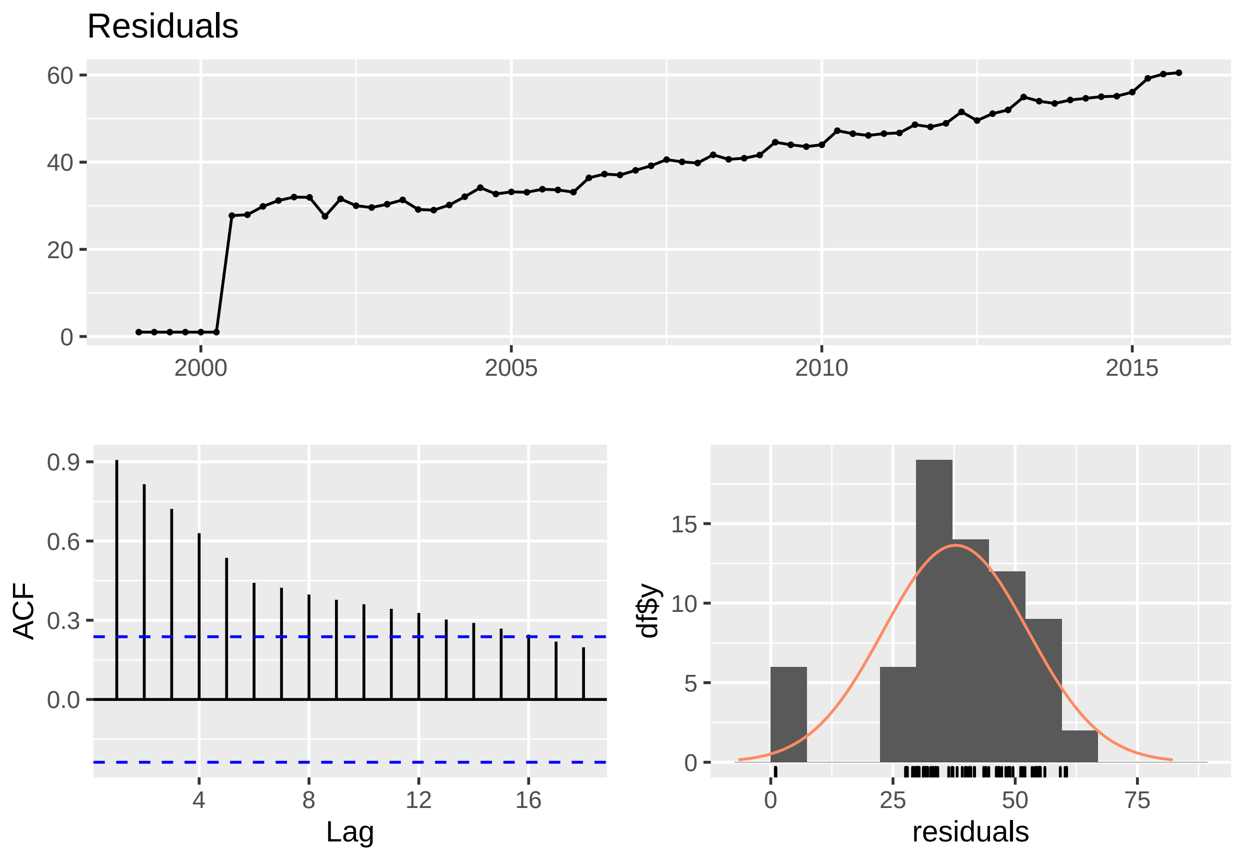
tional tourists in Australia in millions from 1999-2015.

```
## [1] 10.80232
```

```
## [1] 8.317347
```

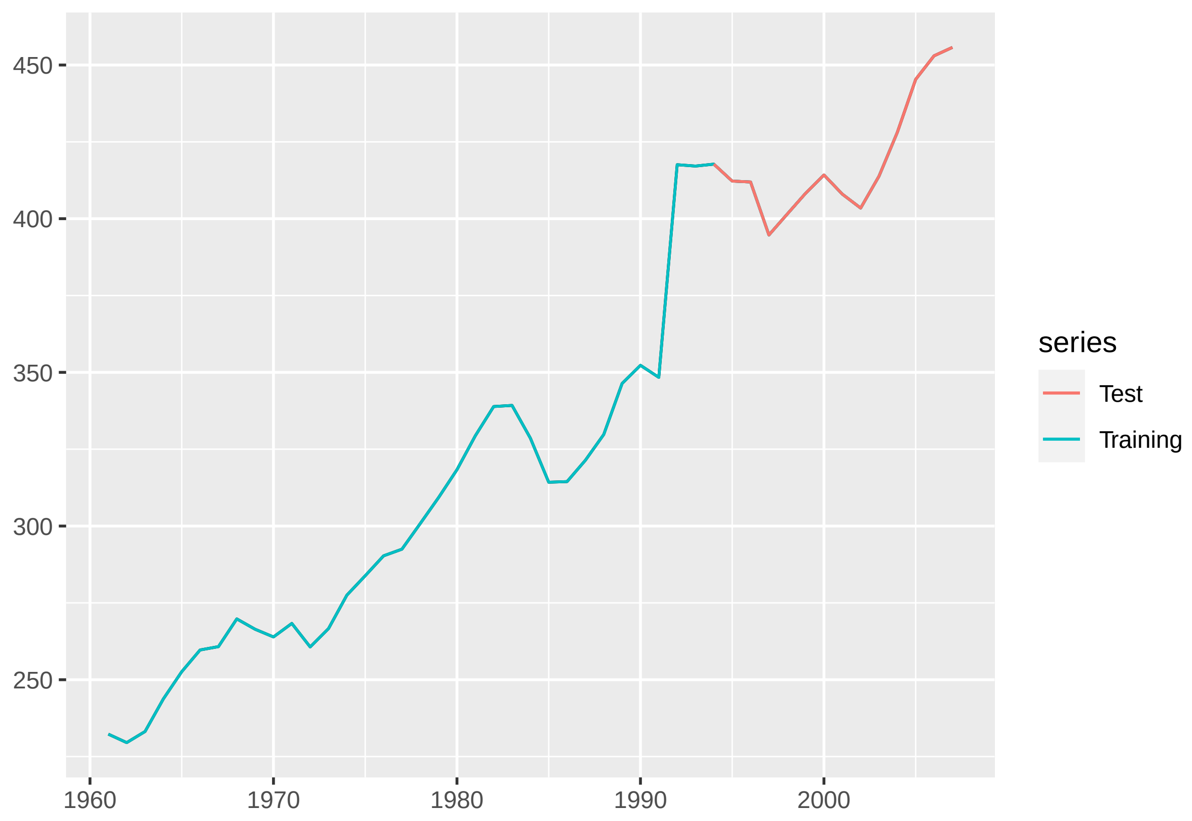
```
## [1] 2.022337
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



The sigmoid function here captures the trend of the data; it fails to capture the spikiness of the seasonality. The holt winters forecast does an amazing job here, with an RMSE of 2, but the other model fail, with a RMSE of 8 for SES and almost 11 for sigmoidmodel. This is not a promising showing for the sigmoid function. Also, the residuals are almost normally distributed, but look right-skewed perhaps. They are autocorrelated, which means not white noise, which is not a plus either.

Lets go to looking at actual forecasts instead of just fitted values. My function now will take the fitted values and spit out forecasts that try to predict the future of the data. Lets look at basic trended data first, with International visitors to Australia from the 1980s until the 2010s.



```
##                ME     RMSE      MAE     MPE     MAPE      ACF1 Theil's U
## Test set 5.211938 15.53893 12.14166 1.16593 2.892794 0.3329795  1.469349
```

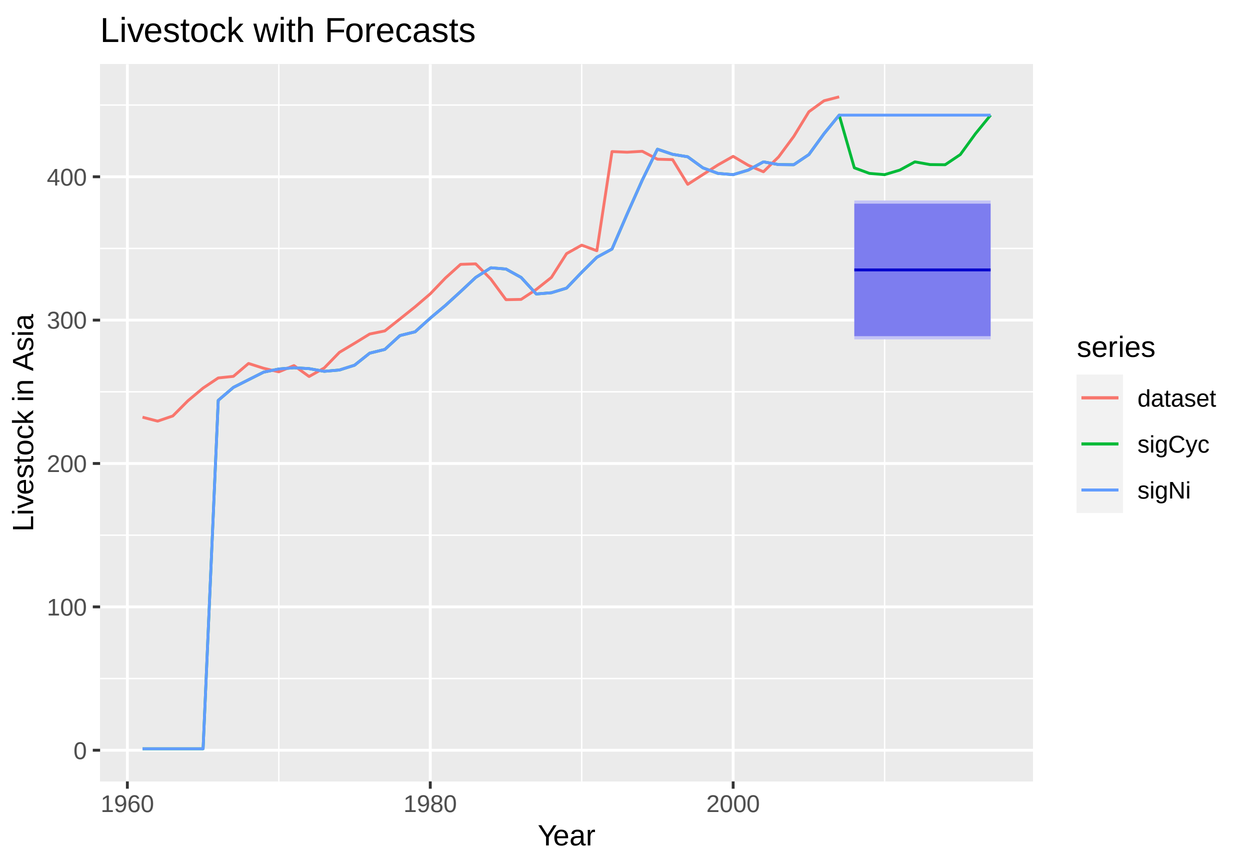
```
##                ME     RMSE      MAE     MPE     MAPE      ACF1 Theil's U
## Test set 5.211938 15.53893 12.14166 1.16593 2.892794 0.3329795  1.469349
```

```
## [1] 173.365
```

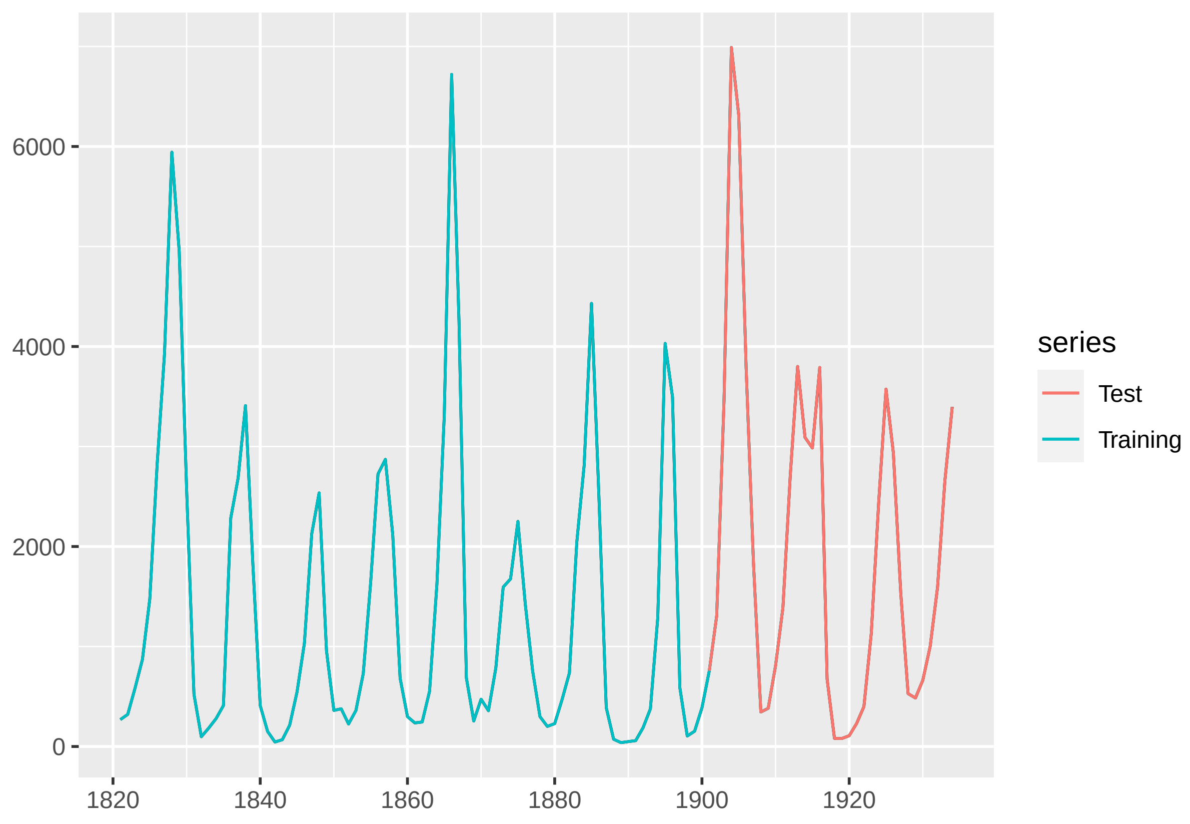
```
## [1] 162.6274
```

```
## [1] 8.803058
```

```
## [1] 8.024192
```



Here, for a linear dataset of livestock sheep in Asia from 1961-2007, This data is trended without seasonality. The two holts have a lower MAE by about 4, which is a decent amount, indicating they may be better than the sigmoid function. This is not suprising to me, because the data is trended, which is exactly what holt mean to accomplish with his forecast. For the plot, it seems like the cyclical one is having a better chance of caputuring a downturn, which is nice, while the other is static, which is playing the odds against regressing to the mean. The mean will always be flat, and the SigNi will be flat. The Cyclical will not be flat for a long while. But eventually it will get there.



Here let's look at

```
##                 ME     RMSE      MAE     MPE     MAPE       ACF1 Theil's U
## Test set 532.3946 2122.035 1655.734 46.5374 89.42713 -0.4961117  1.456005
```

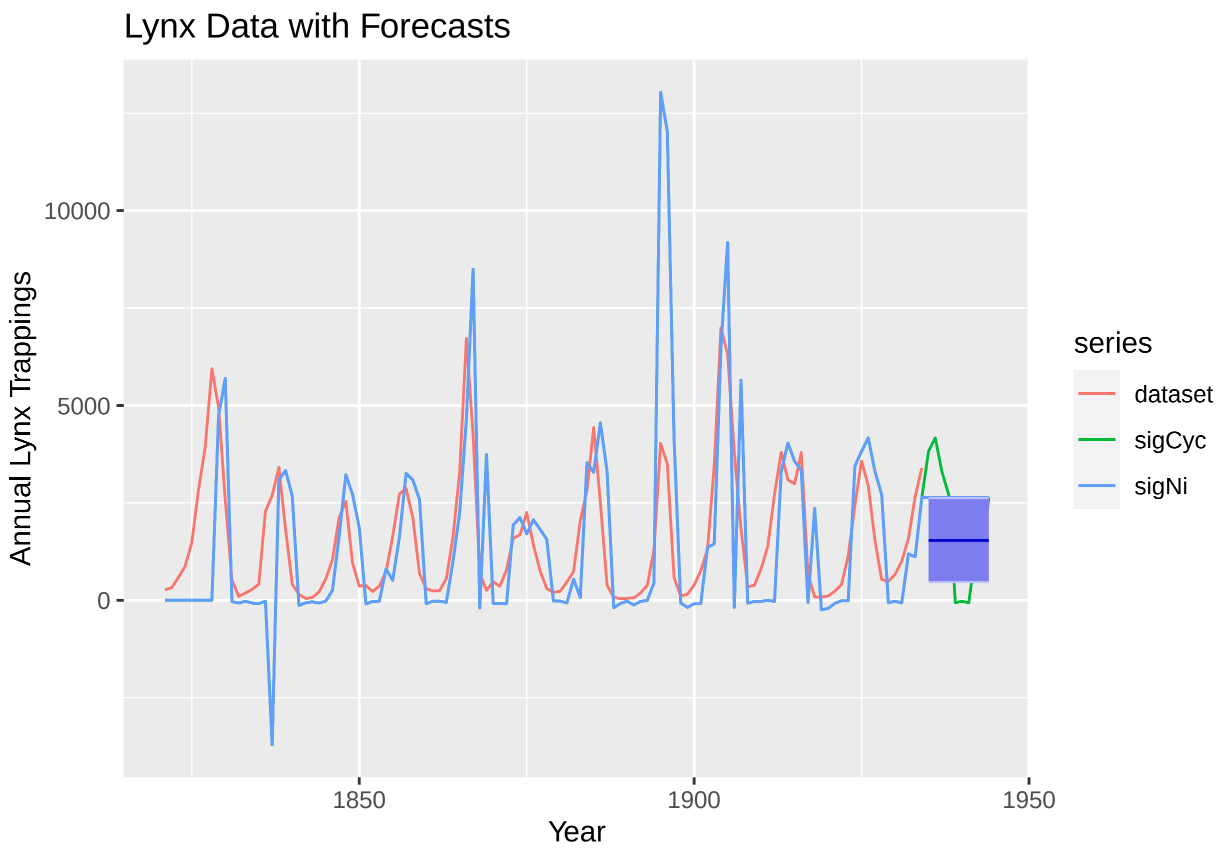
```
##                 ME     RMSE      MAE     MPE     MAPE       ACF1 Theil's U
## Test set 532.3946 2122.035 1655.734 46.5374 89.42713 -0.4961117  1.456005
```

```
## [1] NaN
```

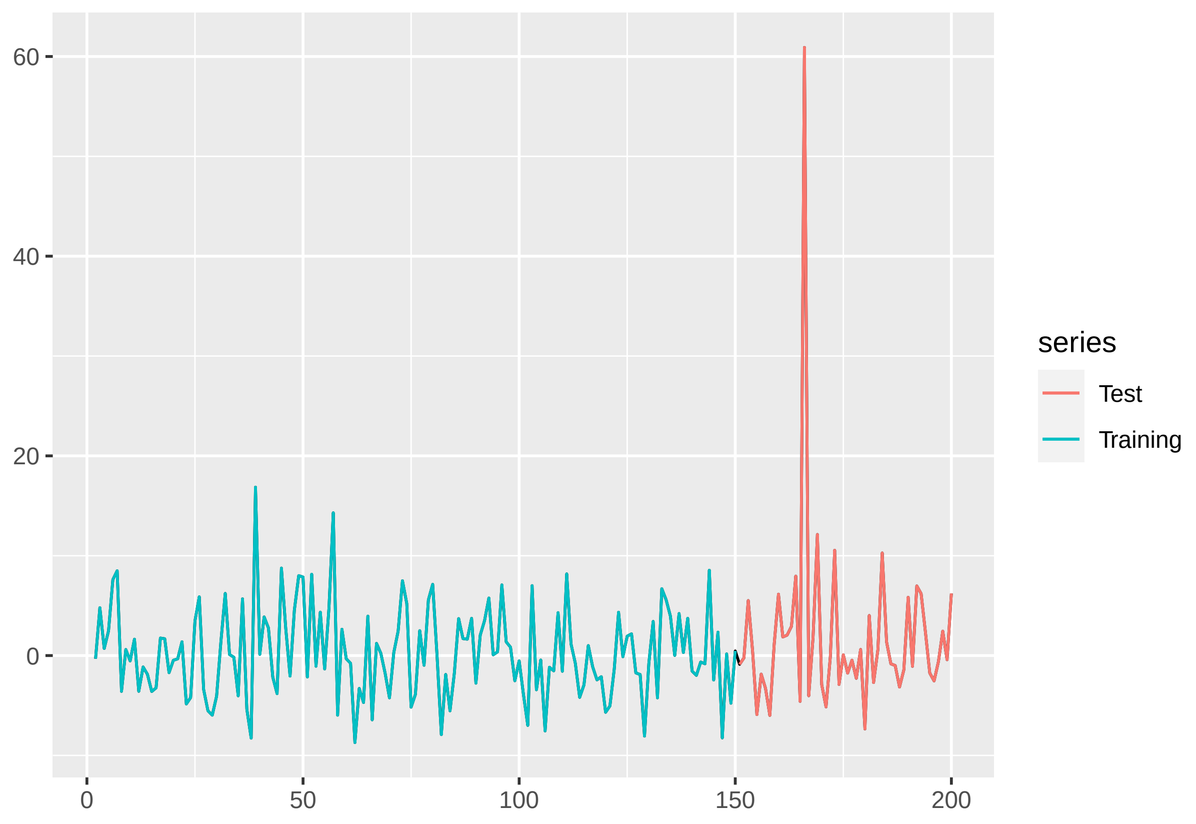
```
## [1] NaN
```

```
## [1] NaN
```

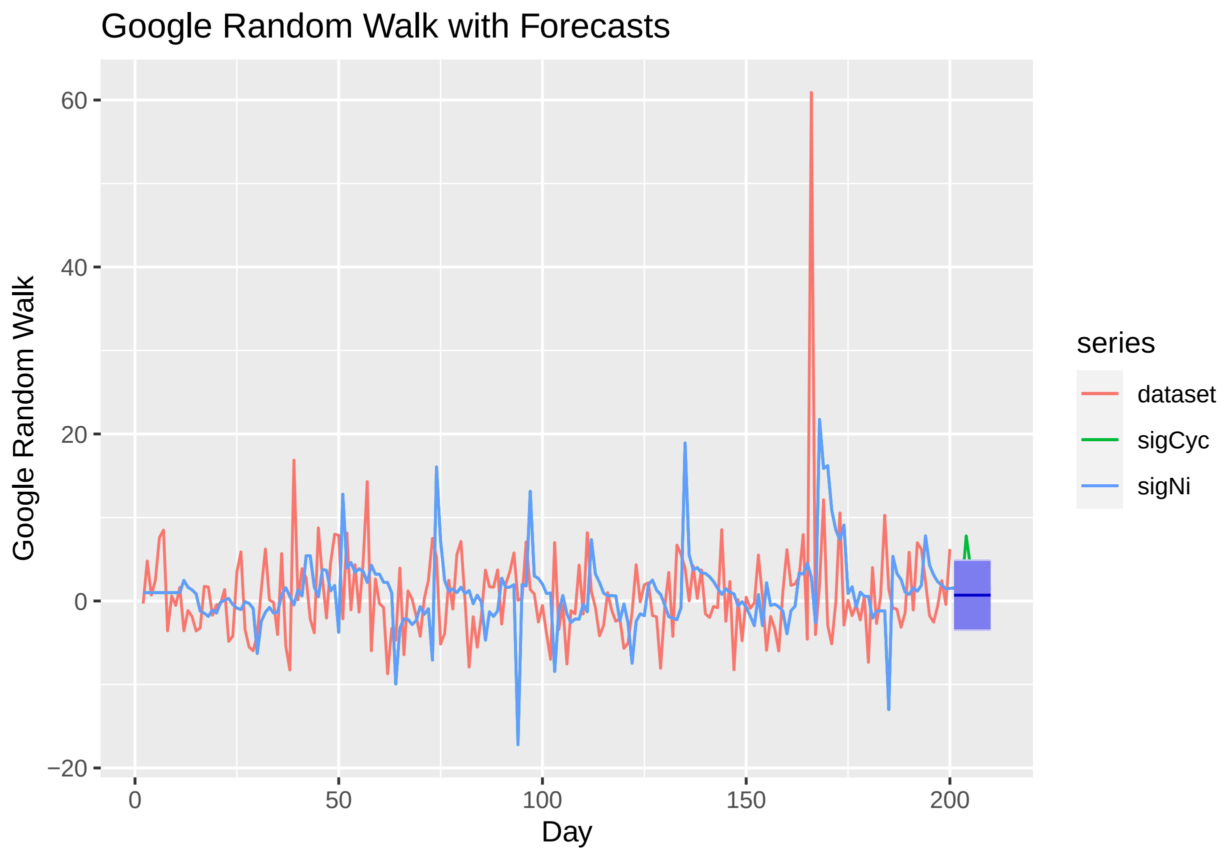
```
## [1] NaN
```



Here, everything is exactly the same for both forecasts I made, however, the error for both of them is still higher even with lynx than the other pre-made data crunchers. This could be because of the strange tendancy of the function to have offshoots in crazy directions. The sigCylical forecast looked almost seasonal, like it was going to make a good prediction! The mean and naive are pretty much the same in this instance.



```
##                  ME     RMSE      MAE      MPE     MAPE
## Test set -0.151162 0.151162 0.151162 17.09499 17.09499

##                  ME     RMSE      MAE      MPE     MAPE
## Test set -0.151162 0.151162 0.151162 17.09499 17.09499

## [1] NaN

## [1] NaN

## [1] NaN

## [1] NaN
```



Here, with the differences Google data set again, we have a sigmoid Cyclical forecast that looks good again, the sigNi looks reasobly centered, and the mean forecast looks okay. The RMSE for each of mine is again lower. This graph is basically white noise, so it is mildly comforting to see a prediction not trying to force things. FOr example, the forecasts here look very reasonable, which for white noise is not the greatest thing ever, but it sufficess, at least in the short term.

As an overall evaluation of the forecast, I recommend it for s-shaped, monotically increasing data. This seems to mirror the nature of the sigmoid curve very well. I also recommend it for cyclical data where there is no trend, such as the lynx dataset. It captured the essence of that dataset very well. I recommend it for trended data as well, where the slope does not change direction, as that could confuse it. It performs poorly on datasets where there is a changing trend. It messes with the window and parameters, to generate massive spikes. Another situation it underperforms is if there is an extremely spiky dataset, or lots of variance about the mean. It performs poorly on data that swings in multiple directions, while it tends to do better on data that gradually changes over a few days, so that its window can matchup with that style. In general, it was worse than Holt methods, Holt-Winters method, and a few other forecasting methods. However, for some datasets it would be better than the mean method, or potentially even better than the naive method. For cyclic data, the Cyclic edition of the forecast may be prudent to use to match the cyclicity at the end of the data, into the start of the forecast. For linearly trended data, the regular Naive version may be best. Ultimately, it is up to the discretion of the chooser to pick which works for the current situation and data, as there are always real-world implications that may not be captured by raw data,