Forecasting Labor Force Participation Rate

Gian Carlo Sanfuego

2024-06-24

## Introduction

For this project, we will look at a univariate time series of Labor Force Participation data in Singapore. Generally, if you want to learn more about the overall employment of a country, there are two ways in how to go about it. The first one is the classic unemployment rate. The size of the pool of unemployed people versus the people in the workforce. The alternative to the unemployment rate is the labor force participation rate. And this rate is harder to manipulate since it starts with a different idea. It calculates the percentage of people in the workforce in relation to all available people in the relevant age range. In this case, this age range will be 25 to 54 years. So we use the data table available over at <https://www.gapminder.org/data/> . The data spans 1980 to 2007 and contains nearly any country but we will only analyze the singapore. Therefore, I will just copy the data on my clipboard and import the data from scratch.

Make sure to copy only the values of the data without headers or anything.

# Input data as a single string  
singapore <- "70.19999695 71.09999847 71.69999695 72.30000305 73.09999847 72.90000153 74.40000153 75.40000153 76 76.90000153 77.40000153 78.19999695 78.90000153 78.69999695 79 78 80 79.80000305 80.30000305 80.5 80.69999695 81.09999847 81.5 81.90000153 82.30000305 82.69999695 83.19999695 83.5"  
  
# Convert the string to a numeric vector  
singapore <- as.numeric(strsplit(singapore, "\\s+")[[1]])  
  
# Print the vector  
print(singapore)

## [1] 70.2 71.1 71.7 72.3 73.1 72.9 74.4 75.4 76.0 76.9 77.4 78.2 78.9 78.7 79.0  
## [16] 78.0 80.0 79.8 80.3 80.5 80.7 81.1 81.5 81.9 82.3 82.7 83.2 83.5

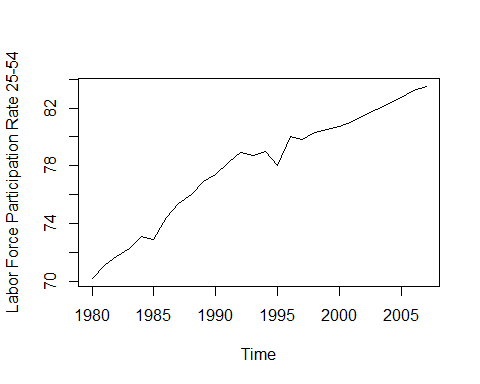
Keep in mind that this is just a vector at this stage, without format, headings, and most importanly no timestamps. And so, we are now going to adjust and convert this vector into a timeseries dataset

# Convert to time series  
singapore = ts(singapore, start = 1980)

We can use Singapore, which makes this an update of the object. We use the ts() function to convert to time series. The dataset is Singapore and we use the Start argument. Now we know from the dataset that the data starts in 1980. Therefore we say start 1980. This dataset has no seasonality or specific cycles, therefore we do not need to use the frequency argument. As a reminder, if you have a dataset that is quarterly, you would have to use frequency equals four as well. That way you would be able to get four observations packed into one year.

Let’s now check the dataset turned into timeseries.

plot(singapore, ylab = "Labor Force Participation Rate 25-54")



Obviously, we will get a line chart with the year on the x axis and the observational values on the Y axis. It is a clearly trending dataset with increasing values over a time span of approximately 30 years. A dataset like this leaves us many options in how to model it. Arima is always an option since it is very versatile and it can handle trends fairly well. But there are also exponential smoothing variations available which can be a nice fit like Holt Linear Trend method. The biggest problem we have to keep in mind here is the fact that this is a rate labor force participation cannot be higher than 100%. In fact, it will even out well before the 100% mark. Therefore, there needs to be some threshold for a linear trend following method, because otherwise the model would go to infinity. Good thing, the Holt method has a nice add on, which is called the damping parameter. With this parameter, it should be possible to control for this problem. Otherwise, strictly linear trend following systems will be useful at least on a short horizon.

Let’s try first with the exponential smoothing.

# Load necessary packages  
library(forecast)

## Warning: package 'forecast' was built under R version 4.3.3

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

Exponential smoothing is primarily covered with the following functions. ses() for simple exponential smoothing. holt() for Holt’s linear trend model, including a damped trend model. hw() for Holt Winters Seasonal method and ets() for an automated parameter selection. With these functions combined with the available function arguments, it is possible to model nearly any time series dataset. As you already know, the labour force participation rate dataset is clearly trending, but there is no seasonality. That means we are only left with the holt() function for a linear trend model. ses() is not permitted since the data is trending. Holt Winters is useless since it is not a seasonal dataset. Of course ETS is always possible since it is an automated model generator which would in any way give us a similar result as the Holt model.

# Exponential smoothing with holt  
holttrend = holt(singapore, h = 5)  
summary(holttrend)

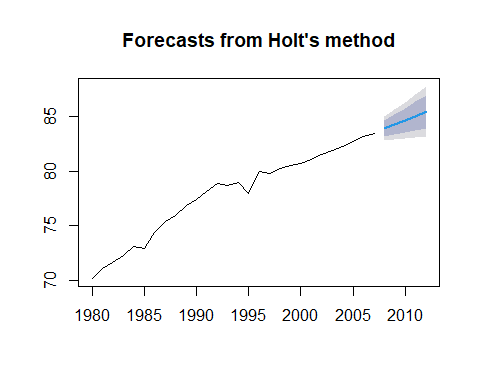
##   
## Forecast method: Holt's method  
##   
## Model Information:  
## Holt's method   
##   
## Call:  
## holt(y = singapore, h = 5)   
##   
## Smoothing parameters:  
## alpha = 0.6378   
## beta = 0.1212   
##   
## Initial states:  
## l = 69.619   
## b = 0.6666   
##   
## sigma: 0.5529  
##   
## AIC AICc BIC   
## 65.79969 68.52697 72.46072   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.08247984 0.5118728 0.3084414 -0.1056525 0.3986079 0.5047223  
## ACF1  
## Training set -0.09190023  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2008 83.90071 83.19216 84.60926 82.81708 84.98435  
## 2009 84.28751 83.39799 85.17703 82.92711 85.64791  
## 2010 84.67431 83.58796 85.76065 83.01289 86.33573  
## 2011 85.06111 83.76361 86.35860 83.07676 87.04545  
## 2012 85.44790 83.92606 86.96975 83.12045 87.77536

Holt’s method is an extension of simple exponential smoothing that allows for forecasting data with a trend but no seasonality. It incorporates two smoothing parameters: Alpha (α): Controls the smoothing of the level. Beta (β): Controls the smoothing of the trend. Gamma which is also a parameter is omitted here since there is no seasonality.

In this summary, we get a lot of crucial info about our model. We get the values for the alpha and beta parameters. Alpha is 0.63 and beta is at 0.12. This indicates that the trend, which is basically the slope of the time series plot, is fairly constant throughout. Even the initial level constant takes several past lags into consideration.

In fact, let’s see the plot. To visualize this scenario, we simply need to run plot on the model and we’re good to go.

# Plot the model  
plot(holttrend)

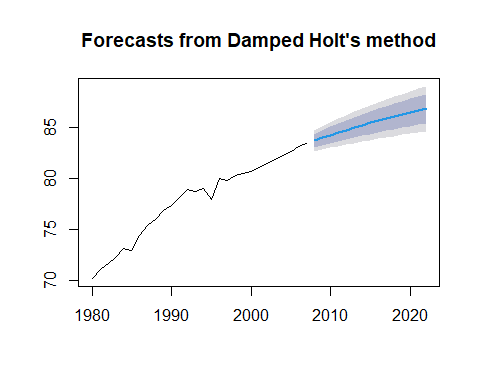


Now, what we get here in blue are the forecasted five years, 2008 to 2012. These two shades are the confidence intervals with 80 and 95% confidence. We can also see that the slope of the forecasted period is nearly the same as with the period between 1998 and 2007. Before 1998, there is a brief period of stagnation, while before that the yearly increase was even steeper. That simply means the model takes a trend slope of at least the last ten years, if not more. Of course, the level value is added to this slope to get the corresponding forecasted value.

Now there is actually a reason why I choose this exact time series. The nature of this particular data contains a classic problem. The trend cannot continue indefinitely. It is simply impossible for a labor force participation rate to cross the 100% mark. A person cannot participate indefinite times in the labor market. In fact, the participation rate will likely come to a halt long before the 100% mark. Think about people being handicapped, unhealthy, undereducated or simply unwilling to participate on the labor market. That means the curve will likely flatten out in the range of 90 to 94% participation.

Fortunately, we can do this fairly easy with a holt function. We just need to use the damped argument. A damped Holt linear trend model assumes that the trend cannot be constant forever. At some point, growth needs to come to an end. The curve needs to flatten out. For that matter, the parameter phi is introduced. If phi is one, it is the same as the standard Holt linear model. Whereas when phi is close to zero, the curve gets flat fairly soon. In practice, the parameter phi is set somewhere between 0.8 and 0.98. When using a phi at around 0.8, you make sure that short run forecasts still contain the trend, whereas the longer forecast lags are at the flatter curve, we can easily adjust our Holt model with the damping parameter. It is possible to have R calculate a parameter value for us or we can set it manually with the phi argument. To show such a Holt model with a damping parameter, I will directly plot the Holt model with an H of 15 so that the flat or damped curve gets optimally visualized. The code itself is the same as with a standard holt model. However, you need to set damped to true or t. That way R is going to select the best suitable parameter phi for you.

# Phi auto generated  
plot(holt(singapore, h = 15, damped = T))



If we run this line, we get the plot, which clearly shows a curve that is more round. The slope is not constant throughout. It gets flat at the end of the trajectory and that is exactly what we do with the damping parameter. We take away the steepness of the slope the more we go into the future. By definition, when you use a damped parameter, the slope of the trend cannot be constant. It changes over time, like here. Now this curve is still not fully flattened out. It would take some more years into the future to reach a totally flat stage in the curve. That is a hint towards a phi that is quite close to one. Remember, if you have a phi of one that is the same as the original slope of the Holt trend. But let’s see what R calculated for us. Let’s run the summary function on the whole thing to extract phi.

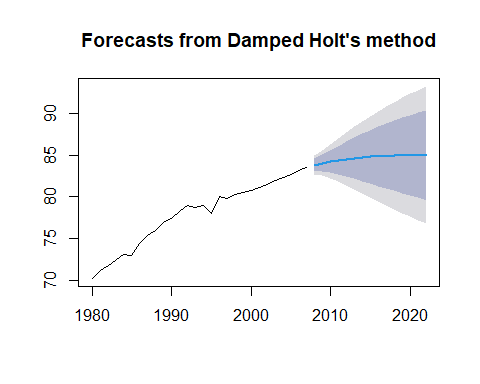
# To see the generated value for Phi  
summary(holt(singapore, h = 15, damped = T))

##   
## Forecast method: Damped Holt's method  
##   
## Model Information:  
## Damped Holt's method   
##   
## Call:  
## holt(y = singapore, h = 15, damped = T)   
##   
## Smoothing parameters:  
## alpha = 0.5149   
## beta = 1e-04   
## phi = 0.9666   
##   
## Initial states:  
## l = 69.5404   
## b = 0.771   
##   
## sigma: 0.5149  
##   
## AIC AICc BIC   
## 62.62649 66.62649 70.61972   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.01419094 0.4667051 0.3250851 0.01746896 0.419571 0.5319574  
## ACF1  
## Training set -0.001595197  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2008 83.72693 83.06701 84.38685 82.71766 84.73620  
## 2009 84.00520 83.26291 84.74750 82.86996 85.14044  
## 2010 84.27418 83.45776 85.09060 83.02557 85.52279  
## 2011 84.53417 83.64979 85.41856 83.18163 85.88672  
## 2012 84.78548 83.83798 85.73298 83.33641 86.23456  
## 2013 85.02840 84.02171 86.03509 83.48880 86.56800  
## 2014 85.26320 84.20060 86.32580 83.63809 86.88831  
## 2015 85.49016 84.37443 86.60589 83.78380 87.19653  
## 2016 85.70954 84.54308 86.87600 83.92560 87.49348  
## 2017 85.92159 84.70651 87.13668 84.06328 87.77990  
## 2018 86.12656 84.86471 87.38841 84.19673 88.05639  
## 2019 86.32469 85.01773 87.63164 84.32587 88.32350  
## 2020 86.51619 85.16563 87.86675 84.45068 88.58170  
## 2021 86.70130 85.30848 88.09412 84.57117 88.83143  
## 2022 86.88023 85.44639 88.31407 84.68736 89.07310

And indeed, as we can see here, next to the smoothing parameters, we get phi of over 0.6. That explains why it takes so long to get the whole curve at a flat trajectory.

But let’s manually set fee to learn how the curve reacts to a lower phi. Let’s say we test a phi of 0.8, which is the lower limit of what is generally recommended. Therefore, we again set damped to true and we also state fee equals 0.8. If we now run this whole thing again as a plot, we get a totally different forecasting curve.

# Manual setting of Phi  
plot(holt(singapore, h = 15, damped = T, phi = 0.8))



As you can clearly see, just within a few years, the curve gets flat. That is due to the low phi of 0.8. It aggressively dampens the whole forecast curve just within a few time lags. Interestingly, the prediction intervals get fairly wide after some time. This is significantly wider than with the are generated fee of 0.96. Generally, it is wise to stick with the damping parameter which R provides for you. However, there might be situations where you want to set it manually.

Now lets try how this performs with arima model. The most important thing to keep in mind is the trend of the dataset. A trend in the Time series means autocorrelation is present. Autocorrelation means that an observation at an earlier time point influences the later observations, and that is exactly what happens in a trending dataset. The values tend to increase or decrease over time. The first of the three parameters of an Arima model is the Autoregressive part, and this one deals with Autocorrelation. Whenever you see a dataset containing a clear trend, chances are that you will get a parameter for the AR part. Therefore, in our case we expect a model which is heavy on the AR part, whereas the moving average is more prevalent with random or rather flat datasets which we do not see here. Now there are Arima models for seasonal datasets, but since this is a yearly dataset without season, a standard Arima model is suitable with a forecast package.

We will use the auto.arima which does the parameter selection for us.

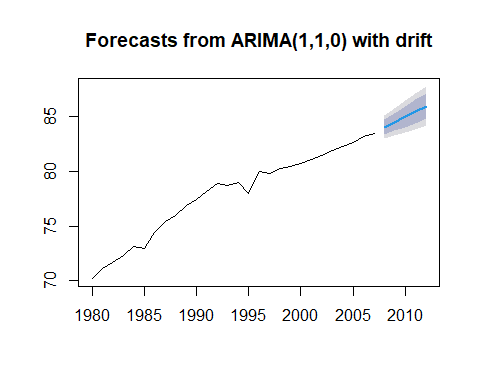
# Arima auto generated  
singapore.arima = auto.arima(singapore)  
summary(singapore.arima)

## Series: singapore   
## ARIMA(1,1,0) with drift   
##   
## Coefficients:  
## ar1 drift  
## -0.3690 0.4904  
## s.e. 0.1763 0.0720  
##   
## sigma^2 = 0.2779: log likelihood = -20.05  
## AIC=46.1 AICc=47.14 BIC=49.99  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.006855948 0.4981113 0.3755194 0.01821962 0.4863707 0.6144862  
## ACF1  
## Training set 0.05505323

So if we execute this line and if we get the summary of it right away, we get an output like this. So that is the model we just created. It is an Arima(1,1,0) with drift, Obviously, there’s an AR part present because of the trend and this is exactly what R recognized as well. The model also went for the first order of Differencing and this is actually a useful method to get some chaos out of the data, which makes it easier to model. With Differencing, the dataset is changed to the gap. So if you have a differenced data set, R does not use the actual observational values anymore. It uses the differences between one lag of observations with these two parameters. R is able to mirror the whole dataset. There is no more need to have an MA moving average part. The coefficient for the autoregressive parameter, including the standard error. We can find here as well as the constant for the drift. Overall, the summary output has the same structure. Like any forecasting model, we get the info criteria for model comparison as well as the accuracy indicators.

Let’s actually take a look at its plot.

# Plotting the auto.arima model  
plot(forecast(singapore.arima, h = 5))



Like always, with this simple plot setup, we have the forecast in blue, including the 80 and 95 prediction intervals. This plot clearly exemplifies that the Arima model is a linear endeavor. There is no damping going on, and since we have no seasonal patterns or other forms of cycles, the model uses the trend from the original dataset and projects it forward. This would go on indefinitely, beyond 100% if I would increase the age to, let’s say, 55. Now, this setup might work fine on a short time span, probably 5 or 7 years, but on a longer time span I would be skeptical due to the nature of this dataset. I would expect some sort of flattening to occur. By the way, with the auto.arima function, there is a little trick you can apply. The function is preset in a way as to handle several time series at once. If you have only one time series like we do in this case, we can adjust and inactivate the parameters, stepwise and approximation to get a more accurate model. That way the calculation might take a bit longer. However, the result is more accurate.

# Exact calculation of Arima parameters  
auto.arima(singapore, stepwise = F, approximation = F)

## Series: singapore   
## ARIMA(1,1,0) with drift   
##   
## Coefficients:  
## ar1 drift  
## -0.3690 0.4904  
## s.e. 0.1763 0.0720  
##   
## sigma^2 = 0.2779: log likelihood = -20.05  
## AIC=46.1 AICc=47.14 BIC=49.99

However, if we execute this line, we can see that the result is still the same.

Let’s now compare all models we just created to see which is the best in forecasting this dataset.

# Overview plot - models  
holttrend = holt(singapore, h = 10)  
holtdamped = holt(singapore, h = 10, damped = T)  
arimafore = forecast(auto.arima(singapore), h = 10)  
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.3.3

Now let’s see how the model performs but this time, with 10 year forecast.

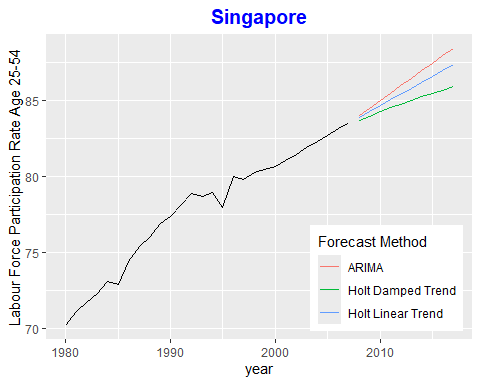
# 3 Forecast Lines as Comparison  
autoplot(singapore) +  
 forecast::autolayer(holttrend$mean, series = "Holt Linear Trend") +  
 forecast::autolayer(holtdamped$mean, series = "Holt Damped Trend") +  
 forecast::autolayer(arimafore$mean, series = "ARIMA") +  
 xlab("year") + ylab("Labour Force Participation Rate Age 25-54") +   
 guides(colour=guide\_legend(title="Forecast Method")) + theme(legend.position = c(0.8, 0.2)) +  
 ggtitle("Singapore") + theme(plot.title=element\_text(family="Times", hjust = 0.5, color = "blue",  
 face="bold", size=15))

## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2  
## 3.5.0.  
## ℹ Please use the `legend.position.inside` argument of `theme()` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## Warning in grid.Call(C\_stringMetric, as.graphicsAnnot(x$label)): font family  
## not found in Windows font database

## Warning in grid.Call(C\_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font  
## family not found in Windows font database

## Warning in grid.Call.graphics(C\_text, as.graphicsAnnot(x$label), x$x, x$y, :  
## font family not found in Windows font database



From this plot, it is quite clear from the start that the Damped Holt model is the most conservative model which tends towards the lowest labor participation rate in the future. The Arima model is the most aggressive one with the highest projected rates and on top of that it is strictly linear, meaning the slope will keep the angle indefinitely.

In summary, the Holt Damped Trend model is considered the best for forecasting the labor force participation rate in this context because it provides a realistic representation of future trends by assuming that the growth rate will eventually slow down. This model balances simplicity with realistic forecasting, making it a robust choice for capturing the expected deceleration in the labor force participation rate.

If you are not satisfied, you can check the accuracy using the accuracy() function.

# Calculate error metrics for training set  
arima\_accuracy <- accuracy(arimafore)  
holt\_linear\_accuracy <- accuracy(holttrend)  
holt\_damped\_accuracy <- accuracy(holtdamped)  
  
arima\_accuracy

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.006855948 0.4981113 0.3755194 0.01821962 0.4863707 0.6144862  
## ACF1  
## Training set 0.05505323

holt\_linear\_accuracy

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.08247984 0.5118728 0.3084414 -0.1056525 0.3986079 0.5047223  
## ACF1  
## Training set -0.09190023

holt\_damped\_accuracy

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.01419094 0.4667051 0.3250851 0.01746896 0.419571 0.5319574  
## ACF1  
## Training set -0.001595197

The Holt Damped Trend model appears to be the best overall model for forecasting the labor force participation rate based on these metrics. It has the lowest RMSE, which indicates it handles large errors better, and a low ME, indicating minimal bias. The ACF1 value close to zero suggests that the residuals do not have significant autocorrelation, which is desirable.