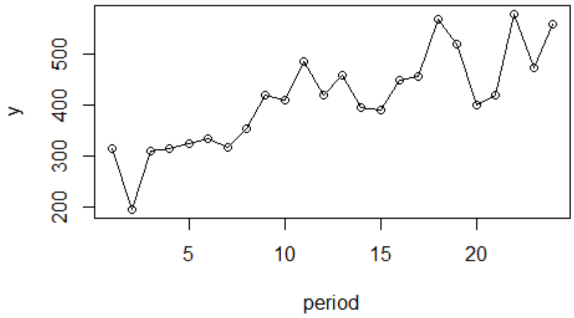
Time series HW4

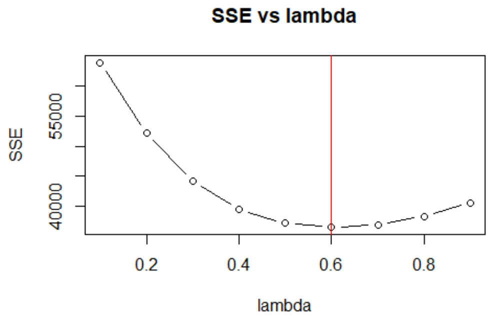
104304005 陶培生

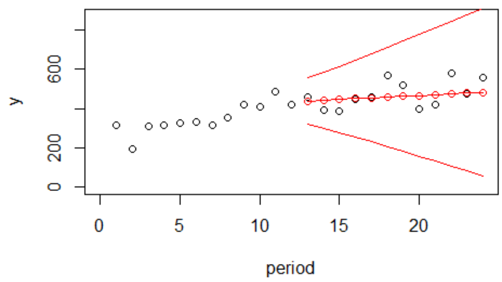
4.8

(a) We can see a trend in the plot



(b, c) Here I first calculated the best lambda, and because the data has a trend so I use the double exponential smoothing.



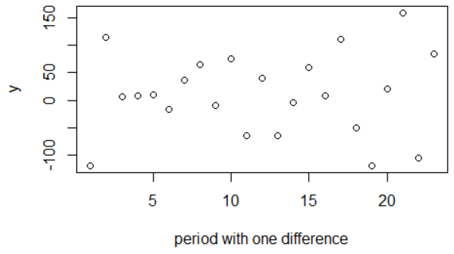


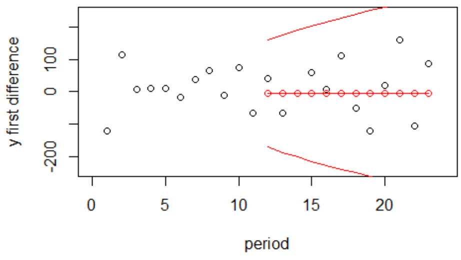
The forecasting error is in the picture below. I think that this forecast seems to work fine, we can see in the plot that the forecast is not too far away from the actual value.

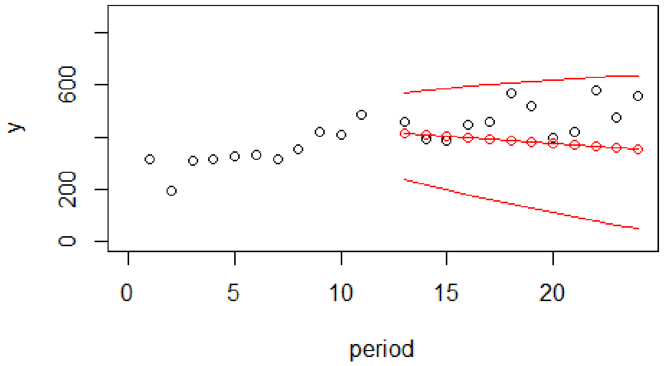
```
> print(thiserror)
 [1]  -20.683340   48.325133   57.333607    1.342080   -2.649447 -110.640974  -56.632501   67.375973   51.384446
[10] -104.607081    4.401392  -76.590135
```

4.9

Since the first difference does not have a trend, I used the simple exponential smoothing instead. We can see in the third plot that its forecast is getting away from the actual trend, but at least in the 95 CI. Last, the forecast errors are larger than the first forecast.



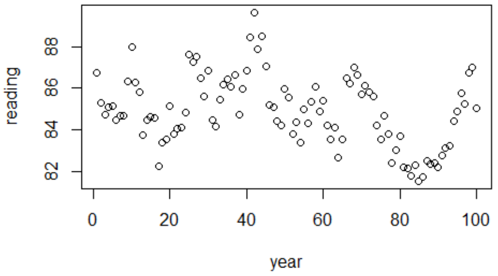


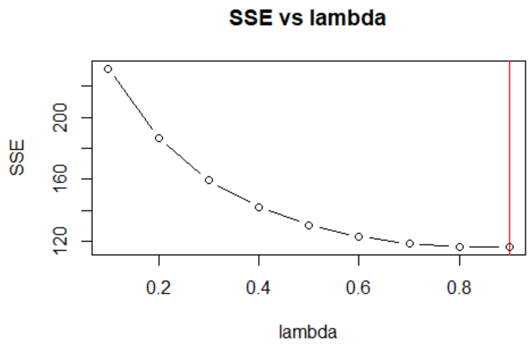



```
> print(thiserror)
 [1]  -45.47512   14.04976   13.57463  -51.90049  -65.37561 -182.85073 -138.32586  -23.80098  -49.27610
[10] -214.75122 -115.22634 -205.70147
```

4.15

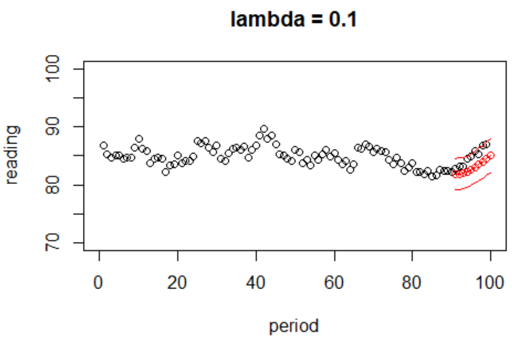
(a) First I calculated the optimum value of lambda = 0.9. Next I compared their errors in the picture below. We can see that the errors are smaller than the ones when lambda=0.1

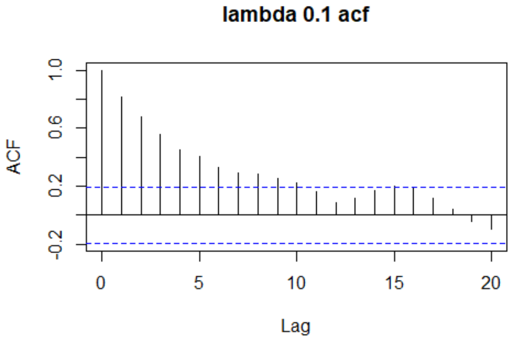




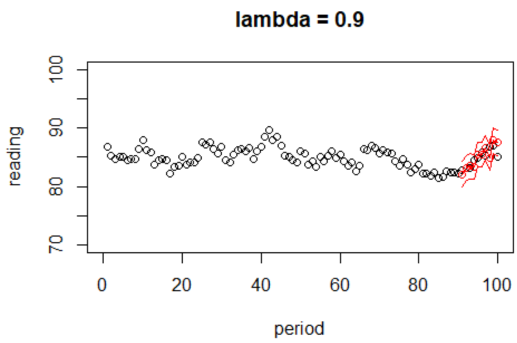

```
> print(measacc(data15[,2],0.1))
       SSE       MAPE        MAD        MSD
231.578730   1.511824   1.283660   2.315787
>   #lambda 0.9 error
> print(measacc(data15[,2],0.9))
        SSE        MAPE         MAD         MSD
116.0241921   1.0043571   0.8550947   1.1602419
```

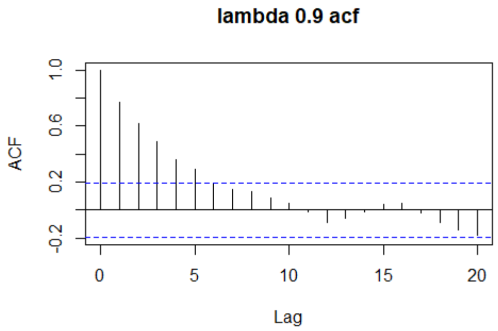
(b, c) In the pictures below are the forecasts, error calculation and ACF for lambda = 0.1 and lambda = 0.9. The forecasts are similar but the second one is slightly more precise, and its calculated errors are smaller. The sample ACF looks the almost the same.






```
> print(thiserror)
 [1] -0.90010710 -1.18646111 -1.14734324 -2.23798434 -2.33182379 -2.76741550 -1.71547063 -2.87794058 -2.56336184
[10] -0.06171944
```

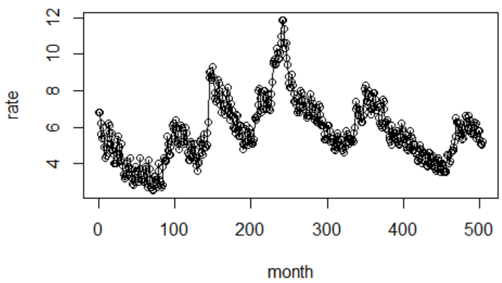


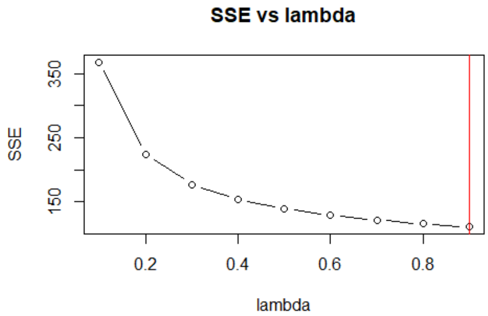



```
> print(thiserror)
 [1] -0.71788013  0.05518192  0.27741519 -1.07886878  0.54865209 -0.26238089  1.31913730 -1.76524873  0.86115888
[10]  2.41108426
```

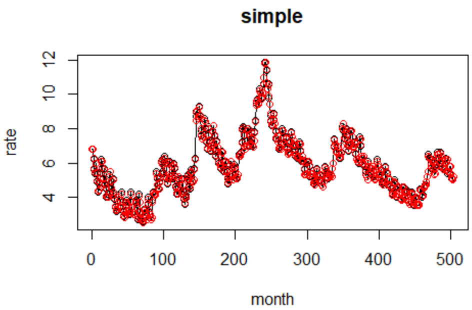
4.24

(a) First, I find optimum value of lambda = 0.9. Then, I use simple exponential smoothing and calculate the errors for lambda = 0.9 and lambda = 0.2. In the third picture we can see that errors are smaller when using the optimum lambda.





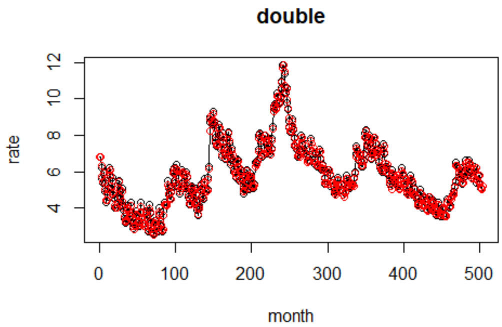

```
> print(measacc(data24[,2],0.2))
        SSE        MAPE         MAD         MSD
224.2225047   9.2513706   0.5187099   0.4448859
>   #lambda 0.9 error
> print(measacc(data24[,2],0.9))
        SSE        MAPE         MAD         MSD
110.6944769   6.4248868   0.3556993   0.2196319
```

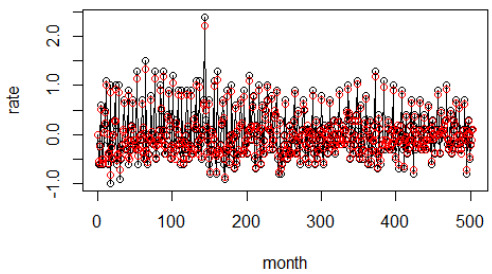


(b) I think that there is a better chance of getting a better forecast using the optimum lambda but it doesn't mean that it will always be better.

(c) In the two plots for simple and double exponential, I think that there is not too much difference between the smoothing results

(d) I think that smoothing for the first difference is also suitable

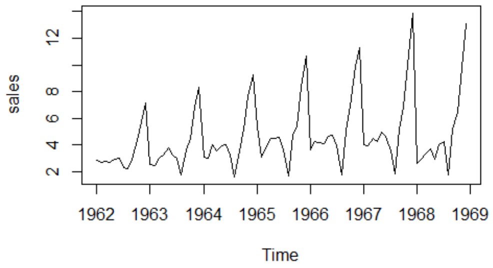


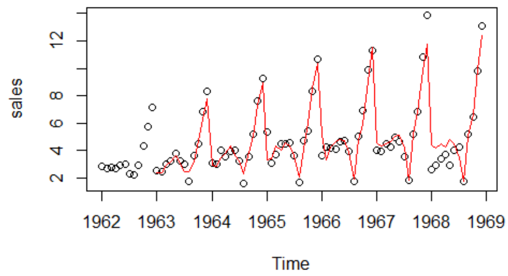


4.29

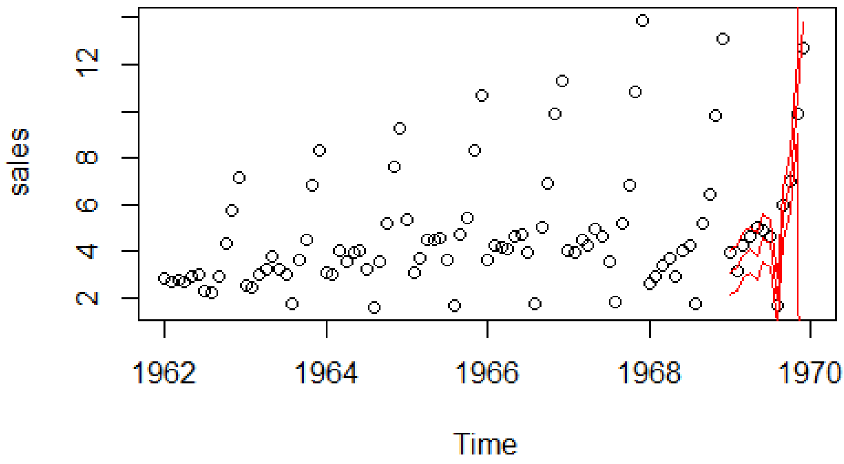
(a) We can see in the plot that this data is seasonal, and the frequency is once an year. I think that the sales for champagne is seasonal by year is because of the different demand in different seasons.

(b) Using Winter's multiplicative method smoothing we can see the results in the second plot. We can see that the smoothing values also are seasonal and that is good.





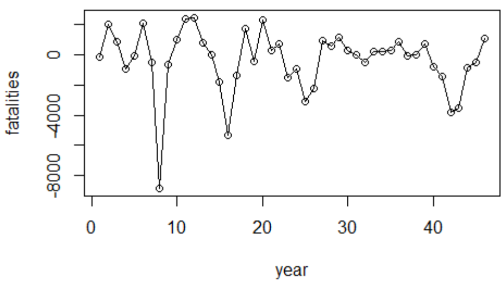
(c) In the forecast, we can see that the actual values are following the forecast interval. In the calculated error below, We can also see that the errors are reasonable.

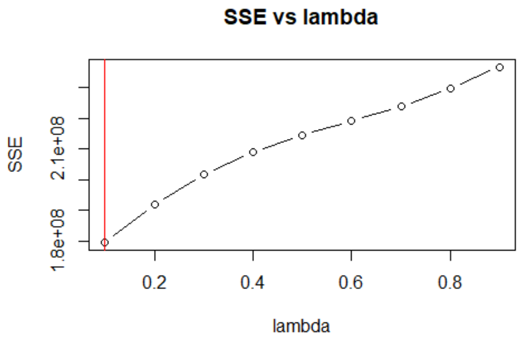



```
> print(thiserror)
 [1] -0.851718287  0.131349212 -0.470747252 -0.594047651 -1.218632439 -0.315128700 -0.306631447  0.228096317
 [9] -0.401184564  0.002506315  0.824714474  1.143603086
```

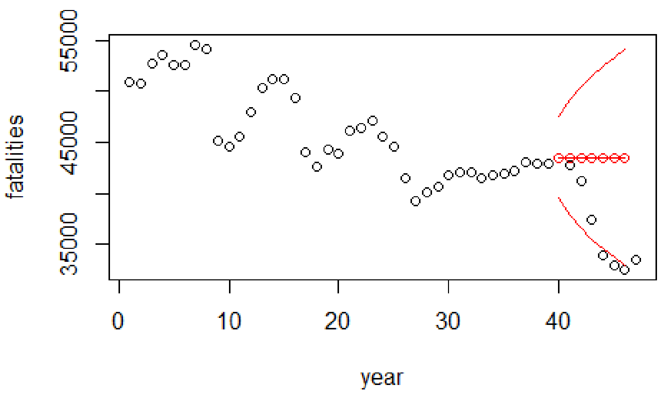
4.53

(a) In the first difference plot, I think that it has a slight trend and is a little seasonal.

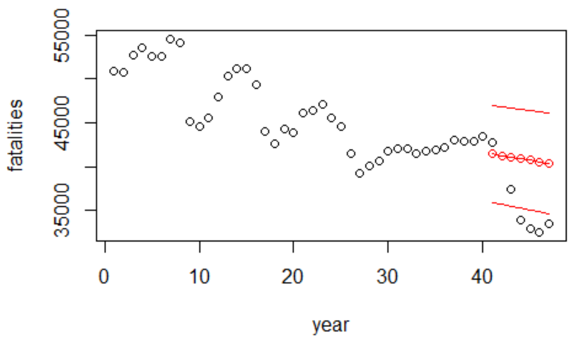




The plot below is simple exponential smoothing and forecasting, the second plot is double exponential smoothing and forecasting.

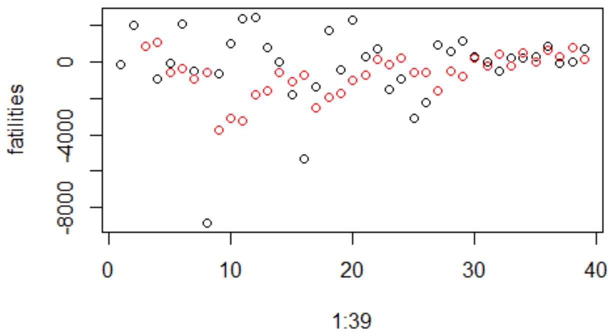



```
> print(thiserror)
[1]   -802   -2251   -6087   -9627  -10511  -11031   -9949
>
```

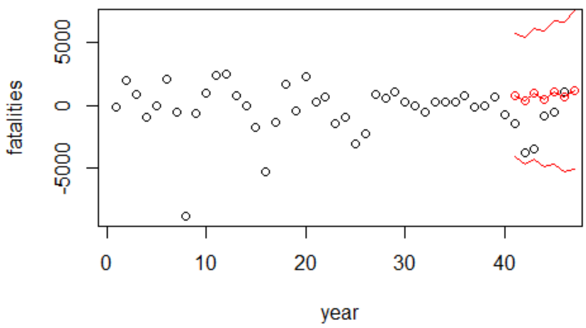



```
> print(thiserror)
[1] -1278.736707    -7.555696  3650.625316  7012.806328  7718.987339  8061.168351  6801.349363
```

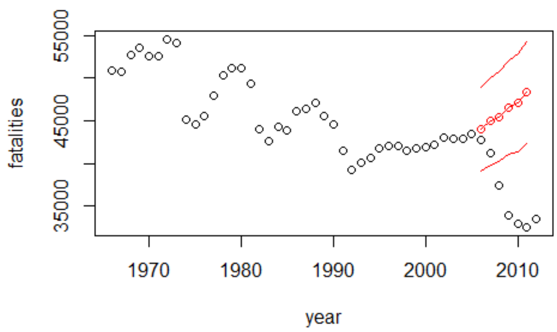
This plot is the smoothing for first difference. Since I think that it is a little seasonal, I used the winter's method.



Forecasting first difference using Winter's method



Actual forecasting plot after adding forecasted difference




```
> print(thiserror)
[1]  1291.997  3663.258  8002.806 12595.973 14114.427 15818.500
```

In the three forecasts, we can see that the double exponential has the best results, second best is simple exponential, last is the Winter's seasoning forecast.