Time Series Project 1

Congyu Hang

2023-10-16

Practice

1. Time series models are used in Finance to describe asset price movements. For this question you will work with Canadian stock prices, examining the Efficient Market/Random Walk hypothesis.

a. [2 marks] Using your student ID# as a seed, pick a random number from 1-60 and download the corresponding stock from the table in the Appendix. **Make sure to use the right data, otherwise you will lose marks.** Use the quantmod library to download and plot the adjusted closing price data over 2010-2020, as in the example below. Note that you need to use the appropriate stock symbol for your company, which should end in ".TO"; you can find that by searching for its ticker at Yahoo! Finance.

```
set.seed(1006740369) # set random see to your student ID
sample(1:60, size =1) # pick a random number from 1-60
```

```
## [1] 53
```

```
#install.packages("quantmod")
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

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

```
getSymbols("TD.TO", from = "2010/12/31", to = "2020/12/31")
```

```
## [1] "TD.TO"
```

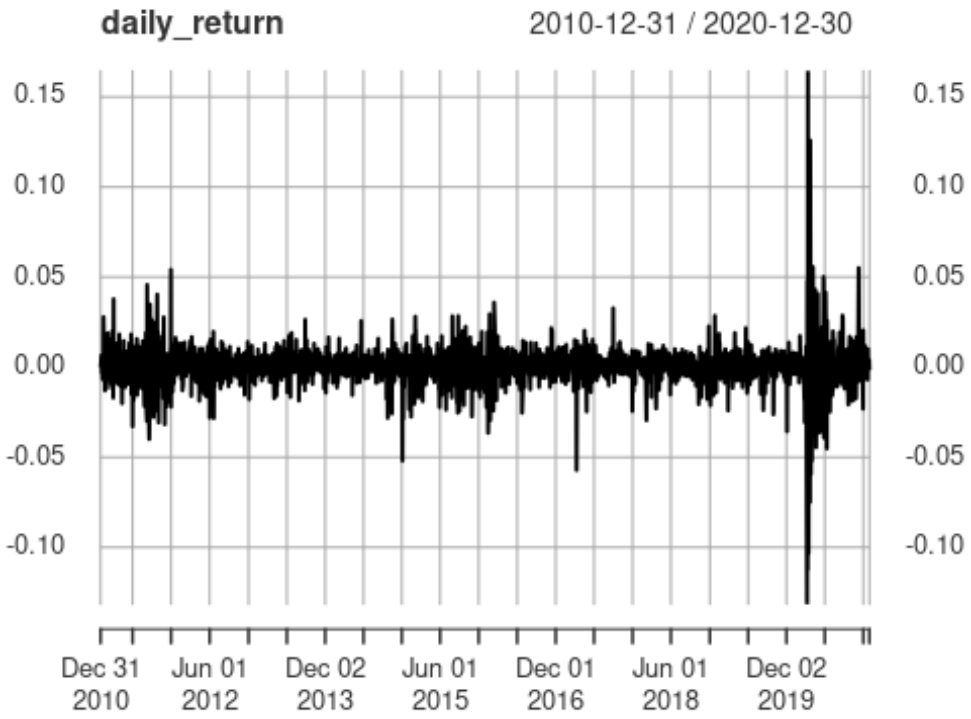
```
plot(TD.TO$TD.TO.Adjusted)
```

b. [2 marks] Calculate and plot the logarithmic returns with the function `quantmod::dailyReturn()` with argument `type = "log"`. Verify that you get the same values by first order differencing of the logarithm of the price series, i.e. $R_t = \nabla \ln(P_t) = \ln(P_t) - \ln(P_{t-1}) = \ln(P_t/P_{t-1})$, where R_t is the log-return and P_t is the adjusted closing price.

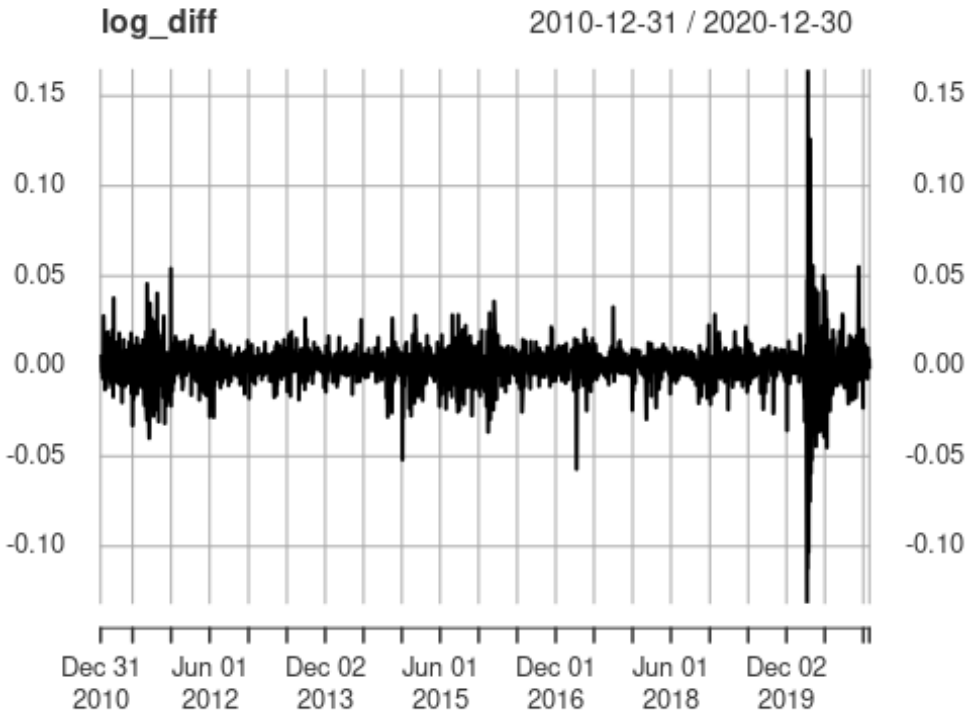
```
daily_return = dailyReturn(TD.TO$TD.TO.Adjusted,type="log")
log_diff = diff(log(TD.TO$TD.TO.Adjusted))

#the 1st element of daily_return is NA, so remove NA to compare
all.equal(as.numeric(na.omit(daily_return[-1])),as.numeric(na.omit(log_diff)))

## [1] TRUE
```

TRUE indicate that I get the same values by first order differencing of the logarithm of the price series.

```
plot(daily_return)
```



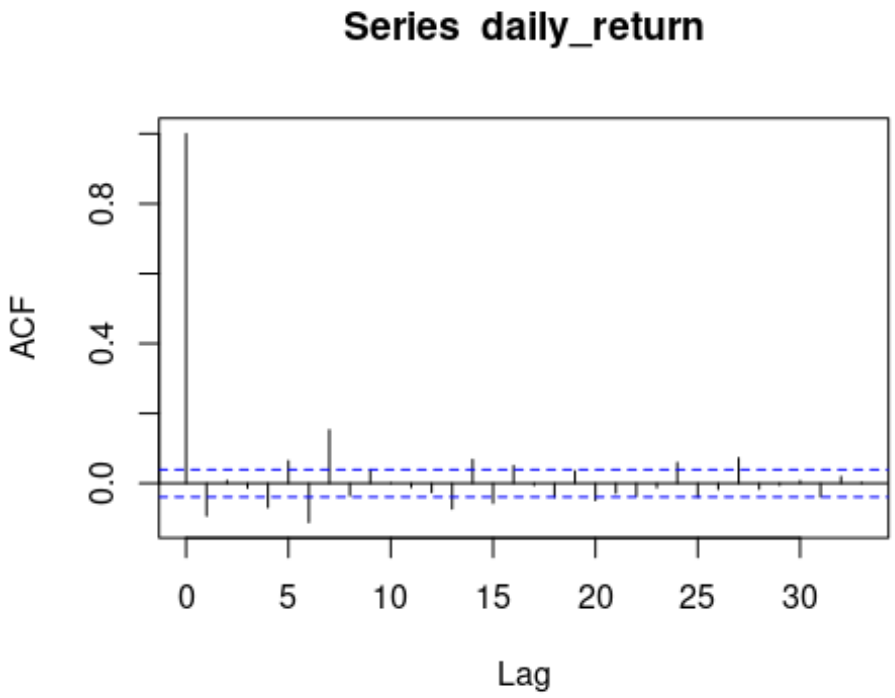

```
plot(log_diff)
```



The 2 plots are the same.

c. [2 marks] The Random Walk hypothesis assumes that log-prices follow a random-walk with drift, i.e. $\ln(P_t) = \mu + \ln(P_{t-1}) + W_t$, where W_t is a White Noise. This implies that the returns $R_t = \mu + W_t$ are **uncorrelated**. Plot the ACF of the returns for your stock and comment on whether they look uncorrelated.

```
acf(daily_return)
```



Comment: They do not look correlated since when lag > 0, the absolute value of correlation is very small and most of them are within the 95%-confidence interval.

d. [4 marks] Consider what would happen if the returns were predictable, let's say following an MA(1) model $R_t = \mu + V_t + .5V_{t-1}$, where $V_t \sim^{iid} N(0, \sigma_v^2)$. First, estimate the values of μ and σ_v^2 based on the mean and variance of your stock's daily returns, by setting $\mu = \bar{R}$ and $\sigma_v^2 = S_R^2/1.25$, where $\bar{R}, S_R^2$ are the sample mean and variance of the returns. (If you are wondering about the division by 1.25, remember that for the MA(1) model $X_t = W_t + \theta W_{t-1}$ we have $V[X_t] = (1 + \theta^2)\sigma_w^2$.)

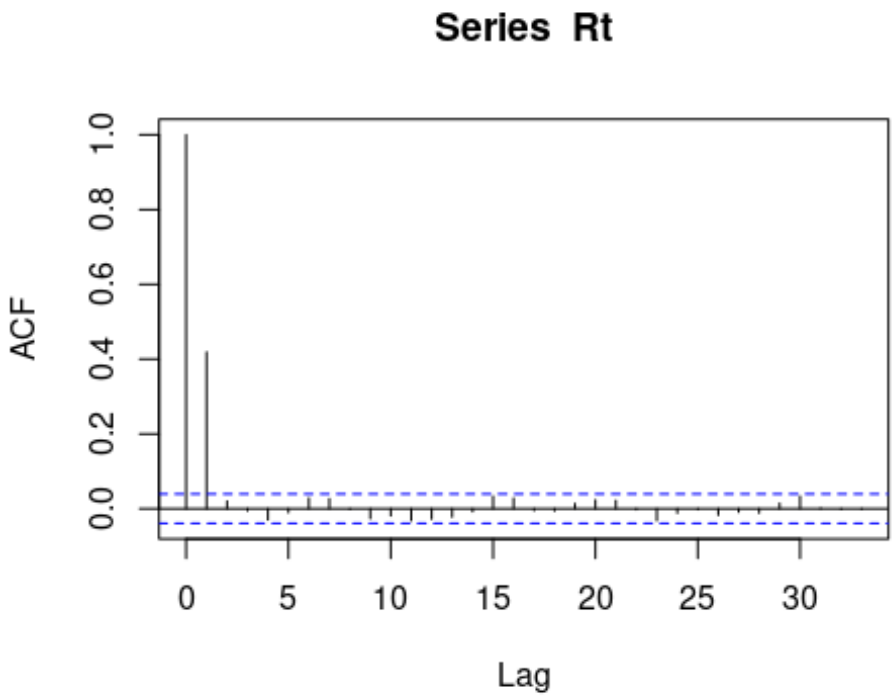
If returns are predictable then it would be easier for people to arbitrage by switching their positions of long and short depending on the signs of previous stock price to make profits.

```
(miu = mean(daily_return))

## [1] 0.0004141086

(sigma2 = var(daily_return)/1.25)

##                daily.returns
## daily.returns  0.0001074572

Vt = rnorm(n=2500,mean = 0,sd=sqrt(sigma2))
#Rt = miu+filter(Vt,c(1,0.5),method = "convolution")
```

miu = 0.0004141087 sigma2 = 0.0001074572

e. [3 marks] Simulate 2500 returns from the model in the previous part and plot their ACF; comment on the type of autocorrelation you observe. Comment: The series looks correlated when lag=1, the ACF is approximately 0.4, which indicates that the stock price of each day is positively correlated to the stock price of previous day. This is why we use the strategy to decide each day's position depending on previous day's stock price.

```
Rt = miu+stats::filter(Vt,c(1,0.5),method = "convolution")
acf(Rt,na.action = na.pass)
```



f. [2 marks] Use the returns you simulated in the previous part to evaluate the following simple trading strategy:

- if today's return is +ve, then buy (long) the stock until tomorrow
- if today's return is -ve, then sell (short) the stock until tomorrow

Assuming no transaction costs, the aggregate return of this strategy is the sum of returns over the days you bought the stock, minus the sum of the returns over the days you sold the stock (i.e. long returns are added, and short returns are subtracted). Calculate the aggregate return of the strategy over the 2500 simulated days, and compare it to the aggregate return of a buy-and-hold strategy (i.e. sum of all returns). What do you observe?

```
amt = as.numeric(as.vector(Rt))
stg = sign(amt)
return_in_days = amt[-1]*stg
sum(return_in_days,na.rm = TRUE)

## [1] 10.03679

#Method of Loop
new_Rt = na.omit(Rt)
rt = 0
for (i in 1:(length(new_Rt)-1)) {
  if(new_Rt[i] > 0){
    rt = rt + new_Rt[i+1]
  }
  else{
    rt = rt - new_Rt[i+1]
  }
```

```
}
sum(rt,na.rm = T)

## [1] 10.03679
```

buy and hold strategy

```
sum(Rt,na.rm = TRUE)

## [1] 1.289785
```

Observation: Both returns are positive, however, if I apply the strategy, my return is much larger than the one I get from buy and hold strategy, i.e. the strategy is more profitable.

2. For this question you will work with Statistics Canada's open data. The data are organized by topic in tables[1], and we will focus on monthly Labour Force Characteristics, and in particular employment numbers, both as unadjusted (raw) and seasonally adjusted series. You can access such data directly within R with the cansim library, using "vectors" to download individual series.

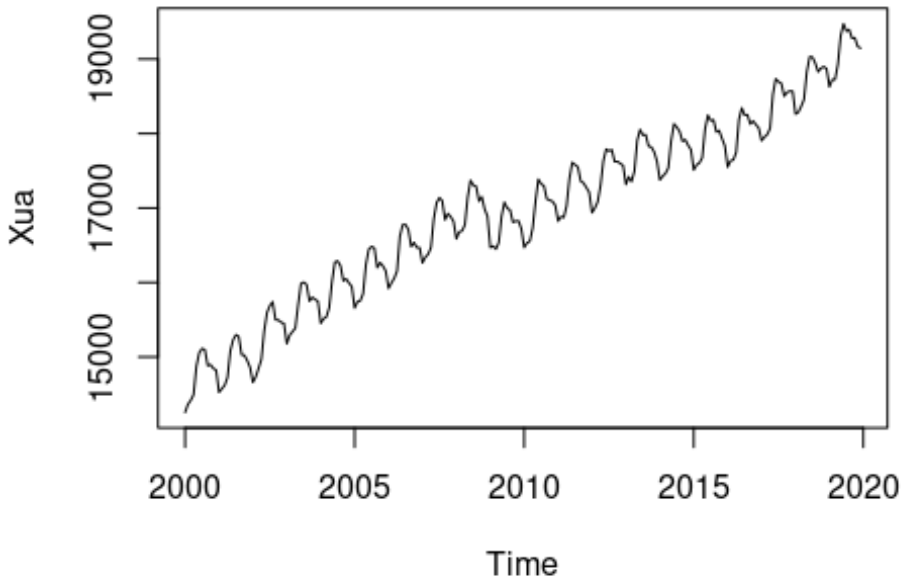
You will be working with employment numbers from a single geographic location. To find your location generate a random number from 1-11 with your ID# as the seed (similar to previous part), and use the corresponding series from the following table:

#	GEO	Seasonally adjusted	Unadjusted
1	Canada	v2062811	v2064890
2	Newfoundland and Labrador	v2063000	v2065079
3	Prince Edward Island	v2063189	v2065268
4	Nova Scotia	v2063378	v2065457
5	New Brunswick	v2063567	v2065646
6	Quebec	v2063756	v2065835
7	Ontario	v2063945	v2066024
8	Manitoba	v2064134	v2066213
9	Saskatchewan	v2064323	v2066402
10	Alberta	v2064512	v2066591
11	British Columbia	v2064701	v2066780

Below is sample code for downloading the unadjusted employment numbers for Canada (vector code v2064890):

```
library(cansim)
library(tidyverse)
# unadjusted series for Canada
Xua = get_cansim_vector( "v2064890", start_time = "2000-01-01", end_time = "2019-12-01")
%>%
  pull(VALUE) %>% ts( start = c(2000,1), frequency = 12)
plot(Xua)
```

[1] see also this brief tutorial

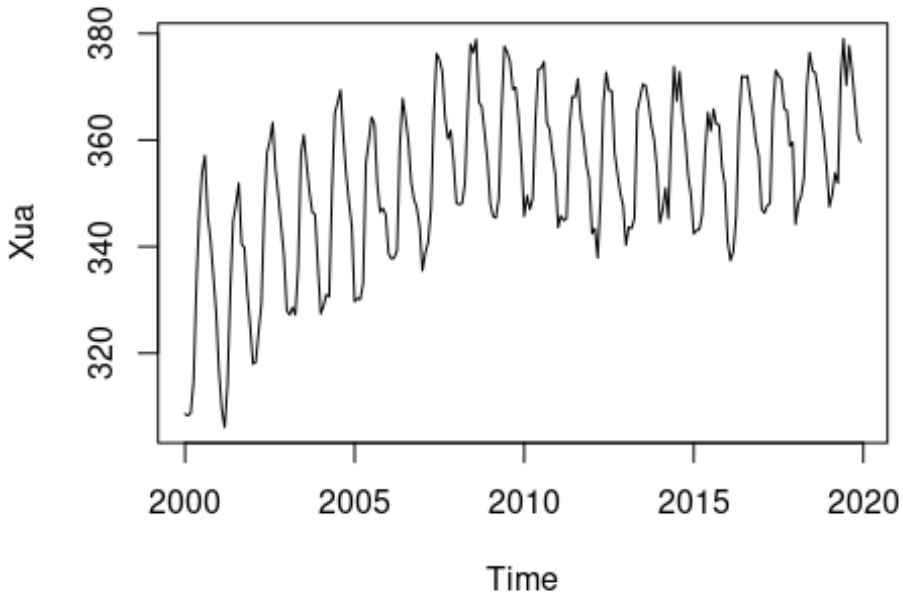



```
set.seed(1006740369) # set random see to your student ID
sample(1:11, size =1) # pick a random number from 1-11

## [1] 5

Xua = get_cansim_vector( "v2065646", start_time = "2000-01-01", end_time = "2019-12-01")
%>%
  pull(VALUE) %>% ts( start = c(2000,1), frequency = 12)

## Accessing CANSIM NDM vectors from Statistics Canada
```

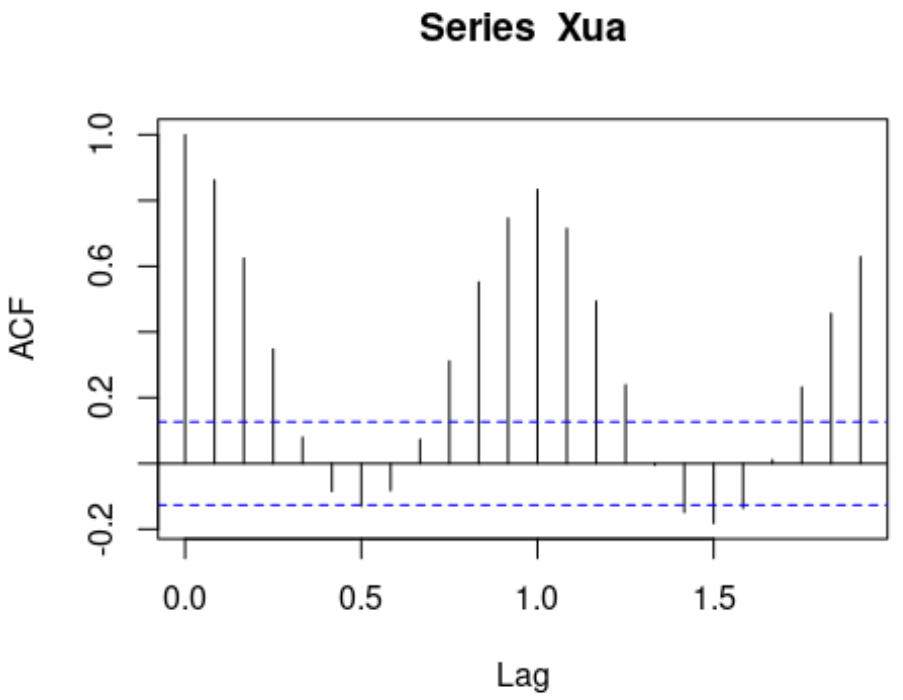
1. [3 marks] Plot the unadjusted series and its ACF, and comment on the following characteristics: trend, seasonality, stationarity.

Comment: The plot shows that the time series has an upward trend with an annual seasonality since ACF reaches peak when lag=0 and lag=1, the peak at lag=1 indicate an annual seasonality. The series is NOT stationary before de-trending and seasonal adjustment, however, after de-trending and seasonal adjustment, the series will be stationary.

```
plot(Xua)
```



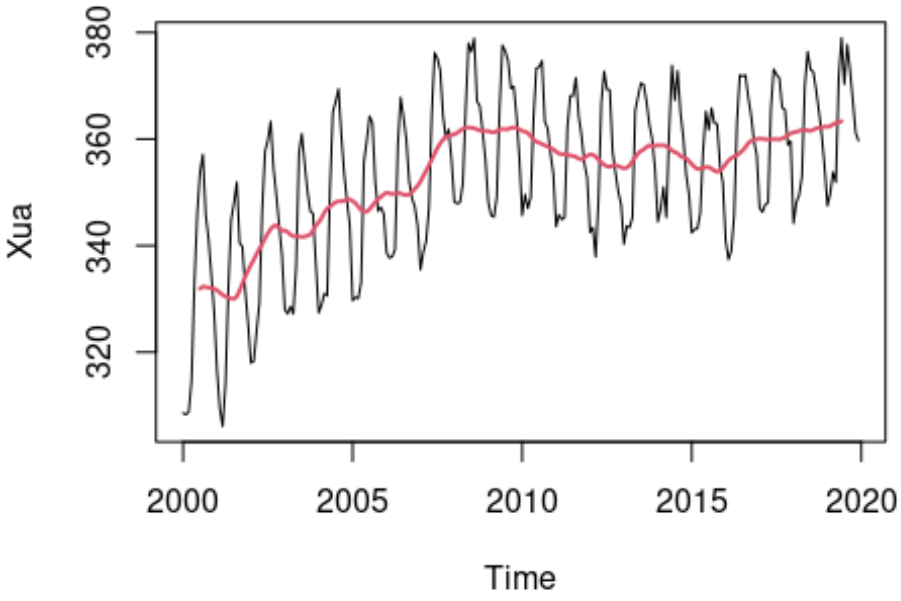

```
acf(Xua)
```



2. [5 marks] Perform a classical *multiplicative* decomposition of the unadjusted series (X_{ua}) into trend (T), seasonal (S), and remainder (R) components (i.e. $X_{ua} = T \times S \times R$): a. First, apply a *12-point MA* to the raw (unadjusted) series to get an estimate of the trend.

```
wgts = c(0.5,rep(1,11),0.5)/12
soif = stats::filter(Xua,sides = 2,filter = wgts)
```

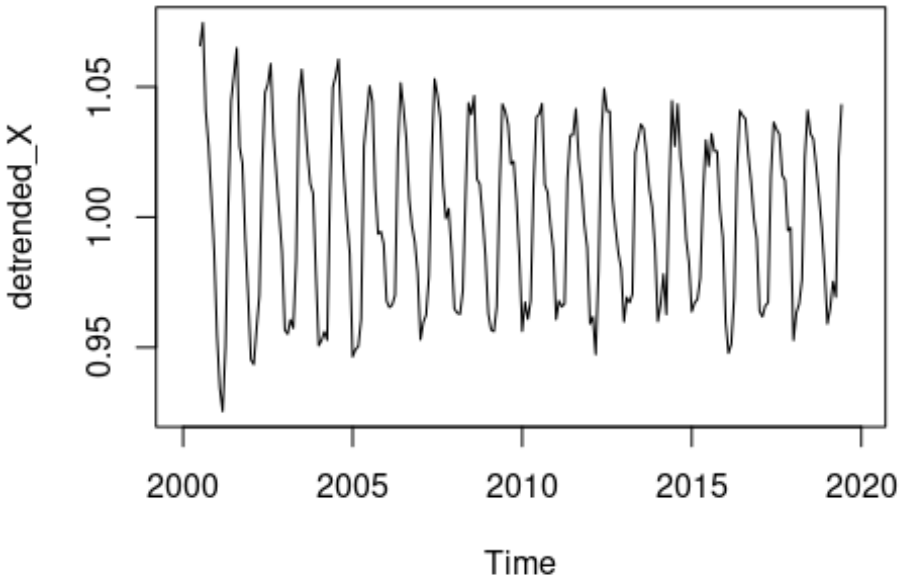
```
plot(Xua)
lines(soif,lwd=2,col=2)
```



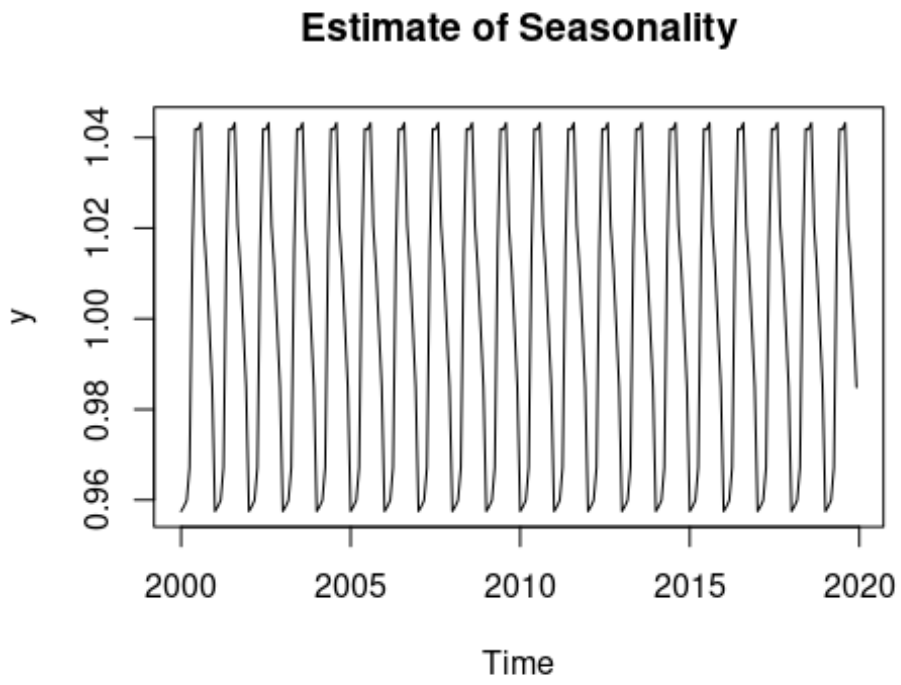
The trend is upward.

b. Then, use the *detrended* data to estimate seasonality: find the seasonal pattern by calculating sample means for each month, and then center the pattern at 1 (i.e divide the pattern by its mean, so that its new mean is 1).

```
detrended_X = Xua/soif
plot(detrended_X)
```



```
a=decompose(Xua,type = "multiplicative")
month_mean = tapply(detrended_X, cycle(detrended_X), mean, na.rm = TRUE)

x = as.numeric(month_mean)
y = ts(rep(x/mean(x),20),start = c(2000,1), frequency = 12)

plot(y,main = "Estimate of Seasonality")
```

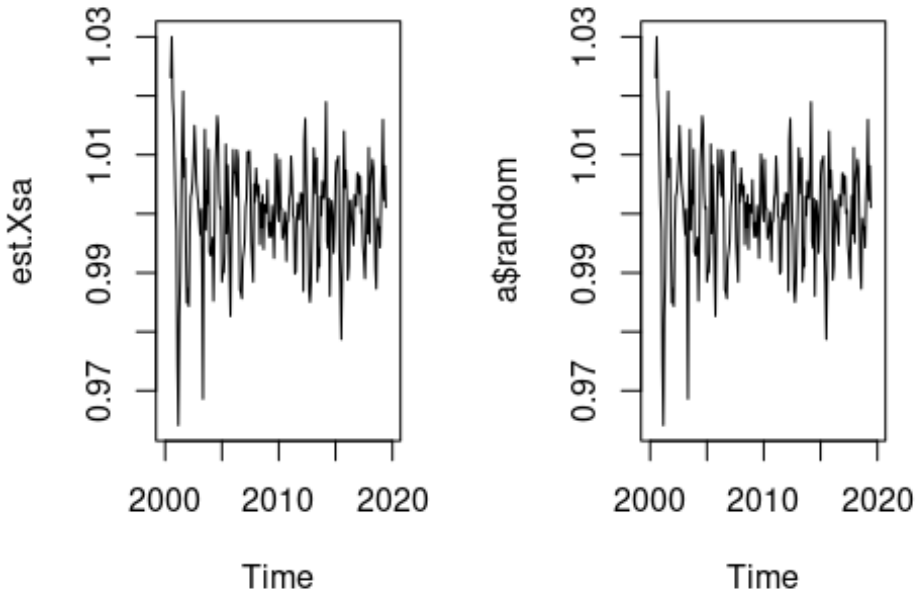



```
identical(y, a$seasonal)

## [1] TRUE
```

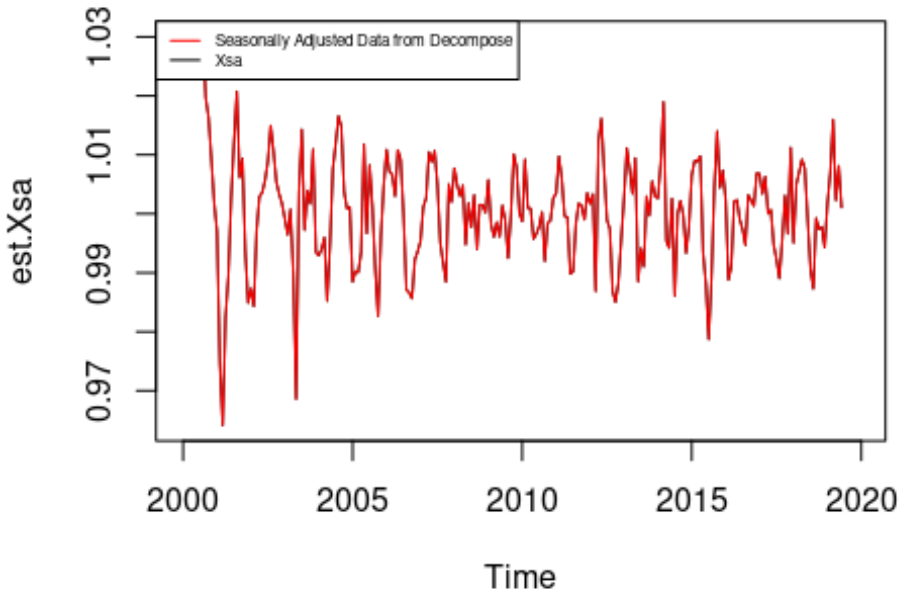
The seasonality estimated and the seasonality from decompose are identical.

c. Finally, calculate the *remainder* component by removing both trend and seasonality from the raw series. Create a time-series plot of all components like the one below.
(*Hint*: you results should perfectly match those of the `decompose` function, which uses the above process)

```
ts_df = data.frame(matrix(detrended_X,nrow = 20,ncol = 12,byrow = T))
par(mfrow=c(1,2))
est.Xsa = Xua/(soif*y)
plot(est.Xsa)
plot(a$random)
```



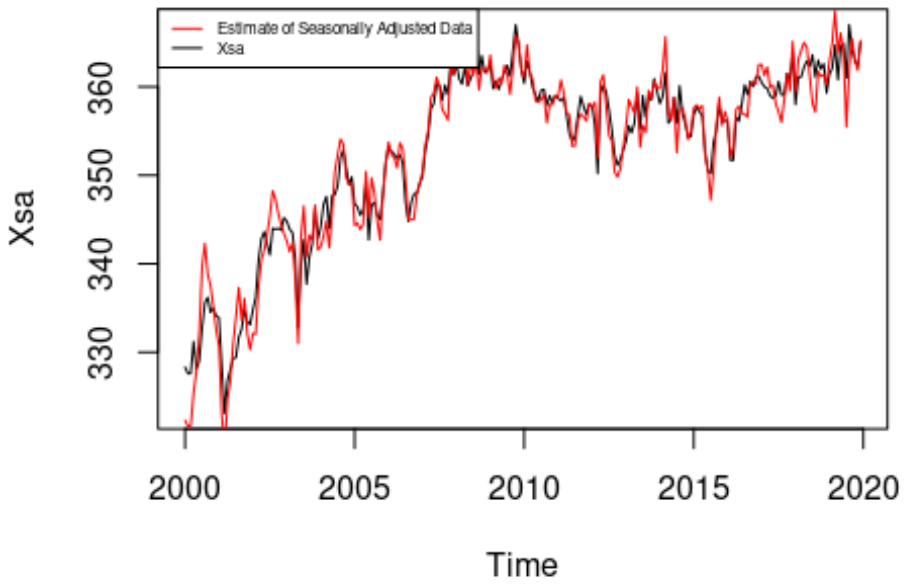

```
a=decompose(Xua,type = "multiplicative")
plot(est.Xsa)
lines(a$random,col="red")
legend("topleft",legend = c("Seasonally Adjusted Data from Decompose","Xsa"),col =
c("red","black"),lty = 1,cex = 0.5)
```



The plots show that the de-trended and centered(seasonally adjusted) series perfectly match.

3. [2 marks] Statistics Canada (StatCan) does their own seasonal adjustment using a more sophisticated method (namely, X-12-ARIMA). Download the corresponding *seasonally adjusted* series and plot together with your own seasonally adjusted data ($X_{sa} = X_{ua}/S = T \times R$) from the previous part. The two versions should be close, but not identical. Report the mean square error (MSE) between the two versions (StaCan's and yours) of seasonally adjusted data.

```
Xsa = get_cansim_vector( "v2063567", start_time = "2000-01-01", end_time = "2019-12-01")
%>%
  pull(VALUE) %>% ts( start = c(2000,1), frequency = 12)

## Accessing CANSIM NDM vectors from Statistics Canada

sa_ts = Xua/y
#par(mfrow=c(1,2))
plot(Xsa)
lines(sa_ts,col="red")
legend("topleft",legend = c("Estimate of Seasonally Adjusted Data","Xsa"),col =
c("red","black"),lty = 1,cex = 0.5)
```



The plot shows that the two versions should be close, but not identical.

```
(mse <- mean((Xsa - sa_ts)^2))

## [1] 4.829642
```

The MSE of my own seasonally adjusted data and StaCan Xsa is 4.829642.

4. [5 marks] The library `seasonal` contains R functions for performing seasonal adjustments/decompositions using various methods. Use the following two methods described in FPP for performing seasonal adjustments (you don't need to know their details):

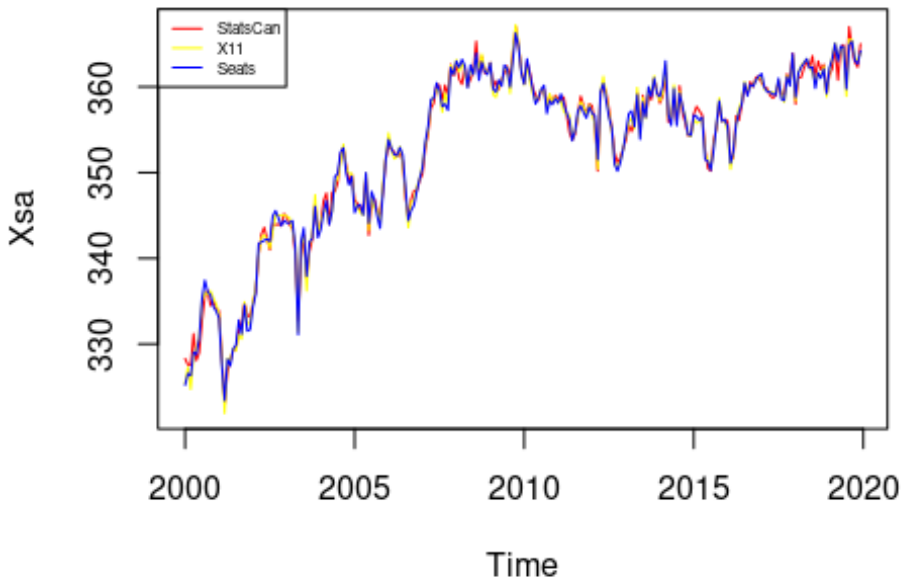
a. X11

```
library(seasonal)
Xua %>% seas(x11="")->x11.fit
```

b. SEATS

Create seasonaly adjusted versions of your raw series based on each method, and plot them together with StaCan's version. Note that the methods (X11 & SEATS) are *multiplicative* by default, and you must use the `forecast` library function `seasadj`, `seasonal`, `trendcycle`, and `remainder` to extract the various components. Which method gives a seasonal adjustment that is closest to StaCan's, based on MSE?

```
Xua %>% seas(seats="",seats.noadmiss="yes")->seats.fit

library(forecast)
plot(Xsa,type="l",col="red",ylim=range(c(seasadj(x11.fit),seasadj(seats.fit),Xsa),na.rm =
TRUE))
lines(seasadj(x11.fit),col="yellow")
lines(seasadj(seats.fit),col="blue")
legend("topleft",legend = c("StatsCan","X11","Seats"),col = c("red","yellow","blue"),lty
= 1,cex = 0.5)
```




```
(x11.mse = mean((seasadj(x11.fit)-Xsa)^2))

## [1] 0.7847131

(seats.mse = mean((seasadj(seats.fit)-Xsa)^2))

## [1] 0.870203
```

MSE of X11 Method has the smallest value, therefore, X11 Method seasonal adjustment is closest to StaCan's.

Appendix

#	SYMBOL	NAME
1	ABX-T	Barrick Gold Corp
2	AEM-T	Agnico Eagle Mines Ltd

#	SYMBOL	NAME
3	AQN-T	Algonquin Power and Utilities Corp
4	ATD-B-T	Alimentation Couche-Tard Inc Cl B Sv
5	BAM-A-T	Brookfield Asset Management Inc Cl A Lv
6	BCE-T	BCE Inc
7	BHC-T	Bausch Health Companies Inc
8	BIP-UN-T	Brookfield Infra Partners LP Units
9	BMO-T	Bank of Montreal
10	BNS-T	Bank of Nova Scotia
11	CAE-T	Cae Inc
12	CAR-UN-T	CDN Apartment Un
13	CCL-B-T	Ccl Industries Inc Cl B NV
14	CCO-T	Cameco Corp
15	CM-T	Canadian Imperial Bank of Commerce
16	CNQ-T	CDN Natural Res
17	CNR-T	Canadian National Railway Co.
18	CP-T	Canadian Pacific Railway Ltd
19	CSU-T	Constellation Software Inc
20	CTC-A-T	Canadian Tire Corp Cl A NV
21	CVE-T	Cenovus Energy Inc
22	DOL-T	Dollarama Inc
23	EMA-T	Emera Incorporated
24	ENB-T	Enbridge Inc
25	FM-T	First Quantum Minerals Ltd
26	FNV-T	Franco-Nevada Corp
27	FTS-T	Fortis Inc
28	GIB-A-T	CGI Group Inc Cl A Sv
29	GIL-T	Gildan Activewear Inc
30	IMO-T	Imperial Oil
31	IPL-T	Inter Pipeline Ltd
32	K-T	Kinross Gold Corp
33	KL-T	Kirkland Lake Gold Ltd
34	L-T	Loblaw CO
35	MFC-T	Manulife Fin
36	MG-T	Magna International Inc
37	MRU-T	Metro Inc
38	NA-T	National Bank of Canada
39	NTR-T	Nutrien Ltd
40	OTEX-T	Open Text Corp
41	POW-T	Power Corp of Canada Sv

#	SYMBOL	NAME
42	PPL-T	Pembina Pipeline Corp
43	QSR-T	Restaurant Brands International Inc
44	RCI-B-T	Rogers Communications Inc Cl B NV
45	RY-T	Royal Bank of Canada
46	SAP-T	Saputo Inc
47	SHOP-T	Shopify Inc
48	SJR-B-T	Shaw Communications Inc Cl B NV
49	SLF-T	Sun Life Financial Inc
50	SNC-T	Snc-Lavalin Sv
51	SU-T	Suncor Energy Inc
52	T-T	Telus Corp
53	TD-T	Toronto-Dominion Bank
54	TECK-B-T	Teck Resources Ltd Cl B
55	TRI-T	Thomson Reuters Corp
56	TRP-T	Tc Energy Corp
57	WCN-T	Waste Connections Inc
58	WEED-T	Canopy Growth Corp
59	WN-T	Weston George
60	WPM-T	Wheaton Precious Metals Corp