Lab 3

Time Series in R[1]

!!!Please read C4 and C5 before.

The ACF and the PACF

1. For the AR(2) process with the parameters Φ =(1,-.9), plot the ACF and the PACF:

```
ACF = ARMAacf(ar=c(1,-.9), ma=0, 30)

PACF = ARMAacf(ar=c(1,-.9), ma=0, 30, pacf=TRUE)

par(mfrow=c(1,2))

plot(ACF, type="h", xlab="lag", ylim=c(-.8,1)); abline(h=0,col=3)

plot(PACF, type="h", xlab="lag", ylim=c(-.8,1)); abline(h=0,col=3)


#equivalent code
w = rnorm(500,0,1)
v = filter(w, filter=c(1,-.9), method="recursive")
acf(v, lag.max=30, na.action = na.pass)
pacf(v, lag.max=30, na.action = na.pass)
#----------------------------------------------------------
```

2. Do the same for the MA(2) process with the parameters Θ=(0.5,-.4) and for the ARMA(2,2) process with the parameters Φ=(1,-.9) and Θ=(0.5,-.4).

3. Compare your results with the table below:

Behavior of the ACF and PACF for ARMA Models[1]

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

[1] Robert H. Shumway, David S. Stoffer. Time Series Analysis and Its Applications – with R examples, Springer 2017

Forecasting

1. Forecast 15 values of an AR(2) process, based on 100 generated observations:

```
x=arima.sim(list(order=c(2,0,0), ar=c(1,-.9)), n=100)

fore = predict(arima(x ,order=c(2,0,0)), n.ahead=15)

par(mfrow=c(1,2))

ts.plot(x, fore$pred, col=1:2, xlim=c(1,120), ylab="AR(2)")

lines(fore$pred, type="p", col=2)
```

   Use fore$se to plot the standard error.

2. Do the same for the MA(2) process with the parameters Θ=(0.5,-.4) and for the ARMA(2,2) process with the parameters Φ=(1,-.9) and Θ=(0.5,-.4).

Estimation

1. The Yule-Walker estimation for AR(p)

```
x=arima.sim(list(order=c(2,0,0), ar=c(1,-.9)), n=500)

x.yw = ar.yw(x, order=2)
x.yw$x.mean    # mean estimate mean(x)
x.yw$ar        # coefficient estimates
sqrt(diag(x.yw$asy.var.coef))        # standard errors of the coef. estimates

x.yw$var.pred        # error variance estimate
```

2. Maximum Likelihood Estimation (MLE)

```
x=arima.sim(list(order=c(2,0,0), ar=c(1,-.9)), n=500)

x.mle = ar.mle(x, order=2)
x.mle$x.mean
x.mle$ar
sqrt(diag(x.mle$asy.var.coef))
x.mle$var.pred
```

   Compare the results of 1 and 2. How are the estimations compared to the true values of the model Φ=(1,-.9)?

We can now use the estimated model to make predictions:

```
x.pr = predict(x.yw, n.ahead=15)
ts.plot(x, x.pr$pred, col=1:2)
```

3. Use arima function to fit an ARIMA model to a univariate time series -- for the MA(2) process with the parameters Θ =(0.5,-.4) and for the ARMA(2,2) process with the parameters Φ =(1,-.9) and Θ =(0.5,-.4).

```
x=arima.sim(list(order=c(0,0,2), ma=c(0.5,-.4)), n=100)
v=arima(x, order = c(0,0,2))

v$coef

v$sigma2
```

We can now use the estimated model to make predictions:

```
v.pr = predict(v, n.ahead=15)
ts.plot(x, v.pr$pred, col=1:2)


#similar for ARMA(2,2)
```

Non-stationarity

1. Random Walk with Drift

$X_t = \delta + X_{t-1} + W_t$

```
set.seed(154)             # so you can reproduce the results

w = rnorm(200)+0.2; x = cumsum(w)

par(mfrow=c(1,1))
plot.ts(x, ylim=c(-5,55), main="random walk", ylab='')

lines(x, col=4)

abline(a=0, b=.2, lty=2)


# differencing with diff to remove the trend

x1=diff(x)

lines(x1, col=3)
```

2. Estimate the trend and the seasonal components using *stl* function

```
# https://stackoverflow.com/questions/40307454/r-deseasonalizing-a-time-series

par(mfrow=c(2,1))
x<-AirPassengers
plot(x,main=(Air~Passengers~1949-1960))
y<-log(x)
plot(y,main=(expression(log(AirPassengers))))
#----------------------------------------------------

decomposed <- stl(y, s.window="periodic")
seasonal <- decomposed$time.series[,1]
trend   <- decomposed$time.series[,2]
remainder <- decomposed$time.series[,3]

par(mfrow=c(2,2))

plot(trend)
plot(seasonal)
plot(remainder)
plot(remainder+trend+seasonal)  # reconstructed data
```

Exercise

In C5 9^{th} slide, you find a suggestion to choose a preliminary value of *p* for an AR(*p*) model. It requires the implementation of the Durbin-Levinson (D-L) algorithm, where you will replace population moments $\gamma(h)$ by the sample moments.

Take an AR(*p*) model (let's say AR(2) with the parameters Φ =(1,-.9)) and generate 200 observations *x*.

Then use *x* to estimate the order of the model, as suggested in C5. Compare then the values of Φ_{mm} from the D-L algorithm with the PACF values returned by the pacf function in R.