

Week 1 exercises

Daniel Alonso

November 18th, 2020

Importing libraries

```
library(ggplot2)
library(foreach)
library(dplyr)
```

Exercise 1

Simulating 100 trajectories of length $n = 1000$ for X and Y .

```
Traj_X <- data.frame()
Traj_Y <- data.frame()
for (k in 1:100) {
  X <- data.frame(x=1:1000, val=rep(0,1000), run=rep(k,1000))
  Y <- data.frame(x=1:1000, val=rep(0,1000), run=rep(k,1000))
  X_r <- rep(0,1000)
  Y_r <- rep(0,1000)
  for (i in 1:1000) {
    if (i > 1) {
      X_r[i] <- 0.5*X_r[i-1]+rnorm(1)
      Y_r[i] <- 2*Y_r[i-1]+rnorm(1)
    }
  }
  X$val <- X_r
  Y$val <- Y_r
  Traj_X <- rbind(Traj_X, X)
  Traj_Y <- rbind(Traj_Y, Y)
}
```

a - Plotting simulated trajectories

Trajectories X

```
ggplot(data = Traj_X, aes(x=x, y=val)) +  
  geom_line(aes(colour=run), show.legend=FALSE)
```

Trajectories Y

The values of Y are too large for ggplot to show.

b - Use the simulated trajectories to estimate the mean and the covariance functions

```
mean_X = rep(0,100)  
mean_Y = rep(0,100)  
for (i in 1:100) {  
  x = Traj_X %>% filter(run == i) %>% select(val)  
  y = Traj_Y %>% filter(run == i) %>% select(val)  
  mean_X[i] = mean(x$val)  
  mean_Y[i] = mean(y$val)  
}
```

The mean of all trajectories of X is the following:

```
mean(mean_X)
```

```
## [1] 0.006358091
```

The mean of all trajectories of Y is the following:

```
mean(mean_Y)
```

```
## [1] 7.706139e+296
```

The mean of the covariances of all combinations of trajectories is the following:

```
covs = rep(0,100*100)  
cnt = 0  
for (i in 1:100) {  
  for (j in 1:100) {  
    x = Traj_X %>% filter(run == i) %>% select(val)  
    y = Traj_Y %>% filter(run == j) %>% select(val)  
    cnt = cnt + 1  
    covs[cnt] = cov(x,y)  
  }  
}  
mean(covs)
```

c - Is the process stationary? Is it weakly stationary?

X is stationary but Y is not

d - If the process is weakly stationary, use the function acf to display the autocorrelation function and compare with your own estimate

```
acf(X[1,])  
L=30  
acfX=numeric(length=L)
```

```
for (i in 1:L){
  acfx[i]=mean(diag(gX[1:(n+1-i),(i+1):(n+1)]))
}
plot(1:L, acfx)
```

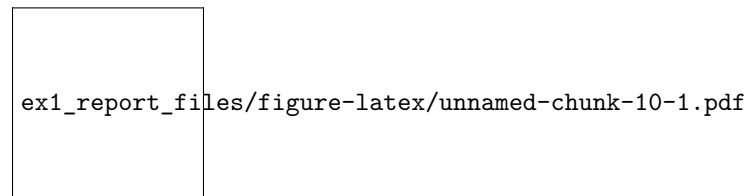
Exercise 2

$I_0 = 5$ and $S_0 = 1000$

$\alpha = 0.0005$

```
alpha <- 0.0001
cases <- c(5)
s <- 1000
i <- 5
while (i > 0) {
  p <- 1 - (1 - alpha)^(i)
  i <- rbinom(1, s, p)
  cases <- c(cases, i)
}
```

```
plot(cases)
```



```
P <- c(0.1, 0.4, 0.2, 0.3,
      0.6, 0, 0.2, 0.2,
      0.1, 0.3, 0.3, 0.3,
      0.5, 0.3, 0.1, 0.1)
P <- matrix(P,nrow=4,byrow=T)
alpha=rep(1/4,4) #uniform initial distribution
```

Exercise 4

Given the transition matrix and initial distribution:

```
P <- c(0.1, 0.4, 0.2, 0.3,
      0.6, 0, 0.2, 0.2,
      0.1, 0.3, 0.3, 0.3,
      0.5, 0.3, 0.1, 0.1)
P <- matrix(P,nrow=4,byrow=T)
alpha=rep(1/4,4) #uniform initial distribution
```

We define the following function to calculate matrix powers (thanks profe!):

```
matrixpower <- function(M,k) {
  # ARGUMENTS:
  # M: square matrix
  # k: exponent
  if(dim(M)[1]!=dim(M)[2]) return(print("Error: matrix M is not square"))
  if (k == 0) return(diag(dim(M)[1])) # if k=0 returns the identity matrix
```

```

    if (k == 1) return(M)
    if (k > 1) return(M %*% matrixpower(M, k-1)) # if k>1 recursively apply the function
}

```

a - $P(X_4 = 3 | X_3 = 4)$

```
P[4,3]
```

```
## [1] 0.1
```

b - $P(X_4 = 3 | X_2 = 4)$

```
matrixpower(P,2)[4,3]
```

```
## [1] 0.2
```

c - $P(X_1 = 1)$

```
(t(alpha)%*%P)[1]
```

```
## [1] 0.325
```

d - $E[X_2]$