Math 611 HW10

Jeff Gould

11/3/2020

```
SampleCasino <- function(bigT){

  X_t <- c("Fair", rep(NA, bigT))

  for (j in 2:(bigT+1)) {
    if(X_t[j-1] == "Fair"){
      X_t[j] = sample(c("Fair", "Cheat"), 1, prob = c(0.95, 0.05))
    }else{
      X_t[j] = sample(c("Fair", "Cheat"), 1, prob = c(0.05, 0.95))
    }

  }

  Y_t <- sapply(X_t, function(x){
    if(x == "Fair"){
      return(sample(1:6, 1))
    }else{
      return(sample(1:6, 1, prob = c(49/250, 49/250, 49/250, 49/250, 49/250, 1/50)))
    }
  })

  return(data.frame(t = c(0:bigT),
                    X = X_t,
                    Y = Y_t))

}

set.seed(1234)
testSample <- SampleCasino(200)
```

a)

See attached for math

```
Forward <- function(Y, theta){

  M <- theta[["M"]]
  G <- theta[["G"]]
  pi <- theta[["pi"]]

  alpha <- matrix(NA, nrow = length(Y), ncol = nrow(M))
```

```
  C  <- sum(pi * G[,Y[1]])

  alpha[1, ] = G[, Y[1] ] * t(pi) / C

  for (t in 2:nrow(testSample)) {
    c_n <- alpha[t-1, ] %*% M %*% G[,Y[t]]
    C <- c(C, c_n)
    alpha[t, ] <- G[,Y[t]] * (alpha[t-1, ] %*% M) / as.numeric(c_n)
  }

  return(cbind(alpha, C))

}

Backward = function(Y, theta, C){

  M <- theta[["M"]]
  G <- theta[["G"]]
  bigT = length(Y)
  m = nrow(M)
  beta = matrix(1, bigT, m)

  for(t in (bigT-1):1){
    tmp = as.matrix(beta[t+1, ] * G[, Y[t+1]])
    beta[t, ] = t(M %*% tmp) / C[t+1]
  }
  return(beta)
}

ForwardBackward <- function(Y, theta){

  alpha <- Forward(Y = Y, theta = theta)
  C <- alpha[,3]
  alpha = alpha[,1:2]
  beta <- Backward(Y = Y, theta = theta, C = C)

  return(alpha * beta)

}
```

```
update <- function(Y, Z){
  N <- length(Y)

  N1 <- sum(Z[-N] == 0) ## The number of times the hidden state is in state 1, less the last step
  N2 <- sum(Z[-N] == 1) ## The number of times the hidden state is in state 2, less the last step

  M11 <- 0
  M12 <- 0
  M22 <- 0
  M21 <- 0


  for (i in 1:(length(Z)-1)) { ## Calculate the number of times the hidden state is in state 1 and stay
```

```
    M11 <- M11 + (Z[i] == 0 & Z[i+1] == 0)
  }

  for (i in 1:(length(Z)-1)) { ## Calculate the number of times the hidden state is in state 1 and move
    M12 <- M12 + (Z[i] == 0 & Z[i+1] == 1)
  }

  for (i in 1:(length(Z)-1)) { ## Calculate the number of times the hidden state is in state 2 and stay
    M22 <- M22 + (Z[i] == 1 & Z[i+1] == 1)
  }

  for (i in 1:(length(Z)-1)) { ## Calculate the number of times the hidden state is in state 2 and move
    M21 <- M21 + (Z[i] == 1 & Z[i+1] == 0)
  }


  M11 <- M11 / N1 ## Divide all to get transition probabilites
  M12 <- M12 / N1
  M22 <- M22 / N2
  M21 <- M21 / N2

  M <- matrix(c(M11, M12,
                M21, M22), byrow = T, nrow = 2)

  Y0 <- Y[Z==0] ## Y for hidden state is fair
  Y1 <- Y[Z==1] ## Y for hidden state is cheating

  g0 <- sapply(1:6, function(x){mean(Y0==x)})  ## get probabilites for each state
  g1 <- sapply(1:6, function(x){mean(Y1==x)})

  #g0 <- (g0 + 0.02) / sum(g0 + 0.02)
  #g1 <- (g1 + 0.02) / sum(g1 + 0.02)

  # g0 <- g0  / sum(g0)
  # g1 <- (g1) / sum(g1)

  G <- matrix(c(g0, g1), nrow = 2, byrow = T)

  pi <- matrix(data = c(Z[1] == 0, Z[1] == 1), nrow = 2) %>% as.numeric()

  theta <- list(M = M,
                G = G,
                pi = pi)
  return(theta)

}



HardEM <- function(Y, starting_Z = NULL, max_iter = 100){
  iter <- 0

  if(is.null(starting_Z)){
```

```
    Z = sample(x = c(0,1), size = length(Y), replace = T)
  }else{
    Z <- starting_Z
  }

  while(iter < max_iter){

    theta <- update(Y = Y, Z = Z)

    Z <- ForwardBackward(Y = Y, theta = theta)
    Z <- round(Z)[,2]

    iter <- iter + 1
  }

  return(list(
    theta = theta,
    Z = Z
  ))



}

Y <- testSample$Y

set.seed(123)
init_Z <- as.numeric(testSample$X == "Cheat")
t <- seq(1,201,20)
init_Z[t] = (init_Z[t] + 1) %% 2

test <- HardEM(Y = Y, starting_Z = init_Z, max_iter = 50)
test$theta
```

```
## $M
##            [,1]       [,2]
## [1,] 0.92207792 0.07792208
## [2,] 0.04878049 0.95121951
##
## $G
##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.07792208 0.1558442 0.2077922 0.1168831 0.2207792 0.2207792
## [2,] 0.30645161 0.1774194 0.1774194 0.2096774 0.1290323 0.0000000
##
## $pi
## [1] 0 1
```

b)

See attached for math

Jeff Gould
Math 611
HW 10

1a) suppose we know $Z(t)$ for $t=0,1,\dots,T$

MLE: $\max_\theta$ ~~log f(z)~~

$$\max_\theta \log\left(\pi_{z(0)} M_{z(0)z(1)} M_{z(1)z(2)} \cdots M_{z(T-1)z(T)} g_{z(0)}(Y(0)) \cdots g_{z(T)}(Y(T))\right)$$

$$= \max_\theta \underbrace{\log(\pi_{z(0)}) + \sum_{t=0}^{T-1} \log M_{z(t)z(t+1)} + \sum_{t=0}^{T} \log g_{z(t)}(Y(t))}_{L(\theta)}$$

$\nabla L(\theta) = 0$, solve for θ

$$\frac{\partial}{\partial M_{ij}} L(\theta) = \sum_{\substack{t=0 \\ z(t)=i \\ z(t+1)=j}}^{T-1} \frac{1}{M_{ij}} = \frac{1}{M_{ij}} \sum_{\substack{t=0 \\ z(t)=i, \\ z(t+1)=j}}^{T-1} 1 = \frac{N_{ij}}{M_{ij}}$$

, note $\sum_{j=1}^{n} M_{ij} = 1$

$n=2$ in this example

$$G(M_{i1}, M_{i2}) = \sum_{j=1}^{2} M_{ij} = 1$$

$$\begin{pmatrix} \frac{\partial L}{\partial M_{i1}}(\theta) \\ \frac{\partial L}{\partial M_{i2}}(\theta) \end{pmatrix} = \lambda \nabla G = \lambda \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \frac{N_{ij}}{M_{ij}} = \lambda, \text{ for } j=1,2$$

$$M_{ij} = \frac{N_{ij}}{\lambda} \to 1 = \sum_{j=1}^{2} M_{ij} = \sum_{j=1}^{2} \frac{N_{ij}}{\lambda} = \frac{N_{i\bullet}}{\lambda} \quad) \Rightarrow \lambda = N_{i\bullet}$$

$$\Rightarrow M_{ij} = \frac{N_{ij}}{N_{i\bullet}} \blacksquare$$

g: constraint - if $g_j(i) = P(Y(t)=i \mid Z(t)=j)$, then $\sum_{i=1}^{6} g_j(i) = 1$

$$\frac{\partial L(\theta)}{\partial g_j(i)} = \sum_{\substack{t=0 \\ Z(t)=j \\ Y(t)=i}}^{T} \frac{1}{g_{z(t)}(Y(t))} = \frac{1}{g_j(i)} \sum_{t=0}^{T} I(Z(t)=j \text{ AND } Y(t)=i)$$

$$1 = \sum_{i=1}^{6} g_j(i) = \sum_{i=1}^{6} \frac{\sum_{t=0}^{T} I(Z(t)=j)}{\lambda}$$

$$\begin{pmatrix} \frac{\partial L(\theta)}{\partial g_j(1)} \\ \vdots \\ \frac{\partial L(\theta)}{\partial g_j(6)} \end{pmatrix} = \lambda \nabla G = \lambda \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \qquad \frac{\sum_{t=0}^{T} I(Z(t)=j \text{ AND } Y(t)=i)}{g_j(i)} = \lambda$$

$$\Rightarrow g_j(i) = \frac{\sum_{t=0}^{T} I(Z(t)=j \text{ AND } Y(t)=i)}{\sum_{t=0}^{T} I(Z(t)=j)} \blacksquare$$

$$\frac{\partial L(\theta)}{\partial \pi} = \frac{1}{\pi_{z(\theta)}}$$

$\pi = (\pi_1, \pi_2)$

$\max \log \pi_{z(0)}$ → clearly set $\pi_i = \begin{cases} 1 & \text{if } Z(0)=i \\ 0 & \text{else} \end{cases}$

1a) continued

$P(Z(t)=i \mid Y(0), \dots Y(\text{seq}))$

Let $\alpha_t(i) = P(Y(0), \dots, Y(t), Z(t)=i)$

$\beta_t(i) = P(Y(t+1), Y(t+2), \dots, Y(T) \mid Z(t)=i)$

$= \dfrac{P(Z(t)=i, Y(0), \dots, Y(T))}{P(Y(0), \dots, Y(T))}$ ← D

$= \dfrac{P(Y(0), Y(1), \dots, Y(t), Z(t)=i) \cdot P(Y(t+1), \dots, Y(T) \mid Z(t)=i)}{D}$

$= \dfrac{\alpha_t(i)\beta_t(i)}{D}$ ⇒ Forward-Backwards Algorithm

---

1b)

Let θ' be the current parameters

$Q(\theta', \theta) = E_{\theta'}[\log P(Z(0), \dots, Z(T), Y(0), \dots, Y(T) \mid \theta)]$

(θ' under the Z's, θ under the Y's)

$\max_\theta Q(\theta', \theta) = \max_\theta E[\log P(\quad)]$

$= \sum_{z(0)=1}^{2} \sum_{z(1)=1}^{2} \sum_{z(2)=1}^{2} \cdots \sum_{z(T)=1}^{2} [\log(\pi_{z(0)}) + \sum_{t=0}^{T-1} \log M_{z(t)z(t+1)} + \sum_{t=0}^{T} \log g_{z(t)}(Y(t))] \cdot P(Z(0), Z(1), \dots Z(T) \mid Y(0), Y(1), \dots, Y(T), \theta')$

Set $\nabla_\theta Q(\theta', \theta) = 0$

$\dfrac{\partial}{\partial M_{ij}} Q(\theta', \theta) = \sum_{z(0)=1}^{2} \sum_{z(1)=1}^{2} \cdots \sum_{z(T)=1}^{2} \sum_{t=0}^{T-1} [\dfrac{1}{M_{ij}} I(Z(t)=i, Z(t+1)=j) \cdot P(Z(0), Z(1), \dots Z(T) \mid Y(0), \dots, Y(T), \theta')]$

$= \sum_{t=0}^{T-1} \sum_{z(t)=1}^{2} \sum_{z(t+1)=1}^{2} \cdots \sum [\dfrac{1}{M_{ij}} I(Z(t)=i, Z(t+1)=j) \cdot P(\quad)]$

$= \sum_{t=0}^{T-1} \sum_{z(t)=1}^{2} \sum_{z(t+1)=1}^{2} \dfrac{1}{M_{ij}} I(Z(t)=i, Z(t+1)=j) P(Z(t)=i, Z(t+1)=j \mid Y(0), \dots, Y(T), \theta')$

$= \sum_{t=0}^{T-1} \sum_{z(t)=1}^{2} \dfrac{1}{M_{ij}} P(Z(t)=i, Z(t+1)=j \mid Y(0), \dots, Y(T), \theta')$

$$\cdot p(z(t)=i, z(t+1)=j \mid Y(0),\dots,Y(T),\theta') \quad g_j(Y(t+1))$$

$$= \underbrace{p(Y(0),\dots,Y(t), z(t)=i)}_{\alpha_t(i)} \underbrace{p(Y(t+2),\dots Y(T)\mid z(t+1)=j)}_{\beta_{t+1}(j)} p(Y(t+1)\mid z(t+1)=j) \cdot \underbrace{p(z(t+1)=j \mid z(t)=i)}_{M'_{ij}}$$

$$= \alpha_t(i)\, g_j(Y(t+1))\, \beta_{t+1}(j)\, M'_{ij} = \psi_t(i,j)$$

$$\Rightarrow \frac{\partial Q(\theta',\theta)}{\partial M_{ij}} - \sum_{t=0}^{T-1} \frac{1}{M_{ij}} I(z(t)=i, z(t+1)=j)\, \psi_t(i,j)$$

following the same steps from 1a)

$$\Rightarrow M_{ij} = \frac{\sum_{t=0}^{T-1} \psi_t(i,j)}{\sum_{t=0}^{T-1}\sum_{k=1}^{2} \psi_t(i,k)}$$

define $\gamma_t(j) = p(z(t)=j \mid Y(0),\dots Y(T),\theta')$

$$\frac{\partial}{\partial g_j(i)} Q(\theta',\theta) = \sum_{\substack{t=0 \\ Y(t)=i}}^{T} \left(\frac{1}{g_{z(t)}(Y(t))}\right) \cdot p(z(t)=j \mid Y(0),\dots,Y(T),\theta'))$$

$$= \sum I(Y(t)=i)\, \gamma_t(j)$$

$$\Rightarrow \frac{\sum_{t=0}^{T} I(Y(t)=i)\, \gamma_t(j)}{\sum_{t=0}^{T} \gamma_t(j)}$$

$\Rightarrow$ constraint $\sum_{j=1}^{6} \gamma_t(j) = 1$

- Lagrange as before

$$\frac{\partial}{\partial \pi_i} Q(\theta',\theta) = 0$$

$$\pi_i = \frac{p(z(0)=i \mid Y(0),\dots,Y(T),\theta')}{\sum_{j=1}^{6} p(z(0)=j \mid Y(0),\dots,Y(T),\theta') = 1} \Big\} = p(z(0)=i \mid Y(0),\dots,Y(T),\theta')$$

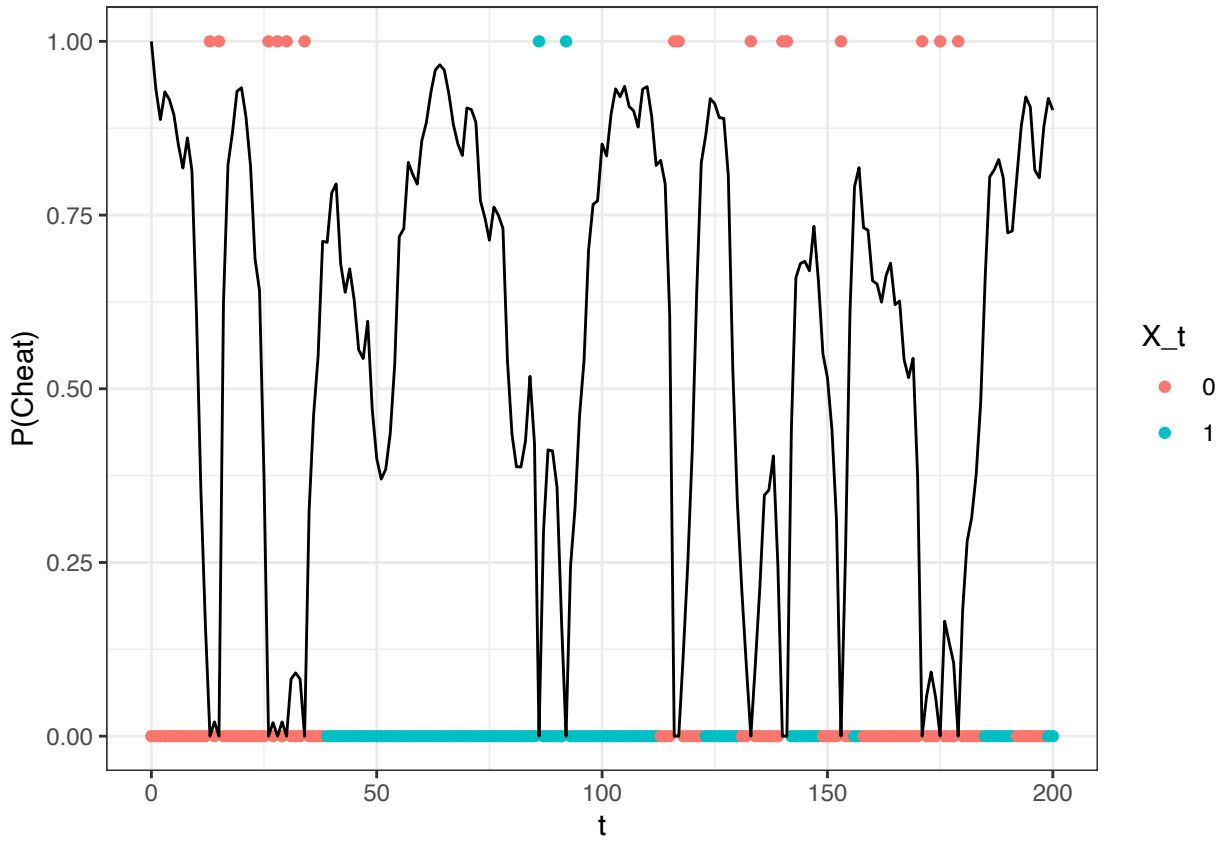
```r
### Initialize theta parameters

M <- matrix(c(0.85, 0.15,
              0.15, 0.85), nrow = 2, byrow = T)
pi <- matrix(c(0.8, 0.2), nrow = 2)
set.seed(1)
g1 <- runif(6)
g2 <- runif(6)

G <- matrix(c((g1 + 1)/sum(g1 + 1), (g2 + 0.05) / sum(g2 + 0.05)), nrow = 2, byrow = T)

init_theta <- list(
  M = M,
  pi = pi,
  G = G
)


updateSoft <- function(Y, Z, theta){

  M_prime <- theta$M
  G_prime <- theta$G
  pi_prime <- theta$pi

  N <- length(Y) - 1

  alpha <- Forward(Y, theta) ## get the forward/backward probabilities
  beta <- Backward(Y, theta, alpha[,3])
  alpha <- alpha[,1:2]

  psi11 <- rep(0, N)

  for (t in 1:N) { ## create psi vector for in state 1 and stays in state 1
    psi11[t] <- as.numeric(alpha[t, 1] * beta[t+1, 1] * M[1,1] * G_prime[1 ,Y[t+1]])
  }

  psi12 <- rep(0, N)

  for (t in 1:N) { ## create psi vector for in state 1 and moves to state 2
    psi12[t] <- as.numeric(alpha[t, 1] * beta[t+1, 1] * M[1,2] * G_prime[2,Y[t+1]])
  }

  M11 <- sum(psi11) / sum(psi11, psi12)  ## normalize
  M12 <- sum(psi12) / sum(psi11, psi12)


  psi22 <- rep(0, N)

  for (t in 1:N) { ## create psi vector for in state 2 and stays in state 2
    psi22[t] <- as.numeric(alpha[t, 2] * beta[t+1, 2] * M[2,2] *G_prime[2,Y[t+1]])
  }

  psi21 <- rep(0, N)
```

```
  for (t in 1:N) {  ## create psi vector for in state 2 and moves to state 1
    psi21[t] <- as.numeric(alpha[t, 2] * beta[t+1, 1] * M[2,1] *  G_prime[1,Y[t+1]])
  }

  M21 <- sum(psi21) / sum(psi21, psi22)
  M22 <- sum(psi22) / sum(psi21, psi22)

  M_update <- matrix(c(M11, M12,
                       M21, M22), byrow = T, nrow = 2)

  g1 <- sapply(1:6, function(x){
    sum(Z[,1] * (Y == x)) / (sum(Z[,1]))
  })

  g2 <- sapply(1:6, function(x){
    sum(Z[,2] * (Y == x)) / (sum(Z[,2]))
  })

  G_update <- matrix(c(g1, g2), byrow = T, nrow = 2)

  pi_update <- matrix(Z[1,], nrow = 2)

  theta_new <- list(M = M_update,
                    G = G_update,
                    pi = pi_update)

  return(theta_new)

}




softEM <- function(Y, start_theta, max_iter){

  iter <- 0
  theta <- start_theta
  while(iter < max_iter){
    Z <- ForwardBackward(Y = Y, theta = theta)

    theta <- updateSoft(Y, Z, theta = theta)

    iter <- iter + 1

  }

  return(list(Z = Z,
              theta = theta))



}
```

```r
testB <- softEM(Y = testSample$Y, start_theta = init_theta, max_iter = 100)

testB$theta
```

```
## $M
##           [,1]      [,2]
## [1,] 0.8369945 0.1630055
## [2,] 0.1264051 0.8735949
##
## $G
##           [,1]      [,2]      [,3]      [,4]      [,5]         [,6]
## [1,] 0.1091667 0.2023434 0.1686101 0.1224041 0.2014999 1.959759e-01
## [2,] 0.3022223 0.1439561 0.2045769 0.2134007 0.1358441 5.152403e-09
##
## $pi
##              [,1]
## [1,] 2.804385e-18
## [2,] 1.000000e+00
```

```r
results <- data.frame(p_cheating = testB$Z[,2],
                      Y_t = testSample$Y,
                      X_t = as.numeric(testSample$X == "Cheat"),
                      t = seq(0, 200))

results %>%
  mutate(is_6 = ifelse(Y_t == 6, 1, 0),
         X_t = as.factor(X_t)) %>%
  ggplot(aes(x = t)) +
  geom_point(aes(y = is_6, color = X_t)) +
  theme_bw() +
  geom_line(aes(y =  p_cheating)) +
  labs( y = "P(Cheat)")
```

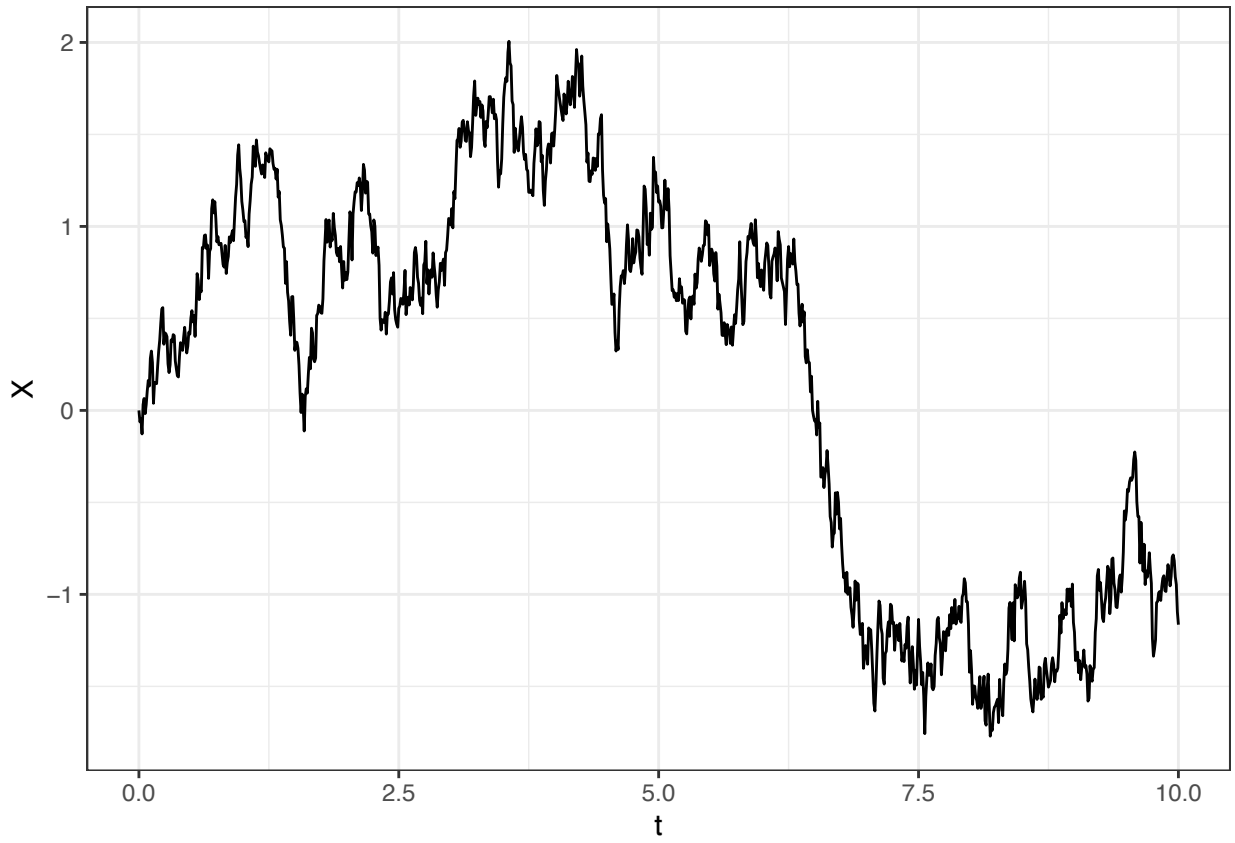


2

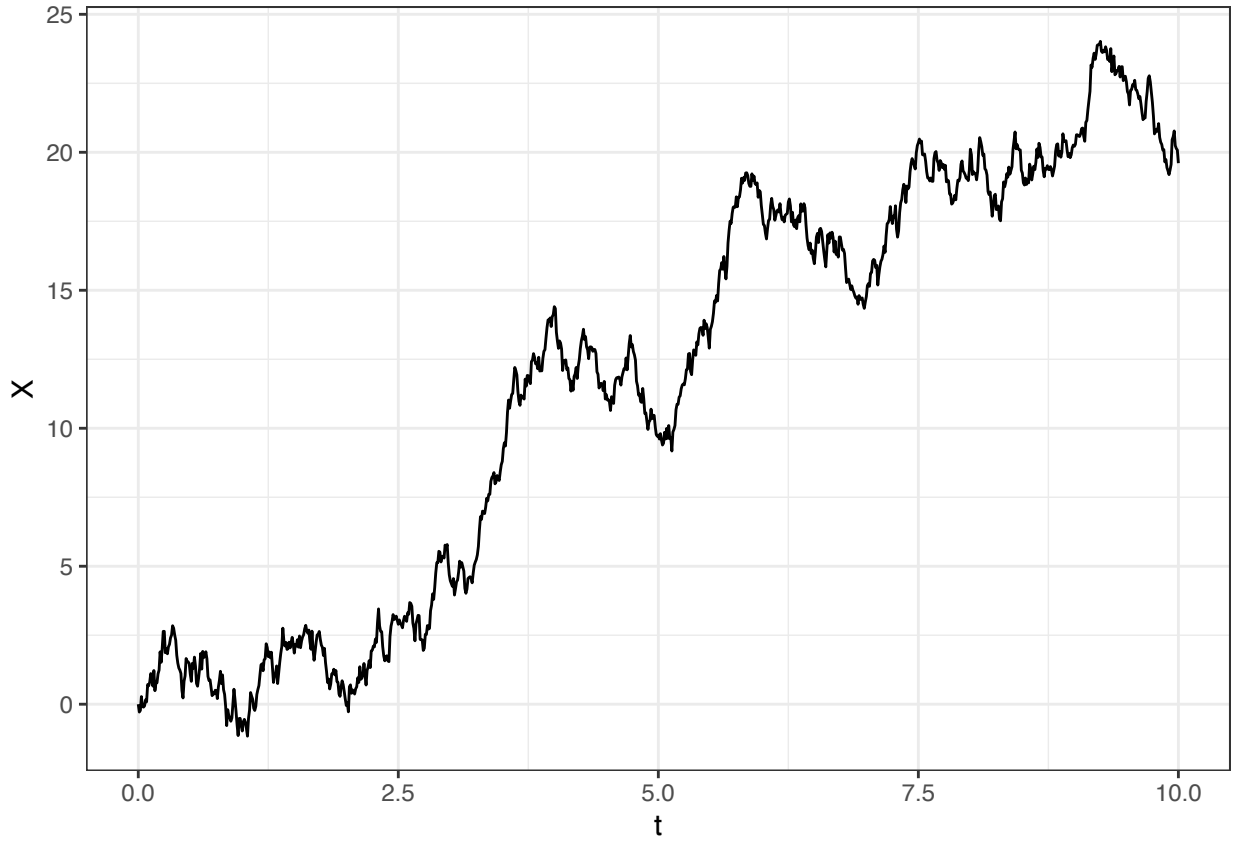
a) To get a feel for the sample paths of Brownian motion, generate a sample of $B(t)$ for i) $\sigma^2 = 1$ and ii) $\sigma^2 = 10$ up to time $t = 10$ using a grid of width $.01$.

```
BrownianMotion <- function(t, delta_t, sigma2){

  intervals <- t / delta_t

  X <- c(0, rnorm(n = intervals, sd = sqrt(sigma2 * delta_t)))

  X <- data.frame(X = cumsum(X),
                  t = seq(0, t, delta_t))

  return(X)



}

set.seed(1)
BM1 <- BrownianMotion(t = 10, delta_t = 0.01, sigma2 = 1)

ggplot(BM1, aes(x = t, y = X)) +
```

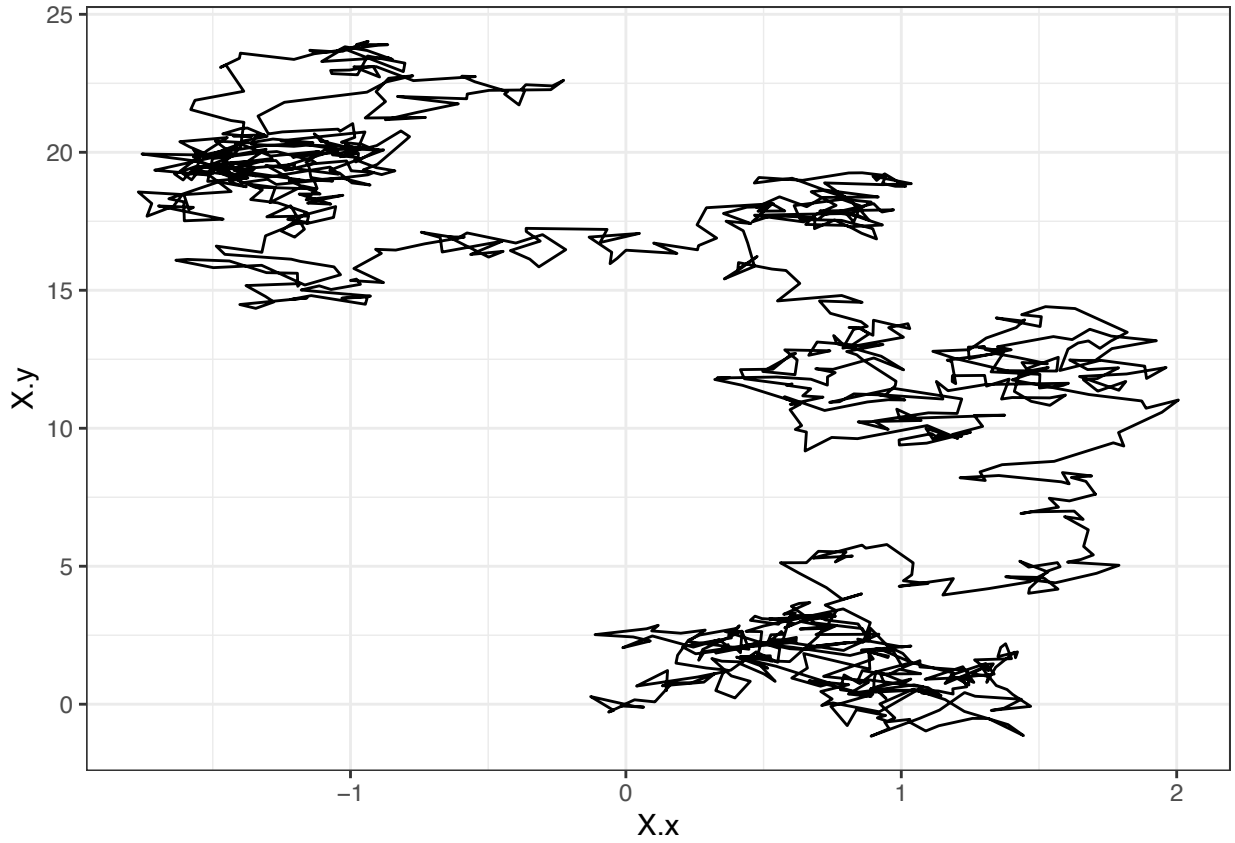
```
  geom_line() +
  theme_bw()
```




```
set.seed(2)
BM2 <- BrownianMotion(t = 10, delta_t = 0.01, sigma2 = 10)


ggplot(BM2, aes(x = t, y = X)) +
  geom_line() +
  theme_bw()
```




```
left_join(BM1, BM2, by = c("t" = "t")) %>%
  ggplot(aes(x = X.x, y = X.y)) +
  geom_path() +
  theme_bw()
```



b) Let $t_1 < t_2 < t_3$.

- Calculate $E[B(t_1)B(t_2)]$.

$B(t_2) = B(t_1) + \mathcal{N}(0, (t_2 - t_1)\sigma^2)$

$E[B(t_1)B(t_2)] = E[B(t_1)(B(t_1)+\mathcal{N}(0, (t_2-t_1)\sigma^2))] = E[B(t_1)B(t_1)]+E[B(t_1)\mathcal{N}(0, (t_2-t_1)\sigma^2))] = E[B(t_1)^2]$

$E[B(t_1)^2] = Var[B(t_1)^2] + E[B(t_1)]^2 = t_1\sigma^2$

- Calculate $E[B(t_1)B(t_2)B(t_3)]$.

$B(t_2) = B(t_1) + [B(t_2) - B(t_1)]$

$B(t_3) = B(t_2) + [B(t_3) - B(t_2)] = B(t_1) + [B(t_2) - B(t_1)] + [B(t_3) - B(t_2)]$

$E[B(t_1)B(t_2)B(t_3)] = E\left[B(t_1)[B(t_1) + [B(t_2) - B(t_1)]][B(t_1) + [B(t_2) - B(t_1)] + [B(t_3) - B(t_2)]]\right]$

Let $x_1 = B(t_1)$, $x_2 = B(t_2) - B(t_1)$, and $x_3 = B(t_3) - B(t_2)$

Then $E[x_1] = E[x_2] = E[x_3] = 0$, and all are independent

$E[B(t_1)B(t_2)B(t_3)] = E\left[B(t_1)[B(t_1) + [B(t_2) - B(t_1)]][B(t_1) + [B(t_2) - B(t_1)] + [B(t_3) - B(t_2)]]\right] =$

$E[x_1(x_1 + x_2)(x_1 + x_2 + x_3)] = E[x_1^3 + 2x_1^2x_2 + x_1x_2^2 + x_1^3x_3 + x_1x_2x_3] = E[x_1^3] + E[2x_1^2x_2] + E[x_1x_2^2] + E[x_1^3x_3] + E[x_1x_2x_3] = E[x_1^3] = E[B(t_1)^3] = 0$

Where $E[B(t_1)^3] = 0$ follows from the third moment of a normal r.v. with mean 0 is 0.