Random Processes Analysis with R
Solution of the exercises

M. Bittelli, R. Olmi, R. Rosa

March 29, 2022

Contents

1 Exercises of Chapter 2: Historical Background

Exercise 2.1

We have seen in Chapter 2 with the `Code 2.1` that in 18000 games with 24 throws the estimated probability of the double six is $\widehat{\mathsf{P}}\{A\}^{[24]} \approx 0.4887$, while with 25 throws it is $\widehat{\mathsf{P}}\{A\}^{[25]} = 0.5113$. Result practically equal to the theoretical one. The difference $\hat{d} = 0.5113 - 0.4887 = 0.0226$ resulted significant.

Write a code to obtain Fig. 2.2, showing if the difference of the means is significant with the bootstrap method. Before you have to read *Appendix A* if you are not familiar with the bootstrap method.

Solution

The first part of the code below is a copy of `Code 2.1 Throw of two dice`, with added prints of some quantities for check.

```
## bootstrap difference of means

################  25 throws
p_25<- 0.5055  # probability of getting two sixes in 25 throws
n.nights<- 180           # number of nights
n.games<-  100           # number of games
n.throws<- 25            # number of throws
spot<- c(1:6)            # spots of a 6-sided die (1, ..., 6)
p_fair<- rep(1/6,6)      # probabilities of a "fair" die
d6<- numeric()
d6T<- numeric()
nseed<- 100
for (j in 1:n.nights)                    {       # loop on the nights
nseed<- nseed+1
set.seed(nseed)
for(l in 1:n.games)      {                   # loop on the games
d6[l]<- 0
for(i in 1:n.throws)   {                        # loop on the throws
die.1<- sample(spot,1,p_fair,replace=T)  # i-th throw with the die 1
die.2<- sample(spot,1,p_fair,replace=T)  # i-th throw with the die 2
s.points<- die.1+die.2
if(s.points == 12) d6[l] <- 1
                     }      # end loop on the throws
                          }          # end loop on the games
d6T[j]<- sum(d6)/n.games
                                        } # end loop on the nights
md6T<-mean(d6T)
md6T
s<- sqrt(md6T*(1-md6T)/n.games)
s
s_theor<- sqrt(p_25*(1-p_25)/n.games)
s_theor

################   24 throws
p_24<- 0.4914  # probability of getting two sixes in 24 throws
n.nights<- 180           # number of nights
n.games<-  100           # number of games
n.throws<- 24            # number of throws
spot<- c(1:6)            # spots of a 6-sided die
p_fair<- rep(1/6,6)      # probabilities of a "fair" die
d6<- numeric()
d6T1<- numeric()
nseed<- 301
for (j in 1:n.nights)                    {       # loop on the nights
nseed<- nseed+1
set.seed(nseed)
for(l in 1:n.games)      {                   # loop on the games
d6[l]<- 0
for(i in 1:n.throws)   {                        # loop on the throws
die.1<- sample(spot,1,p_fair,replace=T)  # i-th throw with the die 1
die.2<- sample(spot,1,p_fair,replace=T)  # i-th throw with the die 2
s.points<- die.1+die.2
if(s.points == 12) d6[l] <- 1
                     }      # end loop on the throws
                          }          # end loop on the games
d6T1[j]<- sum(d6)/n.games
                                        } # end loop on the nights
md6T1<-mean(d6T1)
md6T1
s<- sqrt(md6T1*(1-md6T1)/n.games)
s
s_theor<- sqrt(p_24*(1-p_24)/n.games)
s_theor

###### up to here the code is  Code 2.1 Throw of two dice ######

# for convenience we call:
z<- d6T
y<- d6T1
B<- 1000   # number of bootstrap replications
diffb<- numeric()
for(b in 1:B){           # loop on replications
zb<-sample(z, length(z), replace=T)  # z-th bootstrap replication
zbm<-mean(zb)
yb<-sample(y, length(y), replace=T)  # y-th bootstrap replication
ybm<-mean(yb)
diffb[b]<- zbm-ybm
              }         # end loop on replications
mdiffb<- mean(diffb)
mdiffb
seb<- sd(diffb)         # bootstrap standard error
seb
min0<- sum(diffb<=0)    # are there differences < 0?
min0
perc<- min0/B
perc   #  how many differences are < 0 in percent?
# Compute the observed difference d_obs between the two means
mz<- mean(z)
mz
my<- mean(y)
my
d_obs<- mz-my
d_obs
ymax<- 400
par(lwd=3)
hist(diffb, freq=T,border="black",main="",
xlab=expression(hat(italic("d"))*"*"),ylab="counts",
ylim=c(0,ymax),cex.lab=1.3,font.lab=3)
segments(d_obs,0,d_obs,ymax,col="black",lty=3,lwd=2) # marks d_obs
```

We repeat once again that the difference $\hat{d} = 0.0226$ is significant, since out of $B = 1000$ bootstrap replications $\hat{d}^*$ (`diffb`), none of them is less than 0.

Exercise 2.2

In the dice game Unders and Overs (U&O), two dice are rolled. Players bet on one of the following alternatives:
(1) The result (sum of the dice faces) is below 7
(2) The result is 7
(3) The result is above 7

In cases (1) and (3) the pay off odds are 1:1, i.e. if you bet 1 £ the house give you back your money plus an additional 1 £. In case (2) the odds are 4:1, i.e. betting 1 £ you gain 4 £ (you get 5 £).

Suppose you bet 1 £ on the outcome (1). Which is your expected average win/loss (i.e. in an infinite number of throws)?

Solution

The number of possible outcomes is 36. The number of outcomes satisfying the conditions (1), (2) and (3) are 15, 6 and 15 respectively, as you can verify simply by writing down all possible results:

```
Case (1)
11 12 13 14 15
21 22 23 24
31 32 33 34
41 42
51

Case (2)
16 25 34 43 52 61

Case (3)
26 62
35 36 52 63
44 45 46 64 54
55 65 56
66
```

As you bet on (1), the possible money outcomes M are $M = 0$, if (2) o (3) happens, or $M = 2$ if (1) happens. The probability of the two events are readily computed:

$$\mathsf{P}\{M = 0\} = \frac{21}{36}$$

$$\mathsf{P}\{M = 2\} = \frac{15}{36}$$

Therefore, the expected money outcome is:

$$\mathrm{E}[M] = 2 \times \frac{15}{36} + 0 \times \frac{21}{36} = \frac{5}{6}$$

As you bet 1 £, on the average you get $\frac{5}{6} - 1 = -\frac{1}{6}$ or, in other words, your expected loss is $\frac{1}{6}$ £.

Exercise 2.3

Referring to the previous exercise, write a code to simulate a finite game consisting in 10, 100 or 1000 throws. Discuss the result of the simulations, compared to the theoretical win/loss expectation.
Hint: Use the R function `sample` for sampling a dice face, i.e. an integer number from 1:6

Solution

A possible approach is that of defining an R function *win* which gives a positive or negative return value respectively in case of win or loss:

```
win <- function(x)
{
  if (x<7)
    {ret=2}
  else
    {ret=0}
  return(ret)
}
```

Such a function can be usefully *vectorized* by the R function `Vectorize` (see `help(Vectorize)`) such to compute a vector of the money outcomes associated with a sequence of dice throws, as the following code does.

```
# uncomment one of the following for 10, 100 or 1000 throws
# N <- 10
# N <- 100
# N <- 1000
n1 = 1:6
n2 = 1:6
first = sample(n1,size=N, replace=TRUE)
second = sample(n2,size=N, replace=TRUE)
result = first+second
v_result <- v_win(result)
sum(v_result)/N-1
```

Exercise 2.4

A variant of the U&O game allows the player to bet up to two alternatives (placing 1 £ over each one). How does the win/loss expectation change?

Solution

Suppose you bet on “result is different from 7”. You obtain two possible payoff: $M = 2$, if true, $M = 0$ if the outcome is exactly 7.

$$\mathsf{P}\{M = 0\} = \frac{6}{36}$$
$$\mathsf{P}\{M = 2\} = \frac{30}{36}$$

Therefore $\mathrm{E}[M] = 2 \times \frac{30}{36} = \frac{5}{3}$, and the expect money is $\frac{5}{3} - 2 = -\frac{1}{3}$, because you bet 1 £ on “result less than 7” and 1 £ on “result greater than 7”.

If you bet on “result less or equal to 7” the expected money is still a loss of $\frac{1}{3}$ £ because:

$$\left\{\begin{array}{lcl} \mathsf{P}\{M = 0\} & = & 15/36 \\ \mathsf{P}\{M = 2\} & = & 15/36 \\ \mathsf{P}\{M = 5\} & = & 6/36 \end{array}\right\} \mathrm{E}[M] = 2 \times \frac{15}{36} + 5 \times \frac{6}{36} = \frac{5}{3}$$

Exercise 2.5

Justify the following assertion: the house always wins.
Hint: If you can bet 1 £ on each of the three alternatives, what is the expected outcome?

Solution

Also if you bet 1 £ on all possible outcomes, for a total of 3 £, the expected outcome is:

$$\mathrm{E}[M] = 2 \times \frac{30}{36} + 5 \times \frac{6}{36} = \frac{5}{2}$$

thus the expected money is $\frac{5}{2} - 3 = -\frac{1}{2}$ £.

Exercise 2.6

With reference to exercise 2.5, compare the theoretical result for an infinite number of throws with those obtained in a small number of them, e.g. 10. Simulating the problem in R, in 100 repetition how many times you win and how many you loose your money?

Solution

You can proceed in analogy with exercise 2.3, modifying the function `win`:

```
win <- function(x)
{
  if (x==7)
    {ret=5}
  else
    {ret=2}
  return(ret)
}
```

and the main program:

```
# uncomment one of the following
# N <- 10
# N <- 100
n1 = 1:6
n2 = 1:6
first = sample(n1,size=N, replace=TRUE)
second = sample(n2,size=N, replace=TRUE)
result = first+second
v_result <- v_win(result)
sum(v_result)/N - 3
```

---

2 Exercises of Chapter 3: Introduction to Stochastic Processes

Exercise 3.1

Consider a simple game. In a track like the following.

1	2	3

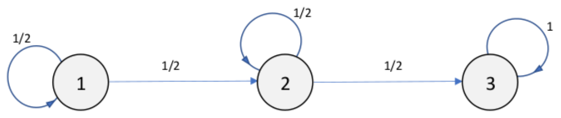
You start on position 1, the goal is to reach position 3. You flip a coin. If you get heads, stay where you are. If you get tails, move forward in the next position. Each player counts the number of coin flips. The winner is who reaches 3 in the lowest number of moves.

Write the transition probability matrix and the state transition diagram for this game.

Solution

The state transition diagram and the transition probability matrix are the following.

$$\begin{pmatrix} 1/2 & 1/2 & 0 \\ 0 & 1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix}$$



Exercise 3.2

Write an R code to simulate the game of exercise 3.1.

Solution

You can modify `Code_3_1.R` as follows:

```
#Function to compute markov chain
markov<- function(x0,n,x,P) {
  s <- numeric()
  s[1] <- x0
  row <- which(x==x0)
  for(i in 2:n) {
    s[i] <- sample(x,1,P[row,],replace=T)
    row <- which(x==s[i])
  }
  return(s)
}

# The game
# max number of moves
n <- 20
# transition matrix
P <- matrix(c(1/2,1/2,0,0,1/2,1/2,0,0,1),nrow=3,ncol=3,byrow=T)

# Two players: x,y
x <- c(1,2,3)
y <- c(1,2,3)
# starting state
x0 <- 1
y0 <- 1

sx <- markov(x0,n,x,P)
sy <- markov(x0,n,x,P)
tmp <- which(sx==3)
x.result <- tmp[1]
tmp <- which(sy==3)
y.result <- tmp[1]

if (x.result<y.result) {
print("X wins")
} else if (x.result>y.result) {
print("Y wins")
} else
print("Tie")
```

Exercise 3.3

Consider a simplified version of the Game of The Goose, a very popular old game attributed to Francesco de Medici (1574−1587). The simplified version consists in a path having only 10 positions (instead of 64), starting from 0 and ending to 9, with a single player and one dye only (instead of tow dice).

The path is like the following:

0	1	2	3	**4**	5	*6*	7	8	9

The rules of the game are the following:

- The player starts on position 0. He rolls a dye and moves forward the number of steps shown by the dye face.
- Position *6* is a goose. If the player lands on it he moves forward the same number of positions. For example, if he was in 5 and the dye shows 1, he goes in 6 and repeats the move, i.e. he goes in position 7.
- Position **4** is the death. The player falling there comes back to position 0.
- The goal is to reach position 9, exactly. If the number of moves is such to exceed that position, the surplus is counted backwards from position 9.

Represent the problem in terms of its transition matrix. In particular, while in general the transition probabilities are 1/6 (due to the dye), rows 4, 6 and 9 of the matrix have peculiar characteristics. Which ones?

Solution

The transition probability matrix is the following.

	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
(0)	0	1/6	1/6	1/6	1/6	1/6	1/6	0	0	0
(1)	0	0	1/6	1/6	1/6	1/6	1/6	1/6	0	0
(2)	0	0	0	1/6	1/6	1/6	1/6	1/6	1/6	0
(3)	0	0	0	0	1/6	1/6	1/6	1/6	1/6	1/6
(4)	1	0	0	0	0	0	0	0	0	0
(5)	1/6	1/6	0	0	0	0	1/6	1/6	1/6	1/6
(6)	1/5	0	0	0	0	1/5	0	1/5	1/5	1/5
(7)	1/6	1/6	1/6	1/6	0	0	0	0	1/6	1/6
(8)	1/6	1/6	1/6	1/6	1/6	0	0	0	0	1/6
(9)	0	0	0	0	0	0	0	0	0	1

Note that from the game rules you can arrive in position 6 only from positions 0, 1, 2, 3 and 5, which is the reason for the 1/5 values in the corresponding matrix row.

Exercise 3.4

Justify the following assertion: an independent and identically distributed sequence of random variables is a Markov chain.
Hint: write the transition matrix

Solution

If the i.i.d. sequence $\{X_n\}$ assumes values in $\{1..k\}$ with probabilities $\mathsf{P}\{X_n = i\} = p_i$, for $i = 1..k$, we have:

$$\sum_{i=1}^{k} p_i = 1$$

and from independence:

$$\mathsf{P}\{X_n = i | X_{n-1} = j\} = \mathsf{P}\{X_n = i\} = p_i$$

Therefore the transition probability matrix is:

$$\begin{pmatrix} p_1 & p_2 & \dots & p_k \\ p_1 & p_2 & \dots & p_k \\ \dots & & & \\ p_1 & p_2 & \dots & p_k \end{pmatrix}$$

In words, the next state in the chain is independent of the previous one. That's a *trivial* example of Markov chain.

Exercise 3.5

Assume the following transition probability matrix for a Markov process describing weather, with possible states:

```
1 = clear sky
2 = cloud covered sky
3 = rain
```

Let P be:

$$\mathbf{P} = \begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.4 & 0.4 \\ 0.1 & 0.6 & 0.3 \end{pmatrix}$$

A meteorologist predicts for tomorrow, Friday, 40 % rain probability and 60 % clear sky. What is the probability of rain on Saturday?
Hint: consider the initial vector probability and refer to Section 3.3

Solution

$$\alpha\mathbf{P} = (0.6\ 0.0\ 0.4)\begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.4 & 0.4 \\ 0.1 & 0.6 & 0.3 \end{pmatrix} = (0.28\ 0.42\ 0.30)$$

Thus the probability of having rain Saturday is 30 %.

Exercise 3.6

Compute the eigenvectors and eigenvalues of the matrix $\mathbf{P}$ given by eqn (3.32).

Solution

To find the eigenvectors, it needs to solve the *characteristic equation*, or eigenvalues equation

$$\lambda^2 - \lambda \operatorname{Tr} \mathbf{P} + \operatorname{Det} \mathbf{P} = 0$$

where $\operatorname{Tr} \mathbf{P}$ e $\operatorname{Det} \mathbf{P}$ are the trace and the determinant of matrix $\mathbf{P}$, respectively, in our case equal to $2 - a - b$ and $1 - b - a$. The equation giving the eigenvalues $\lambda_{1,2}$ is:

$$\lambda_{1,2} = \frac{\operatorname{Tr} \mathbf{P} \pm \sqrt{\operatorname{Tr} \mathbf{P}^2 - 4 \operatorname{Det} \mathbf{P}}}{2} = \frac{\operatorname{Tr} \mathbf{P} \pm \sqrt{\Delta}}{2}$$

where Δ denotes the discriminant. In our case it is:

$$\lambda_{1,2} = \frac{2-a-b}{2} \pm \frac{\sqrt{(b-a)^2 + 4ab}}{2} = \frac{2-a-b}{2} \pm \frac{a+b}{2}$$

from which:

$$\lambda_1 = 1 \quad \text{e} \quad \lambda_2 = (1 - a - b)$$

The roots λ_1 e λ_2 are distinct if $a + b \neq 0$.
Let us derive now the right eigenvectors

$$\mathbf{d}_1 = \begin{pmatrix} d_{11} \\ d_{12} \end{pmatrix} \qquad \text{e} \qquad \mathbf{d}_2 = \begin{pmatrix} d_{21} \\ d_{22} \end{pmatrix}$$

corresponding to λ_1 and λ_2.
The right eigenvectors corresponding to $\lambda_1 = 1$, are the solution of the equation:

$$\mathbf{P}\mathbf{d}_1 = \lambda_1 \mathbf{d}_1 \tag{1}$$

that, in explicit form, becomes:

$$\begin{cases} -ad_{11} + ad_{12} = 0 \\ bd_{11} - bd_{12} = 0 \end{cases}$$

from which $d_{11} = 1$ e $d_{12} = 1$, therefore:

$$\mathbf{d}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

The right eigenvectors corresponding to $\lambda_2 = 1 - a - b$, are the solution of the equation:

$$\begin{cases} bd_{21} + ad_{22} = 0 \\ bd_{21} + ad_{22} = 0 \end{cases}$$

from which $d_{21} = -a$ e $d_{22} = b$, therefore:

$$\mathbf{d}_2 = \begin{pmatrix} -a \\ b \end{pmatrix}$$

Hence, the matrix $\mathbf{A}$ results to be:

$$\mathbf{A} = \begin{pmatrix} 1 & -a \\ 1 & b \end{pmatrix}$$

Now let (e_1, e_2) be the left eigenvector corresponding to λ_1. Then:

$$\begin{pmatrix} e_1 & e_2 \end{pmatrix} \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix} = 1 \begin{pmatrix} e_1 & e_2 \end{pmatrix}$$

$$\begin{pmatrix} e_1[1-a] + e_2 b & e_1 a + e_2[1-b] \end{pmatrix} = \begin{pmatrix} e_1 & e_2 \end{pmatrix}$$

from which:

$$\begin{aligned} e_1(1-a) + e_2 b &= e_1 \\ e_1 a + e_2(1-b) &= e_2 \end{aligned}$$

Given $e_2 = a$, it is $e_1 = b$. We can choice (e_1, e_2), first raw of $\mathbf{A}^{-1}$, in such a way that it is a probability vector. Since the sum $e_1 + e_2$ must add to 1, it needs to divide e_1 and e_2 by $(a + b)$.

Lastly let (e_1, e_2) be the left eigenvector corresponding to λ_2. Then:

$$\begin{aligned} \begin{pmatrix} e_1 & e_2 \end{pmatrix} \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix} &= \begin{pmatrix} e_1[1-a] + be_2 & e_1 a + e_2[1-b] \end{pmatrix} \\ &= (1-a-b) \begin{pmatrix} e_1 & e_2 \end{pmatrix} \end{aligned}$$

from which $e_1 a = -ae_2$ or $e_1 = -e_2$. We can take $e_1 = -1$ and $e_2 = 1$ and, as above, divide by $(a + b)$. Hence:

$$\mathbf{A}^{-1} = \frac{1}{a+b} \begin{pmatrix} b & a \\ -1 & 1 \end{pmatrix}$$

---

3 Exercises of Chapter 4: Poisson Processes

Exercise 4.1

The following is a fun example extracted from the R. McElreath book ”Statistical Rethinking” (CRC Press, 2020). You are in the business with a monastery that you own. The monastery employs 1000 monks, independently working from one another. They copy by hand ancient manuscripts of varying length, and produce every day a variable number of manuscripts, some day more than 3, other day none. You know that on the average 1 over 1000 monks finishes a manuscript.

That is clearly a binomial process, so you can compute the mean μ and variance v by the well known formulae:

$$\mu = Np$$
$$v = Np(1-p)$$

where p = 1/1000 and N = 1000.

Simulate a true manuscript production in 10 years (3650 days) by an R code, to verify how close the "true" average production and variance are to the theoretical quantities. Why you are allowed to affirm that the number of finished manuscripts follow a Poisson distribution?
Hint: use the binomial random number generator `rbinom`

Solution

```
N <- 1000
p <- 1/1000
# theoretical mean and variance
m <- N*p
v <- N*p*(1-p)
# Simulation
y <- rbinom(3650,1000,1/1000)
mean(y)
var(y)
```

The computed values of mean and variance are very close to the theoretical ones, equal to 1 and 0.999 respectively. The quasi coincidence of mean and variance, and the shape of the distribution (obtained by `hist(y)`) are clear evidence of a Poisson distribution. You can convince yourself by comparing `hist(y)` with the histogram of `rpois(3650,lambda=1)`.

Exercise 4.2

With reference to exercise 4.1, suppose that you want to monopolize the market of manuscript production. You are offered to buy a new monastery, but you don't know how many monks work there. The only information you have is that it produces, on the average, 2 manuscripts per day. Use this information to infer the distribution of the number of manuscripts completed every day.

Solution

You can simulate, for example, 1000 days production by the R function `rpois`:

```
y <- rpois(1000,lambda = 2)
hist(y,freq=F,breaks=0:max(y))
```

to have informations about how many books are worked, on the average.

Exercise 4.3

Your sister constantly watches her smartphone, waiting for messages from her friends. Starting from 8 a.m. she gets messages at the rate of 5 per hour. Assuming that the messages arrival is a Poisson process, find:
(1) The probability that she receives exactly 20 messages before eleven o'clock.
(2) The probability of receiving at least 20 messages by the same hour.
(3) The probability she receives exactly 20 messages by 11 a.m. and exactly 40 by 4 p.m.

Solution

(1) The probability is that of receiving 20 messages in 3 hours, knowing that the events are Poisson distributed with parameter $\lambda = 5$:

$$\mathsf{P}\{N_3 = 20\} = \frac{e^{-3\lambda}(3\lambda)^{20}}{20!} = 0.042$$

where N_3 means "the number of messages received in 3 hours".
(2) The required probability is $1 - p$, where p is the probability of receiving up to 19 messages in 3 hours, i.e.

$$p = \sum_{k=0}^{19} \frac{e^{-3\lambda}(3\lambda)^k}{k!}$$

which happens to be about 87 %. Thus the requested probability is 13 %.
(3) If she receives 20 messages between 8 and 11, and 40 between 8 and 4 p.m., she gets 20 messages from 11 a.m. to 4 p.m., thus, using the same notation of answer (1):

$$\mathsf{P}\{N_3 = 20, N_8 = 40\} = \mathsf{P}\{N_3 = 20, N_8 - N_3 = 20\}$$

The last term, because of independence, is:

$$\mathsf{P}\{N_3 = 20, N_8 - N_3 = 20\} = \mathsf{P}\{N_3 = 20\}\,\mathsf{P}\{N_5 = 20\} = \frac{e^{-3\cdot 5}(3\cdot 5)^{20}}{20!}\frac{e^{-5\cdot 5}(5\cdot 5)^{20}}{20!}$$

which gives a value of 0.2 % for the required probability.

Exercise 4.4

Use the R function `dpois` to compute the probabilities of exercise 4.3.

Solution

Question (1). In R, by means of the `dpois` function:
`p <- dpois(20,3*lambda)`, with `lambda <- 5`.

Question (3). In R:

$$\mathsf{P}\{N_3 = 20, N_8 - N_3 = 20\} = \text{dpois}(20, 3 * 5) * \text{dpois}(20, 5 * 5)$$

which gives a value of 0.2 % for the required probability.

Exercise 4.5

A motorway rescue service receives calls according to a Poisson process at the rate of 6 calls per hour.
(1) Find the probability they do not receive calls over a period of 2 hours.
(2) Find the probability of having exactly 6 calls in the next hour and a half.

Solution

Question (1)

The probability of zero calls in the first two hours is that of receiving the first call after that time, so it has an exponential distribution. Denoting by $\mathsf{P}\{N_t\}$ the probability of having the first event (call) at time t:

$$\mathsf{P}\{N_2\} = e^{-\lambda t}$$

with $\lambda = 6$.

Question (2)

The required probability has a Poisson distribution, therefore:

$$\mathsf{P}\{N_{1.5} = 6\} = \frac{e^{-1.5\lambda}(1.5\lambda)^6}{6!}$$

Exercise 4.6

Write an R code to simulate the Poisson process of exercise 4.5 and numerically compute what required in that exercise.
Hint: remember that inter−arrival times are exponentially distributed.

Solution

First of all, we have to simulate a Poisson process with parameter λ. A simple way consists in generating inter-arrival times, that are exponentially distributed. In R code:

```
lambda <- 6
Nt <- 20
X <- rexp(n=Nt, rate=lambda)
# arrival times
S <- cumsum(X)
# plot the inter-arrival times:
N <- 0:Nt
plot(stepfun(x=S, y=N), do.points=F, main="Poisson Process",
      xlab="Time", ylab="Number of events")
```

In order to use that kind of simulation to obtain the probabilities computed in exercise 4.5, we need to generate the above arrival times a large number of times, such to give reliable values of such probabilities. Thus, to solve question (1):

```
lambda <- 6
Nt <- 20
# Generate inter-arrival times NN times
NN <- 10000
# initialize an array to contain the time of the first event
T <- rep(0,NN)
for (ii in 1:NN)
  {
  X <- rexp(n=Nt, rate=lambda)
  # arrival times
  S <- cumsum(X)
  # S[1] is the time of the first event
  T[ii] <- S[1]
  }
# compute the probability of 0 events in 2 hours, counting the number of simulations giving a "first event time"
sum(P>2)/NN
```

We see that this value is always close to zero, as it can also be computed exactly by using the exponential distribution:

```
p_N2 <- exp(-2*lambda)
```

The solution of question (2) is similar. First of all consider that the exact computation can be done using the `dpois()` R function or, equivalently, by the definition of the Poisson distribution:

```
# Probability of exactly 6 calls in 1.5 hours
n_calls <- 6
time <- 1.5
# using dpois
p_6 <- dpois(n_calls,time*lambda)
# or, alternatively
p_6 <- exp(-time*lambda)*(time*lambda)^n_calls/factorial(n_calls)
```

A simulation like that of question (1) can be implemented as follows.

```
lambda <- 6
Nt <- 20
# Generate inter-arrival times NN times
NN <- 10000
T <- rep(0,NN)
for (ii in 1:NN)
  {
  X <- rexp(n=Nt, rate=lambda)
  # arrival times
  S <- cumsum(X)
  if (S[1]<=time) {
    T[ii] <- max(which(S<=time))
  }
}
sum(P==n_calls)/NN
```

---

4 Exercises of Chapter 5: Random Walk

Exercise 5.1

A flea lives on a finite segment of a line, long D. It is able to do jumps towards left or right, except in the starting position $x = 0$ and in the ending position $x = D$, i.e. the two barriers are not surmountable with a jump. The insect jumps to the left with probability p_L, to the right with probability p_R or it decide to rest where it is, with probability p_S. The jump length is such that D is covered with 10 jumps in the same direction (starting from 0). A rather limited world, poor little bugs!

(1) How can you write the transition matrix of this random walk? And, (2) What kind of random walk is it?

Solution

Question (1)

The transition probability matrix is:

$$\begin{pmatrix}
p_S & p_R & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
p_L & p_S & p_R & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & p_L & p_S & p_R & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & p_L & p_S & p_R & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & p_L & p_S & p_R & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & p_L & p_S & p_R & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & p_L & p_S & p_R & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & p_L & p_S & p_R & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & p_L & p_S & p_R \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_L & p_S
\end{pmatrix}$$

Question (2)

It is a random walk with absorbing barriers.

Exercise 5.2

If, in the previous exercise, you take $p_R = 1 - p_L$ and, consequently, $p_S = 0$, what does this new random walk represent? Write its transition matrix.

Solution

Posing $p = p_R$, the transition matrix becomes:

$$\begin{pmatrix} 1 & p & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1-p & 1 & p & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-p & 1 & p & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-p & 1 & p & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-p & 1 & p & 0 & 0 & 0 & 0 \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-p & 1 \end{pmatrix}$$

The above matrix represents the gambler's ruin process.

Exercise 5.3

What kind of random walk is represented by the following transition matrix (q = 1 - p)?

$$\mathbf{P} = \begin{pmatrix} q & p & 0 & \dots & 0 & 0 & 0 \\ q & 0 & p & \dots & 0 & 0 & 0 \\ \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \dots & q & 0 & p \\ 0 & 0 & 0 & \dots & 0 & q & p \end{pmatrix}$$

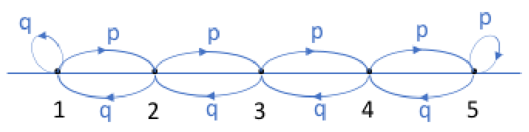
Solution

It represents a random walk with reflecting barriers, for example a game where the potential looser is given back its money to continue playing.

Exercise 5.4

Draw a graph like that in Fig. 5.1 to represent a transition matrix of exercise 5.3 for 5 states.

Solution



Exercise 5.5

We borrow from H. Tijms (“Understanding Probability”, Cambridge University Press, 2007), the depiction of the random walk as the walk of a drunk man. A drunkard moves on a line, at each step going to the left or to the right with equal probability. Call S_n the distance covered by a drunkard on a line, starting from the origin (the pub exit) after n steps. We have seen in Section 5.1 that for n large the average of S_n is zero, because there are as many positive as negative steps. But nevertheless, with the passing of time, the drunkard gets tired of walking because he actually ‘travelled’ for a distance that is

$$D_n = |X_1 + \ldots + X_n|$$

where $X_1 \ldots X_n$ are the n steps taken by the drunkard.

(1) Numerically compute $\mathrm{E}[D_n]$ and $\mathrm{E}[S_n]$ with increasing n, for a random walk simulated with R. (2) For n large, demonstrate that $\mathrm{E}[D_n] \approx \sqrt{\frac{2n}{\pi}}$

Hint: apply the Central Limit Theorem to $X_1 \ldots X_n$

Solution

Question (1)

```
p <- 0.5
x <- numeric()
S <- numeric()
walk <- numeric()
walk[1] <- 0
# number of steps
n <- 100
```

```
# number of simulations
Nmax <- 1000
for (k in 1:Nmax)
{
  u <- runif(n)
  for(i in 1:n)
  {
    if (u[i]<=p)
      x[i] <- +1
    else
      x[i] <- -1
    if (i>1)
      walk[i] <- walk[i-1]+x[i]
  }
  S[k] <- sum(x)
  # plot the walks
  if (k==1)
    plot(1:n,walk,t="l",ylim=c(-50,50),xlab="number of steps")
  else
    lines(walk)
}
# plot S
plot (1:Nmax,S,type="l")
# results: mean of S, mean of |S|, theoretical value of <|S|>
# where S = x1 + x2 + ...
c(mean(S),mean(abs(S)),sqrt(2*n/pi))
```

Question (2)

From the Central Limit Theorem, the random variable $S = X_1 + \ldots + X_n$ has a normal distribution, with mean 0 and variance $\sigma^2 = n$ (see Section 5.1). Therefore, the density of the probability distribution of the random variable $D = |S|$ is:

$$p(z) = \begin{cases} \frac{2}{\sqrt{2\pi n}} \exp\left(-\frac{z^2}{2n}\right) & \text{if } z > 0, \\ 0 & \text{if } z \leq 0 \end{cases}$$

Therefore the $\mathrm{E}\left[D\right]$ is the integral:

$$\mathrm{E}\left[D\right] = \int_0^\infty z p(z) dz = \frac{\sqrt{2n}}{\sqrt{\pi}}$$

Exercise 5.6

A common approach to simulate spatial dispersal of insects with time is to use the random walk. A two−dimensional random walk is used to simulate the degree of insect diffusion over an area with time. Write a code to simulate a two−dimensional movement of a female insect that starts with 100 eggs and deposits 1 egg at each step. Assume an equiprobability for the movement in four directions: left, right, top, bottom. The program ends when the female finishes the eggs. Then assume there are N females and modify the program by setting a variable *steps to oviposit* that is the number of steps the insect must take before depositing one egg. Play with the variables: *number of females*, *number of eggs per female* and *steps to ovideposit*, to see how the distribution of eggs changes.

Solution

First question

```
n.eggs<- 100
xdir<-0
ydir<-0
x<-vector()
x[1]<-xdir
y<-vector()
y[1]<-ydir
for (j in 2:n.eggs) {
  u<-runif(1)
  if(u<=0.25) {xdir<-xdir+1}
  if(u>0.25 & u<=0.5) {xdir<-xdir-1}
  if(u>0.5 & u<=0.75) {ydir<-ydir +1}
  if(u>0.75) {ydir<-ydir-1}
  x[j]<-xdir
  y[j]<-ydir
}   # ending loop on steps

plot(x,y,type="l",xlab="x",ylab="y",main=" ",lwd=1, lty=1, col="black", font.lab=3,cex.lab=1.4)
# starting (red) and ending (blue) positions
text(x[1],y[1],"o",col="red",cex=1)
text(x[n.eggs],y[n.eggs],"o",col="blue",cex=1)
```

The `plot` command shows the eggs distribution, highlighting in red and blue the initial and final positions, respectively.

Second question

```
n.eggs <- 100
n.female <- 50
steps.to.oviposit <- 2
# all females start from the same initial point
x <- rep(0,n.eggs)
y <- rep(0,n.eggs)
# extension of the "world", to be changed if steps.to.oviposit and n.eggs increase
Xmin <- -25
Xmax <- 25
Ymin <- -25
Ymax <- 25
# loop on the number of females
for (k in 1:n.female)
{
  xdir <- 0
  ydir <- 0
  # loop on the number of eggs
  for (j in 2:(n.eggs*steps.to.oviposit)) {
    u<-runif(1)
    if(u<=0.25) {xdir<-xdir+1}
    if(u>0.25 & u<=0.5) {xdir<-xdir-1}
    if(u>0.5 & u<=0.75) {ydir<-ydir +1}
    if(u>0.75) {ydir<-ydir-1}
    x[j]<-xdir
    y[j]<-ydir
  }   # ending loop on steps
  if (k==1)
    plot(x,y,type="l",xlab="x",ylab="y",main=" ",lwd=1, lty=1, col="black", font.lab=3,cex.lab=1.4,xlim=c(Xmin,Xm
  else
    lines(x,y,lty=1,lwd=1)
  # starting (red) and ending (blue) positions
  text(x[1],y[1],"o",col="red",cex=0.6)
  text(x[n.eggs],y[n.eggs],"o",col="blue",cex=0.6)
}
```

Changing *steps.to.oviposit* and *n.eggs* and plotting the result of the simulation we see how those parameters influence the extension of the eggs distribution.

Exercise 5.7

With reference to exercise 5.6, suppose there are two species of ants: black and red. Suppose the queens of both species have 100 eggs, and assign a different number N_s of *steps to oviposit* to the species, for example $N_s = 2$ for black ants, $N_s = 5$ for red ants. On a graph, represent black and red eggs with a coloured dot and see the behaviour of the two distributions of eggs.

Solution

```
n.eggs <- 100
red_Ns <- 5
blk_Ns <- 2
# all females start from the same initial point
x <- rep(0,n.eggs)
y <- rep(0,n.eggs)
# extension of the "world"
Xmin <- -25
Xmax <- 25
Ymin <- -25
Ymax <- 25
# black female
xdir <- 0
ydir <- 0
# loop on the number of eggs
for (j in 2:(n.eggs*blk_Ns)) {
  u<-runif(1)
  if(u<=0.25) {xdir<-xdir+1}
  if(u>0.25 & u<=0.5) {xdir<-xdir-1}
  if(u>0.5 & u<=0.75) {ydir<-ydir +1}
  if(u>0.75) {ydir<-ydir-1}
  x[j]<-xdir
  y[j]<-ydir
}   # ending loop on steps
plot(x,y,type="p",xlab="x",ylab="y",main=" ",
   font.lab=3,cex.lab=1.4,xlim=c(Xmin,Xmax),ylim=c(Ymin,Ymax),pch=".",cex=2)
# red female
xdir <- 0
ydir <- 0
# loop on the number of eggs
for (j in 2:(n.eggs*red_Ns)) {
  u<-runif(1)
  if(u<=0.25) {xdir<-xdir+1}
  if(u>0.25 & u<=0.5) {xdir<-xdir-1}
  if(u>0.5 & u<=0.75) {ydir<-ydir +1}
  if(u>0.75) {ydir<-ydir-1}
  x[j]<-xdir
  y[j]<-ydir
}   # ending loop on steps
points(x,y,col="red",pch=".",cex=2)
```

Exercise 5.8

Simulate a trajectory of a three−dimensional random walk.

Solution

DA FARE

---

5 Exercises of Chapter 6: ARMA Processes

Exercise 6.1

A non stationary process, consisting in a monotonic trend and a white noise, exhibits a slowly decreasing, linear ACF. For example, the process defined by:

$$X_t = \beta t + \epsilon_t$$

with β constant, has an ACF like that in Figure 1. Remembering the definition of autocovariance at lag k:

$$\gamma_k = \mathrm{E}\left[(X_t - \mu)(X_{t+k} - \mu)\right]$$

justify the linear behaviour in that particular case.

Hint: take $\beta = 1$ and consider the difference among subsequent lags, $\gamma_k - \gamma_{k+1}$

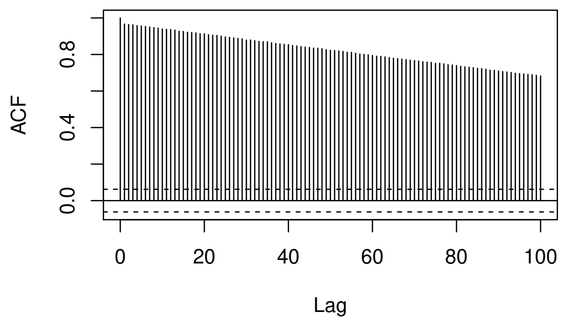


Figure 1: ACF of a linear trend

Solution

Suppose the time series is defined on the closed interval $[0, N]$, so that $\mu = \mathrm{E}[Xt] = N/2$, assuming for simplicity $\beta = 1$ and taking time at integer steps, the covariances at lag k and $k+1$ are:

$$\gamma_k = \frac{1}{N-k} \sum_{i=1}^{N-k} \left(i - \frac{N}{2}\right)\left(i + k - \frac{N}{2}\right)$$

and

$$\gamma_{k+1} = \frac{1}{N-k-1} \sum_{i=1}^{N-k-1} \left(i - \frac{N}{2}\right)\left(i + k + 1 - \frac{N}{2}\right)$$

because $X_t \equiv X_i = i + \epsilon_i$. We are interested in the difference $\gamma_k - \gamma_{k+1}$. With some algebraic manipulations, using the following results on series:

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

$$\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6}$$

we obtain:

$$\gamma_k - \gamma_{k+1} = \frac{k}{3} + \frac{N+1}{6}$$

If in a succession $\{a_k\}$ the difference among two successive terms is linear in k:

$$a_k - a_{k-1} = \alpha k + \beta$$

we know that the succession terms are quadratic in k. Moreover, if $k << N$ or, in other words, if we are interested in the first autocorrelation terms of the ACF, we approximately have:

$$\gamma_k - \gamma_{k+1} \approx \frac{N+1}{6}$$

i.e. $\{\gamma_k\}$ is an arithmetic progression with common difference $-(N+1)/6$, thus its terms linearly decrease with k.

Exercise 6.2

With reference to Exercise 6.1, write an R code for computing the autocorrelation function by implementing:

$$\gamma_k = \frac{1}{N-k} \sum_{i=1}^{N-k} (X_i - \mu)(X_{i+k} - \mu)$$

and remembering that $\rho_k = \gamma_k/\gamma_0$. Use the code to show that a process involving a quadratic trend:

$$X_t = \beta t^2 + \epsilon_t$$

has also a linear, slowly decreasing ACF. Using the following time series data:

```
ti <- 0:1000
beta <- 1e-4
err <- rnorm(1001,sd=2)
x <- beta*ti^2+err
```

show that the 100 lag ACF is very similar to that of the linear trend process.

Solution

We can write an ACF function having the time series x and the lag k as arguments, returning the autocorrelation value at that lag.

```
# autocorrelation function
autoc <- function(x,L)
{
  autocv <- 0
  mu <- mean(x)
  s2 <- var(x)
  N <- length(x)
  for (i in 1:(N-k)){
    autocv <- autocv + ((x[i]-mu)*(x[i+lag]-mu))
  }
  autocv <- autocv/(N-lag)

  return (autocv/s2)
}
```

```
# main program
ti <- 0:1000
beta <- 1e-4
err <- rnorm(1001,sd=2)
Xt <- beta*ti^2+err
# ACF
acf(Xt,lag=100)
# Using autoc()
ac <- rep(0,100)
for (i in 1:100)
  ac[i] <- autoc(i)
plot(ac,t="h",ylim=c(0,1))
```

Exercise 6.3

The following code:

```
n <- 10
x_n <- runif(n)
Nr <- 100
X <- rep(x_n,Nr)
acf(X)
# series length
N <- Nr*n
```

shows that the ACF of a repeated pattern of period n has almost unitary values at lags n and multiples of n. How can you explain it?

Hint: use the definition of autocorrelation, considering that $N - n \approx N$

Solution

Taking $\mu = 0$, for simplicity:

$$\gamma_k = \frac{1}{N-k} \sum_{i=1}^{N-k} X_i X_{i+k}$$

using the condition: $X_{i+np} = X_i$, and computing:

$$\gamma_p = \frac{1}{N-p} \sum_{i=1}^{N-p} X_i^2$$

Thus, $\rho_k = \gamma_k/\gamma_0$ at lags 0, p, etc is:

$$\rho_0 = 1$$

$$\rho_p \approx 1$$

$$\rho_{2p} \approx 1$$

$$\ldots$$

$$\rho_{np} \approx 1$$

for any n such that $N - n \approx N$.

Exercise 6.4

An invertible MA(1) process with parameter θ can be expressed as an infinite AR. In fact, expressing the former in terms of the lag polynomial $\Theta(L) = 1+\theta L$, if $\Psi(L)$ denotes the inverse of Θ, we have:

$$\Psi(L)X_t = \Psi(L)\left(\Theta(L)w_t\right) = w_t$$

because $\Psi\Theta = 1$, with Ψ polynomial of infinite order:

$$\Psi(L) = 1 + \psi_1 L + \psi_2 L^2 + \psi_3 L^3 + ...$$

(i) Obtain the expression of the coefficients ψ_k.

(ii) How many of them are necessary to regain the noise w_t?

(iii) Write an R program for computing the noise, if the MA(1) is given by the following code:

```
N <- 200
theta <- 0.5
w <- rnorm(N)
X <- rep(0,N)
# generate MA(1)
X[1] <- w[1]
for (t in 2:N) X[t] <- w[t] + theta*w[t-1]
```

and verify that the input and output noise are practically identical with just 5-6 terms.

Solution

Question (1).

From $\Psi\Theta = 1$ we have:

$$\left(1 + \psi_1 L + \psi_2 L^2 + \ldots\right)(1+\theta L) = 1$$

by equating the equal powers of L^k:

$k = 1$: $\psi_1 + \theta = 0 \rightarrow \psi_1 = -\theta$
$k = 2$: $\psi_2 + \psi_1\theta = 0 \rightarrow \psi_2 = \theta^2$
$k = 3$: $\psi_3 + \psi_2\theta = 0 \rightarrow \psi_2 = -\theta^3$

...
any k: $\psi_k + \psi_{k-1}\theta = 0 \rightarrow \psi_k = (-1)^k \theta^k$

Question (2).

In principle, an infinite number of terms:

$$w_t = \sum_{k=0}^{\infty} \psi_k L^k X_t = 1 + \psi_1 X_{t-1} + \psi_2 X_{t-2} + \ldots$$

Question (3).

```
# Assumptions
N <- 200
theta <- 0.5
w <- rnorm(N)
X <- rep(0,N)
# generate MA(1)
X[1] <- w[1]
for (t in 2:N) X[t] <- w[t] + theta*w[t-1]

# Plot the time series
plot(as.ts(X),ylab="MA(1) process")
```

Consider the first n terms. Of course the expansion is of infinite order, thus we cannot practically apply the AR infinite expansion, because we would need an infinite series, and we would discard an infinite number of terms.

To see how the inverse lag polynomial work, we compute a few terms and we see that with few terms we practically obtain the original noise (except for the first n terms, of course...).

```
n <- 5
psi <- rep(0,n)
psi[1] <- -theta
for (i in 2:n)
  psi[i] <- -psi[i-1]*theta
eps <- rep(0,N)
eps <- X
for (t in (n+1):N)
  for (i in 1:n)
    eps[t] <- eps[t] + psi[i]*X[t-i]

# Plot the results
plot(eps[n:N],t="l")
lines(w[n:N],col="red")
```

Exercise 6.5

Repeat exercise 6.4, items (i) and (iii), for an invertible MA(2) process with MA coefficients $\theta_1 = 0.8$ and $\theta_2 = 0.5$. In this case, the process can be generated by:

```
N <- 200
theta1 <- 0.8
theta2 <- 0.5
w <- rnorm(N)
X <- rep(0,N)
# generate MA(2)
X[1] <- w[1]
X[2] <- w[2]
for (t in 3:N) X[t] <- w[t] + theta1*w[t-1] + theta2*w[t-2]
```

Solution

Question (1).

The MA(2) process is defined by:

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

therefore the lag polynomial is:

$$\Theta(L) = 1 + \theta_1 L + \theta_2 L^2$$

Formally:

$$\epsilon_t = \Theta(L)^{-1} X_t$$

Solving $\Psi\Theta = 0$, with:

$$\Psi(L) = \Theta(L)^{-1} = \psi_0 + \psi_1 L + \psi_2 L^2 + \dots$$

Following the same lines of exercise 6.4 we obtain the ψ_k coefficients by solving:

$$\left(1 + \psi_1 L + \psi_2 L^2 + \dots\right)\left(1 + \theta_1 L + \theta_2 L^2\right) = 1$$

The explicit values of the coefficient are obtained from:
$\psi_0 = 1$
$\psi_1 + \psi_0\theta_1 = 0$
$\psi_2 + \psi_1\theta_1 + \psi_0\theta_2 = 0$
$\psi_3 + \psi_1\theta_2 + \psi_2\theta_1 = 0$
$\dots$
$\psi_k + \psi_{k+1}\theta_1 + \psi_{k+2}\theta_2 = 0$

Question (2).
The answer is the same of exercise 6.4

Question (3).

```
# Assumptions
N <- 200
theta1 <- 0.8
theta2 <- 0.5
w <- rnorm(N)
X <- rep(0,N)
# generate MA(2)
X[1] <- w[1]
X[2] <- w[2]
for (t in 3:N) X[t] <- w[t] + theta1*w[t-1] + theta2*w[t-2]

# plot time series
plot(as.ts(X),ylab="MA(2) process")
```

Analogously to the previous exercise, we consider the first n terms. Of course the expansion is of infinite order, thus we cannot practically apply the AR infinite expansion, because we would need an infinite series, and we would discard an infinite number of terms.

To see how the inverse lag polynomial work, we compute a few terms and we see that with few terms we practically obtain the original noise (except for the first n terms, of course...).

```
n <- 8
psi <- rep(0,n)
psi[1] <- -theta1
psi[2] <- -theta2 -psi[1]*theta1
for (i in 3:n)
  psi[i] <- -psi[i-2]*theta2-psi[i-1]*theta1
eps <- rep(0,N)
eps <- X
for (t in (n+1):N)
  for (i in 1:n)
    eps[t] <- eps[t] + psi[i]*X[t-i]
# Plot results
plot(eps[n:N],t="l")
lines(w[n:N],col="red")
```

Exercise 6.6

A theorem coming from general linear process theory asserts that an AR(p) process is stationary if and only if the modulus of all the roots of the characteristic polynomial (associated to the lag polynomial) is greater than one. Demonstrate that, for an AR(2) process with coefficients ϕ_1 and ϕ_2 and roots z_1 and z_2:

$$|z_1| > 1 \text{ and } |z_2| > 1$$

implies the following relations:

$$\phi_1 + \phi_2 < 1$$

$$\phi_2 - \phi_1 < 1$$

$$|\phi_2| < 1$$

Hint: consider the roots λ_1 and λ_2 of the "reciprocal" characteristic equation $\lambda^2 - \phi_1\lambda - \phi_2 = 0$ instead of those of $-\phi_1 z - \phi_2 z^2$, and determine conditions on the solutions satisfying $|\lambda_1| < 1$ and $|\lambda_2| < 1$

Solution

If the absolute values of all the roots of the characteristic polynomial are greater than one:

$$1 - \phi_1 z - \phi_2 z^2 = 0$$

has two solutions:

$$z_1 = \frac{-\phi_1 - \sqrt{\phi_1^2 + 4\phi_2}}{2\phi_2}$$

$$z_2 = \frac{-\phi_1 + \sqrt{\phi_1^2 + 4\phi_2}}{2\phi_2}$$

We impose $|z_1| > 1$ and $|z_2| > 1$.

Using the reciprocal of the characteristic polynomial, i.e. the polynomial expressed in terms of $\lambda = 1/z$:

$$\lambda^2 - \phi_1\lambda - \phi_2 = 0$$

its roots λ_1 and λ_2 are:

$$\lambda_1 = \frac{\phi_1 - \sqrt{\phi_1^2 + 4\phi_2}}{2\phi_2}$$

$$\lambda_2 = \frac{\phi_1 + \sqrt{\phi_1^2 + 4\phi_2}}{2\phi_2}$$

and the stationary condition is, in terms of those roots:

$$|\lambda_1| < 1$$

$$|\lambda_2| < 1$$

and, as a consequence $|\lambda_1\lambda_2| = |\phi_2| < 1$.

Moreover, the same condition implies that:

$$|\lambda_1 + \lambda_2| = |\phi_1| < 2$$

Therefore:

$$-1 < \phi_2 < 1$$

and

$$-2 < \phi_1 < 2$$

If λ_1 and λ_2 are real roots:

$$\phi_1^2 + 4\phi_2 \geq 0 \rightarrow -1 < \lambda_2 \leq \lambda_1 < 1$$

and this implies:

$$\phi_1 + \phi_2 < 1$$

and

$$\phi_2 - \phi_1 < 1$$

If $\phi_1^2 + 4\phi_2 < 0$, roots are complex and $-\phi_1^2/4 > \phi_2$.

We obtain a region bounded by:

$\phi_2 = 1 + \phi_1$
$\phi_2 = 1 - \phi_1$
$\phi_2 = -1$

Inside that region, the AR(2) process is stationary.

Exercise 6.7

Write an R code graphically showing that an AR(1) process $X_t = \phi X_{t-1} + \epsilon_t$ gradually changes from random noise ($\phi = 0$) to a random walk ($\phi = 1$), passing through an infinite number of stationary processes (for $0 < \phi < 1$). Which parameter can you compute to distinguish among different AR(1) processes?

Solution

The following code does the work:

```
N <- 100
# Random noise
epsilon <- rnorm(N)
# Plot time series
plot(epsilon,t="l",ylim=c(-5,5))
V <- var(epsilon)
k <- 1
for (phi in seq(0.1,1,0.05)){
  X <- rep(0,N)
  for (i in 2:N) {
    X[i] <- phi*X[i-1] + epsilon[i]
  }
  # Plot time series
  lines(X)
  V[k] <- var(X)
  k <- k+1
}
```

```
# Plot variances
plot(V,t="l")
```

The answer to the final question is in the code. The variance gradually increases from the noise value to a virtually infinite value, because of its mathematical form:

$$\sigma_X^2 = \frac{\sigma_\epsilon^2}{1-\phi^2}$$

described in Section 6.3.1.

Exercise 6.8

Which model does the following expression represent?

$$\Phi_1(L)(1-L)X_t = \Theta_2(L)\epsilon_t$$

with:

$$\Phi_1(L) = 1 - \phi L$$

and

$$\Theta_2(L) = 1 + \theta_1 L + \theta_2^2 L$$

Explicitly write down the model represented by it. Is that a stationary process? If not, how can you get a stationary process from it?

Solution

It represents an ARIMA(1,1,2) process, whose explicit expression is:

$$X_t = X_{t-1} + \phi(X_{t-1} - X_{t-2}) + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2}$$

therefore it intrinsically a non stationary process. You can convince yourself by simulating an ARIMA(1,1,2) process with the `arima.sim()` R function:

```
Y <- arima.sim(list(order = c(1,1,2), ar = 0.7, ma=c(0.2,-0.5)), n = 1000)
plot(Y)
acf(Y)
```

and by plotting its ACF.

If the AR part is stationary (i.e., if the AR coefficient ϕ is less than 1) the process can be made stationary by differencing it. That is, of course, the definition of ARIMA processes! Indeed:

```
plot(diff(Y))
acf(diff(Y))
```

shows that the first difference of the process is ARMA(1,2).

Exercise 6.9

Simulate in R a series of 100 values extracted from an ARMA(1,2) model with autoregressive coefficient $\phi = 0.5$ and moving average coefficients $\theta_1 = 0.4$ and $\theta_2 = -0.3$, using the function `arima.sim`. Refer to *help(arima.sim)* for the proper syntax.

Analyse its plot, and investigate its autocorrelation structure by computing and plotting the ACF and PACF.

Solution

The `arima.sim()` function requires in input the number of data to be generated and the structure of the model, as in the following code:

```
X <- arima.sim(n=100,model=list(ar=0.5,ma=c(0.4,-0.3)))
plot(X)
ACF.X <- acf(X)
PACF.X <- pacf(X)
```

The `acf()` and `pacf()` functions compute and plot the autocorrelation and partial autocorrelation of the generated series.

Exercise 6.10

With reference to exercise 6.9, compare the ACF and PACF plots of the ARMA(1,2) model with those of AR(1) and MA(2) processes with the same parameters. Try to generate data several times and see how the autocorrelation functions change.

Hint: generate AR(1) and MA(2) data with arima.sim, and superimpose ACF and PACF of the three models

Solution

The following code generates tow series of 100 numbers, the first being AR(1) with $\phi = 0.5$, the second MA(2) with $\theta_1 = 0.4$ and $\theta_2 = -0.3$.

```
Xar <- arima.sim(n=100,model=list(ar=0.5))
ACF.Xar<- acf(Xar)
PACF.Xar <- pacf(Xar)
Xma <- arima.sim(n=100,model=list(ma=c(0.4,-0.3)))
ACF.Xma<- acf(Xma)
PACF.Xma <- pacf(Xma)
```

The ARIMA(1,2) series of exercise 6.9 is compared to the AR(1) and MA(2) by:

```
plot(X)
lines(Xar,col="red")
lines(Xma,col="green")
```

The following code graphically compares the autocorrelation structure of the three series.

```
# compare ACF
plot(ACF.X$lag,ACF.X$acf,t="h")
lines(ACF.Xar$lag,ACF.Xar$acf,col="red",t="h")
lines(ACF.Xma$lag,ACF.Xma$acf,col="green",t="h")
# compare PACF
plot(PACF.X$lag,PACF.X$acf,t="h")
lines(PACF.Xar$lag,PACF.Xar$acf,col="red",t="h")
lines(PACF.Xma$lag,PACF.Xma$acf,col="green",t="h")
```

---

6 Exercises of Chapter 7: Spectrum Analysis

Exercise 7.1

Given a zero−mean, stationary real−valued time series $X_t = \{x_t\}$, $t = 1..N$, the power spectral density is defined as:

$$P_{SD}(\omega) = \frac{1}{N}\left|\sum_{k=0}^{N} x_k e^{-i\omega k}\right|^2$$

Prove that:

$$P_{SD} = \frac{1}{N}\sum_{k=-N}^{N} \hat{\gamma}(k)e^{-i\omega k}$$

where $\hat{\gamma}(k)$ is the sample autocovariance:

$$\hat{\gamma}(k) = \frac{1}{N}\sum_{j=0}^{N-k} x_j x_{j+k}$$

for $k \geq 0$, and

$$\hat{\gamma}(k) = \hat{\gamma}(-k)$$

for $k < 0$.

Hint: use the notable relation:

$$\sum_{t=1}^{N}\sum_{s=1}^{N} f(t-s) = \sum_{\tau=-N+1}^{N-1}(N-|\tau|)f(\tau)$$

Solution

$P_{SD(\omega)}$ is:

$$P_{SD}(\omega) = \frac{1}{N}\sum_{p=0}^{N}\sum_{q=0}^{N} x_p x_q e^{-i\omega(q-p)}$$

Can be rearranged as follows:

$$P_{SD}(\omega) = \frac{1}{N}\sum_{k=-N}^{N}\sum_{j=0}^{N-|k|} x_{j+|k|} x_j e^{-i\omega k}$$

Considering that:

$$\hat{\gamma}(|k|) = \frac{1}{N}\sum_{j=0}^{N-|k|} x_j x_{j+|k|}$$

we eventually have:

$$P_{SD}(\omega) = \sum_{k=-N}^{N} \hat{\gamma}(|k|) e^{-i\omega k}$$

Because of the reported notable relation:

$$\sum_{t=1}^{N}\sum_{s=1}^{N} f(t-s) = \sum_{\tau=-N+1}^{N-1} (N-|\tau|) f(\tau)$$

Exercise 7.2

The autocovariance function is defined in terms of the periodogram $\phi(\omega)$, as

$$\gamma(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \phi(\omega) e^{i\omega k} d\omega$$

Prove that $|\gamma(k)| \leq \gamma(0)$ for any $k > 0$.

Solution

Consider that:

$$\gamma(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \phi(\omega) d\omega$$

From complex analysis we know that the integral over a curve ζ of a function $f(z)$ satisfies the following (Darboux) inequality:

$$\left|\int_{\zeta} f(z)dz\right| \leq \int_{\zeta} |f(z)|\, dz$$

Therefore, considering that

$$\left|\phi(\omega)e^{-i\omega k}\right| = |\phi(\omega))|$$

because k is real.

We eventually have:

$$|\gamma(k)| = \left|\frac{1}{2\pi}\int_{-\pi}^{\pi}\phi(\omega)e^{i\omega k}d\omega\right| \leq \frac{1}{2\pi}\int_{-\pi}^{\pi}|\phi(\omega)|\,d\omega = \gamma(0)$$

Exercise 7.3

Prove that for a causal AR(1) process $X_t = \phi X_{t-1} + \epsilon_t$, with noise variance σ^2:

$$P_{SD}(\omega) = \frac{\sigma^2}{1+\phi^2-2\phi\text{cos}\omega}$$

Solution

By definition:

$$P_{SD}(\omega) = \gamma_0 + 2\sum_{k=1}^{\infty}\gamma_k\text{cos}(\omega k)$$

For an AR(1) process we can explicitly compute the autocovariances at any k in terms of the noise variance σ^2:
$\gamma_0 = \frac{\sigma^2}{1-\phi^2}$
$\gamma_1 = \frac{\phi\sigma^2}{1-\phi^2}$
$\ldots$
$\gamma_k = \frac{\phi^k\sigma^2}{1-\phi^2}$

Therefore:

$$P_{SD}(\omega) = \frac{\sigma^2}{1-\phi^2} + 2\sum_{k=1}^{\infty}\frac{\phi^k\sigma^2}{1-\phi^2}\text{cos}(\omega k)$$

By the Euler formula, with some algebra (the series can be summed because $\phi < 1$ for the assumed causality of the AR(1) process):

$$P_{SD}(\omega) = \frac{\sigma^2}{1-\phi^2} + \left(1 + \frac{\phi e^{j\omega}}{1-\phi e^{j\omega}} + \frac{\phi e^{-j\omega}}{1-\phi e^{-j\omega}}\right)$$

which, in terms of the common denominator:

$$\left(1-\phi e^{j\omega}\right)\left(1-\phi e^{-j\omega}\right) \equiv 1+\phi^2-2\phi\omega$$

gives the asserted expression.

Exercise 7.4

If X_t a MA(1) process $X_t = \epsilon_t + \theta\epsilon_{t-1}$, its lag-polynomial is defined by:

$$\Theta(L) = 1 + \theta L$$

Prove that, given the noise variance σ^2:

$$P_{SD}(\omega) = \sigma^2 \left|\Theta\left(e^{-i\omega}\right)\right|^2$$

that is, the power spectral density of a MA(1) process is the squared modulus of the lag-polynomial computed at $e^{-i\omega}$.

Solution

$$\left|\Theta(e^{-j\omega})\right|^2 = \left|1 + \theta e^{-j\theta}\right|^2 = 1 + \theta^2 + 2\theta\text{cos}\omega$$

But, for an MA(1) process:
$\gamma_0 = \sigma^2(1 + \theta^2)$
$\gamma_1 = \sigma^2\theta$
$\gamma_k = 0,\ k > 1$
and

$$P_{SD}(\omega) = \gamma_0 + 2\sum \gamma_k \text{cos}\omega k$$

Exercise 7.5

With reference to exercise 7.4, i can be demonstrated that the relationship between the lag-polynomial and the power spectral density P_{SD} holds for any MA(n) process.

(1) Justify the following assertion:

For a causal AR(1) process, P_{SD} can be expressed as the modulus of a polynomial $\Psi(e^{-i\omega})$:

$$P_{SD}(\omega) = \sigma^2 \left|\Psi\left(e^{-i\omega}\right)\right|^2$$

(2) What is the Ψ polynomial?

Solution

Question (1): an AR(1) process can be expressed as an MA(∞).

Question(2): Ψ is $\Theta(L)$ for an ∞-order MA process.

Exercise 7.6

The relationship of exercise 7.4 can be further generalized for any causal ARMA process. For such a process $\Phi(L)X_t = \Theta(L)\epsilon_t$, with noise variance σ^2, we can write the power spectral density P_{SD} in rational form:

$$P_{SD}(\omega) = \sigma^2 \frac{\left|\Theta\left(e^{-i\omega}\right)\right|^2}{\left|\Phi\left(e^{-i\omega}\right)\right|^2}$$

Using the above rational formula, write the explicit expression of $P_{SD}(\omega)$ for an ARMA(1,1) process of coefficients ϕ and θ.

Solution

$$\left|\Theta\left(e^{-j\omega}\right)\right|^2 = \left|1+\theta e^{-j\omega}\right|^2 = 1+\theta^2+2\theta\text{cos}\omega$$

$$\left|\Phi\left(e^{-j\omega}\right)\right|^2 = \left|1-\phi e^{-j\omega}\right|^2 = 1+\phi^2-2\phi\text{cos}\omega$$

Therefore

$$P_{SD}(\omega) = \sigma^2 \frac{1+\theta^2+2\theta\text{cos}\omega}{1+\phi^2-2\phi\text{cos}\omega}$$

Exercise 7.7

Using the result of exercise 7.6 write an R code to compute the $P_{SD}(\omega)$ of an ARMA(1,1) process with $\phi = 0.5$, $\theta = 0.5$ and $\sigma^2 = 1$. Plot it for $0 \leq \omega \leq \pi$.

Solution

```
phi <- 0.5
theta <- 0.5
sigma2 <- 1
omega <- seq(-pi,pi,0.05)
Psd <- sigma2*(1+theta^2+2*theta*cos(omega))/(1+phi^2-2*phi*cos(omega))
plot(omega,Psd,t="l")
```

Exercise 7.8

Following Section 7.4 and the example codes developed there, perform a singular spectrum analysis on the R dataset airmiles. Write an R code and obtain a plot like Figure 7.26.

What can we say about the trend of that time series?

Solution

Use the `Rssa` library for performing the singular spectrum analysis, as follows.

```
# load data
data(airmiles)
# load Rssa package
library(Rssa)
# Do SSA and plot results
s <- ssa(airmiles)
plot(s,numvalues = 10,main="",ylab="Eigenvalue norms")
plot(s,type="vectors",idx=1:10,main="")
r <- reconstruct(s, groups = list(1, c(2,3)))
plot(r,main="")
# trend part
plot(airmiles); lines(r$F1,col="red")
```

Exercise 7.9

Do the same analysis of exercise 7.8 to the closing prices of the S&P 500 stock index recorded between 1950 and 2019. The datafile, available on the website of the book, can be read with the following code.

```
# change directory to that containing the data
setwd("C:/RPA/code/spectrum_analysis/data")
# read data
dat <- read.csv("snp500.csv")
dat.date <- as.POSIXct(dat$Date, format = "%Y-%m-%d", tz = "GMT")
dat.close <- dat$Close
# plot it
plot(dat.date,dat.close,t="l")
```

Solution

```
# read data and plot time series
setwd("C:/RPA/code/spectrum_analysis/data")
read.csv("snp500.csv") -> xxx
xxx.date <- as.POSIXct(xxx$Date, format = "%Y-%m-%d", tz = "GMT")
plot(xxx.date,xxx$Close,t="l")
xxx.ts <- ts(xxx$Close)
# do SSA and plot results
s <- ssa(xxx.ts)
plot(s,numvalues = 10,main="",ylab="Eigenvalue norms")
plot(s,type="vectors",idx=1:10,main="")
r <- reconstruct(s, groups = list(1, c(2,3)))
plot(r,main="")
# trend part
plot(xxx.date,xxx.ts,t="l"); lines(xxx.date,r$F1,col="red")
# oscillatory part
plot(xxx.date,r$F2,t="l")
```

Exercise 7.10

The file `bike_rent_daily.csv` on the book website contains the daily number of bikes rent in Washington D.C. from January 1st 2011 to December 31st 2012 (Fanaee-T, H. and Gama, J. (2013). Event labeling combining ensemble detectors and background knowledge. Progress in Artificial Intelligence, 115.). Figure 2 shows the result of the SSA analysis.

Write an R code to obtain that figure, and discuss the meaning of the three plots below that of the original time series.

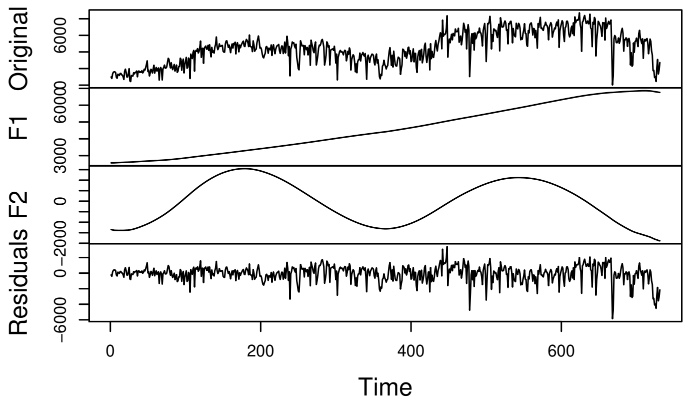


Figure 2: SSA reconstruction of the bike rent data series

Solution

```
# Load SSA package
library(Rssa)
# Bike rent in Washington D.C.
setwd("exercises")
read.csv("bike_rent_daily.csv") -> dat
dat.date <- as.POSIXct(dat$dteday, format = "%Y-%m-%d", tz = "GMT")
# 2 years, daily
# From 01-01-2011 to 31-12-2012
dat$cnt -> X
X <- ts(X)
plot(dat.date,X,t="l")
s <- ssa(X)
plot(s,numvalues = 10,main="",ylab="Eigenvalue norms")
plot(s,type="vectors",idx=1:10,main="")
r <- reconstruct(s, groups = list(1, c(2,3)))
```

```
plot(r,main="")
```

---

7 Exercises of Chapter 8: Markov Chain Monte Carlo

Exercise 8.1

Familiarize yourself with the Monte Carlo method in R. By means of `rnorm` sample N numbers (with N = 100, 1000, 10000) from a normal distribution with mean $\mu = 1.5$ and standard deviation $\sigma = 0.5$, and compare the computed average and standard deviation with μ and σ. Use the density 'counterpart' `dnorm` of the random number generator to compute and plot the theoretical density, and superimpose it to the sample density (plotted by the `hist` command).

Solution

```
# uncomment one of the following definitions of N
# N <- 100
# N <- 1000
N <- 10000
# computed mean and standard deviation
c(mean(sim),sd(sim))
# probability density
hist(sim,xlab="",main="",prob=T,ylim = c(0,1))
curve(dnorm(x,mean=mu,sd=sigma),add=TRUE,col="red")
```

The mean and standard deviation change with N (uncomment the desired line) and, of course, with the simulation session. A possible outcome for the couple (μ, σ) with increasing N is the following:
$N = 100 \rightarrow (1.4703684,\ 0.5187212)$
$N = 1000 \rightarrow (1.4709846,\ 0.4988473)$
$N = 10000 \rightarrow (1.4970172,\ 0.5070764)$

Exercise 8.2

The numerical computation of multidimensional integrals can be a quite demanding task. In addition to the computation time, that can become prohibitive, the geometrical definition of the boundaries can be extremely complicated if the domain of integration is not of simple geometry, but rather a complicated or irregular domain. A Monte Carlo approach greatly simplifies the task. Random numbers are generated within an area of simple boundaries (for instance a square in a two−dimensional domain, or a cube in a three−dimensional one) that contains the area with complicated boundaries. Then, a method is implemented to discriminate if a generated random point of given coordinates is inside or outside the complicated region.

Write a R code using the Monte Carlo method to compute the volume of the sphere $x^2 + y^2 + z^2 = R^2$. Try to change the order of the discretization to see how that affects the accuracy.

Solution

```
N = 1000
R = 10 # radius
vol = 4/3*pi*R^3

# this function checks if a point is inside the sphere
in_sphere <- function(p) {
  x <- p[1]
  y <- p[2]
  z <- p[3]
  return (x^2 + y^2 + z^2 < R^2)
}

n_vol = 0
for (i in 1:N){
  p = c(R*runif(1),R*runif(1),R*runif(1))
  if (in_sphere(p))
        n_vol <- n_vol + (2*R/(N^0.333))^3
}
(vol)
(n_vol)
```

Exercise 8.3

Modify the last section of `Code_8_1.R` to sample 5000 values from a Gamma distribution with parameters: *shape = 2*, *scale = 0.5*.

(1) Try to reproduce the plot in Fig. 3, where the histogram on data coming from the MCMC simulation (excluded a burn-in of 100 values) is compared to the curve obtained by: `curve(dgamma(x,shape=2,scale=0.5),add=T)`
(2) Generate 5000 samples using the R function *rgamma* and compare the results.
(3) How does the number of discharged values (burn-in) affect the result?
Warning: pay attention to the starting value $x(1)$*, remember the peculiar characteristics of the Gamma distribution!*

Solution

```
mh <- function(nsteps,x0,d,pi){
  x<- numeric()
  x[1]<- x0  # initial state of the chain.  Note:  i=1 corresponds to the time t=0
  for(i in 2:nsteps)  {
```

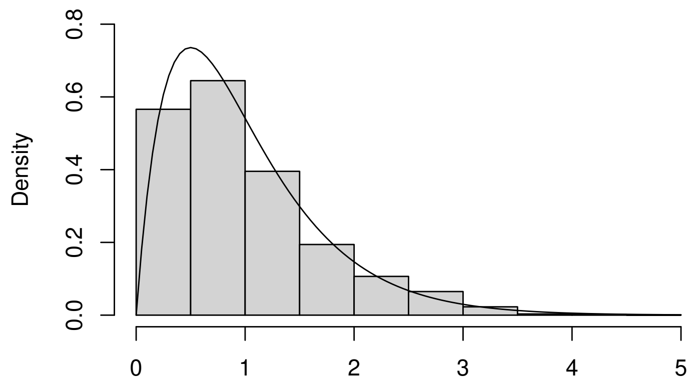


Figure 3: MCMC simulation of a Gamma distribution

```
    # extraction of the candidate y as possible state at the generic time i
    z<-  runif(1,min=-d,max=d)
    y<-x[i-1]+z    # actually this y is y[i-1]

    # will y be the new x[i]?
    alpha<- min(pi(y)/pi(x[i-1]),1)  # alpha: probability of move
    u<- runif(1)  # uniform random number in (0, 1) to check whether  u <= alpha
    if(u <= alpha) x[i]<-  y  else  x[i]<- x[i-1]
  }       # ending loop on the iterations
  return(x)
}  # end function
# target density : gamma
shp<-2
scl<-0.5
target<-function(x){
  dgamma(x,shape=shp,scale=scl)
}
set.seed(245)
nsteps<- 5000
x0<- 2
delta<- 0.75
x<- mh(nsteps,x0,delta,target)

# discard first 100 numbers
hist(x[100:5000],xlab="",main="",prob=T,ylim=c(0,0.8))
curve(dgamma(x,shape=shp,scale=scl),add=T)
# from rgamma
x.rgamma <- rgamma(5000,shape=shp,scale=scl)
hist(x.rgamma,add=TRUE,col="red",prob=T)
```

Exercise 8.4

Sampling by the MCMC algorithm is also allowed from a discrete distribution, with very simple changes to `Code_8_1.R` concerning the target distribution and the proposal one. Suppose, for example, to have to sample from a distribution of 6 integers (from 1 to 6) given by a table like the following:

1	2	3	4	5	6
0.15	0.30	0.20	0.10	0.17	0.08

Obtain 5000 samples for such a distribution and verify that the frequency of the obtained values is consistent with those of the above table.

Hint: change the target distribution to one based on the list `c(0.15,0.30,0.20,0.10,0.17,0.08)` *and use a 'fair' die roll for the proposal distribution*

Solution

```
mh <- function(nsteps,x0,pi){
  x<- numeric()
  x[1]<- x0  # initial state of the chain.  Note:  i=1 corresponds to the time t=0
  for(i in 2:nsteps)  {
    # extraction of the candidate y as possible state at the generic time i
    z <- sample(1:6,1)
    y<-z    # actually this y is y[i-1]

    # will y be the new x[i]?
    alpha<- min(pi(y)/pi(x[i-1]),1)  # alpha: probability of move
    u<- runif(1)  # uniform random number in (0, 1) to check whether  u <= alpha
    if(u <= alpha) x[i]<-  y  else  x[i]<- x[i-1]
  }       # ending loop on the iterations
  return(x)
}  # end function
# target density : not fair dye
dtgt = c(0.15,0.30,0.20,0.10,0.17,0.08)
target<-function(x){
  dtgt[x]
}
# number of steps
nsteps<- 5000
x0<- 1
x<- mh(nsteps,x0,target)

p1 <- sum(x==1)/nsteps
p2 <- sum(x==2)/nsteps
p3 <- sum(x==3)/nsteps
p4 <- sum(x==4)/nsteps
p5 <- sum(x==5)/nsteps
p6 <- sum(x==6)/nsteps
# Display result
c(p1,p2,p3,p4,p5,p6)
```

Exercise 8.5

Run the R code written to solve exercise 8.4 with different number of steps: 100, 1000, 10000. Compute the relative frequencies of the six possible outcomes and compare them in a table with the values describing the discrete target distribution. How do the initial value and the burn-in period affect the results?

Solution

The answer to the first question can be given by running a code like the following, with different *nsteps*.

```
mh <- function(nsteps,x0,pi){
  x<- numeric()
  x[1]<- x0  # initial state of the chain.  Note:  i=1 corresponds to the time t=0
  for(i in 2:nsteps)  {
    # extraction of the candidate y as possible state at the generic time i
    z <- sample(1:6,1)
    y<-z    # actually this y is y[i-1]

    # will y be the new x[i]?
    alpha<- min(pi(y)/pi(x[i-1]),1)  # alpha: probability of move
    u<- runif(1)  # uniform random number in (0, 1) to check whether  u <= alpha
    if(u <= alpha) x[i]<-  y  else  x[i]<- x[i-1]
  }       # ending loop on the iterations
  return(x)
}  # end function
# target density : not fair dye
dtgt = c(0.15,0.30,0.20,0.10,0.17,0.08)
target<-function(x){
  dtgt[x]
}
# number of steps
# nsteps <- 100
# nsteps <- 1000
nsteps <- 10000
# number of steps to be discarded
# change burn-in as you want
nburnin <- 0
x0<- 1
x<- mh(nsteps,x0,target)
# discard burnin
x1 <- x[(nburnin+1):nsteps]

p1 <- sum(x1==1)/nsteps
p2 <- sum(x1==2)/nsteps
p3 <- sum(x1==3)/nsteps
p4 <- sum(x1==4)/nsteps
p5 <- sum(x1==5)/nsteps
p6 <- sum(x1==6)/nsteps
c(p1,p2,p3,p4,p5,p6)
```

A possible result with increasing *nsteps* is the following table:

N	p_1	p_2	p_3	p_4	p_5	p_6
true	0.15	0.30	0.20	0.10	0.17	0.08
100	0.14	0.18	0.30	0.10	0.23	0.05
1000	0.137	0.255	0.189	0.089	0.161	0.069
10000	0.1490	0.3086	0.1922	0.1006	0.1574	0.0722

The answer to the other question is easily obtained by running the above code with different initial value and burn-in period.

---

8 Exercises of Chapter 9: Bayesian inference and stochastic processes

Exercise 9.1

If H_1, H_2, ..., H_n are mutually-exclusive hypotheses, the Bayes theorem asserts that the probability of a particular hypothesis H_i after the verification of an event A is given by:

$$\mathsf{P}\{H_i|A\} = \frac{\mathsf{P}\{H_i\}\,\mathsf{P}\{A|H_i\}}{\mathsf{P}\{A\}}$$

where:

$$\mathsf{P}\{A\} = \sum_{i=1}^{n} \mathsf{P}\{H_i\}\,\mathsf{P}\{H_i|A\}$$

Demonstrate that the Bayes theorem is a corollary of the multiplication probability theorem, stating that the probability (or, also, the probability density) of an event C being the product of two events A and B is given by the product between the probability of A and that of B *conditioned* on A.
Hint: Use the the multiplication theorem to express the probability of every event AH_i

Solution

Denote by $H_1 \dots H_n$ a **complete** set of incompatible hypotheses: this means that one, and only one, of them must be true. For every $i \in [1, n]$, $\mathsf{P}\{AH_i\}$ is the probability that both A and H_i occur, i.e. that the event A occurs and the hypothesis H_i is true.

The multiplication theorem tells us that both the following assertions are true for ani i:

$\mathsf{P}\{AH_i\} = \mathsf{P}\{A\}\,\mathsf{P}\{H_i|A\}$
$\mathsf{P}\{AH_i\} = \mathsf{P}\{H_i\}\,\mathsf{P}\{A|H_i\}$

equating the above expressions we obtain:

$$\mathsf{P}\{A\}\,\mathsf{P}\{H_i|A\} = \mathsf{P}\{H_i\}\,\mathsf{P}\{A|H_i\}$$

which, solved with respect to $\mathsf{P}\{H_i|A\}$, gives:

$$\mathsf{P}\{H_i|A\} = \frac{\mathsf{P}\{H_i\}\,\mathsf{P}\{A|H_i\}}{\mathsf{P}\{A\}}$$

Exercise 9.2

In an archery club two archers A and B shoot an arrow each on the same target. Based on the previous performances of the two archers, we know that the probabilities of centering the target are 0.8 for archer A and 0.5 for archer B.

If only one arrow is stuck in the target board, what is the probability that it comes from archer B?

Hint: enumerate all possible mutually−exclusive hypotheses, for instance "both archers hit the target" etc.

Solution

Following the suggestion, the possible cases are:

- H_1: both archers miss the target
- H_2: both archers hit the target
- H_3: A hits the target, B does not
- H_4: A misses the target, B hits it

The probability of the above hypotheses are:

$\mathsf{P}\{H_1\} = 0.2 \times 0.5 = 0.1$
$\mathsf{P}\{H_2\} = 0.8.5 = 0.4$
$\mathsf{P}\{H_3\} = 0.8 \times 0.5 = 0.4$
$\mathsf{P}\{H_4\} = 0.2 \times 0.5 = 0.1$

Conditioning to the event E: "one arrow in the target", only hypothesis 3 and 4 are compatible, i.e.

$\mathsf{P}\{E|H_1\} = \mathsf{P}\{E|H_2\} = 0$

while

$\mathsf{P}\{E|H_3\} = \mathsf{P}\{E|H_4\} = 1$

Therefore, after E has occurred, we have (from Bayes theorem):

$$\mathsf{P}\{H_3|E\} = \frac{0.4 \times 1}{0.4 \times 1 + 0.1 \times 1} = \frac{4}{5}$$

$$\mathsf{P}\{H_4|E\} = \frac{0.1 \times 1}{0.4 \times 1 + 0.1 \times 1} = \frac{1}{5}$$

There is 80 % probability that the arrow stuck in the target belongs to the first archer.

Exercise 9.3

A typical application of the Bayes formula is in medical diagnostics. Suppose a diagnostic test T for lycanthropy, executed on a blood sample, is 99% effective. This means that if you actually transform into a wolf in full moon nights, test T on your blood has 99% chance to result positive: $\mathsf{P}\{T|H_L\} = 0.99$, where H_L denotes the hypothesis "the tested individual is a lycanthrope".

On the other side, suppose that the probability of a false positive test, $\mathsf{P}\{T|\overline{H_L}\}$ is 10%: you are <u>not</u> a werewolf but the test is anyway positive. Knowing that lycanthropy is (luckily!) a rare condition, let say that its incidence in the population is 1%, if your brother Jack results positive to the test T estimate the probability that you better stay away from him especially in full moon nights.

Solution

Let us define the following:

L = lycanthropy disease
H_L = (hypothesis to be tested) Jack has the disease L
T = (fact) the test for L is positive on Jack

The probability of the hypothesis H_L conditioned on the event T is given by:

$$\mathsf{P}\{H_L|T\} = \frac{\mathsf{P}\{T|H_L\}\,\mathsf{P}\{H_L\}}{\mathsf{P}\{T\}}$$

where: $\mathsf{P}\{T\} = \mathsf{P}\{T|H_L\}\,\mathsf{P}\{H_L\} + \mathsf{P}\{T|\overline{H_L}\}\,\mathsf{P}\{\overline{H_L}\}$, is the unconditional probability of a positive test. The test effectiveness is 99%, or: $\mathsf{P}\{T|H_L\} = 0.99$.

Given the incidence of the disease L in the population (1 %):

$\mathsf{P}\{H_L\} = 0.01$
$\mathsf{P}\{\overline{H_L}\} = 0.99$

The problem states that the probability of a false positive test, $\mathsf{P}\{T|\overline{H_L}\}$ is 10%, i.e. : $\mathsf{P}\{T|\overline{H_L}\} = 0.1$ Therefore we compute:

$$\mathsf{P}\{H_L|T\} = \frac{0.99 \times 0.01}{0.99 \times 0.01 + 0.1 \times 0.99} = 0.09$$

Resulting positive to the test, Jack has 9 % probability of being a Lycanthrope.

Exercise 9.4

With reference to exercise 9.3, what is the probability that Jack is actually a lycanthrope, if the test T on his blood gave a negative result?

Solution

Remember that a test is positive or is negative, no other result. So the probability of a "positive or negative test" is always 1, both if you are a lycanthrope or not. In mathematical terms:

$$\mathsf{P}\{T|H_L\} + \mathsf{P}\{\overline{T}|H_L\} = 1$$

and

$$\mathsf{P}\{T|\overline{H_L}\} + \mathsf{P}\{\overline{T}|\overline{H_L}\} = 1$$

This time we want to compute:

$$\mathsf{P}\{H_L|\overline{T}\} = \frac{\mathsf{P}\{\overline{T}|H_L\}\,\mathsf{P}\{H_L\}}{\mathsf{P}\{\overline{T}\}}$$

It is straightforward to compute:

$$\mathsf{P}\{H_L|\overline{T}\} = \frac{0.9 \times 0.01}{0.01 \times 0.01 + 0.9 \times 0.99} = 0.01$$

Exercise 9.5

A bag contains an unknown number of dice, which you know can be of three types:

C: conventional, with faces enumerated from 1 to 6

E: even, with faces enumerated with even numbers from 2 to 12

R: repeated, two faces enumerated with 1, two with 2, two with 3

You blindly extract a die from the bag, and throw it four times obtaining the sequence 2 2 1 3. Assuming a uniform prior for the three different *hypotheses* ($\mathsf{P}\{H_C\} = \mathsf{P}\{H_E\} = \mathsf{P}\{H_R\} = 1/3$), and knowing the likelihood L (the probability of a given result):

(1) Fill a table like the following describing how the prior is updated after each result of the sequence or, which is the same, reporting the posterior probability after each outcome.

(2) Compute the posterior probability that the die is "C" after the full sequence.

(3) Compute the posterior probability that the die is "E" after the full sequence.

The table could be something like:

		outcome = 2		outcome = 2		outcome = 1		outcome = 3	
H	Prior	ℓ	Post	ℓ	Post	ℓ	Post	ℓ	Post
H_C	1/3	1/6							
H_E	1/3	1/6							
H_R	1/3	1/3							

Hint: the likelihood of a particular outcome is the probability of that result, given the die, and of course it is zero if that outcome is impossible

Solution

The likelihoods of the first outcome $D = 2$ are:

$$\ell_{1C} \equiv \mathsf{P}\{D|H_C\} = 1/6$$
$$\ell_{1E} \equiv \mathsf{P}\{D|H_E\} = 1/6$$
$$\ell_{1R} \equiv \mathsf{P}\{D|H_R\} = 2/6$$

Thus the normalization constant is:

$$\mathsf{P}\{H_C\}\,\mathsf{P}\{D|H_C\}+\mathsf{P}\{H_E\}\,\mathsf{P}\{D|H_E\}+\mathsf{P}\{H_R\}\,\mathsf{P}\{D|H_R\} = \tfrac{1}{3}\tfrac{1}{6}+\tfrac{1}{3}\tfrac{1}{6}+\tfrac{1}{3}\tfrac{1}{3} = \tfrac{2}{9}$$

And the posterior probabilities are:

$\mathsf{P}\{H_C|D\} = 1/4$
$\mathsf{P}\{H_E|D\} = 1/4$
$\mathsf{P}\{H_R|D\} = 1/2$

The above posterior probability become the new *priors* before the second outcome. Applying the same procedure to the following outcomes $D = 2$, $D = 1$ and $D = 3$ we eventually obtain the following table.

		outcome = 2		outcome = 2		outcome = 1		outcome = 3	
H	Prior	ℓ	Post	ℓ	Post	ℓ	Post	ℓ	Post
H_C	1/3	1/6	1/4	1/6	1/6	1/6	1/9	1/6	1/17
H_E	1/3	1/6	1/4	1/6	1/6	0	0	0	0
H_R	1/3	1/3	1/2	1/3	2/3	1/3	8/9	1/3	16/17

This result is consistent with the intuitive consideration that the outcome of numbers between "1" and "3", once the impossible hypothesis has been removed, is more probable on the third die that in the first one.

Exercise 9.6

How the results of exercise 9.5 change if you use a different prior? For example, suppose you are almost certain that dice of the "E" type are not present in the bag, and someone told you that there are presumably twice conventional dice with respect to repeated ones.

Solution

Suppose to translate "almost impossible" with probability 1/100. The initial priors are:

$prH_C = 66/100$
$\mathsf{P}\{H_E\} = 1/100$
$\mathsf{P}\{\}\, H_R = 33/100$

while the initial likelihoods remain unchanged:

$\ell_{1C} \equiv \mathsf{P}\{D|H_C\} = 1/6$
$\ell_{1E} \equiv \mathsf{P}\{D|H_E\} = 1/6$
$\ell_{1R} \equiv \mathsf{P}\{D|H_R\} = 2/6$

The procedure is the same of exercise 9.5.

Exercise 9.7

With reference to the bus waiting-time of section 9.3, modify the R code to compute:

(1) The probability $p(\lambda_B > \lambda_A)$ for the given simulated data.

(2) The quantiles of the empirical probability distribution of $\lambda_B / \sum \lambda_i$

Solution

Add to Code_9_10.R the following code chunks.

Question (1)

```
greater <- lam2>lam1
# p(lam2>lam1)
sum(greater)/50000
```

The result is: 0.28862.

Question (2)

```
empirical_B <- lam2/(lam1+lam2+lam3)
quantile(empirical_B)
```

The result is:

```
         0%         25%         50%         75%        100%
0.04589579 0.22138145 0.27654098 0.33518579 0.65055859
```

Exercise 9.8

With reference to exercise 9.7 how do things change assuming a uniform prior probability instead of $1/\lambda$?

Solution

From Section 9.3, che choice of an uniform prior brings to a posterior that is $Gam(n+1, t)$. Therefore, Code_9_10.R only needs to be modified in the section defining the prior, as follows:

```
set.seed(111)
lam1 <- rgamma(50000,N1+1,T1)
set.seed(222)
lam2 <- rgamma(50000,N2+1,T2)
set.seed(333)
lam3 <- rgamma(50000,N3+1,T3)
```

The results are not so different from those obtained with the Jeffreys prior:

0.29544

```
        0%        25%        50%        75%       100%
0.04627538 0.22826246 0.28018968 0.33636804 0.70986088
```

As it was presumable (and desirable) the result does not depend on the choice of the prior.

Exercise 9.9

The JAGS language is a “natural” extension of R for MCMC calculations. It is therefore useful to be acquainted with it. With reference to section B.0.2 of Appendix B write an R code to compare normally-distributed random data generated in R with a sample extracted in JAGS, for example comparing the two histograms.

Note: despite the identical name, `dnorm` *in JAGS is defined in terms of the precision instead of the standard deviation*

Solution

Here is the code.

```
# in R
N <- 10000
mu <- 2.4
sigma <- 1.8
Z <- rnorm(N,mu,sigma)
par(mfrow=c(1,2))
hist(Z,col="red",main="",xlab="in R")
summary(Z)

# in JAGS
model <- textConnection("model {
      Y ~ dnorm(Mu,Prec);
}")

# Init
jags.inits <- NULL
# Data
jags.data <- list("Mu"=mu,"Prec"=1/sigma^2)
# Perform Bayesian analysis using JAGS
N <- 10000
model <- jags.model(model, data=jags.data, inits=jags.inits, n.chains=1)
update(model, n.iter=N)

# Parameters
jags.params <- c("Y") # parameters to be monitored
```

```
samples <- coda.samples(model, jags.params,N)

out <- do.call(rbind.data.frame, samples)
names(out)
summary(out$Y)
hist(out$Y,col="blue",xlab="in JAGS")

c(summary(Z),summary(out$Y))
```

Exercise 9.10

We have treated the problem of a fair coin at the beginning of this Chapter. Suppose you want to decide about the fairness of a coin by an experiment consisting in counting the number of heads obtained in n flips. You know that the likelihood $P(\theta|h)$, the probability of obtaining h heads in n flips is the $Binomial(n,h)$ given the probability θ of a head:

$$P(\theta|h) = \left(\begin{array}{c} n \\ h \end{array} \right) \theta^h (1-\theta)^{n-h}$$

Your prior expresses the information you have about the coin. If you have no reason to think the coin is fair, you can assume a uniform prior or, which is the same, a $Beta(1,1)$ distribution. If you think the coin is fair, you assign a more "informative" prior, such as $Beta(k,k)$ with $k > 1$. The greater is k, the narrower is the prior distribution around the value $\theta = 0.5$.

Obtain the posterior distribution as a function of k and write down an R code to compute and plot likelihood, prior and posterior for $n = 100$ and $h = 40$. Maintaining the same ratio h/n, comment how things change if $n = 10$ or $n = 1000$.

Solution

The code can be like the following.

```
n <- 100
h <- 40
k <- 1
lik <- n*dbinom(0:n,n,h/n)
prior <- dbeta(seq(0,1,length=n+1),k,k)
post <- dbeta(seq(0,1,length=n+1),k+h,n-h+k)

p <- seq(0,1,length=n+1)
plot(p,post/max(post),t="l",col="black")
lines(p,prior,col="red")
lines(p,lik/max(lik),col="green",lty=2)
# line type for plots
tpl <- 2
# loop from k=5 to k=30
for (k in seq(5,30,5)){
  prior <- dbeta(seq(0,1,length=n+1),k,k)
  post <- dbeta(seq(0,1,length=n+1),k+h,n-h+k)

  p <- seq(0,1,length=n+1)
  lines(p,post/max(post),col="black",lty=tpl)
  tpl <- tpl+1
  lines(p,prior/max(prior),col="red",lty=tpl)
}
```

For the purpose of comparing likelihood, prior and posterior distributions it is convenient to nomalise them to their maximum values. From the computation we see that our belief about the fairness of the coin, measured by the order the *Beta* function, pushes the posterior towards a probability having a maximum in p=0.5. But, to actually have p=0.5 we must really be convinced about the fairness, otherwise a result of 40 heads over 100 coin flips suggests an unfair one.

Concerning the final question, the total number of flips intuitively influences our conclusions about fairness: if you get 4 heads in 10 flips you cannot say much about the actual fairness, but if you get 400 heads in 1000 flips, fairness appears to be rather improbable. The following code shows that for the choice of a prior $Beta(20, 20)$, i.e. one reflecting a strong belief about the fairness of the coin.

```
# Very few coin flips
n <- 10
h <- 4
k <- 20
lik <- n*dbinom(0:n,n,h/n)
prior <- dbeta(seq(0,1,length=n+1),k,k)
post <- dbeta(seq(0,1,length=n+1),k+h,n-h+k)
p <- seq(0,1,length=n+1)
plot(p,post/max(post),t="l",col="black")
lines(p,prior/max(prior),col="red")
lines(p,lik/max(lik),col="green",lty=2)

# Large number of coin flips
n <- 1000
h <- 400
k <- 20
lik <- n*dbinom(0:n,n,h/n)
prior <- dbeta(seq(0,1,length=n+1),k,k)
post <- dbeta(seq(0,1,length=n+1),k+h,n-h+k)
p <- seq(0,1,length=n+1)
plot(p,post/max(post),t="l",col="black")
lines(p,prior/max(prior),col="red")
lines(p,lik/max(lik),col="green",lty=2)
```

Plotting the results of the two simulations you see that in the first case (4 head in 10 flips) the small number of flips is not sufficient to change our mind: the posterior practically coincides with the prior. In the second case (400 heads in 1000 flips) also if our prior belief was about fairness, the occurrence of a proportion of 2/5 heads pushes the posterior to coincide with the likelihood, i.e. data always win.

Exercise 9.11

Repeat exercise 9.10 using JAGS. The JAGS model to be used in `rjags` is something like

```
H ~ dbin(theta,N)
theta ~ dbeta(k,k)
```

to be inserted into a `textConnection` command or to be read from a text file. Here H is the number of heads, whose distribution identifies the likelihood, and *theta* is the searched probability.

Experiment by changing the number of *burn–in* steps (using the rjags function `update`) and the number of extracted samples (via the `coda.samples` function).

Plot the posterior distribution and compare it to the analytical one obtained in the previous exercise.

Solution

We do not repeat the whole exercise 9.10, rather we limit ourselves to write down the code for simulating the problem in JAGS.

```
library(rjags)

mod <- textConnection("model {
  H ~ dbin(theta,N)
  # uniform prior
  # theta ~ dbeta(1,1)
  # Alternatively, belief in coin fairness:
  theta ~ dbeta(20,20)
}")

n <- 100
h <- 40
jags.data <- list("N"=n,"H"=h)
jags.params <- c("theta") # parameters to be monitored
jags.inits <- list(theta = 0.5)

model <- jags.model(mod, data=jags.data, inits=jags.inits, n.chains=1)
update(model, n.iter=4000)
samples <- coda.samples(model, jags.params,10000)
# transform list into dataframe
tmp <- do.call(rbind.data.frame, samples)
tmp.hist <- hist(tmp$theta,prob=TRUE,breaks=20)
plot(tmp.hist$mids,tmp.hist$density/max(tmp.hist$density),t="l")
# most probable value
mean(tmp$theta)
```

Exercise 9.12

This problem concerns with a probability puzzle based on a (in)famous television game, known as Monty Hall. There are three doors A, B, C. Behind one door is a car, behind the others, goats. The contestant picks a door, say A, which remains closed. The host, who knows what's behind the doors, opens another one of the remaining doors, say C, where he knows there is a goat. The host asks the contestant ”Do you want to maintain your first choice A or change your mind and pick the door B?”. The answer is not indifferent, even though at a first sight it seems so. Prove via Bayes theorem that it is advantageous for the contestant to switch his choice. Try also to simulate the game.

Solution

If I switch or not the door, I have to compute the probabilities of the events $\mathbf{P}(B)$ (the car is behind B) and $\mathbf{P}(\bar{B})$ (the car is not behind B).

Let $\mathbf{P}(B)$ be the probability of a good choice, that is I have the car: $\mathbf{P}(A) = 1/3$, while $\mathbf{P}(\bar{A})$ is the probability of a bad choice. Recall the the law (or formula) of total probability with two events:

$$\mathbf{P}(A) = \mathbf{P}(A|B)\,\mathbf{P}(B) \,+\, \mathbf{P}(A|\bar{B})\,\mathbf{P}(\bar{B})$$

or also:

$$\mathbf{P}(B) = \mathbf{P}(B|A)\,\mathbf{P}(A) \,+\, \mathbf{P}(B|\bar{A})\,\mathbf{P}(\bar{A}) \,=\, 0 \times 1/3 + 1 \times 2/3 = 2/3$$

Indeed, if the car is behind A, the probability that it is behind B is 0. If the car is not behind A, then it is surely in B.

If I do not switch the door, then:

$$\mathbf{P}(\bar{B}) = \mathbf{P}(\bar{B}|A)\,\mathbf{P}(A) \,+\, \mathbf{P}(\bar{B}|\bar{A})\,\mathbf{P}(\bar{A}) \,=\, 1 \times 1/3 + 0 \times 2/3 = 1/3$$

The probability is halved.

Code to search for convergence to probability = 2/3.

```
####  To simulate the Monty Hall game
doors<-c(1:3)
prob<-rep(1/3,3)
ris<- rep(0,30000)
set.seed(1)
# I suppose that door n. 1 is always with the car
for(i in 1:length(ris)){
door<-sample(doors,1,prob,replace=T)
newscat<-numeric()
if  (door==1)
{newdoor=2}
if (door==2)
{newdoor=1}
if (door==3)
{newdoor=1}
if (newdoor==1)
{ris[i]=1
} else {ris[i]=0}
door<-0
newdoor<-0
}
p<-sum(ris)/length(ris)
p
#   let us look at the convergence
freq<- 0
for(i in 1:length(ris))   {
freq[i]<- mean(ris[1:i])}
ns <- seq(1,length(ris),1)
plot(ns,freq,ylim=c(0.6,0.7),xlab='no. games',ylab='frequency',type="l",lwd=1)
abline(2/3,0,col="red",lty=4,lwd=2)
```

---

9 Exercises of Chapter 10: Genetic Algorithms: an EvolutionaryBased Global Random Search

Exercise 10.1

The following is a toy-problem, obviously not needing GA to be solved. Suppose you want to find the positive root of the polynomial:

$$f(x) = x^2 - 4x - 12$$

i.e. the value $x = 6$.

Your population consists of binary strings of 4 bits. The solution is, of course, the string `0110`. Applying by hand a very rough GA, without mutation and with fixed crossover point in the middle of the chromosome, suppose you have obtained `0101` as the fittest individual.

Remembering the mating procedure introduced in Section 10.3 which, given two individuals $X_1X_2X_3X_4$ and $Y_1Y_2Y_3Y_4$, generates the new chromosomes $X_1X_2Y_3Y_4$ and $Y_1Y_2X_3X_4$, how many chromosomes (among the 16 possible ones) would give rise to the proper solution through crossover and mating with `0101`?

Solution

The chromosomes are all strings `**10`, i.e. their number is 4 over the total of 16.

Exercise 10.2

With reference to exercise 10.1, define a fitness function suitable to be maximized such, for instance:

$$fitness(x) = 1/(10 + abs(x^2 - 4x - 12))$$

Write an R code to compute the fitness value of all possible chromosomes, and plot it.

Solution

The code is:

```
# solution
x <- 6
fit.true <- 1/(10+abs(x^2-4*x-12))
fit.true
# Fitness of all possible chromosomes
fitt <- rep(0,16)
for (i in 1:16)
  fitt[i] <- 1/(10+abs((i-1)^2-4*(i-1)-12))
fitt
```

Exercise 10.3

With reference to exercise 10.2, implement a real-coded genetic algorithm based on the GA R library, taking the example in Section 10.5.4 as a starting point. Begin with a population of 10 individuals, 100 generations, cross-over and mutation probabilities of 0.7 and 0.1 respectively, then play with numbers to see how the values of the four parameters affect the result. Compare the fitness histories for 10, 100 and 1000 individuals with a proportional number of generations, using `GA@summary[,1]`.

Solution

```
# Initialize
pop.size<- 10
max.gen <- 100
xover.prob <- 0.7
mut.prob <- 0.1
# Define fitness function
f <- function(x)
{
  x.val <- x^2-4*x-12

  1/(10+abs(x.val))
}
# GA solution
library(GA)

GA <- ga(type = "real-valued", fitness = f, lower = 0.0, upper = 10.0, maxiter=max.gen,popSize = pop.size)
ga.sol <- unname(GA@solution)
ga.sol

plot(GA@summary[,1],t="l",xlab="Generation",ylab="Fitness")
```

Exercise 10.4

Download the data for the example in Section 10.5.4 and modify the fitness function. Does the result significantly change if fitness is computed in terms of the absolute residuals instead of the residual sum of squares? How convergence is affected, if any, by the choice of the fitness function?

Solution

```
# init
pop.size<- 100
max.gen <- 200
xover.prob <- 0.7
mut.prob <- 0.1
# data
setwd("C:/RPA/code/genetic_algorithm/data")
dat <- read.table("exp_data.txt")
X <- dat$V1
Y <- dat$V2
# fitness function
f <- function(x)
{
  a <- x[1]
  b <- x[2]
  c <- x[3]
  N <- length(X)
  ret <- 0
  for (i in 1:N)
    #ret <- ret + (Y[i] - a*exp(-b*X[i])+c)^2
    ret <- ret + abs(Y[i] - a*exp(-b*X[i])+c)
  1/ret
}
# test
fexp <- function(x,p)
{
  a <- p[1]
  b <- p[2]
  c <- p[3]

  a*exp(-b*x)+c
}
# GA
library(GA)
GA <- ga(type = "real-valued", fitness = f, lower = c(0.,0.,0.), upper = c(1,1,1),maxiter=max.gen,popSize = pop.s
ga.sol <- unname(GA@solution)
ga.sol
plot(X,Y)
curve(fexp(x,ga.sol),from=0,to=10,add=TRUE,col="red",lty=2)

plot(GA@summary[,1],t="l",xlab="Generation",ylab="Fitness")
```

Exercise 10.5

Following the lines of the previous exercises, write a GA code for fitting data generated by the following:

```
# Generate random signal, composition of sine and cosine
t <- seq(0,10,1/10)
A1 <- runif(1)
A2 <- runif(1)
T1 <- 1
T2 <- 0.5
f1 <- runif(1)/T1
f2 <- runif(1)/T2

s <- A1*sin(2*pi*f1*t)+A2*cos(2*pi*f2*t)
plot(t,s,t="l",xlab="t (s)",ylab="s(t)")
```

where the functional form of the signal s is supposed to be known:

$$s(t) = A_1 sin(2\pi f_1 t) + A2 cos(2\pi f_2 t)$$

but A_1, A_2, f_1, f_2 are unknown.

Write down the fitness function, and try to obtain a solution with a population of 100 chromosomes, with 100 generations. Repeat several times the whole code, to study how well different sine/cosine compositions can be managed.

Hint: we have four parameters, A_1, A_2, f_1, f_2, instead of the three in the code of Section 10.5.4. The fitness function must be accordingly modified.

Solution

```
# Generate random composition of sin and cos
t <- seq(0,10,1/10)
A1 <- runif(1)
A2 <- runif(1)
T1 <- 1
T2 <- 0.5
f1 <- 1/T1*runif(1)
f2 <- 1/T2*runif(1)

s <- A1*sin(2*pi*f1*t)+A2*cos(2*pi*f2*t)

plot(t,s,t="l",xlab="t (s)",ylab="s(t)")
# init
pop.size<- 100
max.gen <- 100
xover.prob <- 0.7
mut.prob <- 0.1
# data
X <- t
Y <- s
# fitness function
f <- function(x)
{
  A1 <- x[1]
  A2 <- x[2]
  f1 <- x[3]
  f2 <- x[4]

  N <- length(X)
  ret <- 0
  for (i in 1:N){
    Ycalc <- A1*sin(2*pi*f1*X[i])+A2*cos(2*pi*f2*X[i])
    ret <- ret + (Y[i] - Ycalc)^2
  }
  1/ret
}
# test
ftest <- function(t,p)
{
  A1 <- p[1]
  A2 <- p[2]
  f1 <- p[3]
  f2 <- p[4]

  A1*sin(2*pi*f1*t)+A2*cos(2*pi*f2*t)
}
# GA
library(GA)
GA <- ga(type = "real-valued", fitness = f, lower = c(0.,0.,0.01/T1,0.01/T2), upper = c(1,1,1/T1,1/T2),maxiter=ma
ga.sol <- unname(GA@solution)
ga.sol
plot(X,Y,t="l")
curve(ftest(x,ga.sol),from=0,to=10,add=TRUE,col="red",lty=2)

plot(GA@summary[,1],t="l",xlab="Generation",ylab="Fitness")
```

Exercise 10.6

Section 10.7 shows how a quasi−optimal solution can be obtained for 15 cities. In a simpler case, say 5 cities including the starting one, the possible routes can be enumerated and all the relative distances can be computed. Modify the code to generate only 5 cities and compare the result of the heuristic algorithm to the true best solution. Write a simple R code to make that work.

Solution

```
library(GA)

# 5 cities: 1 2 3 4 5
distances <- matrix(rep( 0, len=25), nrow = 5)
# city coordinates
coords <- matrix(nrow=5,ncol=2)
coords[1,] <- c(1,5)
coords[2,] <- c(2,4)
coords[3,] <- c(4,7)
coords[4,] <- c(3,1)
coords[5,] <- c(5,4)
```

```
for (i in 1:5){
  for (j in 1:5){
    distances[i,j] <- sqrt((coords[i,1]-coords[j,1])^2+(coords[i,2]-coords[j,2])^2)
  }
}
# Plot cities
plot(0:10,0:10,type="n")
for (i in (1:5))
  text(coords[i,1],coords[i,2],labels=i,col="red")

# route example: 1 2 3 4 5
route <- c(1,2,3,4,5)
for (i in 1:4)
  segments(coords[route[i],1],coords[route[i],2],coords[route[i+1],1],coords[route[i+1],2],col="green")
segments(coords[route[5],1],coords[route[5],2],coords[route[1],1],coords[route[1],2],col="green",lty=2)

myfun <- function(x) {
    dist.tot <- 0
    N <- length(x)
    for (i in 1:(N-1))
      dist.tot <- dist.tot + distances[x[i],x[i+1]]
    dist.tot <- dist.tot + distances[x[N],x[1]]
    1/dist.tot
}

myfitness <- function(x) {
    N <- length(x)
    dist.tot <- distances[1,x[1]]
    for (i in 1:(N-1))
      dist.tot <- dist.tot + distances[x[i],x[i+1]]
    dist.tot <- dist.tot + distances[x[N],1]
    1/dist.tot
}

# compute ALL possible routes
permutations <- function(n){
    if(n==1){
        return(matrix(1))
    } else {
        sp <- permutations(n-1)
        p <- nrow(sp)
        A <- matrix(nrow=n*p,ncol=n)
        for(i in 1:n){
            A[(i-1)*p+1:p,] <- cbind(i,sp+(sp>=i))
        }
        return(A)
    }
}

routes <- permutations(4)+1
routes <- cbind(1,routes)

# fitness values
fitval <- rep(24,0)
for (i in 1:24)for (i in 1:4)
  segments(coords[route[i],1],coords[route[i],2],coords[route[i+1],1],coords[route[i+1],2],col="green")
segments(coords[route[5],1],coords[route[5],2],coords[route[1],1],coords[route[1],2],col="green",lty=2)

fitval[i] <- myfun(routes[i,])

# True best route
route <- routes[which.max(fitval),]
for (i in 1:4)
  segments(coords[route[i],1],coords[route[i],2],coords[route[i+1],1],coords[route[i+1],2],col="red")
segments(coords[route[5],1],coords[route[5],2],coords[route[1],1],coords[route[1],2],col="red",lty=2)

# Compute best route via GA
tmp <- ga(type = "permutation", fitness = myfitness, lower = 2, upper = 5,
    seed = 123, elitism = 1, maxiter = 100, popSize = 100)


route.best <- unname(tmp@solution)
route.best <- c(1, route.best[1,])

for (i in 1:4)
  segments(coords[route.best[i],1],coords[route.best[i],2],coords[route.best[i+1],1],coords[route.best[i+1],2],co
```

---

10 Exercises of Chapter 11: The Problem of Accuracy

Exercise 11.1

Write a code to estimate by means of the bootstrap the standard error of the standard error both plug−in and correct.

Solution

```
# standard error of standard error (plug-in)

set.seed(1)
thb<-numeric()
B<-1400
x<-c(80,82,78,81,80,79,84)
nx<-length(x)
x_bar<-mean(x)
x_bar
se_mean<- sd(x)/sqrt(length(x)) # correct standard error
# se_mean<-sum((x-x_bar)^2/(nx*(nx-1)))^(1/2) # explicit formula
# the plug-in standard error by the function:
sig2.pl<-function(x){var(x)*(length(x)-1)/length(x)}
se_pl<-sqrt(sig2.pl(x)/length(x)) # plug-in stand. error
se_pl
for(b in 1:B){
xb<-sample(x,length(x),replace=TRUE)
# thb[b], bootstrap replications
# standard error estimated on the sample
thb[b]<-sqrt(sig2.pl(xb)/length(xb))
```

```
             }
seb<-sqrt(var(thb))
mthb<-mean(thb)
seb
mthb

hist(thb) # histogram if whished
quantile(thb, probs = c(0.025, 0.1587, 0.5, 0.8413, 0.975),qua = TRUE)

#------------------------------------------------

# standard error of standard error (correct)

set.seed(1)
thb<-numeric()
B<-1400
x<-c(80,82,78,81,80,79,84)
nx<-length(x)
# correct stand. error estimated on the observed sample
th<- sd(x)/sqrt(length(x))
th
for(b in 1:B){
xb<-sample(x,length(x),replace=TRUE)
thb[b]<-sd(xb)/length(x)           }
seb<-sqrt(var(thb))
mthb<-mean(thb)
seb
mthb
hist(thb) # histogram if whished
quantile(thb, probs = c(0.025, 0.1587, 0.5, 0.8413, 0.975),qua = TRUE)
```

Exercise 11.2

Demonstrate the expression in eqn. (11.7).
Hint: use the operator notation for the variance of $\hat{\sigma}^2_{bm}$

Solution

For simplicity of notation, we can derive the expression of the variance of the sample variance of a random variable X, in other words, we apply the variance operator to S^2 given by:

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \hat{\mu})^2$$

explicitly, we aim to determine:

$$v_S = \text{Var}\left[S^2\right]$$

and obtain from that the standard deviation $s_S = \sqrt{v_S}$. It is clear that the same procedure (with a heavier formalism) applies to the standard error of $\hat{\sigma}_{bm}$

defined in Section 11.3.

Note that by definition:

$$\begin{aligned}\mathrm{Var}\left[S^2\right] =& \frac{1}{N(N-1)}\sum_{i=1}^{N}\mathrm{Var}\left[(X_i-\hat{\mu})^2\right] = \\ & \frac{1}{N(N-1)}\sum_{i=1}^{N}\left(\mathrm{E}\left[(X_i-\hat{\mu})^4\right]-\mathrm{E}\left[(X_i-\hat{\mu})\right]^2\right) = \\ & \frac{1}{N(N-1)}\left[ND_4 - N\sigma^4\right] = \frac{1}{N-1}\left(D_4-\sigma^4\right)\end{aligned}$$

with the fourth moment defined in Section 11.3.

Now, it can be demonstrated that the variance of a function $f(Y)$ of a random variable Y is approximately given by the Taylor expansion of f around $\mathrm{E}[Y]$:

$$\mathrm{Var}\left[f(Y)\right] \approx \left(f'(E[Y])\right)^2 \mathrm{Var}\left[Y\right]$$

Therefore, in our case posing $Y = S^2$ and $S = \sqrt{Y}$ (in other words $f() \equiv \sqrt{()}$) we have:

$\mathrm{E}\left[Y\right] = \mathrm{E}\left[S^2\right] = \sigma^2$

$f'(E[Y]) = \frac{1}{2\sqrt{\mathrm{E}[S^2]}} = \frac{1}{2}(\sqrt{\sigma^2})^{-2}$

and, eventually:

$$\mathrm{Var}\left[S\right] \approx \frac{D_4-\sigma^4}{4(N-1)\sigma^2}$$

the standard deviation of S is the square root of the above expression.

Exercise 11.3

Write a code to compare the distribution of means of a set of trajectories with that of a unique trajectory but obtained with bootstrap.

Solution

```
###### distribution of means
###### comparison between a set of trajectories
######                and one trajectory with bootstrap

# population distribution: # normal standard
plot(function(x) dnorm(x),-4,4,lty=1,col="black",ylim=c(0,0.5),
font.lab=3,cex.lab=1.2,lwd=2,ylab="normal distribution")
```

```
set.seed(1)
n<- 100
# simulating random variates having a normal distribution
x1<-rnorm(n)   # "observed"  data set
m1<- mean(x1)
m1
sd(x1)
hist(x1,10,freq=F,add=T,font.lab=3,cex.lab=1.2,lwd=1.2 )
segments(m1,0,m1,0.5,lty=3,lwd=3,col="black") # marks the mean m1
# extracting a further sample with a different seed
set.seed(2)    # new seed
n<- 100
x2<-rnorm(n)
m2<- mean(x2)
m2
sd(x2)
# uncomment to overlap the new histogram
#hist(x2,10,freq=F,add=T,border="blue")
#segments(m2,0,m2,0.5,lty=3,lwd=3,col="blue")
# further sample again
set.seed(3)   # new seed
n<- 100
x3<-rnorm(n)
m3<- mean(x3)
m3
sd(x3)
# # uncomment to overlap the new histogram
#hist(x3,10,add=T,freq=F,lty=3,border="magenta")
#segments(m3,0,m3,0.5,lty=3,lwd=3,col="magenta")

# as expected the means are different:
# m1 = 0.1088874, m2 =  -0.03069816, m3 = 0.01103557

# the distribution of the means is required
# Nsample of dimension n are extracted
# the mean of each of them is saved

set.seed(1)
y<- numeric()
n<- 100
Nsample<- 1000
for(i in 1:Nsample) {
x<- rnorm(n)
y[i]<- mean(x)
                  }
my<- mean(y)
my
sd(y)
# new plot
hist(y,10,freq=F,font.lab=3,cex.lab=1.2,lwd=1.2,main=" " )
segments(my,0,my,5,lty=3,lwd=3,col="black") # marks the mean my
# the distribution has a normal shape, as expected from the
# central limit theorem. See also:
# function giving a normal QQ plot of the values in y.
qqnorm(y)
#adding a line to a normal QQ plot through the 1 and 3 quartiles.
qqline(y)
```

```
###################### BOOTSTRAP #################################

###  B from 100 to 1000 (Nsample above is = 1000)
# the unique ("observed") sample is resampled B times
set.seed(1)
thb<-numeric()  # vector of bootstrap replications
B<-100          # number of replications
z<- x1  # "observed" data set
nz<-length(z)
mean.oss<- mean(z)
mean.oss       # it must be = m1
sd.oss<- sd(z)
sd.oss
for(b in 1:B)           {   # bootstrap loop
# sample(...) extracts samples (with reintroduction) from vector z
zb<-sample(z,length(z),replace=TRUE)
thb[b]<- mean(zb)    #  bootstrap replications
                                      }   # ending bootstrap loop

seb<-  sd(thb)        # bootstrap stand.error
mthb<- mean(thb)      # mean of the bootstrap replications
mthb
seb

hist(y,10,freq=F,font.lab=3,cex.lab=1.2,lwd=1.2,main=" " )
segments(m1,0,m1,5,lty=3,lwd=3,col="black")
segments(mthb,0,mthb,6,lty=2,lwd=3,col="black")

qqnorm(thb)
qqline(thb)

# mean of the observed sample:          m1=0.1088874
# mean of the distribution of the means: y=-0.002244083
# bootstrap mean:                      mthb=0.1113563

# sd of the distribution of the means sd(y)=0.09670521
# in agreement with seb=0.09092471
# if B=1000, we see that mthb and seb chance a little
# even though the shape is nearer to a normal
# (mthb= 0.1115907 e seb=0.08590306)
```

Exercise 11.4

Write a code to assign the standard error to the estimate by the batch means method and by the Moving Block Bootstrap method. The target density of the MCMC algorithm is the standard normal.

Solution

```
####  Algorithm M(RT)^2
##    target density : standard normal
##    to assign the standard error to the estimate
```

```
burn.in<-1000
mh <- function(nsteps,x0,d,pi){
x<- numeric()
x[1]<- x0  # initial state of the chain
t<-0
#Note:  i=1 corresponds to the time t=0
for(i in 2:nsteps)  {
# extraction of the candidate y as possible state at the generic time i
z<-  runif(1,min=-d,max=d)
y<-x[i-1]+z    # actually this y is y[i-1]
# will y be the new x[i]?
alpha<- min(pi(y)/pi(x[i-1]),1)  # alpha: probability of move
u<- runif(1)  # to check whether  u <= alpha
if(u <= alpha) x[i]<-  y  else  x[i]<- x[i-1]
                           }       # ending loop on the iterations
return(x)
                                    }  # end function
# target density : standard normal
mu<-0
sigma<-1
target<-function(x){
dnorm(x,mean=mu,sd=sigma)
}
set.seed(11)
nsteps<- 11000
x0<- -10
delta<- 0.75
x<- mh(nsteps,x0,delta,target)
ta<- burn.in
tb<- length(x)
lt<- length(x[ta:tb] )
mtemp<-  mean(x[ta:(tb-1)] )
mtemp
se.temp<-  sqrt(var(x[ta:tb])/lt)
se.temp
hist(x[ta:(tb-1)],freq=F,ylim=c(0,0.5),
cex.lab=1.2,cex.main=1.1,,main="",font.lab=3)
# freq=F to compare with analytical density by 'curve(.)')
curve (dnorm(x, mean=mu, sd=sigma),add=T,lty=2,col="black",lwd=2)

### ==================  batch means  ================

mbatch<- numeric()
se.batch<- numeric()
se.se<- numeric()
amp<- numeric()
mj<- numeric()
## the interval [ta,tb] is divided in batches
## each batch has h observations, i.e., h = batch size
h<-  c(800,600,500,400,300,200,100,50,40,30,20,15,12,10)
# one can choose the initial value of l (lbmi) and the final one (lbmf)
hi<- 1
hf<- 14
amp<- floor((tb-ta)/h )      # batches length
tb
ta
```

```
amp
# starting most external loop on the batch length
for (l in hi:hf)  {
amp[l]
j<- ta-amp[l]
for (k in 1:h[l])     {     # loop on h batches
j<- j+amp[l]
mj[k]<-mean(x[j:(j+(amp[l]-1))])   # mean in each batch
                      }     # end loop on h batches
mbatch[l]<-mean(mj[1:h[l]])
se.batch[l]<- sd(mj[1:h[l]])/sqrt(h[l])
se.se[l]<- se.batch[l]/sqrt(2.*(amp[l]-1))
                  }     # ending more external loop
# summary of quantities in [lbmi:lbmf]
h[hi:hf]
amp[hi:hf]
mbatch[hi:hf]    #  check: they must be all equal
se.batch[hi:hf]
se.se[hi:hf]
par(mai=c(1.02,1.,0.82,0.42)+0.1)
plot(amp[hi:hf],se.batch[hi:hf], type="b",
xlab="batch size",ylim=c(0.02,0.06),
ylab=expression(hat(sigma)[bm]),
font.lab=3,lwd=2,cex.lab=1.2)
arrows(amp,se.batch, amp,se.batch+se.se, length=0.05, angle=90)
arrows(amp,se.batch, amp,se.batch-se.se, length=0.05, angle=90)

### ===================  MBB ================

sig2<-function(x){var(x)}
nmax<- length(x[ta:(tb-1)])
x[1:nmax]<- x[ta:(tb-1)]
xb<-    numeric()
xbb<-   numeric()
mb<-    numeric()
mj<-    numeric()
nbl<-   numeric()
seb<-   numeric()
mmbb<-  numeric()
sembb<- numeric()
se.sembb<- numeric()
set.seed(1)               # reset random numbers
B<- 100                   # number of replications
##  lmb = block length, i.e., each block has l observations
lmb<-c(1,10,50,100,150,200,300,400,500,600)
llmb<- length(lmb)
# one can choose the initial value of lmb (lmbi) and the final one (lmbf)
lmbi<- 1
lmbf<- 10
# starting most external loop on the block length
for (l in lmbi:lmbf)  {
# ... periodic boundary conditions:
# ... each x[i] has to be appeared lmb times
#  the blocks in the whole are nmax
# total observations after adding the first lmb-1 observations
ntot<- nmax+(lmb[l]-1)
nmax1<- nmax+1
if(lmb[l]>1){   # if l[k]=1 the loop to form the "new series" is skipped
ii=0
for (i in nmax1:ntot)  {
ii<- ii+1
x[i]<- x[ii]
             }     # ending "if l[k]=1"
                        }
# x is the  "new series" = the original one + (lmb-1) initial realizations
# number of blocks to form the resampled series
nbl[l]<- ceiling(nmax/lmb[l])
#print(x)   # to see  the "new series"
# starting loop on the  B replications
for (b in 1:B) {
        nn<- 0
        for (k in 1:nbl[l]) { # loop  on the number of sampled blocks
# beginning of a block picked at random
        i<- sample(nmax,1,replace=TRUE)
# a block with length l is formed  (initial 'i' included)
              for (ll in i:(i+(lmb[l]-1)))  {
              nn<- nn+1
              xb[nn]<- x[ll]
           } # # the block is finished
                    } #  ending loop on the number of sampled blocks
# print(xb)   # to print the replications
mb[b] <- mean(xb)
# print(mb[b]) # to print the mean of each replication
seb[b]<- sqrt(sig2(xb)/length(xb))
# print(seb[b])  # to print the s.e. of each replication
                  }          # ending loop on the replications
        mmbb[l]<- mean(mb)
        sembb[l]<- sqrt(var(mb))
        se.sembb[l]<- sembb[l]/sqrt(2.*(nbl[l]-1))
        se.sembb[l]<- se.sembb[l]*sqrt(2./3.)
                  }  # ending most external loop on the length of blocks
# summary of quantities in [lmbi:lmbf]
lmb[lmbi:lmbf]
nbl[lmbi:lmbf]
mmbb[lmbi:lmbf]
sembb[lmbi:lmbf]
se.sembb[lmbi:lmbf]
par(mai=c(1.02,1.,0.82,0.42)+0.1)
plot(lmb[lmbi:lmbf],sembb[lmbi:lmbf], type="b",ylim=c(0,0.06),
xlab="block length",font.lab=3,lwd=2,cex.lab=1.3,
ylab=expression(hat(sigma)*"*"))
arrows(lmb,sembb, lmb,sembb+se.sembb, length=0.05, angle=90)
arrows(lmb,sembb, lmb,sembb-se.sembb, length=0.05, angle=90)
```

---

11 Exercises of Chapter 12: Spatial Analysis

Exercise 12.1

Demonstrate equation (12.11), i.e. that for a second-order stationary process the semivariogram is expressed in terms of the covariance $C(h)$ and the variance

$C(0)$:

$$\gamma(h) = C(0) - C(h) \tag{2}$$

where:

$$C(h) = \operatorname{Cov}\left[Z_{s+h}, Z_s\right]$$

and

$$C(0) = \operatorname{Var}\left[Z_s\right]$$

Solution

By definition:

$$\gamma(h) = \frac{1}{2}\operatorname{Var}\left[Z_{s+h} - Z_s\right]$$

and:

$$\operatorname{Var}\left[Z_{s+h} - Z_s\right] = \operatorname{E}\left[(Z_{s+h} - Z_s)^2\right] - \left(\operatorname{E}\left[Z_{s+h} - Z_s\right]\right)^2$$

From stationarity

$\operatorname{E}\left[Z_{s+h} - Z_s\right] = 0$

Therefore:

$$\begin{aligned}\gamma(h) =& \frac{1}{2}\operatorname{E}\left[(Z_{s+h} - Z_s)^2\right] = \frac{1}{2}\operatorname{E}\left[Z_{s+h}^2 + Z_s^2 - 2Z_{s+h}Z_s\right] = \\ & \frac{1}{2}\operatorname{Var}\left[Z_{s+h}\right] + \frac{1}{2}\operatorname{Var}\left[Z_s\right] - \operatorname{Cov}\left[Z_{s+h}, Z_s\right] = \\ & \operatorname{Var}\left[Z_s\right] - \operatorname{Cov}\left[Z_{s+h}, Z_s\right]\end{aligned}$$

Exercise 12.2

If a variogram has a ”sill”, what does it mean in terms of the covariance function (when it is defined, of course)?

Solution

The “sill” corresponds to $\gamma(h) \to C(0)$, therefore to $C(h) \to 0$.

Exercise 12.3

Replicate for T_{min} the analysis of section 12.3.2 on the data *20201121_weather.csv* conducted for T_{max}, limited to the computation of the sample and model variogram. Determine "by eye" reasonable parameters for a linear semivariogram model.

Solution

Use exactly the same code `Code_12_3.R`, substituting T_{min} for T_{max}, and adjusting the coefficients for the semivariogram model, like the following:

```
# Libraries
library(gstat)
library(sf)

# Read data
setwd("C:\RPA\code\spatial_analysis\data")
geo_data = read.csv("20201121_weather.csv")
xy <- geo_data[,c(1,2)]

# Semivariogram cloud
Tmin.vgm.cloud <- variogram(Tmin ~ 1, locations = ~UTM.X + UTM.Y, data = geo_data, cloud=TRUE)
plot(Tmin.vgm.cloud)

# Convert dataframe geo_data into object of the sf class
geo_data.sf <- st_as_sf(geo_data,coords=c("UTM.X","UTM.Y"),crs="EPSG:32632")
vgm <- variogram(Tmin~1,geo_data.sf)
plot(vgm)

# Anisotropy
vgm.aniso <- variogram(Tmin~1,geo_data.sf, alpha = c(0, 45, 90, 135))
plot(vgm.aniso)

# Linear dependence
Tmin.fit <- lm(Tmin~Alt,data=geo_data)
plot(Tmin~Alt,geo_data,xlab="Altitude (m)",ylab=expression(paste(T[min])))
abline(Tmin.fit,col="red")

# Work on residuals of linear model
# Semivariogram cloud
Tmin.res <- Tmin.fit$residuals
Tmin.vgm.cloud <- variogram(Tmin.res~1,locations = ~UTM.X + UTM.Y, data = geo_data, cloud=TRUE)
plot(Tmin.vgm.cloud)
# Empirical semivariogram of residuals
Tmin.vgm <- variogram(Tmin.res~1,locations = ~UTM.X + UTM.Y, data = geo_data)
plot(Tmin.vgm)
Tmin.vgm.model <- vgm(3, "Lin", 150000, 2.5)
plot(Tmin.vgm,Tmin.vgm.model)
```

Exercise 12.4

With reference to exercise 12.3, guess an exponential model instead of a linear one and use `fit.variogram` to refine its parameters.

Solution

Same code as the previous exercise, but substitute the line:

```
Tmin.vgm.model <- vgm(3, "Lin", 150000, 2.5)
plot(Tmin.vgm,Tmin.vgm.model)
```

with:

```
Tmin.vgm.model <- vgm(5, "Exp", 150000, 2)
plot(Tmin.vgm,Tmin.vgm.model)
fit.variogram(Tmin.vgm,Tmin.vgm.model) -> vgm.model.fit
plot(Tmin.vgm,vgm.model.fit)
```

Exercise 12.5

Perform the spatial kriging procedure on the T_{min} data of the last two exercises, and compare the interpolation results obtained by the linear and exponential variogram model. Use `Code_12_4.R` as follows.

Solution

First case: linear variogram model. Add the following lines to exercise 12.3.

```
# Load the stars package
library(stars)
library(ggplot2)
# import DEM data of Emilia-Romagna
ER.DEM <- read_stars("dem450_ER.tif")
# give the name Alt to the data column
names(ER.DEM) <- 'Alt'
# Set the CRS
st_crs(ER.DEM) <- st_crs(geo_data.sf)
# Show Altitude map
ggplot() + geom_stars(data = ER.DEM) +
  coord_equal()+
theme_void() +
scale_fill_steps(n.breaks=8,low="lightyellow2",high="red",na.value="white") +
scale_x_discrete(expand=c(0,0))+
scale_y_discrete(expand=c(0,0))+
labs(fill="Alt") +
theme(panel.border = element_rect(linetype = "solid", fill = NA),
legend.key.size = unit(0.95, 'cm'))+ theme(plot.margin=unit(c(1,1,1,1),"cm"))

# Kriging
res.kriged <- krige(Tmin.res~1,geo_data.sf,ER.DEM,model=Tmin.vgm.model)
```

```
# show kriging results (predicted residuals)
ggplot() + geom_stars(data = res.kriged) +
  coord_equal()+
  theme_void() +
  scale_fill_steps(n.breaks=8,low="blue",high="orange",na.value="white") +
  scale_x_discrete(expand=c(0,0))+
  scale_y_discrete(expand=c(0,0))+
  labs(fill="Pred") +
  theme(panel.border = element_rect(linetype = "solid", fill = NA),
legend.key.size = unit(0.95, 'cm'))+ theme(plot.margin=unit(c(1,1,1,1),"cm"))
```

Second case: linear variogram model. Add the following lines to exercise 12.4. The code is the same as above except for the `Kriging` section that must be substituted with the following.

```
# Kriging
res.kriged <- krige(Tmin.res~1,geo_data.sf,ER.DEM,model=vgm.model.fit)
# show kriging results (predicted residuals)
ggplot() + geom_stars(data = res.kriged) +
  coord_equal()+
  theme_void() +
  scale_fill_steps(n.breaks=8,low="blue",high="orange",na.value="white") +
  scale_x_discrete(expand=c(0,0))+
  scale_y_discrete(expand=c(0,0))+
  labs(fill="Pred") +
  theme(panel.border = element_rect(linetype = "solid", fill = NA),
legend.key.size = unit(0.95, 'cm'))+ theme(plot.margin=unit(c(1,1,1,1),"cm"))
```

Exercise 12.6

Using the precipitation data in *20201121_weather.csv* (variable *Prec*) try to conduct a variogram analysis like that developed for T_{max} in the Chapter, or for T_{min}. What you can tell about the dependence on Altitude?

Solution

The modifications to the above code are straightforward:

```
# Libraries
library(gstat)
library(sf)

# Read data
#setwd("C:\RPA\code\spatial_analysis\data")
setwd("/Users/olmi/Desktop/libro Bittelli Olmi Rosa/Exercises_book")
geo_data = read.csv("20201121_weather.csv")
xy <- geo_data[,c(1,2)]

# Semivariogram cloud
Prec.vgm.cloud <- variogram(Prec ~ 1, locations = ~UTM.X + UTM.Y, data = geo_data, cloud=TRUE)
plot(Prec.vgm.cloud)
```

```
# Convert dataframe geo_data into object of the sf class
geo_data.sf <- st_as_sf(geo_data,coords=c("UTM.X","UTM.Y"),crs="EPSG:32632")
vgm <- variogram(Prec~1,geo_data.sf)
plot(vgm)

# Anisotropy
vgm.aniso <- variogram(Prec~1,geo_data.sf, alpha = c(0, 45, 90, 135))
plot(vgm.aniso)

# Linear dependence
Prec.fit <- lm(Prec~Alt,data=geo_data)
plot(Prec~Alt,geo_data,xlab="Altitude (m)",ylab="Precipitation")
abline(Prec.fit,col="red")

# Work on residuals of the linear model
# Semivariogram cloud
Prec.res <-Prec.fit$residuals
Prec.vgm.cloud <- variogram(Prec.res~1,locations = ~UTM.X + UTM.Y, data = geo_data, cloud=TRUE)
plot(Prec.vgm.cloud)
# Empirical semivariogram of residuals
Prec.vgm <- variogram(Prec.res~1,locations = ~UTM.X + UTM.Y, data = geo_data)
plot(Prec.vgm)
Prec.vgm.model <- vgm(3, "Lin", 150000, 65)
plot(Prec.vgm,Prec.vgm.model)
```

Precipitation appear to be almost independent of altitude. Moreover the semivariogram is nearly isotropic.

Exercise 12.7

Use a linear variogram model in exercise 12.6 and compute a spatial kriging. If `res.kriged` is the result of analysis on the residuals of linear dependence of *Prec* on *Alt*:

```
krige(Prec.res~1,geo_data.sf,ER.DEM,model=Prec.vgm.model) -> res.kriged
```

Plot both “prediction” and “variance” like follows, and discuss what you see:

```
plot(res.kriged['var1.pred'])
plot(res.kriged['var1.var'])
```

Solution

```
# DEM model
ER.DEM <- read_stars("dem450_ER.tif")
# give the name Alt to the data column
names(ER.DEM) <- 'Alt'
# Set the CRS
st_crs(ER.DEM) <- st_crs(geo_data.sf)
# Kriging
res.kriged <- krige(Prec.res~1,geo_data.sf,ER.DEM,model=Prec.vgm.model)
# Plot "prediction" and variance
plot(res.kriged["var1.pred"])
plot(res.kriged["var1.var"])
```