Midterm

2019150432 임효진

April 29, 2021

1

(a)

```
library(PolynomF)
library(Deriv)

coef=numeric(11)

for(k in 1:length(coef)){
    d=Deriv(atan, nderiv = k)
    coef[k]=d(0)/factorial(k)
}

polynom1=function(x, coef){
  poly=polynom(c(atan(0), coef))
  return(poly(x))
}
```

(b)

```
pi1=6*(polynom1(1/sqrt(3), coef))
pi2=4*polynom1(1, coef)

pi1
```

```
## [1] 3.141309
```

```
pi2
```

```
## [1] 2.976046
```

$\hat{\pi}_1$ estimates more accurately.

(c)

```
polynom2=function(x, coef){
  n=length(coef)
  c=numeric(n)
  c[n]=coef[n]
  while(n>1){
    n=n-1
    c[n]=c[n+1]*x+coef[n]
  }
  return(c[1])
}
```

(d)

```
6*(polynom2(1/8, coef))+2*(polynom2(1/57, coef))+
  polynom2(1/239, coef)
```

```
## [1] 8.968829
```

(e)

```
system.time(
  sapply(c(seq(0.001, 0.999, by=0.001),
           -seq(0.001, 0.999, by=0.001)), polynom1, coef))
```

```
##    user  system elapsed
##    0.06    0.00    0.08
```

```
system.time(
  sapply(c(seq(0.001, 0.999, by=0.001),
           -seq(0.001, 0.999, by=0.001)), polynom2, coef))
```

```
##    user  system elapsed
##       0       0       0
```

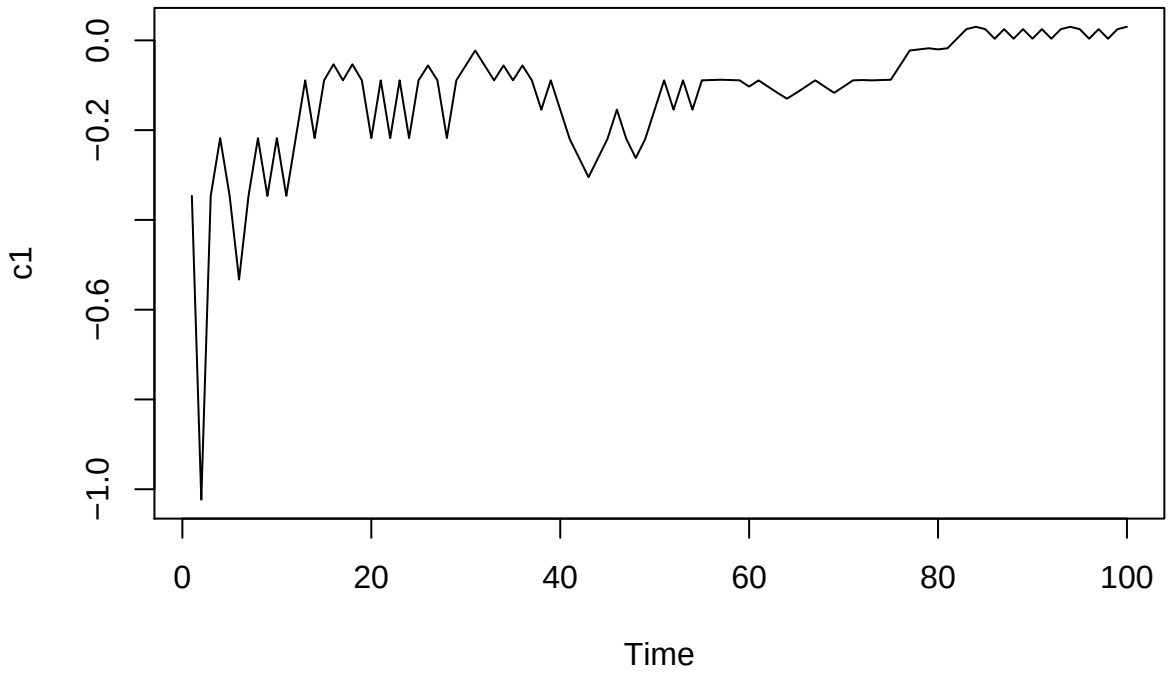
When considering the system.time polynom2 is more efficient.

2

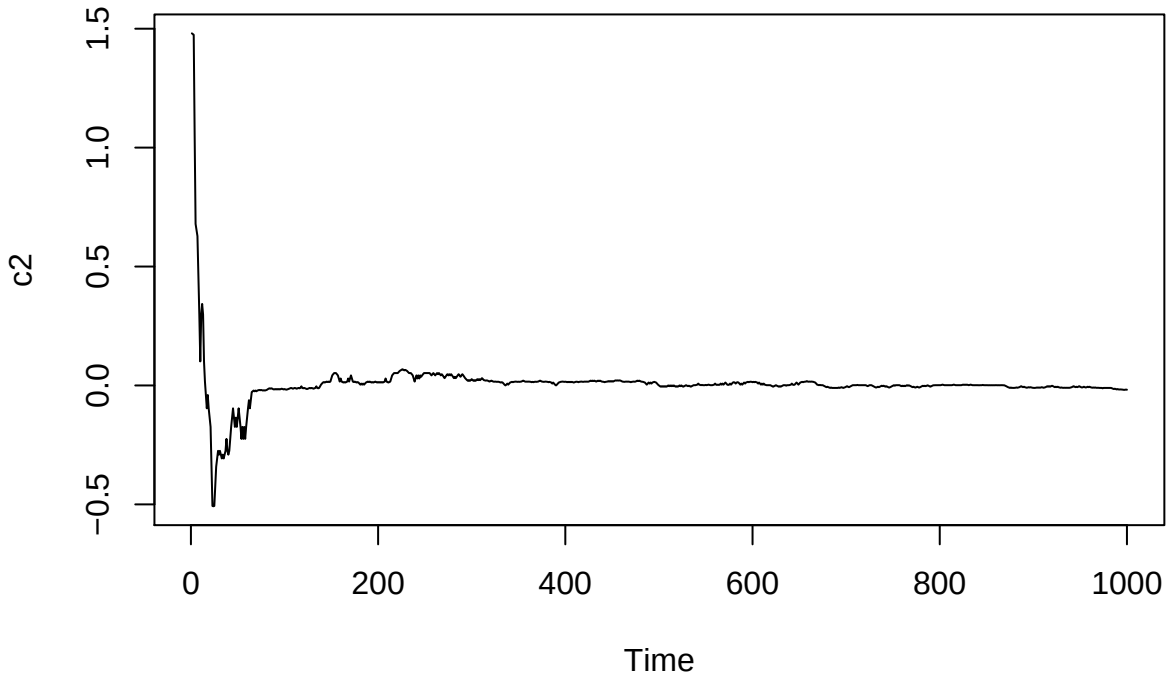
##(a)

```
cumMed=function(x){
    out=numeric(length(x))
    for(i in 1:length(x)){
        temp=sort(x[1:i])
        out[i]=ifelse(i%%2==1,
                      temp[(i+1)/2], mean(temp[i/2+0:1]))
```

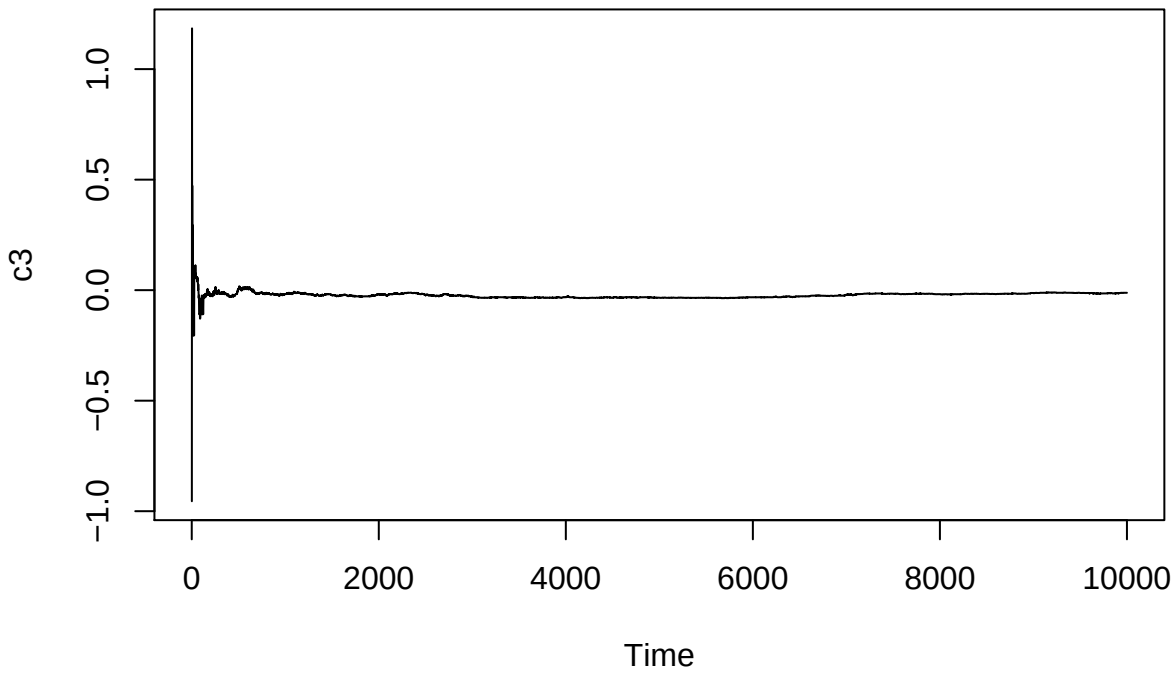
```
    }
    return(out)
}

c1=cumMed(rnorm(100))
c2=cumMed(rnorm(1000))
c3=cumMed(rnorm(10000))

ts.plot(c1)
```




```
ts.plot(c2)
```




```
ts.plot(c3)
```



(b)

```
cumMed2=function(x){
    out=matrix(x, nrow=length(x), ncol=length(x),
           byrow = T)
  out[upper.tri(out)]=NA
  apply(out, 1, median, na.rm=T)
}
```

3

(a)

```
arithMean=function(n, r){
    temp1=combn(n, r)
    temp2=apply(temp1, 2, min)
    return(mean(temp2))
}

arithMean(10, 6)
```

```
## [1] 1.571429
```

(b)

```
sim=function(n, r){
    temp=sample(n, r)
    return(min(temp))
}

res1=replicate(100, sim(20, 7))
mean(res1)
```

```
## [1] 2.29
```

4

(a)

```
cofacDet=function(n, p, q){
    mat=matrix(0, n, n)
    for(i in 2:(n-1)){
        mat[i,]=c(rep(0, i-2), q, p, p-q, rep(0, (n-1)-i))
    }
    mat[1, c(1, 2)]=c(p, p-q)
    mat[n, c(n-1, n)]=c(q, p)
    temp <- numeric(length(mat[1,]))
    for (i in 1:n){
        temp[i] <-(-1)^{i+1}*mat[1,i]*det(mat[2:n,-i])
        }
    return(sum(temp))
}

for(i in 10:20){
    print(cofacDet(i, 1/2, 1/3))
}
```

```
## [1] 3.385364e-05
## [1] 1.12873e-05
## [1] 3.762893e-06
## [1] 1.254374e-06
## [1] 4.181376e-07
## [1] 1.393813e-07
## [1] 4.646079e-08
## [1] 1.548699e-08
## [1] 5.16234e-09
## [1] 1.720782e-09
## [1] 5.735941e-10
```

(b)

$A_{n\times n}$ is not a transition matrix because the rows do not sum to 1. Therefore we can modify the matrix by correcting element (1, n):

$$\widetilde{A}_{n\times n} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & \dots & 0 & 0 \\ . & . & & & & & \\ . & . & & & & & \\ . & . & & & & & \\ 0 & 0 & \dots & & 0 & 0.5 & 0.5 \\ 0.5 & 0 & \dots & & & 0 & 0.5 \end{bmatrix}$$

The above markov chain has a sequence of random variables $X_1, X_2 \dots, X_n, \dots$ that take values 1, 2, ⋯, 10 which implies $S = \{1, 2, ..10\}$. From the transition matrix we can conclude that the probability of moving to the next state is 0.5 and remaining in the state it is in is also 0.5 with exception of the current state being n.

```
library(matrixcalc)
mat=matrix(c(0.5, 0.5, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0.5, 0.5, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0.5, 0.5, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0.5, 0.5, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0.5, 0.5, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0.5, 0.5, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0.5, 0.5, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 0,
             0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0.5),
           nrow=10, ncol=10, byrow = T)

powerMat=matrix.power(mat, 19)
sum(powerMat[1,]*c(0, 0, 0, 0, 0, 0, 0, 1, 0, 0))
```

```
## [1] 0.09922028
```

```
# by simulation
sim=function(mat, trial, current){
    x=numeric(trial)
    x[1]=current
    for(j in 2:trial){
        x[j]=sample(1:10, size = 1, prob = mat[x[j-1],])
    }
    return(x[trial])
}

res1=replicate(100, sim(mat, 20, 1))
res1=factor(res1, levels=1:10)
(table(res1)/100)[8]
```

```
##    8
## 0.11
```

5

(a)

The suitable state space is S is $S = \{1, ..9\}$. From this we can calculate a transition matrix.

```
mat=matrix(c(0, 1/2, 0, 0, 0, 1/2, 0, 0, 0,
             1/3, 0, 1/3, 0, 1/3, 0, 0, 0, 0,
             0, 1/2, 0, 1/2, 0, 0, 0, 0, 0,
             0, 0, 1/3, 0, 1/3, 0, 0, 0, 1/3,
             0, 1/4, 0, 1/4, 0, 1/4, 0, 1/4, 0,
             1/3, 0, 0, 0, 1/3, 0, 1/3, 0, 0,
             0, 0, 0, 0, 0, 1/2, 0, 1/2, 0,
             0, 0, 0, 0, 1/3, 0, 1/3, 0, 1/3,
             0, 0, 0, 1/2, 0, 0, 0, 1/2, 0),
           nrow=9, byrow = T)
```

(b)

Since from an even number state the processed state is an odd number and from an odd number state the processed state the processed state is only even numbered, we can conclude that the Marchov chain is not regular.

(c)

```
Rmat=matrix(1, nrow=9, ncol=9)
zerovec=matrix(0, nrow=1, ncol=9)
Rmat[, 1:8]=mat[, 8]-diag(9)[, 1:8]
bvec=matrix(1, nrow=1, ncol=9)
bvec[, 1:8]=zerovec[, 1:8]
bvec%*%solve(Rmat)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.1176471 0.1176471 0.1176471 0.1176471 0.1176471 0.1176471 0.1176471
##           [,8]       [,9]
## [1,] 0.1176471 0.05882353
```

6

(a)

The process described above is a markov chain having a sequence of random variables $X_0, X_1, ...$ that take values 0, 1, ⋯, N which implies $S = \{0, 1, ..N\}$. These random variables is a sequence of random variables that exhibits one-step dependence in which the probability of each event depends only on the state attained in the previous event.

(b)

The possible transition probability when assuming $X_n = a$ is:

- $X_{n+1} = a$ (R from left R from right): $(a/N) * ((N-a)/N)$
- $X_{n+1} = a - 1$ (R from left B from right): $(a/N) * (a/N)$
- $X_{n+1} = a + 1$ (B from left R from right): $((N-a)/N) * ((N-a)/N)$
- $X_{n+1} = a$ (B from left B from right): $((N-a)/N) * (a/N)$

This implies that:

- $p_{aa} = 2 * ((N-a)/N) * (a/N)$
- $p_{a,a+1} = ((N-a)/N)^2$
- $p_{a,a-1} = (a/N)^2$
- $p_{a,b} = 0$ if $b \neq a-1, a, a+1$

Therfore the transition matrix can be expressed as follows:

$$P = (1/N^2) \begin{bmatrix} 0 & N^2 & 0 & 0 & \dots & 0 & 0 \\ 1 & 2(N-1) & (N-1)^2 & 0 & \dots & 0 & 0 \\ 0 & 2^2 & 4(N-2) & (N-2)^2 & \dots & 0 & 0 \\ . & . & & & & & \\ . & . & & & & & \\ . & . & & & & & \\ 0 & 0 & \dots & & (N-1)^2 & 2(N-1) & 1 \\ 0 & 0 & \dots & & & N^2 & 0 \end{bmatrix}$$

(c)

```
mat=matrix(c(0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0.01, 0.18, 0.81, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0.04, 0.32, 0.64, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0.09, 0.42, 0.49, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0.16, 0.48, 0.36, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0.25, 0.5, 0.25, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0.36, 0.48, 0.16, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0.49, 0.42, 0.09, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0.64, 0.32, 0.04, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0.81, 0.18, 0.01,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1),
           nrow=11, byrow=T)

powerMat=matrix.power(mat, 19)

sim=function(mat, trial, current){
    x=numeric(trial)
    x[1]=current
    for(j in 2:trial){
        x[j]=sample(1:11, size = 1, prob = mat[x[j-1],])
```

```
    }
    return(x[trial])
}

res1=replicate(100, sim(mat, 20, 1))
res1=factor(res1, levels=0:10)
table(res1)/100
```

```
## res1
##    0    1    2    3    4    5    6    7    8    9   10
## 0.00 0.00 0.00 0.01 0.07 0.22 0.30 0.29 0.11 0.00 0.00
```