# 2023 秋季本科时间序列

# 第 6 次作业答案

11 月 19 日

1. (a)

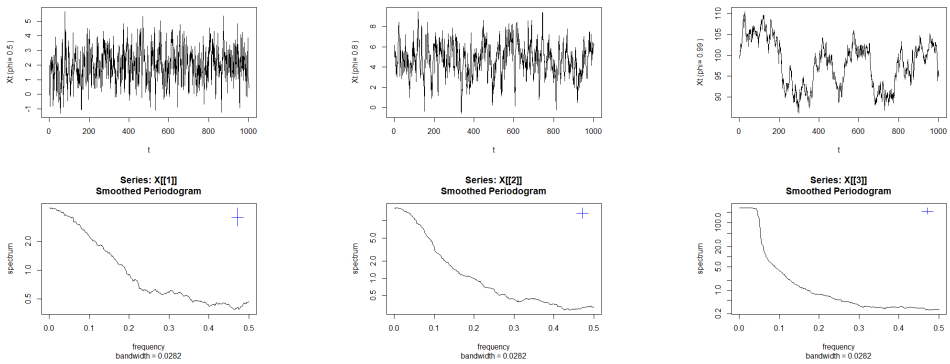
$$\boldsymbol{M} = \mathbb{E}\begin{bmatrix} 1 \\ \boldsymbol{X}_{t-1} \end{bmatrix}\begin{bmatrix} 1 & \boldsymbol{X}_{t-1} \end{bmatrix} = \begin{bmatrix} 1 & \mathbb{E}\boldsymbol{X}_{t-1} \\ \mathbb{E}\boldsymbol{X}_{t-1} & \mathbb{E}\boldsymbol{X}_{t-1}^2 \end{bmatrix}, det(\boldsymbol{M}) = \mathrm{var}\boldsymbol{X}_{t-1}$$

$$\mathrm{var}\boldsymbol{X}_t = \mathrm{var}(\mu + \phi\boldsymbol{X}_{t-1} + \epsilon) = \phi^2\mathrm{var}\boldsymbol{X}_{t-1} + \sigma_\epsilon^2$$

因为 AR(1) 过程 $\boldsymbol{X}_t$ 平稳，故 $\mathrm{var}\boldsymbol{X}_{t-1} = \mathrm{var}\boldsymbol{X}_t = \frac{1}{1-\phi^2} > 0, \boldsymbol{M}$ 满秩

(b) 3 个模拟序列及其样本谱密度函数的估计代码如下：

```
mu <- 1
phi <- c(0.5,0.8,0.99)
X <- tibble()
for(i in 1:3){
 X[1, i] = mu/(1 - phi[i])
 for(j in 1:1000){
  X[j+1, i] = mu + phi[i]*X[j, i] + rnorm(1,0,1)
 }
 plot(X[[i]], type = "l", xlab = "t",
 ylab = paste('Xt (phi=', phi[i],')'))
}
spec.pgram(X[[1]],span=100, taper = 0.2)
spec.pgram(X[[2]],span=100, taper = 0.1)
spec.pgram(X[[3]],span=100, taper = 0.2)
```




理论谱密度函数: $\gamma(0)=\frac{1}{1-\phi^2}, \gamma(k)=\phi\gamma(k-1)=\phi^k\gamma(0)=\frac{\phi^k}{1-\phi^2}$

$$S_X(\omega)=\sum_{k=-\infty}^{\infty}\gamma(k)e^{-i2\pi\omega k}$$

由于常数项不影响谱密度函数，即求 AR(1) 过程 $\boldsymbol{X}_t=\phi\boldsymbol{X}_{t-1}+\varepsilon_t$ 的谱密度函数，

$\boldsymbol{X}_t-\phi\boldsymbol{X}_{t-1}=\varepsilon_t, A(\mathcal{L})\boldsymbol{X}_t=\varepsilon_t$，则 $\boldsymbol{X}_t=A^{-1}(\mathcal{L})\varepsilon_t, A^{-1}(\mathcal{L})=\frac{1}{1-\phi\mathcal{L}}$

给定滤波多项式 $C(z)=\frac{1}{1-\phi z}$，按照第五讲滤波序列的谱表示，增益函数 $G(\omega)=\sqrt{C(e^{-i2\pi\omega})C(e^{i2\pi\omega})}$

则 $S_X(\omega)=G^2(\omega)S_\varepsilon(\omega), S_\varepsilon(\omega)=1$，代入得

$$S_X(\omega)=\frac{1}{(1-\phi e^{-i2\pi\omega})(1-\phi e^{i2\pi\omega})}=\frac{1}{1-2\phi\cos(2\pi\omega)+\phi^2}$$

把 $\phi=0.5,0.8,0.99$ 代入得

$$S_X(\omega)|(\phi=0.5)=\frac{1}{1.25-\cos(2\pi\omega)}$$

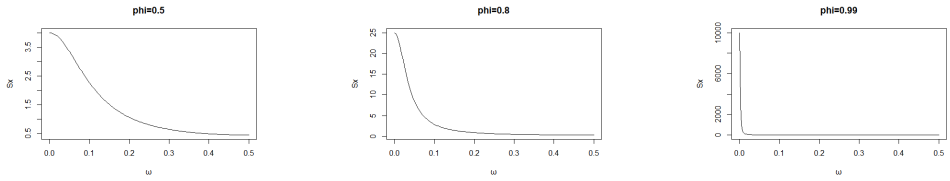
$$S_X(\omega)|(\phi=0.8)=\frac{1}{1.64-1.6\cos(2\pi\omega)}$$

$$S_X(\omega)|(\phi=0.99)=\frac{1}{1.9801-1.98\cos(2\pi\omega)}$$

代码如下：

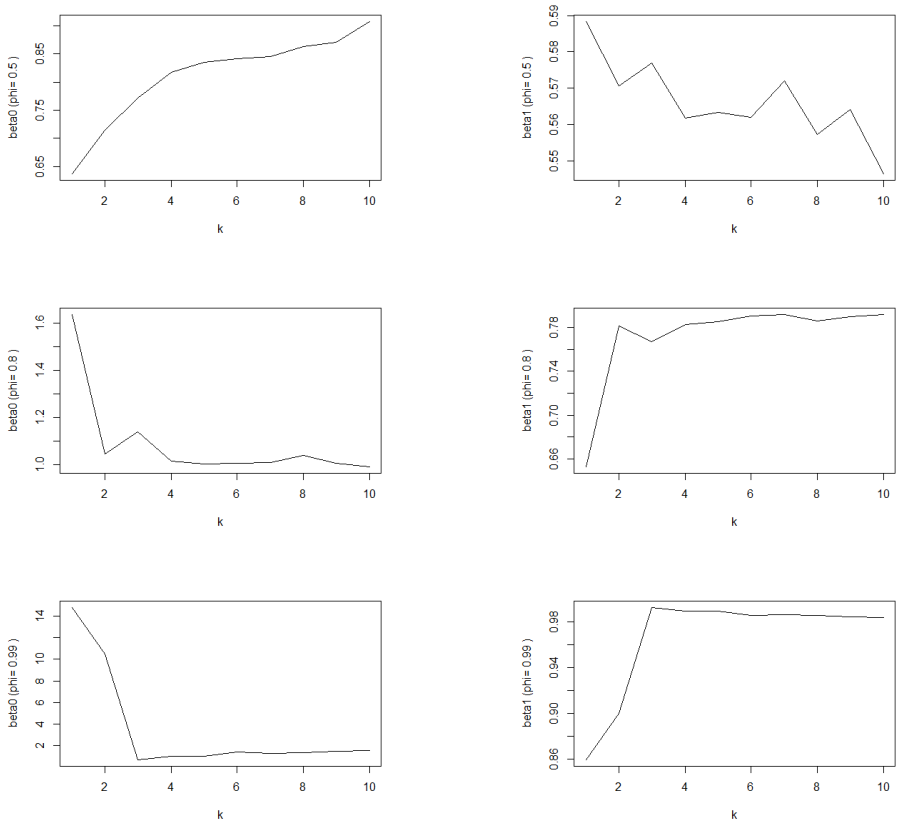
```
x <- seq(0,0.5,0.0025)
s1 <- vector()
s2 <- vector()
s3 <- vector()
length(x)
for(i in 1:length(x)) {
 s1[i] <- 1/(1.25-cos(2*pi*x[i]))
 s2[i] <- 1/(1.64-1.6*cos(2*pi*x[i]))
 s3[i] <- 1/(1.9801-1.98*cos(2*pi*x[i]))
```

```
}
plot(x,s1,type="l",main="phi=0.5",xlab="w",ylab="Sx")
plot(x,s2,type="l",main="phi=0.8",xlab="w",ylab="Sx")
plot(x,s3,type="l",main="phi=0.99",xlab="w",ylab="Sx")
```




(c) 代码如下：

```
beta0 <- tibble()
beta1 <- tibble()
for(i in 1:3){
 for(k in 1:10){
  y <- unlist(X[2:(100*k+1), i])
  x <- unlist(X[1:(100*k), i])
  ols <- coefficients(lm(y ~ x))
  beta0[k, i] = ols[1]
  beta1[k, i] = ols[2]
 }
 plot(beta0[[i]], type = "l", xlab = "k",
 ylab = paste('beta0 (phi=', phi[i],')'))
 plot(beta1[[i]], type = "l", xlab = "k",
 ylab = paste('beta1 (phi=', phi[i],')'))
}
```



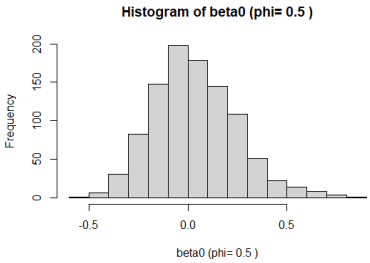

随着 $\phi$ 的增大，OLS 估计随样本量在 100k 增加时的收敛性逐渐增强。

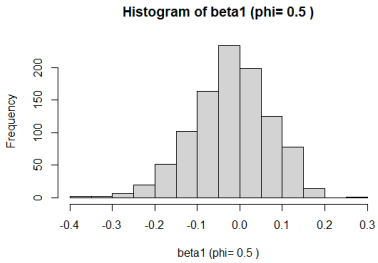
(d) 代码如下：

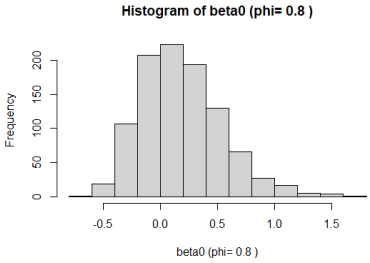
```
X1 <- tibble()
for(i in 1:1000){
 X1[1, i] = mu/(1 - phi[1])
 for(j in 1:100){
  X1[j+1, i] = mu + phi[1]*X1[j, i] + rnorm(1,0,1)
 }
}
# phi = 0.8
X2 <- tibble()
for(i in 1:1000){
 X2[1, i] = mu/(1 - phi[2])
 for(j in 1:100){
  X2[j+1, i] = mu + phi[2]*X2[j, i] + rnorm(1,0,1)
 }
}
```

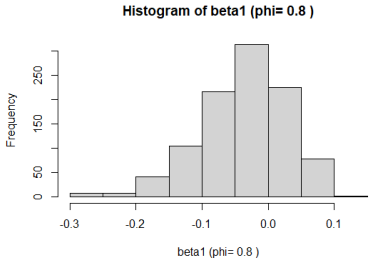
```r
# phi = 0.99
X3 <- tibble()
for(i in 1:1000){
 X3[1, i] = mu/(1 - phi[3])
 for(j in 1:100){
  X3[j+1, i] = mu + phi[3]*X3[j, i] + rnorm(1,0,1)
 }
}
X <- list(X1, X2, X3)

beta0 <- tibble()
beta1 <- tibble()
b0 <- tibble()
b1 <- tibble()
sd_beta0 <- vector("double",2)
sd_beta1 <- vector("double",2)
for(i in 1:3){
 for(j in 1:1000){
  y <- unlist(X[[i]][2:101, j])
  x <- unlist(X[[i]][1:100, j])
  ols <- coefficients(lm(y ~ x))
  beta0[j, i] <- ols[1]
  beta1[j, i] <- ols[2]
 }
 hist(beta0[[i]]-mu, xlab = paste('beta0(phi=',
 phi[i],')'), main = paste('Histogram of beta0(phi=',
 phi[i],')'))
 hist(beta1[[i]]-phi[i], xlab = paste('beta1(phi=',
 phi[i],')'), main = paste('Histogram of beta1(phi=',
 phi[i],')'))

 sd_beta0[i] <- sd(beta0[[i]])
 sd_beta1[i] <- sd(beta1[[i]])
}

```

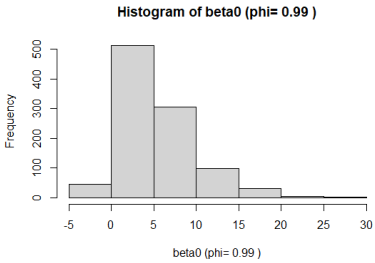
```r
sd_beta0
## [1] 0.2108055 0.3540542 4.3524750
sd_beta1
## [1] 0.09132497 0.06635243 0.04314966
for(i in 1:3){
 print(ggplot ()+ geom_histogram (data=NULL ,
 mapping=aes(x=beta0 [[i]],y=..density..))
 + xlab(paste('beta0 (phi=',phi[i],')'))
 + geom_function(fun = dnorm ,color="red",
 args=list(mean=mu ,sd=sd_beta0[i])))
 print(ggplot ()+ geom_histogram (data=NULL ,
 mapping=aes(x=beta1 [[i]],y=..density..))
 + xlab(paste('beta1 (phi=',phi[i],')'))
 + geom_function(fun=dnorm,color="red",
 args=list(mean=phi[i],sd=sd_beta1[i])))
}
```

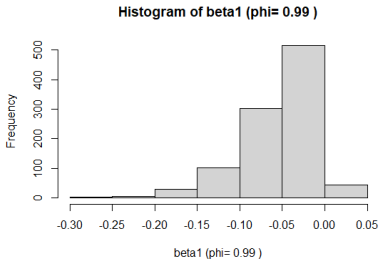

Histogram of beta0 (phi= 0.5 )
Frequency
beta0 (phi= 0.5 )

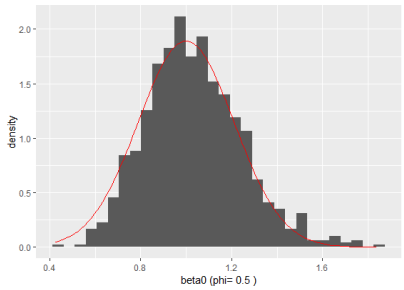

Histogram of beta1 (phi= 0.5 )
Frequency
beta1 (phi= 0.5 )

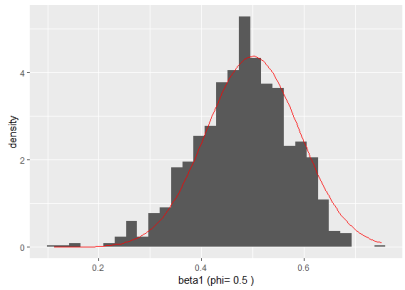

Histogram of beta0 (phi= 0.8 )
Frequency
beta0 (phi= 0.8 )

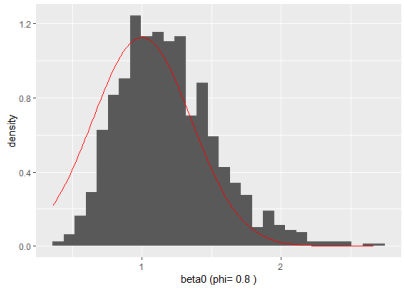

Histogram of beta1 (phi= 0.8 )
Frequency
beta1 (phi= 0.8 )

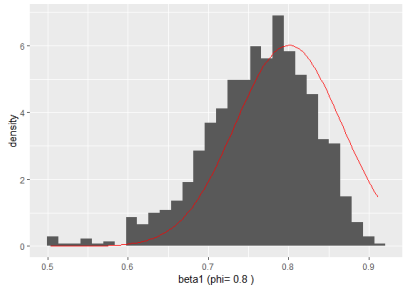

Histogram of beta0 (phi= 0.99 )
Frequency
beta0 (phi= 0.99 )

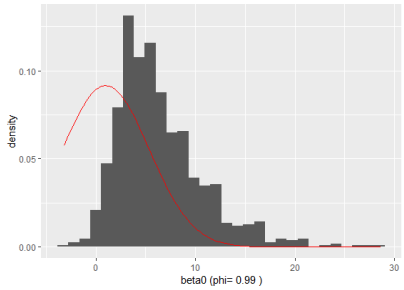

Histogram of beta1 (phi= 0.99 )
Frequency
beta1 (phi= 0.99 )

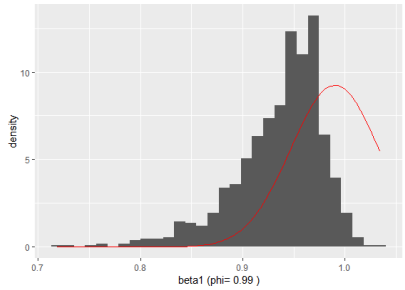

density
beta0 (phi= 0.5 )


density
beta1 (phi= 0.5 )


density
beta0 (phi= 0.8 )


density
beta1 (phi= 0.8 )


density
beta0 (phi= 0.99 )


density
beta1 (phi= 0.99 )

当 $\phi$ 趋近于 1 时，样本分布逐渐偏离正态分布。

(e) 代码如下：

```
#理论渐近标准误
mu <- 1
sigma <- 1
T_size <- 100
AR1_CM <- function(phi, mu, sigma){
 EX <- mu / (1 - phi)
 EX2 <- (EX)^2 + 1 / (1 - phi^2)
 M <- matrix(c(1, EX, EX, EX2),
 ncol = 2)
 CM <- solve(M) * (sigma ^2) * (1/T_size)
 return(CM)
}

AR1_CM(phi[1], mu = 1, sigma = 1)
#        [,1]    [,2]
#[1,]  0.040 -0.0150
#[2,] -0.015  0.0075
AR1_CM(phi[2], mu = 1, sigma = 1)
#        [,1]    [,2]
#[1,]  0.100 -0.0180
#[2,] -0.018  0.0036
AR1_CM(phi[3], mu = 1, sigma = 1)
#         [,1]      [,2]
#[1,]  2.0000 -0.019900
#[2,] -0.0199  0.000199
sd_theory <- vector(mode = "list", length = 3)
for(i in 1:3){
 sd_theory[[i]] <- c(sd_beta0 = sqrt(AR1_CM(phi[i], mu=1,
 sigma = 1)[1,1]), sd_beta1 =sqrt(AR1_CM(phi[i], mu = 1,
 sigma = 1)[2,2]))
}
names(sd_theory) <- c("phi=0.5","phi=0.8","phi=0.99")
sd_theory
```

```r
# $phi=0.5
# sd_bata0     sd_beta1
# 0.20000000  0.08660254
#
# $phi=0.8
# sd_bata0    sd_beta1
# 0.3162278  0.0600000
#
# $phi=0.99
# sd_bata0     sd_beta1
# 1.41421356  0.01410674

#样本渐近标准误
x <- vector("double",101)
for (i in 1:3) {
 x[1] <- mu/(1 - phi[i])
 for (j in 1:100) {
  x[j+1] <- mu + phi[i]*x[j] + rnorm(1,0,1)
 }
 X <- cbind(rep(1,100), x[2:101])
 M <- t(X) %*% X / 100    # 求M矩阵
 CM<- 1/100*1*solve(M)
 print(CM)
 sd_sample<-c(sd_beta0=sqrt(CM[1,1]),sd_beta1=sqrt(CM[2,2]))
 print(sd_sample)
}
#             [,1]          [,2]
#[1,]  0.04012951 -0.015470419
#[2,] -0.01547042  0.007943503
#sd_beta0    sd_beta1
#0.20032352 0.08912633
#             [,1]         [,2]
#[1,]  0.11983710 -0.02330555
#[2,] -0.02330555  0.00494504
#sd_beta0    sd_beta1
```

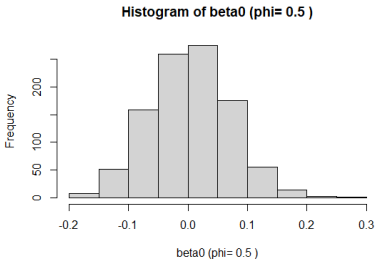
```
#0.34617495  0.07032098
#             [,1]          [,2]
#[1,]   2.35308469 -0.0252535546
#[2,] -0.02525355  0.0002721805
#sd_beta0    sd_beta1
#1.53397676  0.01649789
```

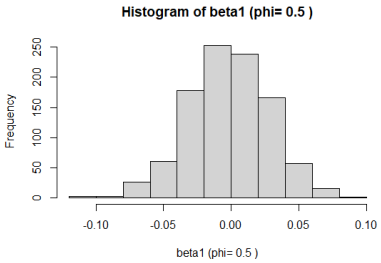
渐进标准误接近样本标准差的值。

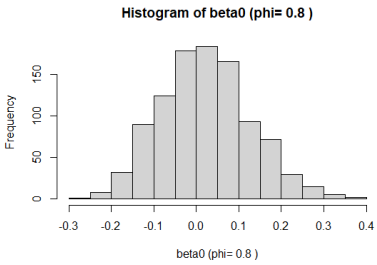
(f) 代码如下：

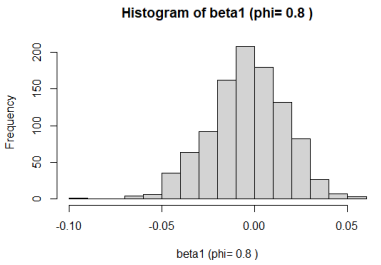
```
# phi = 0.5
X1 <- tibble()
for(i in 1:1000){
 X1[1, i] = mu/(1 - phi[1])
 for(j in 1:900){
  X1[j+1, i] = mu + phi[1]*X1[j, i] + rnorm(1,0,1)
 }
}
# phi = 0.8
X2 <- tibble()
for(i in 1:1000){
 X2[1, i] = mu/(1 - phi[2])
 for(j in 1:900){
  X2[j+1, i] = mu + phi[2]*X2[j, i] + rnorm(1,0,1)
 }
}
# phi = 0.99
X3 <- tibble()
for(i in 1:1000){
 X3[1, i] = mu/(1 - phi[3])
 for(j in 1:900){
  X3[j+1, i] = mu + phi[3]*X3[j, i] + rnorm(1,0,1)
 }
}
X <- list(X1, X2, X3)
beta0 <- tibble()
beta1 <- tibble()
```

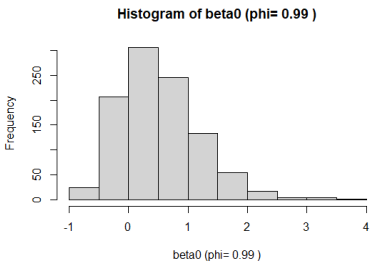
```r
b0 <- tibble()
b1 <- tibble()
sd_beta0 <- vector("double",2)
sd_beta1 <- vector("double",2)
for(i in 1:3){
 for(j in 1:1000){
  y <- unlist(X[[i]][2:901, j])
  x <- unlist(X[[i]][1:900, j])
  ols <- coefficients(lm(y ~ x))
  beta0[j, i] <- ols[1]
  beta1[j, i] <- ols[2]
 }
 hist(beta0[[i]]-mu, xlab = paste('beta0 (phi=',
 phi[i],')'),main=paste('Histogram of beta0 (phi=',
 phi[i],')'))
 hist(beta1[[i]]-phi[i],xlab=paste('beta1(phi=',
 phi[i],')'),main=paste('Histogram of beta1 (phi=',
 phi[i],')'))
 sd_beta0[i] <- sd(beta0[[i]])
 sd_beta1[i] <- sd(beta1[[i]])
}
sd_beta0
## [1] 0.06656101 0.10590085 0.68341522
sd_beta1
## [1] 0.029616538 0.020192899 0.006804025
for(i in 1:3){
 print(ggplot()+geom_histogram(data=NULL,mapping=
 aes(x=beta0[[i]],y=..density..))+xlab(paste('beta0
 (phi=',phi[i],')'))+geom_function(fun=dnorm,color="red",
  args=list(mean=mu,sd=sd_beta0[i])))
 print(ggplot()+geom_histogram(data=NULL,mapping=
 aes(x=beta1[[i]],y=..density..))+xlab(paste('beta1
 (phi=',phi[i],')'))+geom_function(fun=dnorm,color="red",
 args=list(mean=phi[i],sd=sd_beta1[i])))
}
```

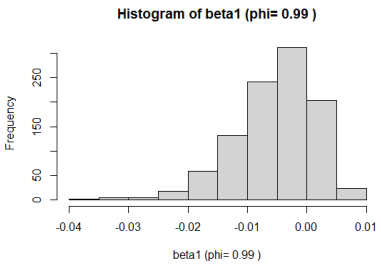

Histogram of beta0 (phi= 0.5 )
Frequency
beta0 (phi= 0.5 )

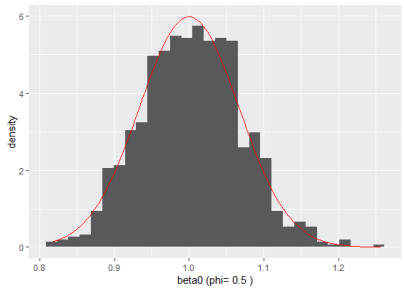

Histogram of beta1 (phi= 0.5 )
Frequency
beta1 (phi= 0.5 )

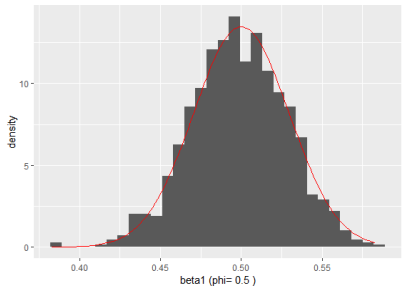

Histogram of beta0 (phi= 0.8 )
Frequency
beta0 (phi= 0.8 )

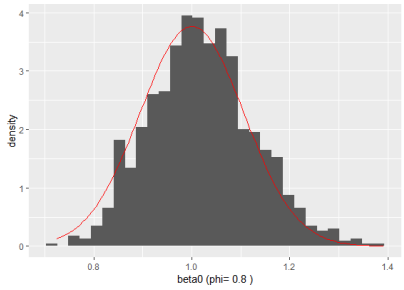

Histogram of beta1 (phi= 0.8 )
Frequency
beta1 (phi= 0.8 )

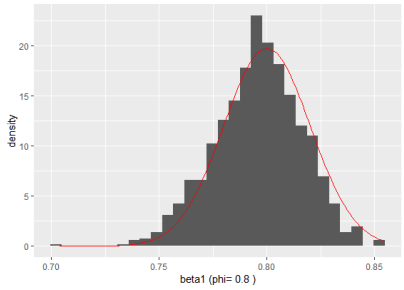

Histogram of beta0 (phi= 0.99 )
Frequency
beta0 (phi= 0.99 )

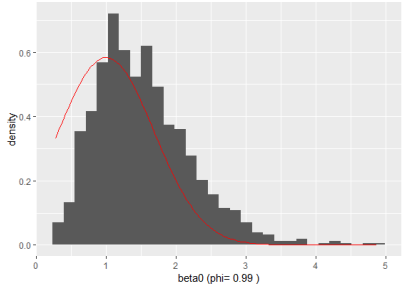

Histogram of beta1 (phi= 0.99 )
Frequency
beta1 (phi= 0.99 )

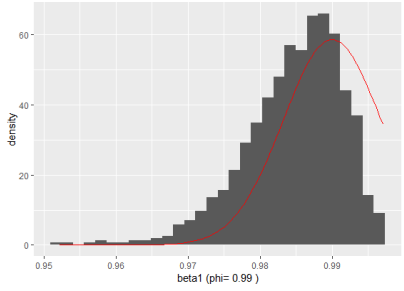

density
beta0 (phi= 0.5 )


density
beta1 (phi= 0.5 )


density
beta0 (phi= 0.8 )


density
beta1 (phi= 0.8 )


density
beta0 (phi= 0.99 )


density
beta1 (phi= 0.99 )

```r
#理论渐近标准误
mu <- 1
sigma <- 1
T_size <- 900
AR1_CM <- function(phi, mu, sigma){
 EX <- mu / (1 - phi)
 EX2 <- (EX)^2 + 1 / (1 - phi^2)
 M <- matrix(c(1, EX, EX, EX2),
 ncol = 2)
 CM <- solve(M) * (sigma ^2) * (1/T_size)
 return(CM)
}
AR1_CM(phi[1], mu = 1, sigma = 1)
#              [,1]          [,2]
# [1,]  0.004444444 -0.0016666667
# [2,] -0.001666667  0.0008333333
AR1_CM(phi[2], mu = 1, sigma = 1)
#              [,1]    [,2]
# [1,]  0.01111111 -2e-03
# [2,] -0.00200000  4e-04
AR1_CM(phi[3], mu = 1, sigma = 1)
#               [,1]          [,2]
# [1,]  0.222222222 -2.211111e-03
# [2,] -0.002211111  2.211111e-05
sd_theory <- vector(mode = "list", length = 3)
for(i in 1:3){
 sd_theory[[i]] <- c(sd_beta0 = sqrt(AR1_CM(phi[i], mu = 1,
 sigma = 1)[1,1]), sd_beta1 =sqrt(AR1_CM(phi[i],
 mu = 1,sigma = 1)[2,2]))
}
names(sd_theory) <- c("phi=0.5","phi=0.8","phi=0.99")
sd_theory
# $phi=0.5
# sd_beta0   sd_beta1
# 0.06666667 0.02886751
```

```
#
# $phi=0.8
# sd_beta0   sd_beta1
# 0.1054093  0.0200000
#
# $phi=0.99
# sd_beta0     sd_beta1
# 0.471404521  0.004702245

#样本渐近标准误
x <- vector("double",901)
for (i in 1:3) {
 x[1] <- mu/(1 - phi[i])
 for (j in 1:900) {
  x[j+1] <- mu + phi[i]*x[j] + rnorm(1,0,1)
 }
 X <- cbind(rep(1,900), x[2:901])
 M <- t(X) %*% X / 900    # 求M矩阵
 CM<- 1/900*1*solve(M)
 print(CM)
 sd_sample<-c(sd_beta0=sqrt(CM[1,1]),sd_beta1=sqrt(CM[2,2]))
 print(sd_sample)
}
#              [,1]          [,2]
#[1,]  0.004203435 -0.0015511974
#[2,] -0.001551197  0.0007781246
#sd_beta0    sd_beta1
#0.06483391 0.02789488
#              [,1]          [,2]
#[1,]  0.011050169 -0.0020451716
#[2,] -0.002045172  0.0004208374
#sd_beta0    sd_beta1
#0.10511978 0.02051432
#              [,1]          [,2]
#[1,]  0.239969153 -2.392425e-03
```

```
#[2,] -0.002392425  2.396276e-05
#sd_beta0     sd_beta1
#0.489866464 0.004895177
```

此时估计系数的样本标准差和渐进标准误均近似为先前的 1/3。