Final_Exam_5206_4206_Fall_2021

Lili Tan lt2846

12/17/2021

Part 0: Instructions

The STAT GU4206/GR5206 final is open notes, open book(s), open computer and online resources are allowed. Students are **not** allowed to communicate with any other people regarding the exam with the exception of the instructor (Gabriel Young) and the TAs. This includes emailing fellow students, using WeChat and other similar forms of communication. If there is any suspicion of one or more students cheating, further investigation will take place. If students do not follow the guidelines, they will receive a zero on the exam and potentially face more severe consequences. The exam will be posted on Friday, 12/17/2021 at 2:00PM (ET). Students are required to submit both the .pdf (or .html) and .rmd files on Canvas by Saturday, 12/18/2021 at 11:59PM (ET). Students will be given the whole 34 hour time frame to complete and upload their knitted file.

A few more recommendations follow:

- Don't forget to submit both the correct .rmd file and at least one of your .html or .pdf files.
- Save your .rmd regularly to avoid any problems if your computer crashes.
- Please try and knit somewhat regularly!
- Please ensure your output is tidy. Do not print pages and pages of data. Doing so will result in points deducted.
- Please stop working on the exam at least 20 min before it's deadline to make sure your RMarkdown file knitts properly.
- If you have a question, please include both the instructor and TA in the email thread.

The **tidyverse** is not formally assessed on the fall 2021 STAT GR5206/GU4206 final exam. Students have the choice to use **tidyverse** or **base R** for all of the following problems. This also applies to any required plotting exercises, i.e., you can choose between **ggplot2** or **base R** plotting functions.

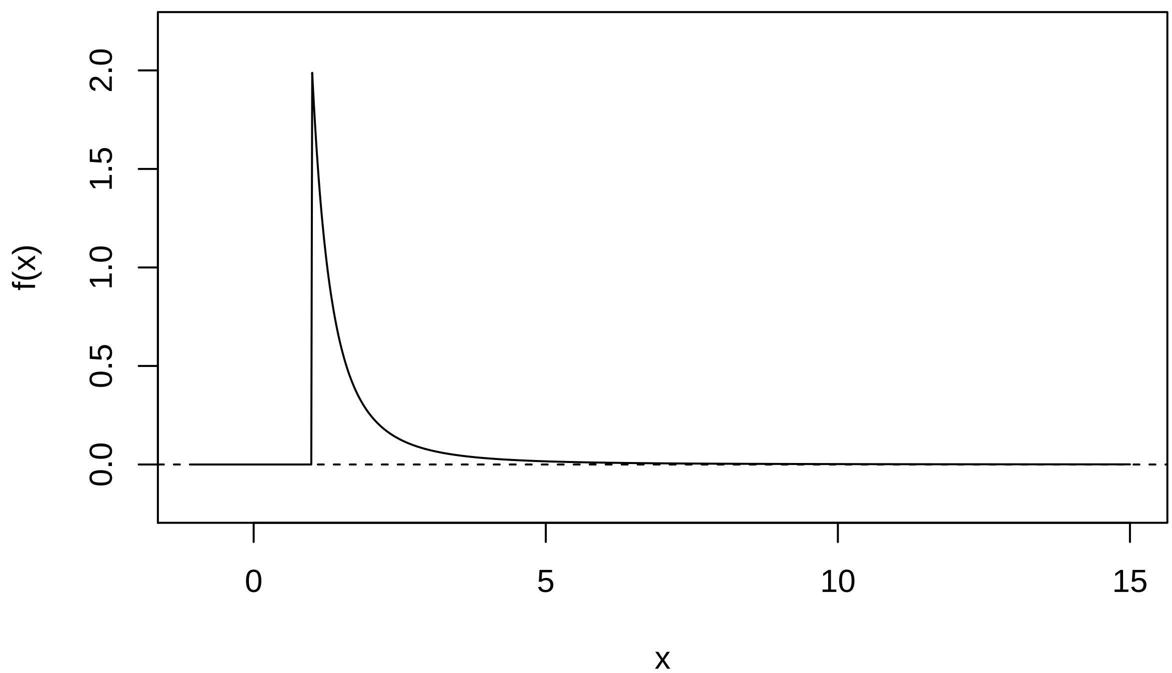
Part I: Inverse Transform Method

Consider the continuous random variable X with density function:

$$f(x) = \begin{cases} \frac{2}{x^3}, & x \geq 1 \\ 0, & \text{otherwise} \end{cases}$$

```
f <- function(x) {
  out <- ifelse(x<1,0,2/x^3)
  return(out)
}
x.plot <- seq(-1,15,length=1000)
```

```r
plot(x.plot,f(x.plot),type="l",ylim=c(-.2,2.2),xlab="x",ylab="f(x)")
abline(h=0,lty=2)
```



Problem 1

Analytically derive the cdf of X, i.e., find $F(x)$. Plot the cdf $F(x)$ over the interval $[-1, 15]$. You only have to show the plot of $F(x)$ for submission.

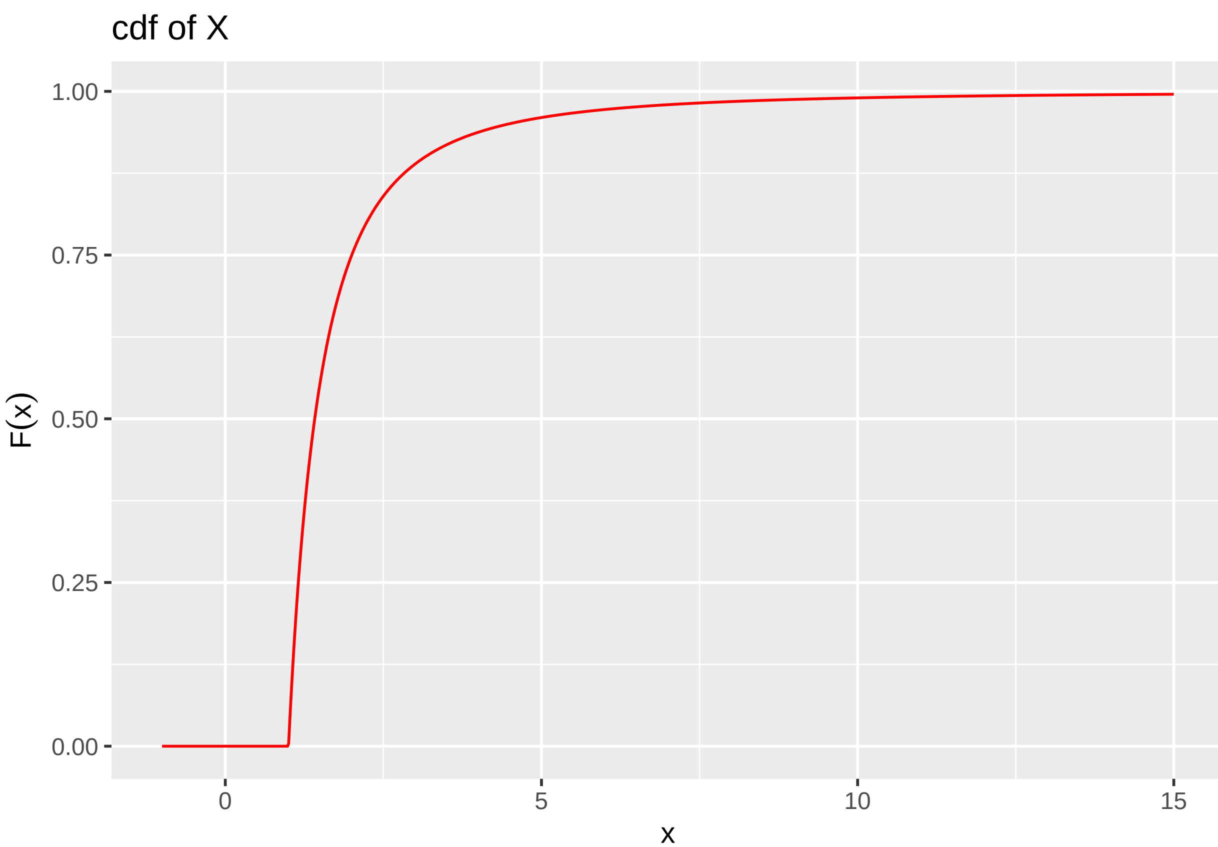
Note: The below derivation is not required for submission but is included for reference.

$$F(x) = \int_1^x \frac{2}{t^3} dt = \begin{cases} 1 - \frac{1}{x^2}, & x \geq 1 \\ 0, & \text{otherwise} \end{cases}$$

Solution

```r
### Solution goes here -------------------------
set.seed(0)

F.cdf <- function(x){
  ifelse(x>=1,1-1/(x^2),0)
}

plot_data_cdf <- data.frame(x=x.plot,F_cdf=F.cdf(x.plot))
```

```
library(ggplot2)
ggplot(data = plot_data_cdf) +
  geom_line(mapping = aes(x = x, y = F_cdf),color="red")+
  labs(title ="cdf of X",y=expression(F(x)))
```



Problem 2

Simulate 10,000 cases from $f(x)$ using the inverse transform method. Plot a histogram of the simulated cases with the true cdf overlayed on the plot. Also, on a separate graphic, plot the empirical cdf of your simulated cases. **Note:** make sure to truncate your x-axis by setting **xlim=c(0,15)**.

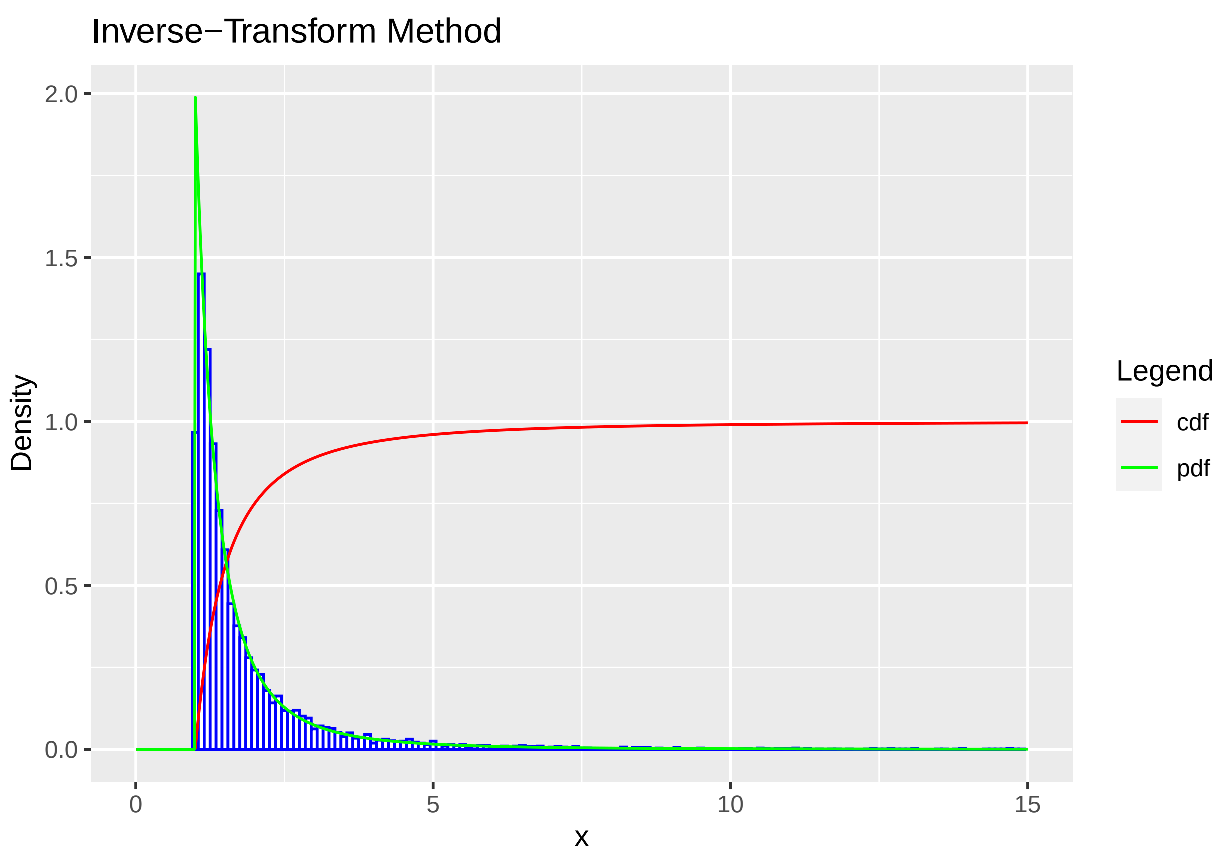
Solution

```
### Solution goes here -------------------------
F.cdf.inv <- function(x){
    ifelse(x<0,0,sqrt(1/(1-x)))
}

inv.sim.x <- F.cdf.inv(runif(10000))
inv.sim.x[1:10]
```

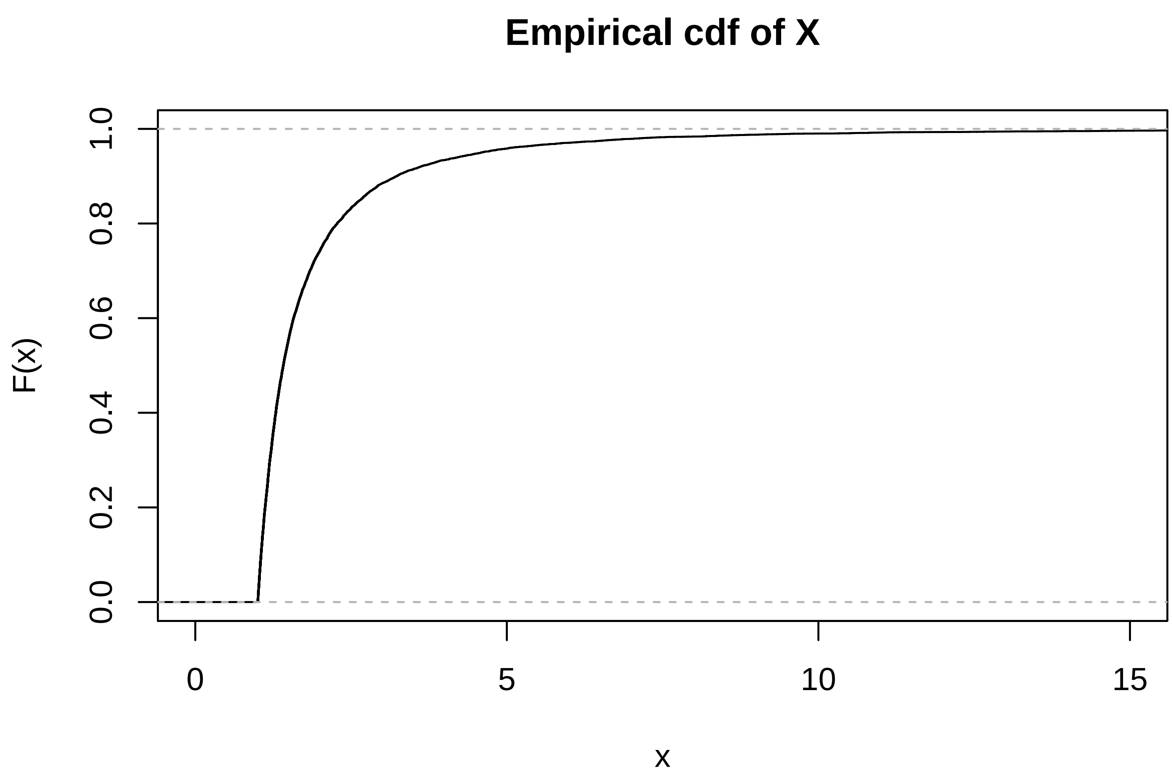
```
##  [1] 3.111315 1.166828 1.262011 1.530071 3.300632 1.119211 3.137120 4.251482
##  [9] 1.717001 1.642025
```

```r
hist_data <- data.frame(x.var=inv.sim.x)
plot_data <- data.frame(x=x.plot, cdf=F.cdf(x.plot), f=f(x.plot))
color <- c("cdf"="red","pdf"="green")

ggplot(hist_data)+
  geom_histogram(mapping=aes(x=x.var,y=..density..),
                 col="blue", fill="white",binwidth=.1)+
  geom_line(plot_data,mapping = aes(x=x, y=cdf,col="cdf"))+
  geom_line(plot_data,mapping = aes(x=x, y=f,col="pdf"))+
  xlim(low=0,high=15)+
  labs(title = "Inverse-Transform Method",
       x="x",y="Density", color="Legend") +
  scale_color_manual(values = color)
```

```
## Warning: Removed 39 rows containing non-finite values (stat_bin).

## Warning: Removed 2 rows containing missing values (geom_bar).

## Warning: Removed 63 row(s) containing missing values (geom_path).

## Warning: Removed 63 row(s) containing missing values (geom_path).
```



```r
plot(ecdf(inv.sim.x),xlim=c(0,15),main="Empirical cdf of X",ylab="F(x)")
```



Part II: Monte Carlo Integration and Accept-Reject

Problem 3

Consider the probability density function

$$f(x) = \begin{cases} \frac{3}{2}cos(x)(sin(x))^2, & 3\pi/2 \leq x \leq 5\pi/2 \\ 0, & \text{otherwise.} \end{cases}$$

Define the function $f(x)$ and evaluate the points $f(2\pi), f(7)$. Note that $f(x)$ is not the same density as problems 1-2.

Solution

```r
### Solution goes here -------------------------
f <- function(x){
  ifelse((x<=(5*pi)/2 & x>=((3*pi)/2)),(3/2)*cos(x)*(sin(x))^2,0)
}

f(2*pi)
```

```
## [1] 8.998559e-32
```

```r
f(7)
```

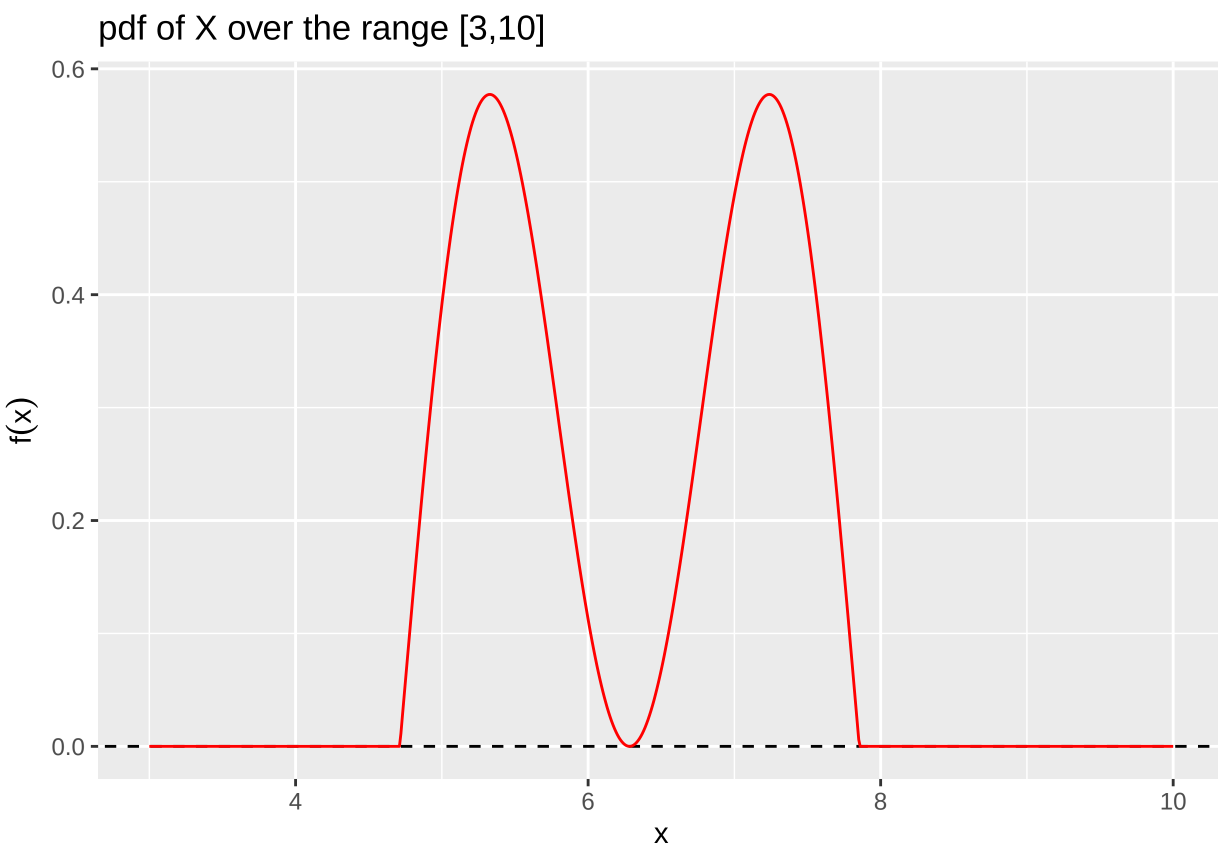
```
## [1] 0.4881118
```

Problem 4

Plot $f(x)$ over the interval $[3, 10]$.

Solution

```r
### Solution goes here -------------------------
x <- seq(3,10,0.01)

plot_data <- data.frame(x=x,f=f(x))

ggplot(data = plot_data) +
  geom_abline(slope=0,intercept=0,linetype = "dashed")+
  geom_line(mapping = aes(x=x, y=f),color="red")+
  labs(title = "pdf of X over the range [3,10]", y=expression(f(x)))
```



Problem 5

Use Monte Carlo Integration to show that $f(x)$ is a valid probability density function, i.e., show that

$$\int_{3\pi/2}^{5\pi/2} f(x)dx = 1.$$

Note the Monte Carlo method will approximate the above integral and your result should be very close to 1. Pick n large enough so that the Monte Carlo integral is within .001 of the truth.

Solution

```
### Solution goes here -------------------------
n <- 1000000
X <- runif(n,(3*pi)/2,(5*pi)/2)
g.over.p <- function(x){
  return(f(x)*pi)
}
mean(g.over.p(X))
```

```
## [1] 0.9997943
```

Problem 6

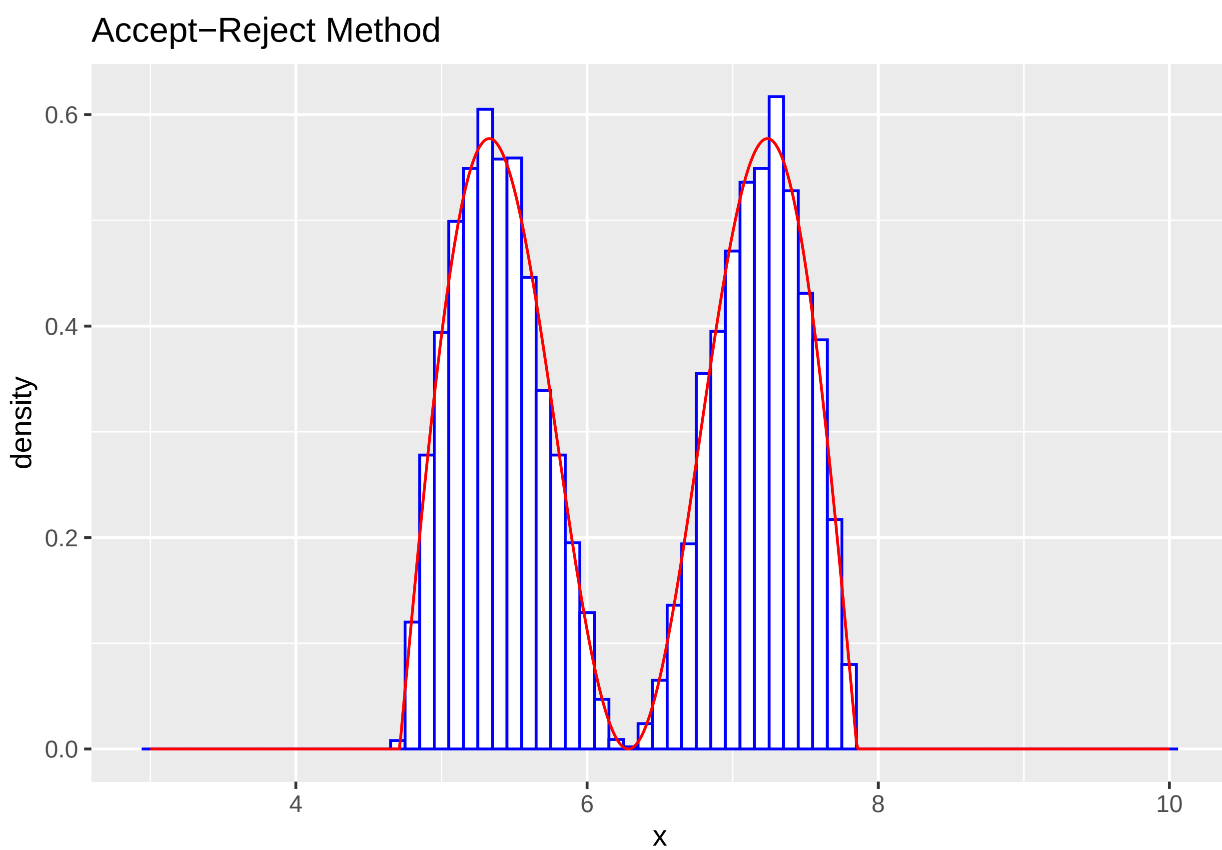
Use the accept-reject method to simulate 10,000 draws from $f(x)$. Your envelope function must satisfy $e(x) \geq f(x)$ for all x but it does not have to be perfect.

Solution

```
### Solution goes here -------------------------
#choose g(x) to be uniform((3*pi)/2, (5*pi)/2)
f.max <- max(f(X))

e <- function(x) {
  ifelse((x < (3*pi)/2 | x > (5*pi)/2), Inf, f.max) }

x.accept.reject <- function(numb){
  n.samps <- numb
  n <- 0
  samps <- numeric(n.samps)
  while (n<n.samps){
  y <- runif(1,(3*pi)/2,(5*pi)/2)
  u <- runif(1)
  if(u < f(y)/e(y)){
    n <- n+1
    samps[n] <- y
  }
  }
  return(samps)
}

x.accept.reject.sim <- x.accept.reject(10000)
x.accept.reject.sim[1:10]
```

```
##  [1] 5.751511 5.485677 7.253906 7.552184 7.680949 7.298716 7.135175 4.951489
##  [9] 5.669487 5.581970
```

```r
hist_data <- data.frame(x.var=x.accept.reject.sim)
plot_data <- data.frame(x=x,f=f(x))

ggplot(hist_data)+
  geom_histogram(mapping=aes(x=x.var,y=..density..),
                 col="blue", fill="white",binwidth=.1)+
  geom_line(plot_data,mapping = aes(x = x, y = f),col="red")+
  labs(title = "Accept-Reject Method",x="x")
```



Part III: Simulating an AR(1) Proccess

Consider the autoregressive lag 1 model (AR(1)) defined by

$$\epsilon_t = \phi\epsilon_{t-1} + z_t, \quad t = 2, 3, \ldots, n,$$

and

$$z_t \overset{iid}{\sim} N(0, \sigma^2), \quad \epsilon_1 = \text{constant}.$$

The autoregressive parameter ϕ must satisfy $|\phi| < 1$, otherwise the series will exhibit unpredictable behavior, i.e., random walk ($|\phi| = 1$) or explosive ($|\phi| > 1$).

Problem 7

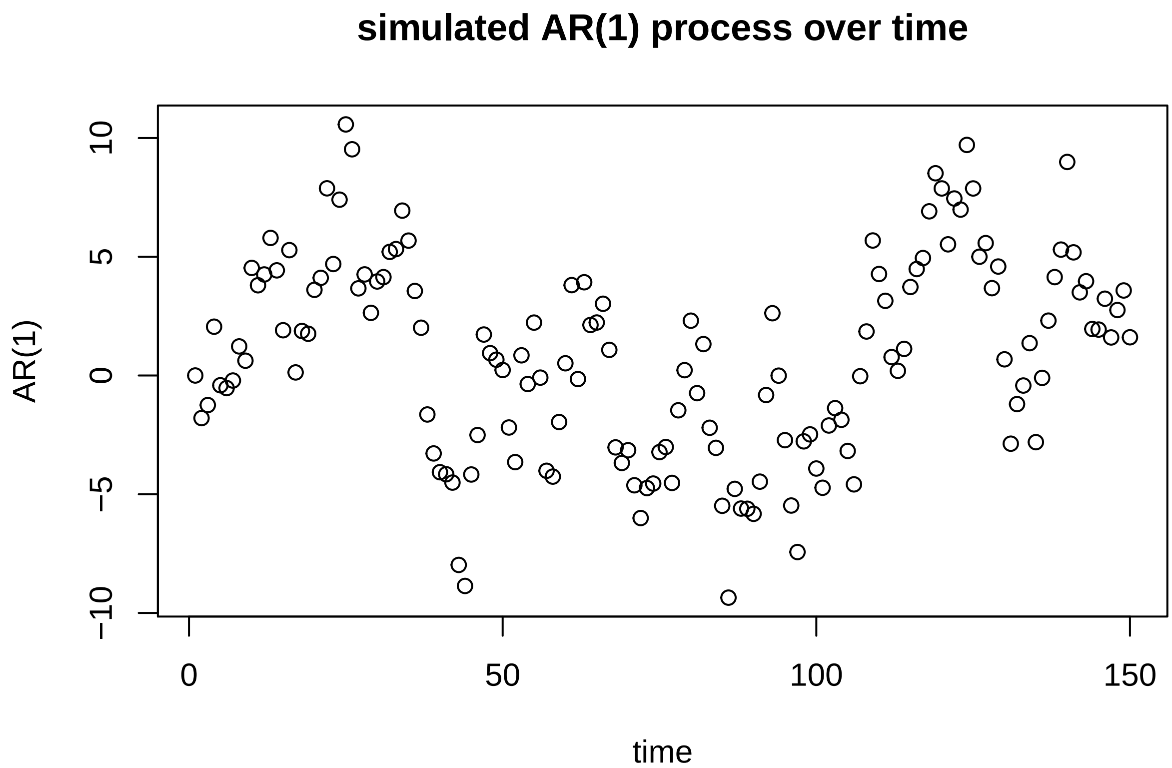
Create a function named **my__AR1** that simulates n realizations from the AR(1) model. Your function should have inputs **n, phi, sigma, e1**; where **n** is the number of simulated cases, **phi** is the AR(1) parameter, **sigma** is the noise standard deviation and **e1** is the initial value with default **e1=0**. Test your function using **set.seed(2)** and inputs **my__AR1(n=150,phi=.9,sigma=2,e1=0)**. Plot your simulated AR(1) process over time, i.e., plot your AR(1) process over the sequence **1:150**.

Solution

```r
### Solution goes here -------------------------
my_AR1 <- function(n,phi,sigma,e1=0){
  stopifnot(abs(phi)<1)
  n.samps <- n
  n <- 0
  samps <- numeric(n.samps)
  samps[1] <- e1
  for (i in 2:n.samps){
    z <- rnorm(1,0,sigma)
    e <- samps[i-1]
    samps[i] <- phi*e+z
  }
  return(samps)
}

set.seed(2)

AR1_sim <- my_AR1(n=150,phi=0.9,sigma=2,e1=0)
AR1_sim[1:10]
```

```
##  [1]  0.0000000 -1.7938291 -1.2447478  2.0554176 -0.4108755 -0.5302914
##  [7] -0.2124217  1.2247299  0.6228609  4.5295226
```

```r
plot(1:150,AR1_sim,main="simulated AR(1) process over time",xlab="time",ylab="AR(1)")
```



Part IV: Maximum Likelihood Estimation

Consider estimating the noise variance σ^2 and the AR(1) parameter ϕ using maximum likelihood estimation. This problem is non-trivial and requires some heavy prerequisite knowledge. The following definitions lack complete descriptions and students must take these formulas on faith. You can learn more about these relations in a time series course.

The AR(1) likelihood function of interest can be formulated by:

$$\mathcal{L}(\sigma^2, \phi; \epsilon_1, \cdots, \epsilon_n) = [\prod_{t=2}^{n} f(\epsilon_t|\epsilon_{t-1}; \sigma^2, \phi)] * f(\epsilon_1; \sigma^2, \phi)$$

Further, the conditional distribution of $\epsilon_t|\epsilon_{t-1}$ is

$$\epsilon_t|\epsilon_{t-1} \sim N(\phi\epsilon_{t-1}, \sigma^2),$$

and the distribution of ϵ_1 is

$$\epsilon_1 \sim N(0, \sigma^2/(1-\phi^2)).$$

The likelihood can be formulated by a product of normal densities (**Hint: dnorm()**).

Problem 8

Define the function **L.AR1** which computes the likelihood $\mathcal{L}(\sigma^2, \phi; \epsilon_1, \cdots, \epsilon_n)$ of the AR(1) model. **L.AR1** should be a function of the parameter vector **theta** and data vector **ts_data**. Note that **theta** must be a vector of length 2, i.e., $\theta = (\sigma^2, \phi)$.

Test your function **L.AR1** on the simulated time series **AR1_sim** from problem 7 using inputs **L.AR1(theta=c(5,.9),ts_data=AR1_sim)**. Note that your function should return a very small number on the order of **10^{-145}**.

Solution

```
### Solution goes here -------------------------
L.AR1 <- function(theta, ts_data){
  stopifnot(length(theta)==2)
  pre <- 1
  for (i in 2:length(ts_data)){
    pre <- pre*dnorm(ts_data[i],ts_data[i-1]*theta[2],sqrt(theta[1]))
  }
  pre*dnorm(ts_data[1],0,sqrt(theta[1]/(1-theta[2]^2)))
}

L.AR1(theta=c(5,.9),ts_data=AR1_sim)
```

```
## [1] 1.300037e-145
```

Problem 9

Define the function **neg.ll.AR1** which computes the negative log-likelihood of the AR(1) model. Similar to problem 8, **neg.ll.AR1** should be a function of the parameter vector **theta** and data vector **ts_data**. Test your function **neg.ll.AR1** on the simulated time series **AR1_sim** from problem 7 using inputs **neg.ll.AR1(theta=c(5,.9),ts_data=AR1_sim)**. Note that your function should return a number around 333.

Solution

```
### Solution goes here -------------------------
neg.ll.AR1 <- function(theta,ts_data=AR1_sim){
  stopifnot(length(theta)==2)
  pre <- 0
  for (i in 2:length(ts_data)){
    pre <- pre+dnorm(ts_data[i],ts_data[i-1]*theta[2],sqrt(theta[1]),log=T)
  }
  -(pre+dnorm(ts_data[1],0,sqrt(theta[1]/(1-theta[2]^2)),log=T))
}

neg.ll.AR1(theta=c(5,.9),ts_data=AR1_sim)
```

```
## [1] 333.6124
```

Problem 10

Use the **nlm()** function to minimize the negative log-likelihood **neg.ll.ar1** based on the simulated AR(1) model **AR1_sim**. Minimizing **neg.ll.ar1** will yield the maximum likelihood estimates of noise variance σ^2 and autoregressive parameter ϕ. Before minimizing the negative log-likelihood, make sure to center the **AR1_sim** dataset and use starting value **p=c(5,.9)**. Note that there will be at least one warning from the **nlm()** output, which is suppressed using **warning=F**.

Compare your estimated values MLE values $\hat{\sigma}_{mle}$ and $\hat{\phi}_{mle}$ to the true parameters $\sigma = 2$ and $\phi = .9$. Also compare the sample variance of **AR1_sim** with the quantity

$$\frac{\hat{\sigma}^2_{mle}}{1 - \hat{\phi}^2_{mle}}.$$

Solution

```
### Solution goes here -------------------------
#center AR1_sim
AR1_sim.center <- AR1_sim-mean(AR1_sim)
theta.mle <- nlm(neg.ll.AR1,ts_data=AR1_sim.center,p=c(5,.9))$estimate

sigma.sqr.mle <- theta.mle[1]
sigma.sqr.mle
```

```
## [1] 4.892679
```

```
phi.mle <- theta.mle[2]
phi.mle
```

```
## [1] 0.8410903
```

```
var(AR1_sim.center)
```

```
## [1] 17.35734
```

```
sigma.sqr.mle/(1-phi.mle^2)
```

```
## [1] 16.72327
```

MLE values $\hat{\sigma}_{mle} = \sqrt{4.892679} = 2.21194$ and $\hat{\phi}_{mle} = 0.8410903$, and they are close to the true parameters $\sigma = 2$ and $\phi = .9$

sample variance of **AR1_sim** is 17.35734 which is really close to the quantity $\frac{\hat{\sigma}^2_{mle}}{1-\hat{\phi}^2_{mle}} = 16.72327$

Part V: SPLIT/APPLY/COMBINE

Suppose that you are data scientist working at Boeing Airlines (BA). Further, suppose that you are interested in how your company's stock (BA) correlates with other major cooperate entities included in the Dow Jones Industrial Average (DJIA). The csv file **close_data.csv** includes the daily closing prices of the Dow Jones Industrial Average (DJIA) recorded from 2021-01-01 to 2021-11-02. Each stock is recorded over 210 trading days. Among the 30 companies in the DJIA, Boeing (BA) is removed, resulting in 29 tickers. The closing prices of Boeing (BA) are summarized in a separate csv file **BA.csv**.

```
BA <- read.csv("BA.csv")
dim(BA)
```

```
## [1] 210   3
```

```r
close_data <- read.csv("close_data.csv")
dim(close_data)
```

```
## [1] 6090    3
```

```r
names(close_data)
```

```
## [1] "day"    "ticker" "close"
```

Problem 11

Create a new dataframe named **df_AXP** that contains only the rows related to ticker **AXP**. Your sub-dataframe should have dimension (210×3). Show the **head** and **dim** of the dataframe **df_AXP**.

Solution

```r
### Solution goes here -------------------------
df_AXP <- close_data[close_data$ticker=="AXP",]
dim(df_AXP)
```

```
## [1] 210   3
```

```r
head(df_AXP)
```

```
##      day ticker  close
## 2731   1    AXP 118.04
## 2732   2    AXP 118.67
## 2733   3    AXP 123.06
## 2734   4    AXP 121.66
## 2735   5    AXP 121.78
## 2736   6    AXP 121.06
```

Problem 12

For a given day (t), the returns (R_t) of a financial object are defined by

$$R_t = \frac{P_{t+1} - P_t}{P_t}, \quad t = 1, \ldots, n-1.$$

In our setting, the closing price represents P_t and the data is recorded over $n = 210$ trading days, i.e., $t = 1, 2, \ldots, 209$. Define a function **compute_return** that computes a vector of the returns of a specific stock and appends this column onto your existing dataframe. Test your function **compute_return** on the **df_AXP** dataframe from problem 11, which should yield in a new dataframe of dimension (209×4). Show the **head** and **dim** of **compute_return(df_AXP)**.

Solution

```r
### Solution goes here -------------------------
compute_return <- function(df){
  p <- df$close
  n <- length(p)
  R <- numeric(n)
  for (t in 1:(n-1)){
    R[t] <- (p[t+1]-p[t])/p[t]
  }
  df$returns <- R
  return(df[-n,])
}

df_AXP.return <- compute_return(df_AXP)
head(df_AXP.return)
```

```
##      day ticker  close       returns
## 2731   1    AXP 118.04  0.0053371484
## 2732   2    AXP 118.67  0.0369933435
## 2733   3    AXP 123.06 -0.0113765157
## 2734   4    AXP 121.66  0.0009863143
## 2735   5    AXP 121.78 -0.0059123091
## 2736   6    AXP 121.06  0.0046258468
```

```r
dim(df_AXP.return)
```

```
## [1] 209   4
```

Problem 13

Create a new **BA** dataframe by evaluating **compute_return(BA)**. Assign the updated BA dataframe as **BA_new**. Again, show the **head** and **dim** of **compute_return(BA)**.

Solution

```r
### Solution goes here -------------------------
BA_new <- compute_return(BA)
head(BA_new)
```

```
##   day ticker  close      returns
## 1   1     BA 202.72  0.043952269
## 2   2     BA 211.63 -0.002835165
## 3   3     BA 211.03  0.007960991
## 4   4     BA 212.71 -0.013210535
## 5   5     BA 209.90 -0.014816585
## 6   6     BA 206.79  0.007834088
```

```r
dim(BA_new)
```

```
## [1] 209   4
```

Problem 14

Run the **SPLIT/APPLY/COMBINE** model on the full dataset **close_data**, i.e., split by **ticker** and apply the function **compute_return** on each sub-dataframe. To solve this problem, use an appropriate function from the **plyr** package in conjunction with your **compute_return** function. Assign your new dataframe as **close_data_new** and compute its dimension. Note that $209 * 29 = 6061$.

Solution

```
library("plyr")
### Solution goes here -------------------------
close_data_new <- ddply(close_data,.(ticker),compute_return)
dim(close_data_new)
```

```
## [1] 6061    4
```

Part VI: More SPLIT/APPLY/COMBINE

Suppose that Boeing has a fixed amount of money to invest into itself (Y) and another company (X). We must decide what fraction (α) of money to invest in $Y = BA$ and $(1-\alpha)$ in stock X. The total returns (W) are modeled by the relation

$$W = \alpha X + (1-\alpha)Y.$$

To determine the optimal percentage α, we minimize the variance of W, i.e., minimize the expression

$$var(W) = \alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2\alpha(1-\alpha)cov(X,Y).$$

The univariate derivative of total variance $var(W)$ with respect to α is:

$$\frac{d}{d\alpha}var(W) = 2\alpha\sigma_X^2 + 2\alpha\sigma_Y^2 - 2\sigma_Y^2 + 2cov(X,Y) - 4\alpha cov(X,Y)$$

Consequently, the optimal value of α is

$$\alpha = \frac{\sigma_Y^2 - cov(X,Y)}{\sigma_X^2 + \sigma_Y^2 - 2cov(X,Y)}.$$

Hence, we can choose a reasonable estimator of α as

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - c\hat{o}v(X,Y)}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2c\hat{o}v(X,Y)}.$$

In **R**, you can easily estimate α using:

```
#(var(Y)-cov(X,Y))/(var(X)+var(Y)-2*cov(X,Y))
```

Problem 15

Compute the estimated α based on **Y=BA** versus **X=AXP**. Conclude what percentage to invest in each stock.

Solution

```r
### Solution goes here --------------------------
Y <- BA_new$returns
X <- df_AXP.return$returns
alpha.hat <- (var(Y)-cov(X,Y))/(var(X)+var(Y)-2*cov(X,Y))
alpha.hat
```

```
## [1] 0.7133757
```

```r
1-alpha.hat
```

```
## [1] 0.2866243
```

we get $\alpha = 0.7133757$ which means we invest 71.34% in AXP's stock and 28.66% in BA's stock.

Problem 16

Now suppose that Boeing is interested in a stock buy-back but also wants to invest in one other company. To determine which company to invest in, the data scientist compares the sample correlations of BA versus all other tickers in the DJIA. Further, you also store the estimated α value for each DJIA ticker verses **BA**. Define a function **BA_relationships** with inputs: **ticker**, **BA** and **data**. The **BA_relationships** function should output a vector of length 2 with the sample correlation and estimated alpha value. Test **BA_relationships** on the ticker **AXP**.

Solution

```r
### Solution goes here --------------------------
BA_relationships <- function(ticker,BA=BA_new$returns,df=close_data_new){
  Y <- BA
  X <- df[df$ticker==ticker,4]
  correlation <- cor(X,Y)
  alpha.hat <- (var(Y)-cov(X,Y))/(var(X)+var(Y)-2*cov(X,Y))
  result <- c(correlation,alpha.hat)
  names(result) <- c("sample correlation","estimated alpha")
  return(result)
}

BA_relationships(ticker="AXP",df=close_data_new)
```

```
## sample correlation    estimated alpha
##          0.4640197          0.7133757
```

Problem 17

Run a SPLIT/APPLY/COMBINE procedure to store sample correlations of BA verses all stocks and the corresponding estimated α values. Display the head and dimension of your resulting dataframe after sorting its rows by correlation. Note that your resulting dataframe should have dimension (29×2) and **GS** has the highest correlation with **BA** among all tickers in the DJIA.

Solution

```r
### Solution goes here -------------------------
ticker.names <- levels(factor(close_data_new$ticker))
result <- ldply(ticker.names,BA_relationships)
rownames(result) <- ticker.names
colnames(result) <- c("sample correlation","estimated alpha")
result[order(result$`sample correlation`,decreasing=T),]
```

```
##      sample correlation estimated alpha
## GS          0.540726090      0.80270004
## HON         0.505522294      0.95874395
## CAT         0.502046578      0.75439366
## DOW         0.483934972      0.61474649
## CVX         0.473727824      0.74516543
## AXP         0.464019708      0.71337575
## DIS         0.404057257      0.76289780
## TRV         0.339749797      0.82412935
## JMP         0.319403745      0.07119362
## INTC        0.319067012      0.51662150
## MCD         0.290008934      0.92167584
## WBA         0.275219065      0.62181335
## WMT         0.239301339      0.88490435
## IBM         0.236217709      0.68060083
## V           0.231540951      0.73882409
## AAPL        0.211040933      0.69029398
## CSCO        0.199752967      0.84083927
## HD          0.170504556      0.79596456
## MMM         0.162762613      0.80429869
## MSFT        0.149401959      0.76365741
## CRM         0.148253042      0.67126305
## NKE         0.144252813      0.60294784
## AMGN        0.113581167      0.76302881
## KO          0.086739017      0.87594124
## VZ          0.080791587      0.87347032
## JNJ         0.067253276      0.86699714
## UNH         0.035408540      0.76177485
## MRK         0.023706507      0.72606222
## PG          0.002398583      0.84694613
```

```r
dim(result)
```

```
## [1] 29  2
```

Part VII: MCMC of the AR(1) Model

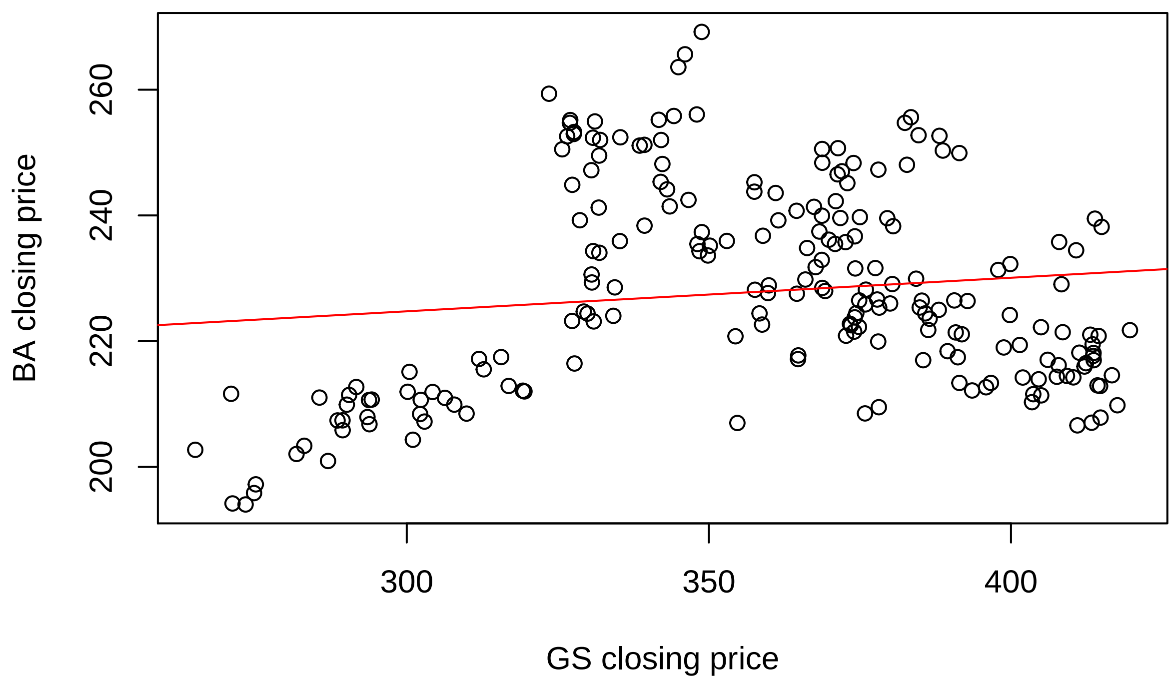
Consider the linear regression model:

$$Y_t = \beta_0 + \beta_1 X_t + \epsilon_t, \quad t = 1, 2, \ldots, n,$$

where Y_t is the closing price of Boeing (**BA**) and X_t is the closing price of Goldman Sachs (**GS**), each measured on day t. Below shows the linear regression analysis on Y versus X using the **lm()** function. The vector **res** represents the residuals of your linear model.

Note: we are using the **closing price** and not the **returns** for this analysis.

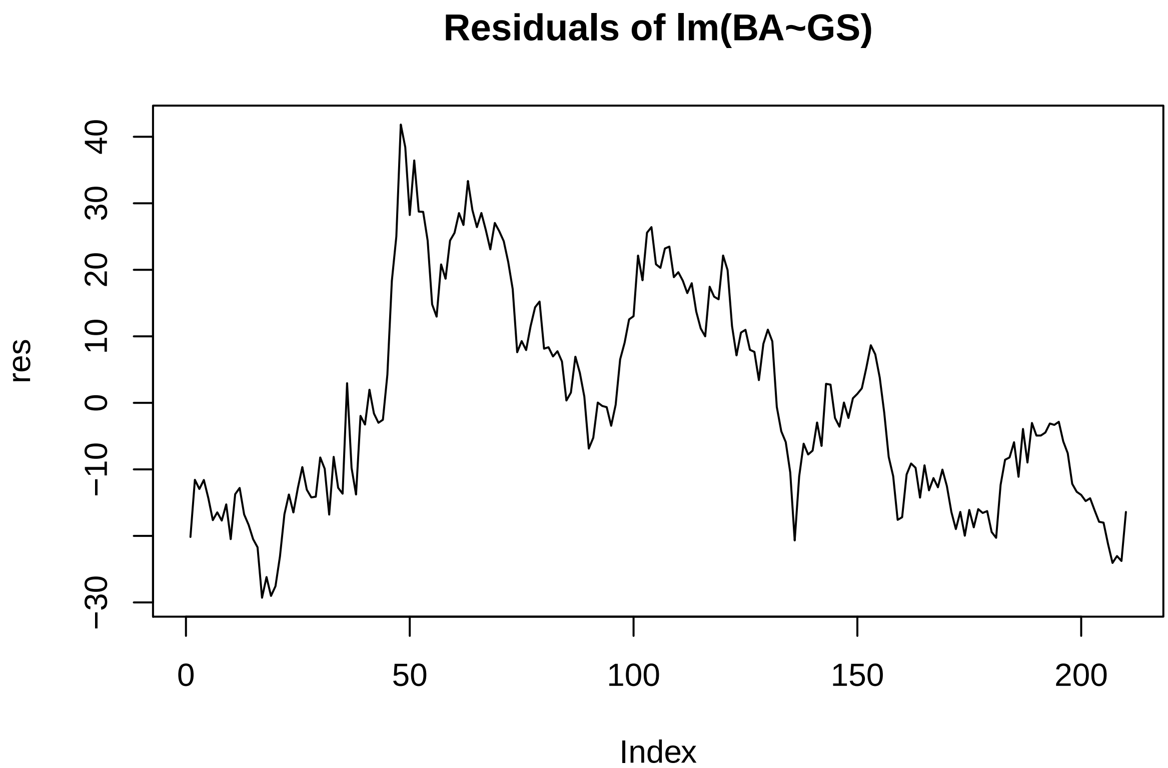
```r
# Define Y
Y <- BA$close

# Define X
GS_data <- close_data[close_data$ticker=="GS",]
X <- GS_data$close

# Run linear model
model <- lm(Y~X)

# Plot Y versus X
plot(X,Y,xlab="GS closing price",ylab="BA closing price")
abline(model,col="red")
```




```r
# define residuals
res <- residuals(model)

# Display coefficients
model$coefficients
```

```
##   (Intercept)            X
## 208.73839322   0.05337265
```

```r
# Plot residuals
plot(res,type="l",main="Residuals of lm(BA~GS)")
```



Further, suppose that the error structure ϵ_t is modeled by an autoregressive lag 1 (AR(1)) process defined by

$$\epsilon_t = \phi\epsilon_{t-1} + z_t, \quad t = 2, 3, \ldots, n z_t \overset{iid}{\sim} N(0, \sigma^2)$$

The full statistical model is comprised 4 parameters: $\beta_0, \beta_1, \sigma^2, \phi$. For Problems 18 and 19, we restrict our attention to just the noise variance σ^2 and the AR(1) parameter ϕ, i.e., $\theta = (\sigma^2, \phi)$.

Problem 18

Estimate the noise variance σ^2 and the AR(1) parameter ϕ of the residuals **res** via maximum likelihood. You can easily solve this problem by minimizing **neg.ll.ar1** from problems 9 and 10. Display your estimated parameters based the **nlm()** output.

Solution

```r
### Solution goes here -------------------------
nlm(neg.ll.AR1,ts_data=res,p=c(5,.9))$estimate
```

```
## [1] 21.4451749  0.9579138
```

Problem 19

Consider using a Bayesian approach to model the AR(1) error structure introduced in Part VII. Assuming the priors

$$\sigma^2 \sim \text{gamma}(\alpha = 5, \beta = 4),$$

and

$$\phi \sim \text{unif}(0, 1),$$

your goal is to estimate the posterior $\pi(\theta|e_1, e_2, \ldots, e_n)$, where θ is the parameter vector $\theta = (\sigma^2, \phi)$ and $e_1, e_2, \ldots, e_n$ are the residuals **res**. Using Markov Chain Monte Carlo, estimate the posterior using the Metropolis Hastings algorithm from page 65 of SET11 lecture notes. Your simulated posterior $\theta_{(t)}$ should be estimated using 100,000 MCMC iterations and discarding a 20% burn-in period. The resulting matrix $\theta_{(t)}$ will have dimension (2×80000).

Note: If your chain fails to run, try running it a few times to see if it recovers from the initial draw $\theta^{(0)}$. You should be able to set some seed so that your Markdown file always knits.

Display traceplots and histograms of chains $\sigma^2_{(t)}$ and $\phi_{(t)}$.

Solution

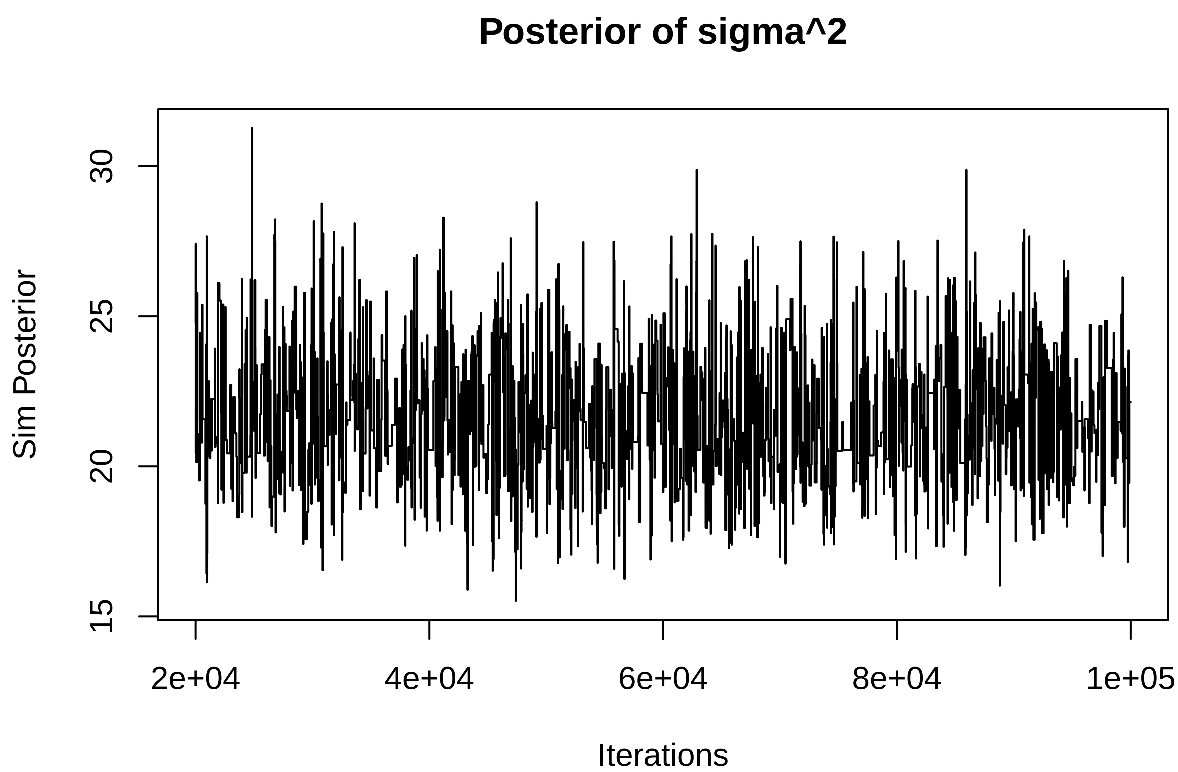
```r
### Solution goes here -------------------------
set.seed(1)

sim.theta <- function(n){
  sigma.sqr <- rgamma(n,shape=5,scale=4)
  phi <- runif(n)
  return(c(sigma.sqr,phi))
}

R <- 100000
beta_t_matrix <- matrix(0,nrow=2,ncol=R+1)
beta_t_matrix[,1] <- sim.theta(1)
for (r in 1:R) {
beta_star <- sim.theta(1)
beta_t <- beta_t_matrix[,r]
R_MH <- L.AR1(beta_star,res)/L.AR1(beta_t,res)
Sample.index <- sample(c(1,2),1,prob=c(min(R_MH,1),1-min(R_MH,1)))
if(Sample.index==1) {
  beta_t_matrix[,r+1] <- beta_star}
else {
  beta_t_matrix[,r+1] <- beta_t}
}

#posterior Bayes' estimators
theta_final <- beta_t_matrix[,-c(1:20001)]
BE_sigma.sqr <- mean(theta_final[1,])
BE_sigma.sqr
```

```
## [1] 21.63132
```

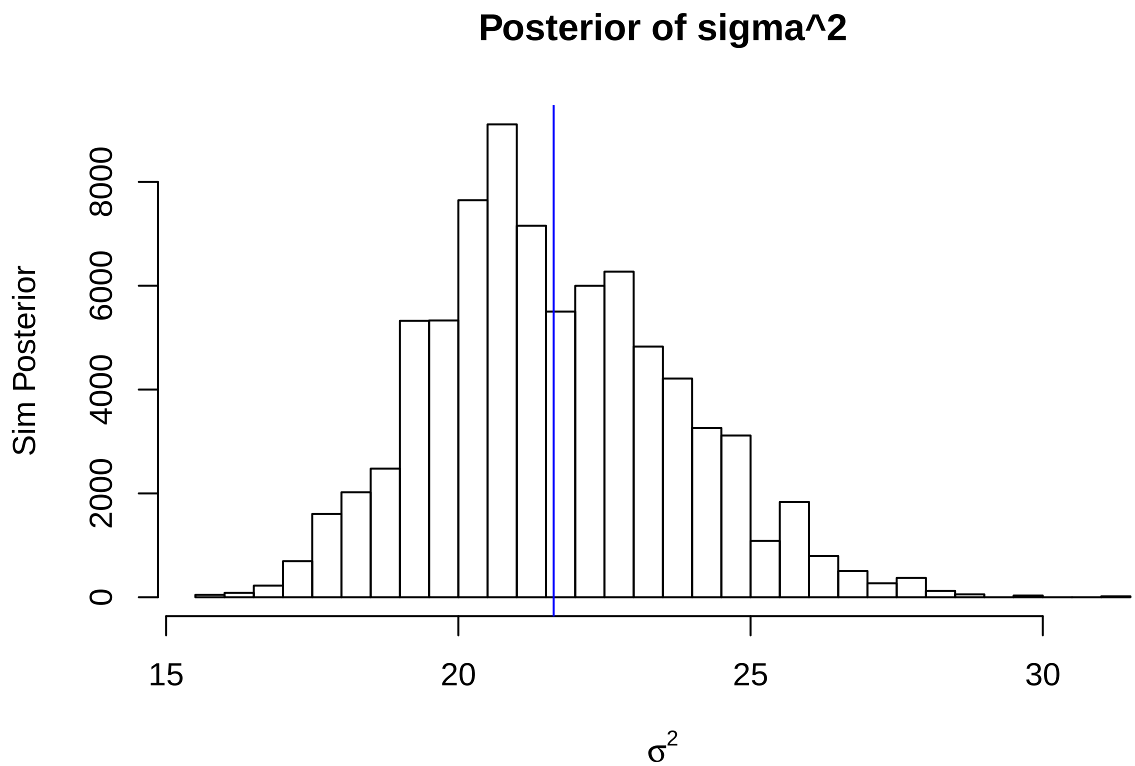
```r
BE_phi <- mean(theta_final[2,])
BE_phi
```

```
## [1] 0.9565255
```

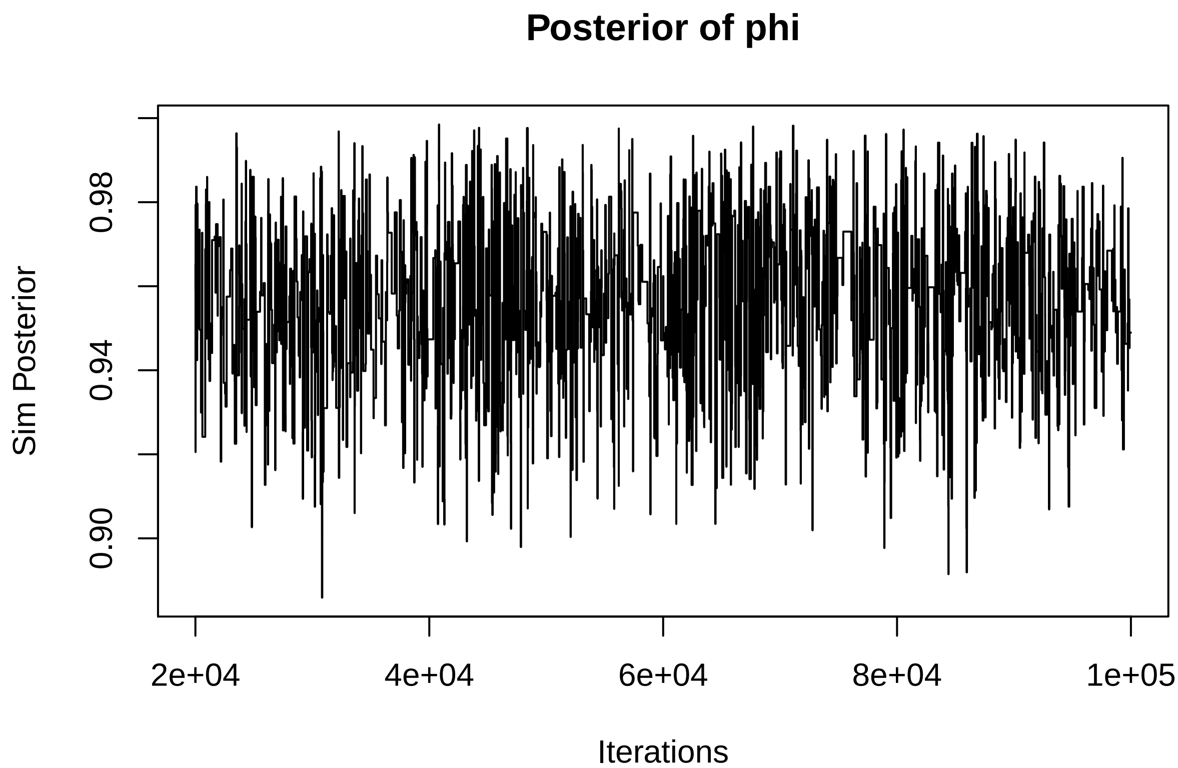
```r
#plot
plot(20002:(R+1),theta_final[1,],type="l",main="Posterior of sigma^2",
     xlab="Iterations",ylab="Sim Posterior")
```



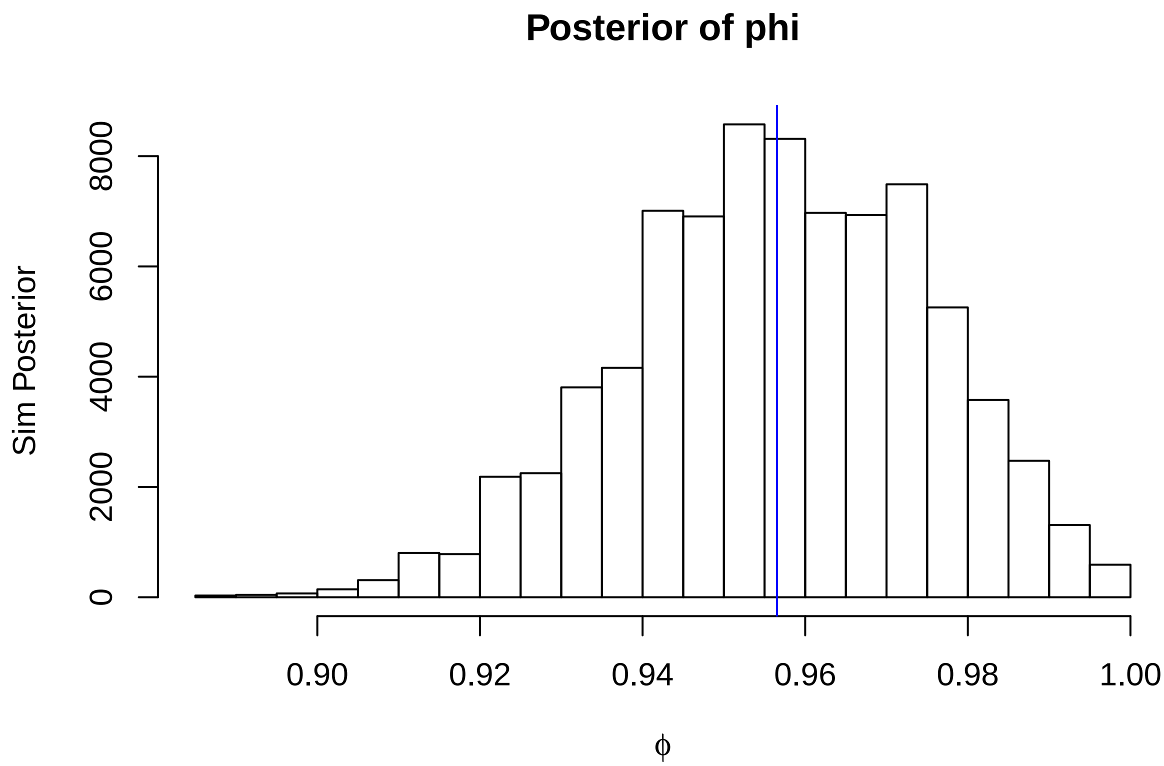

```r
hist(theta_final[1,],xlab=expression(sigma^2),breaks=40,
     ylab="Sim Posterior",main="Posterior of sigma^2")
abline(v=BE_sigma.sqr ,col="blue")
```




```r
plot(20002:(R+1),theta_final[2,],type="l",main="Posterior of phi",
     xlab="Iterations",ylab="Sim Posterior")
```




```r
hist(theta_final[2,],xlab=expression(phi),breaks=40,
     ylab="Sim Posterior",main="Posterior of phi")
abline(v=BE_phi ,col="blue")
```



Problem 20

Compute the Bayes' estimates for both σ^2 and ϕ based on your simulated chain, after discarding the 20% burn-in period. **Note:** Simply take the sample mean of your simulated chains!

Solution

```
### Solution goes here -------------------------
theta_final <- beta_t_matrix[,-c(1:20001)]
BE_sigma.sqr <- mean(theta_final[1,])
BE_sigma.sqr
```

```
## [1] 21.63132
```

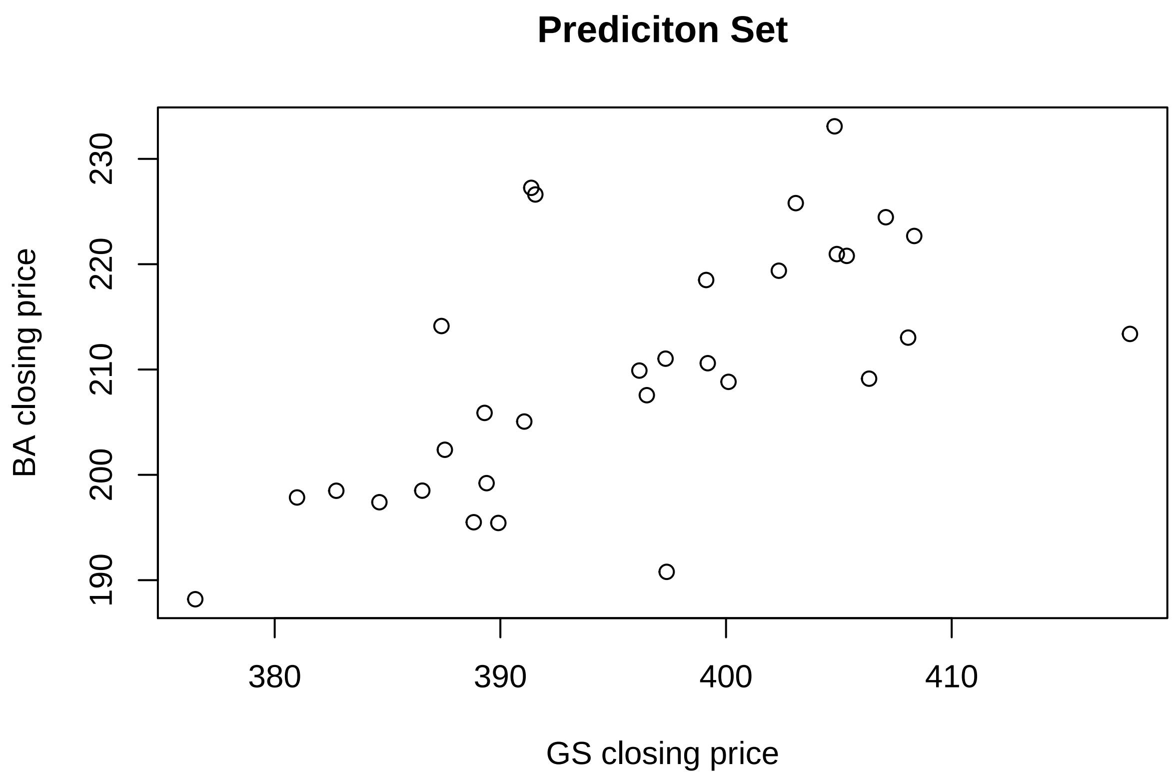
```
BE_phi <- mean(theta_final[2,])
BE_phi
```

```
## [1] 0.9565255
```

Part VIII: Forecasting

Consider the csv file **Pred_BA_GS.csv**, which includes the closing prices from **Y=BA** and **X=GS** recorded over the time period 2021-11-03 to 2021-12-30. The dataframe has dimension (31×2).

```r
pred_data <- read.csv("Pred_BA_GS.csv")
#dim(pred_data)
Y_test <- pred_data$BA
X_test <- pred_data$GS
plot(X_test,Y_test,xlab="GS closing price",ylab="BA closing price",
     main="Prediciton Set")
```



Problem 21

For this task, you will forecast the closing price of Boeing (**BA**) based on the linear regression model

$$Y_t = \beta_0 + \beta_1 X_t + \epsilon_t,$$

with AR(1) errors

$$\epsilon_t = \phi\epsilon_{t-1} + z_t,$$

$$z_t \overset{iid}{\sim} N(0, \sigma^2)$$

To accomplish this task: 1) Manually compute the linear component's prediction of $Y = BA$ using $Y_{pred} = \hat{\beta}_0 + \hat{\beta}_1 X$, where X represents the test cases of GS. This should result in a vector of length 31. 2) Compute the vector of estimated errors or residuals **res_pred <- Y_test - Y_pred**. 3) Forecast Y by combining Y_{pred} from step (1) and the estimated errors from step (2). 4) Compute the mean square prediction error of your forecasted BA values versus the true values **Y_test**. Compare this number to $\hat{\sigma}^2$. 5) Create a graphic showing the difference between your forecasted BA values and true values **Y_test**.

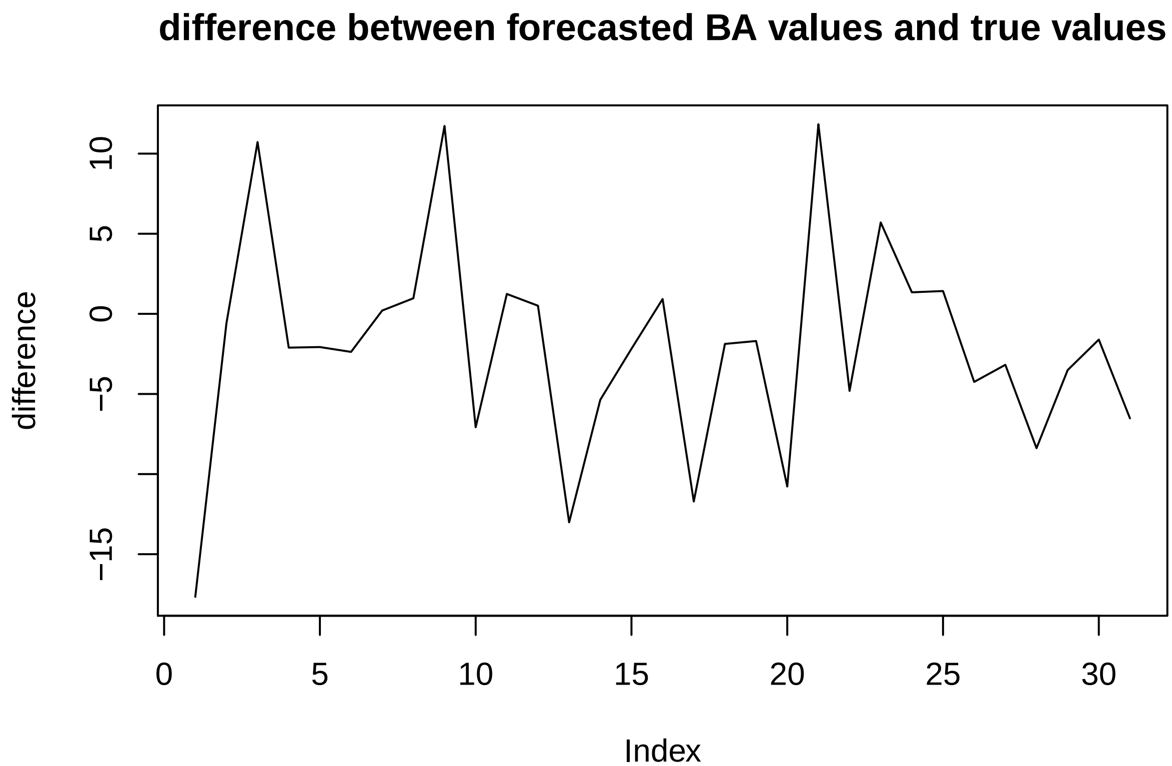
Notes: Do not over-complicate this problem! You will directly use $\hat{\beta}_0$ and $\hat{\beta}_1$ from the linear model introduced in **Part VII**. Further you can choose the estimated AR(1) coefficient as the MLE from Problem 18 or the Bayes' estimate from problem 20. If you failed to complete 18 or 19, use $\hat{\phi} = .9$.

Solution

```
### Solution goes here -------------------------
#step1
beta_0 <- model$coefficients[1]
beta_1 <- model$coefficients[2]
Y_pred <- beta_0+beta_1*X_test

#step2
res_pred <- Y_test-Y_pred

#step3
n <- length(res_pred)
#From Q18 sigma^2=21.63132  phi=0.9565255
#forecast Y
Y_for <- numeric(n)
Y_for[1] <- Y_pred[1]
for (i in 2:n){
  Y_for[i] <- Y_pred[i]+0.9565255*res_pred[i-1]
}

#step4
#prediction error of forecasted BA values versus the true values Y_test
er <- Y_test-Y_for
#MSE we have 2 parameters:beta0, beta1
sum((er)^2)/(n-2)
```

```
## [1] 50.03067
```

```
#this number is much larger than the sigma^2=21.63132

#step5
plot(er,type="l",main="difference between forecasted BA values and true values",
     ylab="difference")
```



difference between forecasted BA values and true values
difference
Index
10
5
0
−5
−15
0
5
10
15
20
25
30