Question 1: Be careful when comparing

Consider the following two R code snippets

```
x1<-1/3;x2<-1/4
if (x1-x2==1/12){
print("Subtraction is correct")
}else{
print("Subtraction is wrong")
}
```

and

```
x1<-1;x2<-1/2
if (x1-x2==1/2){
print("Subtraction is correct")
}else{
print("Subtraction is wrong")
}
```

1. Check the results of the snippets. Comment what is going on.
2. If there are any problems, suggest improvements.

Question 2: Derivative

From the defintion of a derivative a popular way of computing it at a point x is to use a small ϵ and the formula

$$f'(x) = \frac{f(x+\epsilon) - f(x)}{\epsilon}.$$

1. Write your own R function to calculate the derivative of $f(x) = x$ in this way with $\epsilon = 10^{-15}$.
2. Evaluate your derivative function at $x = 1$ and $x = 100000$.
3. What values did you obtain? What are the true values? Explain the reasons behind the discovered differences.

Question 3: Variance

A known formula for estimating the variance based on a vector of n observations is

$$\mathrm{Var}(\vec{x}) = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right)$$

1. Write your own R function, `myvar`, to estimate the variance in this way.
2. Generate a vector $x = (x_1, \ldots, x_{10000})$ with 10000 random numbers with mean 10^8 and variance 1.
3. For each subset $X_i = \{x_1, \ldots, x_i\}$, $i = 1, \ldots, 10000$ compute the difference $Y_i = \texttt{myvar}(X_i) - \texttt{var}(X_i)$, where $\texttt{var}(X_i)$ is the standard variance estimation function in R. Plot the dependence Y_i on i. Draw conclusions from this plot. How well does your function work? Can you explain the behaviour?
4. How can you better implement a variance estimator? Find and implement a formula that will give the same results as `var()`?

Question 4: Linear Algebra

The Excel file "tecator.xls" contains the results of a study aimed to investigate whether a near–infrared absorbance spectrum and the levels of moisture and fat can be used to predict the protein content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry. The worksheet you need to use is "data" (or file "tecator.csv"). It contains data from 215 samples of finely chopped meat. The aim is to fit a linear regression model that could predict protein content as function of all other variables.

1. Import the data set to R
2. Optimal regression coefficients can be found by solving a system of the type $\mathbf{A}\vec{\beta} = \vec{b}$ where $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ and $\vec{b} = \mathbf{X}^T\vec{y}$. Compute $\mathbf{A}$ and $\vec{b}$ for the given data set. The matrix $\mathbf{X}$ are the observations of the absorbance records, levels of moisture and fat, while $\vec{y}$ are the protein levels.
3. Try to solve $\mathbf{A}\vec{\beta} = \vec{b}$ with default solver `solve()`. What kind of result did you get? How can this result be explained?
4. Check the condition number of the matrix $\mathbf{A}$ (function `kappa()`) and consider how it is related to your conclusion in step 3.
5. Scale the data set and repeat steps 2–4. How has the result changed and why?

CompStatsLab1

Omkar Bhutra

7 December 2018

Question 1:

Be careful when comparing

```
## [1] "subtraction is wrong"
```

Due to underflow the subtraction is displaying the same number although when the digits are increased using options we can see that the number is actually different. Underflow is the loss of significant digits.

```
## [1] "subtraction is correct"
```

```
## [1] "subtraction is correct"
```

Evaluating the results of the 2 snippets we see that in the first occasion we get the wrong print of the if-else statement.The problem lies to the fact that float numbers that have infinite numbers of decimals can't be represented exactly in the binary system in computers due to memory storage limitation.Using print(x1-x2,digits=16) and print(1/12,digits=16) we will see that the resulting floats are (0.08333333333333331,0.08333333333333333) respectfully and they are not the same causing the condition of unerflow which leads to the failure of the if statement and evaluation of else. We can adress this problem using the "all_equals()" in the if statement instead of "==" to compare the numbers and we will see that the if statement will be executed and the correctly print message will be outputed. The second statement is evaluated correctly and we get the correct print output because 1/2 has finite numbers of decimals so we don't have the occurence of underflow here.

Question 2:

Derivative

```
## [1] 1.1102230246251565
```

```
## [1] 0
```

The value of the derivative for when x=1 is 1.11022 while the value obtained for the derivative when x=100000 is 0. Looking at the equation algebraically it seems that the answer should be 0. But in the first case, when x=1 the addition of a small value epsilon retains its effect compared to the 2nd case and hence produces a different result than the expected value of 0.

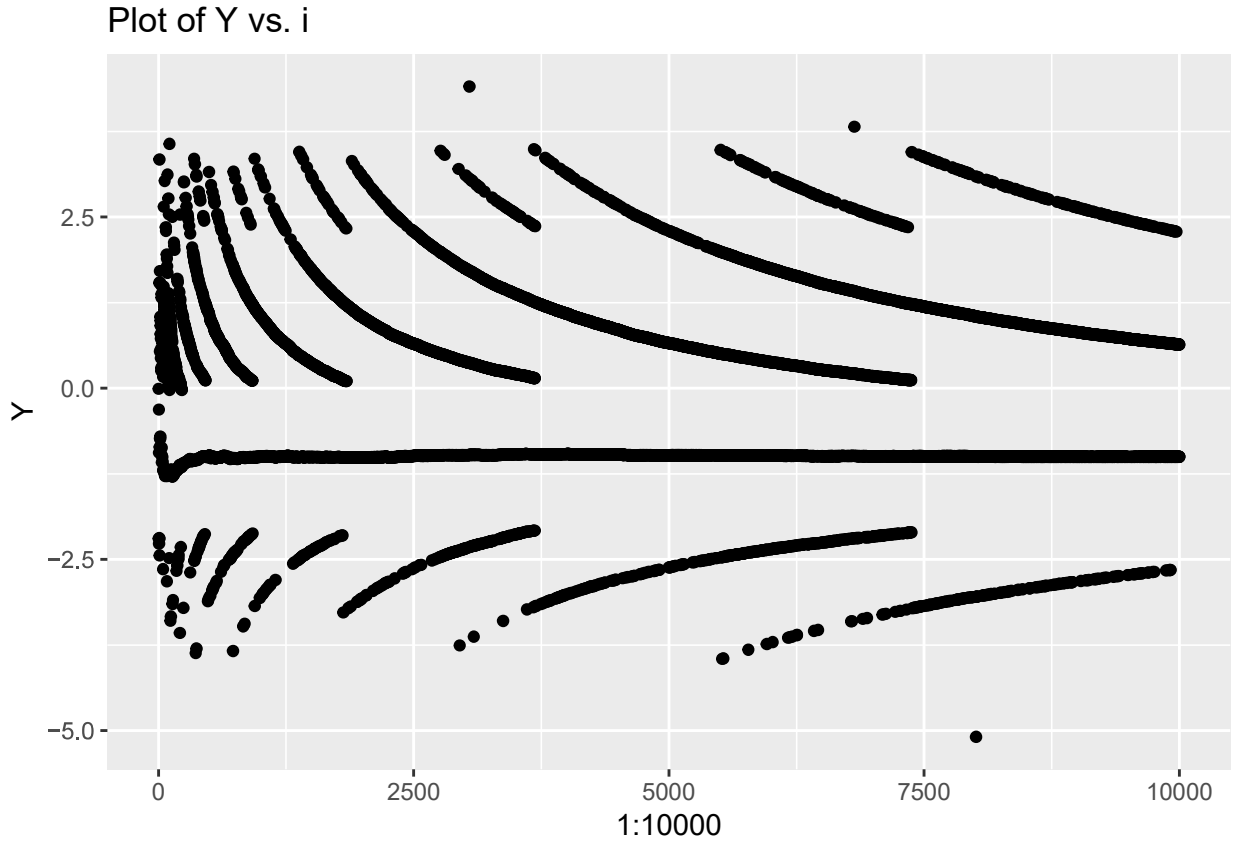
The true value for the function using the function $f(x) = x$ is $f'(x) = \frac{f(x+\epsilon)-f(x)}{\epsilon} = \frac{(x+\epsilon)-x}{\epsilon} = 1$ is always constant with value 1.Regarding the result of the derivative function we see that for $x = 100000$ R doesn't take into account the decimals after a specific number of x and rounds the number to the nearest integer which is 100000 due to underflow occurance so the numeretor of the derivative formula becomes 0 leading finally to 0.When instead $x = 1$ the numerator evaluated is 1.1102230246251565e-15 and the devision with epsilon 10^-15 is just discards the last 15 decimals resulting 1.1102230246251565.

Question 3:

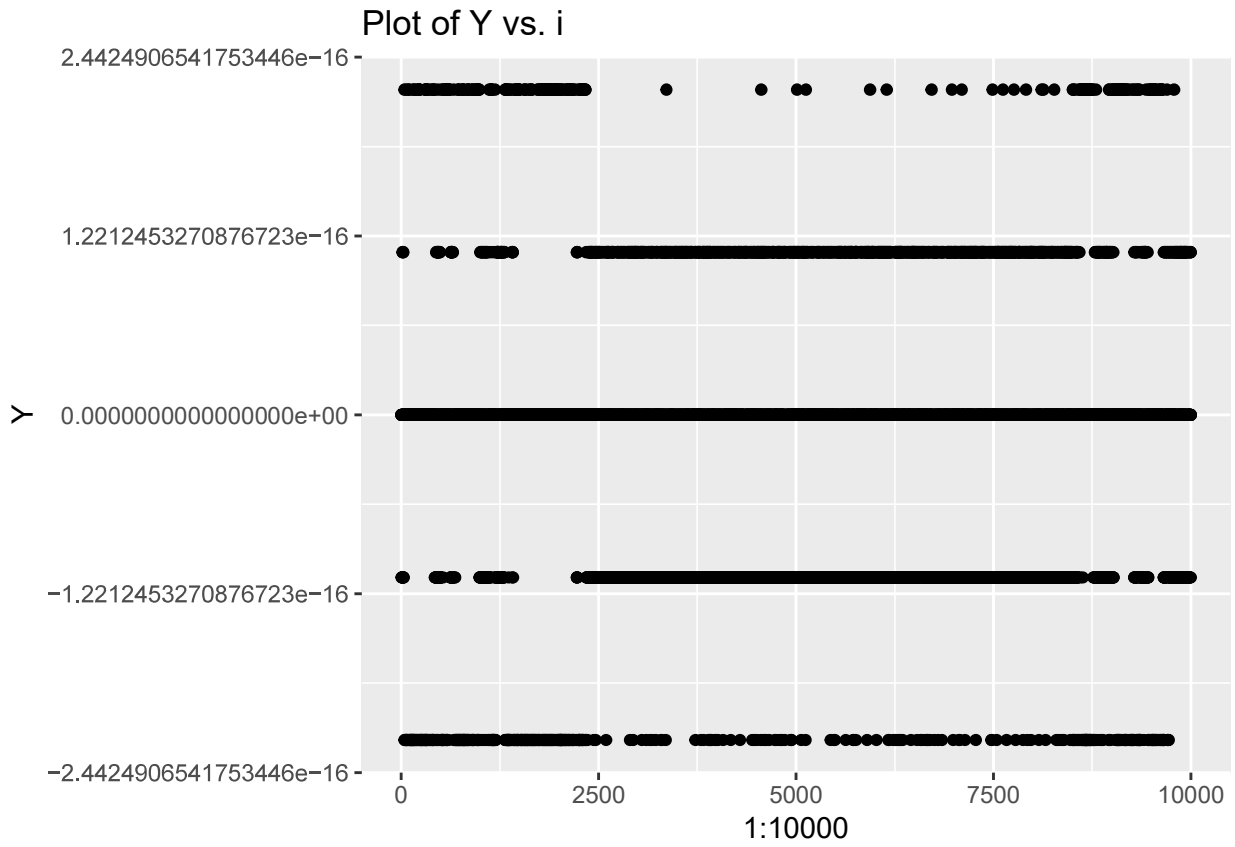
Variance

```
## [1] 1.6385638563856386
```

The plot above shows the dependence Y_i on i with the formula $Var(x) = \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n-1}$ where μ is the mean. Using the new formula where we center the points arround the mean we see that we have an improvement in the range of the errors and the deviation of the errors is steady and we can see an upper and a lower band with few errors lie beyond these linear bands represented with red in the plot.Also we can observe that the range of the errors is much smaller with means the formula used almost as good as the var() basic function in R.




```
## [1] 0.99969989923081803
```



The function does not perform as well as expected due to the numerical precision of the expression myvar(Xi)-var(Xi) which shows us negative values on most occurances.

Question 4:

Linear Algebra

```
## [1] 1157834236871692.2
```

Error in solve.default(A, b) : system is computationally singular: reciprocal condition number = 7.13971e-17

Printing the number of kappa for the value of A matrix we see that is very big and that implies that the matrix is said to be ill-conditioned a very small change in matrix A will cause a large error in b and makes the solution unstable.

```
##                              [,1]
## Channel1    -110.64200663177356887
## Channel2    -221.18999213628123357
## Channel3     378.00670489934344687
## Channel4    -129.70382900682116656
## Channel5     413.41802724878726849
## Channel6     -79.75199462292130193
## Channel7    -203.00600103836293897
## Channel8      82.79496481288938980
## Channel9    -132.38107625535101874
## Channel10    255.82331130982933587
```

```
## Channel65   -399.25583177909453525
## Channel66    364.86655737276095124
## Channel67   -367.16148297452463112
## Channel68    243.92206215756519327
## Channel69    -76.29483445491032967
## Channel70   -318.19160711413701392
## Channel71    327.66533461201169075
## Channel72   -178.52315275400727046
## Channel73    119.18564986588171450
## Channel74    445.11500447600673169
## Channel75    -20.01273187169612910
## Channel76   -642.75099614710870810
## Channel77    369.48095078598106511
## Channel78    -74.90113656891863059
## Channel79    -23.48543216535520983
## Channel80   -676.86153344610886506
## Channel81   1013.45380290573928050
## Channel82   -889.76231887539347554
## Channel83    403.00656326222281223
## Channel84    424.08483028785201441
## Channel85   -801.09561546783777430
## Channel86    655.01342015748275571
## Channel87    659.18297852899979716
## Channel88  -2150.83256466095554060
## Channel89   1671.80888532636413402
## Channel90    298.69770682030031139
## Channel91   -332.17277624942408920
## Channel92   -487.36897587493234596
## Channel93    278.62773677083617940
## Channel94    201.66273526775202640
## Channel95   -609.50814557014871298
## Channel96    565.28517886262272896
## Channel97   -133.34075951392054549
## Channel98   -368.00872501373430623
## Channel99    238.20159678039942719
## Channel100    24.64181878308056639
## Fat           -1.66664028405493303
## Moisture      -0.93410994774249811
```

This happens because the tolerence returned is larger than the default threshold set by the function solve (argument tolerence) so an error returned and we cannot get a solution.The torrelance is related to conditon number by the function $tolerance = \frac{1}{conditionnumber}$ so in our case $tolerance = \frac{1}{kappa(A)} = 7.425326e-16$ and it is bigger that the threshold of $7.425326e-17$ that is set by solve function as we see in the printed error resulting the end of execution of the function. Using the scaled data we where able to solve the linear system and get coefficients for every feature value. Printing the number of kappa again we can see that is still high but much less that the previous used with the unscaled data and we where able to solve the linear system and get coefficient values.

When we scale the data we see that the linear system did not get any better or worse the linear dependences of the column features are still present but we manage to make the value of condition number smaller with scaling.This is happening because If we look at the definition of the condition number $k(A) = ||A|| * ||A^{-}1||$ and just by making the range of the columns smaller the magnitude got smaller leading to a smaller value of condition number which is below threshold value of solve function and we manage to get the solution.The tolerence now is $tolerance = \frac{1}{kappa(A1)} = \frac{1}{490471518993} = 2.038854e-12$ which is smaller than the default

$7.425326e-17$ set by solve so now we are able to get a solution.

Apendix

```
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
library(plotly)
library(ggplot2)
library(xlsx)
library(readxl)
library(boot)
library(kableExtra)
library(knitr)
library(testthat)
options(digits=22)
x1<-1/3;x2<-1/4
if(x1-x2==1/12){
  print("subtraction is correct")
}else{
  print("subtraction is wrong")
}
x1<-1/3;x2<-1/4
if(all.equal((x1-x2),(1/12))){
  print("subtraction is correct")
}else{
  print("subtraction is wrong")
}
x1<-1;x2<-1/2
if(x1-x2==1/2){
  print("subtraction is correct")
}else{
  print("subtraction is wrong")
}
derivative<-function(x){
  f<-function(x){
    return(x)
  }
  epsilion<-10^-15
  x<-(f(x+epsilion)-f(x))/epsilion
  return(x)
}

derivative(x=1)

derivative(x=100000)

set.seed(12345)
myvar<-function(x){
n=length(x)
var<-(1/(n-1))*(sum(x^2)-((1/n)*(sum(x)^2)))
return(var)
}
```

```r
x<-rnorm(n=10000,mean=10^8,sd=sqrt(1))
myvar(x)
Y <- c()
for (i in 1:10000){
options(digits = 22)
Y[i] <- myvar(x[1:i])-var(x[1:i])
}

p1<-ggplot()+ geom_point(aes(1:10000,Y))+ labs(title="Plot of Y vs. i")
p1
set.seed(12345)
varfun <- function(x){
vari <- (1/(length(x) - 1)) * sum((x - mean(x))^2)
return(vari)
}

x<-rnorm(n=10000,mean=10^8,sd=sqrt(1))
varfun(x)

Y <- c()
for (i in 1:10000){
Y[i] <- varfun(x[1:i])-var(x[1:i])
}

p2<-ggplot() + geom_point(aes(1:10000,Y))+ labs(title="Plot of Y vs. i")
p2
tecator = read_excel("tecator.xls",sheet ="data" )
X<-as.matrix(tecator[,c(2:102,104)])
Y<-as.matrix(tecator[,c(103)])
A<-t(X)%*%X
b<-t(X)%*%Y
kappa(A)
#solve(A,b)
tecatorscale <- as.matrix(scale(tecator))
Xscale <- as.matrix(tecatorscale[,-c(1,103)])
yscale <- as.matrix(tecatorscale[,103])
Ascale <- t(Xscale)%*%Xscale
bscale <- t(Xscale)%*%yscale
solve(Ascale,bscale)
```

Computer Lab 2
Computational Statistics

Linköpings Universitet, IDA, Statistik

2019/01/25

Kurskod och namn:	732A90 Computational Statistics
Datum:	2019/01/24—2019/02/07 (lab session 25 January 2019)
Delmomentsansvarig:	Krzysztof Bartoszek and Eric Herwin
Instruktioner:	This computer laboratory is part of the examination for the Computational Statistics course Create a group report, (that is directly presentable, if you are a presenting group), on the solutions to the lab as a **.PDF** file. Be concise and do not include unnecessary printouts and figures produced by the software and not required in the assignments. **All R code should be included as an appendix into your report.** A typical lab report should 2-4 pages of text plus some amount of figures plus appendix with codes. In the report reference **ALL** consulted sources and disclose **ALL** collaborations. The report should be handed in via LISAM (or alternatively in case of problems e–mailed to krzysztof.bartoszek@liu.se or Eric Herwin erihe068@student.liu.se or Sara Johansson sarjo775@student.liu.se), by **23:59 7 February 2019** at latest. Notice there is a final deadline of **23:59 1 April 2019** after which no submissions nor corrections will be considered and you will have to redo the missing labs next year. The seminar for this lab will take place **7 March 2019**. The report has to be written in English.

Question 1: Optimizing a model parameter

The file `mortality_rate.csv` contains information about mortality rates of the fruit flies during a certain period.

1. Import this file to `R` and add one more variable `LMR` to the data which is the natural logarithm of `Rate`. Afterwards, divide the data into training and test sets by using the following code:

```
n=dim(data)[1]
set.seed(123456)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]
```

2. Write your own function `myMSE()` that for given parameters λ and list `pars` containing vectors `X`, `Y`, `Xtest`, `Ytest` fits a LOESS model with response Y and predictor X using `loess()` function with penalty λ (parameter `enp.target` in `loess()`) and then predicts the model for `Xtest`. The function should compute the predictive MSE, print it and return as a result. The predictive MSE is the mean square error of the prediction on the testing data. It is defined by the following Equation (for you to implement):

$$\text{predictive MSE} = \frac{1}{\texttt{length(test)}} \sum_{i\text{th element in test set}} (\texttt{Ytest[i]} - \texttt{fYpred(X[i])})^2,$$

   where `fYpred(X[i])` is the predicted value of `Y` if `X` is `X[i]`. Read on `R`'s functions for prediction so that you do not have to implement it yourself.

3. Use a simple approach: use function `myMSE()`, training and test sets with response LMR and predictor Day and the following λ values to estimate the predictive MSE values: $\lambda = 0.1, 0.2, \ldots, 40$

4. Create a plot of the MSE values versus λ and comment on which λ value is optimal. How many evaluations of `myMSE()` were required (read `?optimize`) to find this value?

5. Use `optimize()` function for the same purpose, specify range for search $[0.1, 40]$ and the accuracy 0.01. Have the function managed to find the optimal MSE value? How many `myMSE()` function evaluations were required? Compare to step 4.

6. Use `optim()` function and BFGS method with starting point $\lambda = 35$ to find the optimal λ value. How many `myMSE()` function evaluations were required (read `?optim`)? Compare the results you obtained with the results from step 5 and make conclusions.

Question 2: Maximizing likelihood

The file `data.RData` contains a sample from normal distribution with some parameters μ, σ. For this question read `?optim` in detail.

1. Load the data to `R` environment.

2. Write down the log-likelihood function for 100 observations and derive maximum likelihood estimators for μ, σ analytically by setting partial derivatives to zero. Use the derived formulae to obtain parameter estimates for the loaded data.

3. Optimize the minus log–likelihood function with initial parameters $\mu = 0$, $\sigma = 1$. Try both Conjugate Gradient method (described in the presentation handout) and BFGS (discussed in the lecture) algorithm with gradient specified and without. Why it is a bad idea to maximize likelihood rather than maximizing log–likelihood?

4. Did the algorithms converge in all cases? What were the optimal values of parameters and how many function and gradient evaluations were required for algorithms to converge? Which settings would you recommend?

Lab2 Computational Statistics

Andreas Christopoulos Charitos (andch552) Omkar Bhutra (omkbh878)

27 jan 2019

Question 1

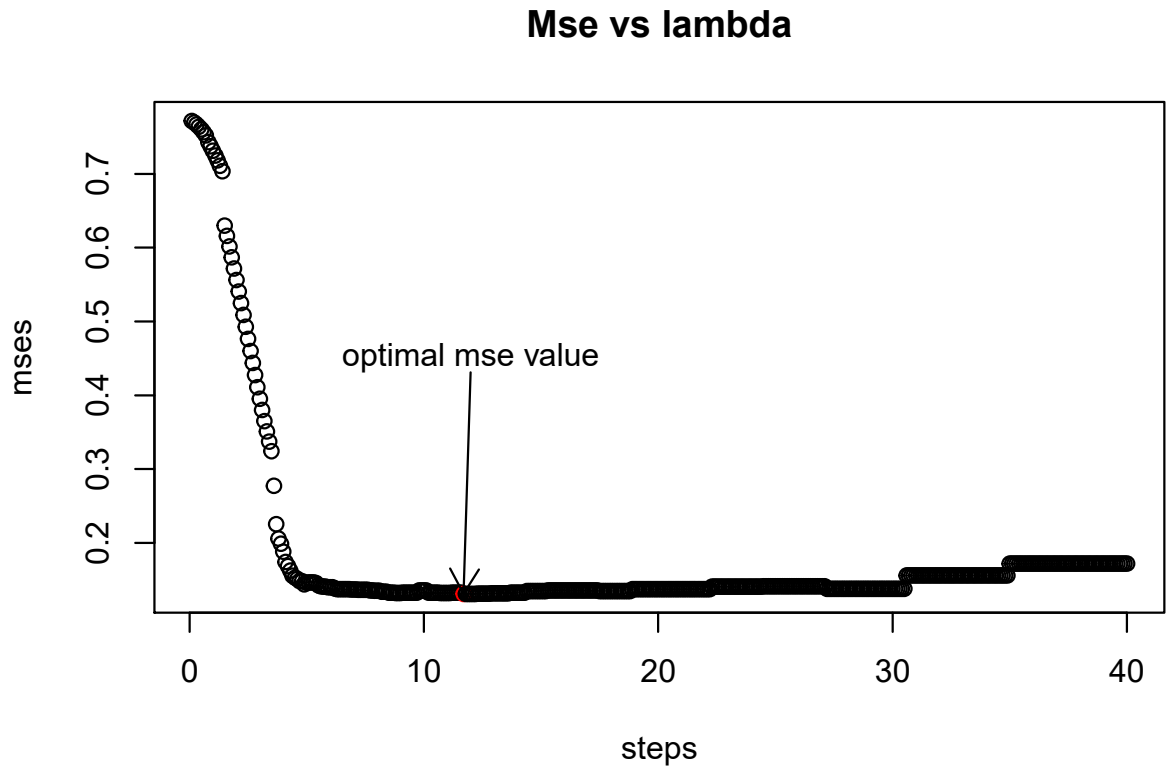
1

```
#importing and diving data
mortality<-read.csv2("mortality_rate.csv")
mortality$LMR<-log(mortality$Rate)

n=dim(mortality)[1]
set.seed(123456)
id=sample(1:n , floor(n*0.5))
train=mortality[id, ]
test=mortality[-id, ]
```

```
#defining the function "myMSE"

myMSE<-function(lambda,pars,iterCounter = T){
  model<-loess(pars$Y~pars$X,enp.target = lambda)
  preds<-predict(model,newdata=pars$X_test)
  mse<-sum((pars$Y_test-preds)^2)/length(pars$Y_test)

  # If we want a iteration counter
  if(iterCounter){
    if(!exists("iterForMyMSE")){
      # Control if the variable exists in the global environemnt,
      # if not, create a variable and set the value to 1. This
      # would be the case for the first iteration
      # We will call the variable 'iterForMyMSE'
      assign("iterForMyMSE",
             value = 1,
             globalenv())
    } else {
      # This part is for the 2nd and the subsequent iterations.
      # Starting of with obtaining the current iteration number
      # and then overwrite the current value by the incremental
      # increase of the current value
      currentNr <- get("iterForMyMSE")
      assign("iterForMyMSE",
             value = currentNr + 1,
             globalenv())
    }
  }
  return(mse)
}
```

```
set.seed(123456)
steps=seq(0.1,40,0.1)
```

```r
mses<-double(length(steps))

mypars<-list(X=train$Day,Y=train$LMR,X_test=test$Day,Y_test=test$LMR)

for(i in steps){
  mses[which(i==steps)]<-myMSE(i,mypars)

}

optimal_l=steps[which.min(mses)]
optimal_mse=min(mses)

plot(x=steps,y= mses,main = "Mse vs lambda",
      col=ifelse(steps==optimal_l,"red","black"))
arrows(optimal_l+0.3, optimal_mse+0.3, x1 = optimal_l, y1 = optimal_mse,
               length = 0.15, angle = 30)
text(optimal_l+0.3,optimal_mse+0.32,labels=c("optimal mse value"))
```




```r
iters1=iterForMyMSE

results1=list("par"=optimal_l,"value"=optimal_mse)

cat("The number of iterations using brute force are :", iters1,"\n",
    "The results from the brute force are :\n")
```

```
## The number of iterations using brute force are : 400
```

```
##  The results from the brute force are :
```

```
results1
```

```
## $par
## [1] 11.7
##
## $value
## [1] 0.131047
```

5

```
set.seed(123456)
remove("iterForMyMSE")
xmin <- optimize(myMSE,pars=mypars,
                 interval=c(0.1,40),maximum  = FALSE,tol=0.01)
iters2=iterForMyMSE
cat("The number of iterations using lambda in [0.1,40] and accuracy 0:01 are :", iters2,"\n",
    "The output of optimize function is : \n")
```

```
## The number of iterations using lambda in [0.1,40] and accuracy 0:01 are : 18
##  The output of optimize function is :
```

```
## $minimum
## [1] 10.69361
##
## $objective
## [1] 0.1321441
```

We can see comparing the results with the previous step that the number of iterations needed for the algorithm to converge decreased dramatically from 400 to only 18.

6

```
set.seed(123456)
remove("iterForMyMSE")
xmin1=optim(c(35), myMSE,pars=mypars,
      method = c( "BFGS"))

iters3=iterForMyMSE
cat("The new number of iterations using BFGS algorithm and lambda = 35 are :", iters3,"\n",
    "The output of optimize function is : \n")
```

```
## The new number of iterations using BFGS algorithm and lambda = 35 are : 3
##  The output of optimize function is :
```

```
## $par
## [1] 35
##
## $value
## [1] 0.1719996
##
## $counts
## function gradient
##        1        1
##
```

```
## $convergence
## [1] 0
##
## $message
## NULL
```

Comparing again the number of iterations with the previous step we can see that the iterations decreased from 18 to only 3 using BFGS.

The table below summarises the results of the 3 methods used

```
library(knitr)

sumtable<-data.frame("Brute Force Method"=c(optimal_l,optimal_mse,iters1),
                     "Optimize function"=c(xmin$minimum,xmin$objective,iters2),
                     "Optim function BFGS"=c(xmin1$par,xmin1$value,iters3))
rownames(sumtable)<-c("lambda","mse","iters")

kable(sumtable)
```

	Brute.Force.Method	Optimize.function	Optim.function.BFGS
lambda	11.700000	10.6936107	35.0000000
mse	0.131047	0.1321441	0.1719996
iters	400.000000	18.0000000	3.0000000

As we can see from the table above the BFGS algorithm is able to converge with only 3 iterations which are lower compared to the other 2 methods but with a little higher mse value.

Question 2

2

Maximum Likehihood estimation

The probability function is given by the formula

$$f(x) = \frac{1}{\sqrt{(2\pi\sigma^2)}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

Assuming that the observations from the sample are i.i.d. the likelihood formula is given by

$$f(x_1, x_2, ..., x_n|\mu, \sigma^2) = \prod_{j=1}^{n} (2\pi\sigma^2)^{-(n/2)} exp(-\frac{(x_j-\mu)^2}{2\sigma^2})$$

which is equavalent to :

$$f(x_1, x_2, ..., x_n|\mu, \sigma^2) = (2\pi\sigma^2)^{-(n/2)} exp(-\frac{\sum_{j=1}^{n}(x_j-\mu)^2)}{2\sigma^2})$$

The loglikekihood function is then

$$L = \ln[(2\pi\sigma^2)^{(-n/2)} exp(-(\frac{1}{2\sigma^2})\sum_{j=1}^{n}(x_j-\mu)^2)] =$$

$$\ln((2\pi\sigma^2)^{(-n/2)}) + \ln(exp(-(\frac{1}{2\sigma^2})\sum_{j=1}^{n}(x_j-\mu)^2)) =$$

$$-\frac{n}{2}\ln(2\pi\sigma^2) - (\frac{1}{2\sigma^2})\sum_{j=1}^{n}(x_j-\mu)^2 =$$

$$-\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - (\frac{1}{2\sigma^2})\sum_{j=1}^{n}(x_j-\mu)^2$$

Getting the derivative with respect to μ to we get :

$$\frac{\partial L}{\partial \mu} => -\frac{1}{2\sigma^2}(\sum_{j=1}^{n}(x_j^2 - 2x_j\mu + \mu^2))' =>$$

$$-\frac{1}{2\sigma^2}(\sum_{j=1}^{n}x_j^2 - 2\sum_{j=1}^{n}x_j\mu + n\mu^2)' =>$$

$$-\frac{1}{2\sigma^2}(-2\sum_{j=1}^{n}x_j + 2n\mu) => -\frac{1}{2\sigma^2}(\sum_{j=1}^{n}x_j - n\mu)(-2)(e.1)$$

Setting $(e.1)$ equal to zero we get

$$\frac{\partial L}{\partial \mu} = 0 => \mu = \frac{\sum_{j=1}^{n}x_j}{n}$$

Which is the estimator for the parameter $\hat{\mu}$

Getting the derivative with respect to σ we get :

$$\frac{\partial L}{\partial \sigma} => -\frac{n}{2}\frac{1}{\sigma^2}2\sigma - \frac{1}{2}\sum_{j=1}^{n}(x_j-\mu)^2\frac{\theta L}{\theta\sigma}(\sigma^2)^{-1} => -\frac{n}{\sigma} - \frac{1}{2}\sum_{j=1}^{n}(x_j-\mu)^2\frac{-2}{(\sigma^3)} => -\frac{n}{\sigma} + \sum_{j=1}^{n}(x_j-\mu)^2\frac{1}{(\sigma^3)}(e.2)$$

Setting $(e.2)$ equal to zero we get

$$\frac{\partial L}{\partial \sigma} = 0 => \sum_{j=1}^{n}(x_j-\mu)^2 = \frac{n\sigma^3}{\sigma} => \sigma^2 = \frac{\sum_{j=1}^{n}(x_j-\mu)^2}{n} => \sigma = \sqrt{\frac{\sum_{j=1}^{n}(x_j-\mu)^2}{n}}$$

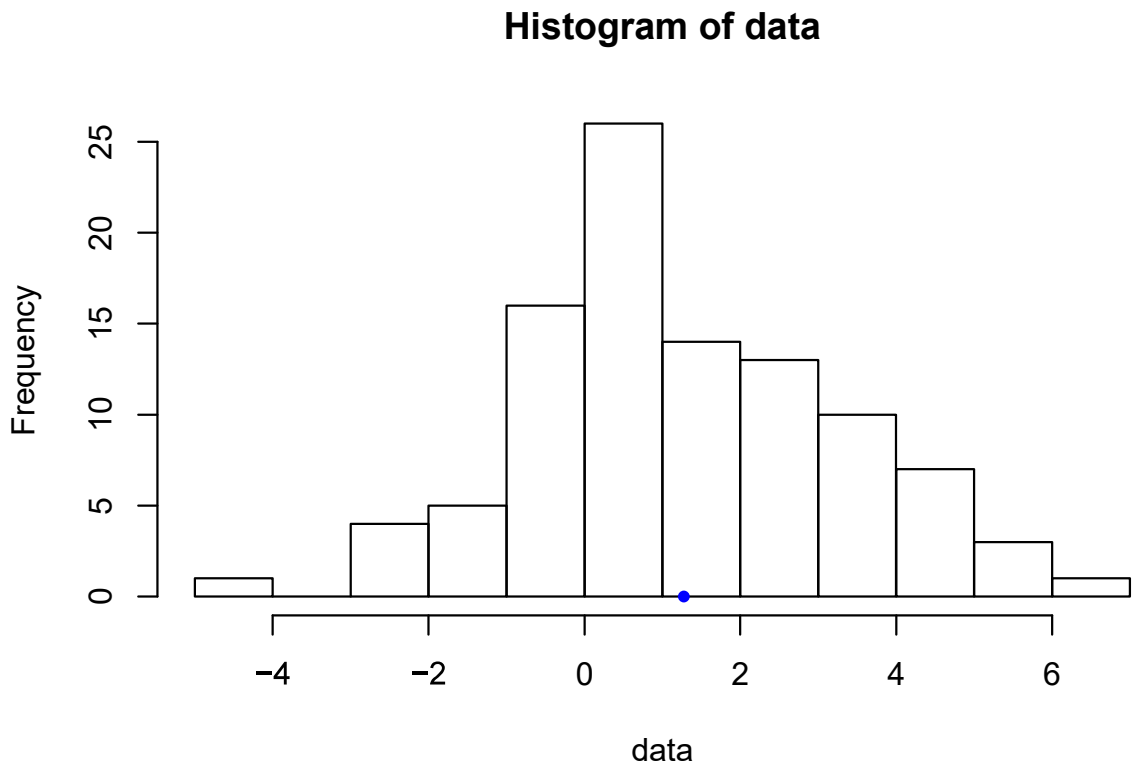
We can use the obtained formulas to obtain the mean $\hat{\mu}$ and variance $\hat{\sigma}$ analytically.

```
#load data
load("data.RData")

#make estimators function
estimators<-function(data){
  n<-length(data)
  mean<-sum(data)/n
  sigma<-sum((data-mean)^2)/n
  return(c("mean"=mean,"sd"=sqrt(sigma)))
```

```
  }


res=estimators(data)

cat("============================================================\n",
    "The obtained mean is :",res["mean"],"and the sd is :",res["sd"])
```

```
## ============================================================
##  The obtained mean is : 1.275528 and the sd is : 2.005976
```

```
hist(data)
points(res["mean"],0,col="blue",pch=20,
       main="Histogram of data")
```



The above histogram shows the distribution of our data and the blue point is indicating the mean value.

3

First we optimize minus loglikehood without specifing gradient

```
loglike<-function(args,x){

  n<-length(x)
  logminus<- (-n*log(2*pi*args[2]^2)/2)-(sum((args[1]-x)^2)/(2*args[2]^2))
  logminus<- -1*logminus
  return(logminus)
```

```
}


#with BFGS
myres1<-optim(c(0,1),fn=loglike,x=data,method = "BFGS")

#with Conjugate Gradient
myres2<-optim(c(0,1),fn=loglike,x=data,method = "CG")


cat("====================================\n",
    "The output of optim with BFGS is :\n")
```

```
## ====================================
##  The output of optim with BFGS is :
```

```
myres1
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       41       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
cat("\n")
```

```
cat("================================================\n",
    "The output of optim with Conjugate Gradient is :\n")
```

```
## ================================================
##  The output of optim with Conjugate Gradient is :
```

```
myres2
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##      208       35
##
## $convergence
## [1] 0
```

```
##
## $message
## NULL
```

Second we optimize minus loglikehood specifing gradient

The gradient with respect to μ is given by the formula (e.1) we calculated erlier $\frac{\partial L}{\partial \mu} = \frac{\sum_{j=1}^{n} x_j - n\mu}{\sigma^2}$ and the derivative with respect to σ is given by the formula (e.2) $\frac{\partial L}{\partial \sigma} = -\frac{n}{\sigma} + \frac{\sum_{j=1}^{n}(x_j - \mu)}{\sigma^3}$

```
gradient<-function(args,x){
  n<-length(x)
  gr_mu<- (sum(x)-n*args[1])/(args[2]^2)

  gr_sigma<- sum((x-args[1])^2)/args[2]^3-n/args[2]

  return(c(-gr_mu,-gr_sigma))
}

#with BFGS
myres3<-optim(c(0,1),fn=loglike,gr=gradient,x=data,method = "BFGS")

#with Conjugate Gradient
myres4<-optim(c(0,1),fn=loglike,gr=gradient,x=data,method = "CG")


cat("==================================================\n",
    "The output of optim with BFGS with gradient is :\n")
```

```
## ==================================================
##  The output of optim with BFGS with gradient is :
```

```
myres3
```

```
## $par
## [1] 1.275528 2.005977
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       39       15
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
cat("\n")
```

```
cat("==================================================================\n",
    "The output of optim with Conjugate Gradient with gradient is :\n")
```

```
## ==================================================================
##  The output of optim with Conjugate Gradient with gradient is :
```

```
myres4
```

```
## $par
## [1] 1.275528 2.005976
##
## $value
## [1] 211.5069
##
## $counts
## function gradient
##       53       17
##
## $convergence
## [1] 0
##
## $message
## NULL
```

As we can see loglikelihood is monotonically increasing function and is has the same relations of order as the likelihood $p(x|\theta_1) > p(x|\theta_2) <=> \ln(p(x|\theta_1)) > \ln(p(x|\theta_2))$ so maximizing likelihood is equivalent to maximizing log likelihood.The reason why we use $ln(x)$ is for computational convenience as we see in the calculations of loglikehood we did before.Using $ln()$ we where able to discard the exponent and also use the $ln(ab) = \ln(a) + \ln(b)$ we manage to replace multiplication with summation which is more coninient and we are able to maximize just by setting the derivatives to 0.

4

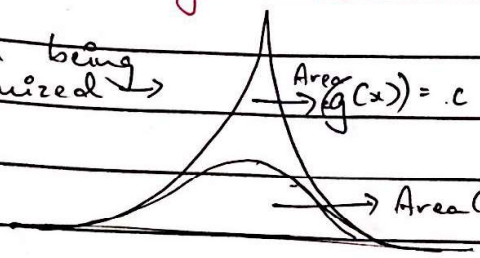
```
library(knitr)

sumdat<-data.frame("BFGS_without_gr"=c(myres1$par[1],myres1$par[2],
                                       myres1$counts[1],myres1$counts[2],myres1$convergence),
                   "CG_without_gr"=c(myres2$par[1],myres2$par[2],
                                     myres2$counts[1],myres2$counts[2],myres2$convergence),
                   "BFGS_with_gr"=c(myres3$par[1],myres3$par[2],
                                    myres3$counts[1],myres3$counts[2],myres3$convergence),
                   "CG_with_gr"=c(myres4$par[1],myres4$par[2],
                                  myres4$counts[1],myres4$counts[2],myres4$convergence)
                   )

rownames(sumdat)<-c("mean","sd","iterations function","iterations gradient","convergence")
sumdat[5,]<-ifelse(sumdat[5,]==0,"yes","no")

kable(sumdat)
```

	BFGS_without_gr	CG_without_gr	BFGS_with_gr	CG_with_gr
mean	1.27552755151932	1.27552771909709	1.27552755040258	1.27552759112531
sd	2.00597696486639	2.00597650338868	2.00597654945241	2.00597647249389
iterations function	41	208	39	53
iterations gradient	15	35	15	17
convergence	yes	yes	yes	yes

The table provides summary information using the 2 algorithms ("BFGS","CG") with and without specifing gradient function in optim function. All the algorihms converge as we can see and the results are almost

Area of maj. func. =

Rej. rate is the prob. that a pt. is outside target distr.



Exp. Rej rate = Rej. region / Total region

$\Rightarrow \frac{c-1}{c}$

~~Lab 4~~

Lab 2-

Q2)

$$P(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(x_i-\mu)^2}$$

when no particular distribution is mentioned, take normal distr.

Likelihood:

$$L(\mu,\sigma|X) = \prod_{n=1}^{N} P(X_n|\mu,\sigma)$$

$$= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{(x_n-\mu)^2}{2\sigma^2}\right)}$$

Log Likelihood:

$$\log L(\mu,\sigma|X) = \sum_{n=1}^{N} (\log(c_1) - \log(2\pi\sigma^2)^{1/2}) - \sum_{n=1}^{N} \frac{(x_n-\mu)^2}{2\sigma^2}$$

$$= -\frac{1}{2}\sum_{n=1}^{N} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 \quad \text{---(1)}$$

Partially diff. (1) w.r.t μ & set to 0

$$\frac{\partial \log L(\mu,\sigma|X)}{\partial \mu} = -\frac{1}{2\sigma^2}\cdot(-2)\sum_{n=1}^{N}(x_n-\mu) \Rightarrow \sum_{n=1}^{N}\frac{(x_n-\mu)}{\sigma^2} = 0$$

Gradient of μ

$$\sum_{n=1}^{N}(x_n-\mu)=0$$

$$\sum_{n=1}^{N}x_n=n\mu$$

$$\therefore \boxed{\mu=\frac{\sum_{n=1}^{N}x_n}{n}}$$

Partially diff. ① w.r.t. σ & set to 0,

$$\frac{\partial}{\partial\sigma}\log L(\mu,\sigma|x)=\frac{-1}{2}\cdot\frac{n\cancel{(2)}\cancel{2\pi}\sigma}{\cancel{2\pi}\sigma^{\cancel{3}}}\left(\frac{-\cancel{2}}{\cancel{2}}\frac{1}{\sigma^3}\sum_{n=1}^{N}(x_n-\mu)^2\right)$$

$$0=\frac{-n}{\sigma}+\frac{\sum_{n=1}^{N}(x_n-\mu)^2}{\sigma^3}$$ } Gradient of σ

$$0=-n\sigma^2+\sum_{n=1}^{N}(x_n-\mu)^2$$

$$\sigma^2=\frac{\sum_{n=1}^{N}(x_n-\mu)^2}{n}$$

$$\boxed{\sigma=\sqrt{\frac{\sum_{n=1}^{N}(x_n-\mu)^2}{n}}}$$

Computational Statistics - Lab 3

Yusur Al-Mter (yusal621), Lakshidaa Saigiridharan (laksa656)

2/8/2019

Contents

QUESTION 1 : Cluster sampling

An opinion pool is assumed to be performed in several locations of Sweden by sending interviewers to this location. Of course, it is unreasonable from the financial point of view to visit each city. Instead, a decision was done to use random sampling without replacement with the probabilities proportional to the number of inhabitants of the city to select 20 cities. Explore the file population.xls. Note that names in bold are counties, not cities.

TASK 1.1:

Import necessary information to R.

```
# TASK 1.1

# Importing the file
data <- read.csv2("population.csv")
```

TASK 1.2:

Use a uniform random number generator to create a function that selects 1 city from the whole list by the probability scheme offered above (do not use standard sampling functions present in R).

```
# TASK 1.2

# Function which uses a uniform random number generator to select a city from a list
city_selection <- function(data){
data$cumsum <- cumsum(data$Population)/sum(data$Population)
selected <- which(runif(1) <= data$cumsum)[1]
```

```
city_selected <- data$Municipality[selected]
as.numeric(rownames(data[selected,]))
}
```

TASK 1.3:

Use the function you have created in step 2 as follows:

(a) Apply it to the list of all cities and select one city

(b) Remove this city from the list

(c) Apply this function again to the updated list of the cities

(d) Remove this city from the list

(e) . . . and so on until you get exactly 20 cities.

```
# TASK 1.3

select <- data.frame()

# List of all cities stored in data1
data1 <- data

# Selecting 20 cities
for(i in 1:20){
  selected <- which(city_selection(data1) == as.numeric(rownames(data)))
  select <- rbind(select, data1[selected,])
  # Removing the selected city from the list
  data1 <- data1[-c(selected),]
}
```

TASK 1.4

Run the program. Which cities were selected? What can you say about the size of the selected cities?

```
# TASK 1.4

# Dataframe of 20 selected cities
cat("Selected cities : \n")
```

```
## Selected cities :
```

```
select$Municipality
```

```
##  [1] Borgholm         Sollefte\345     Osby             Torsby
##  [5] Sigtuna          Bjurholm         Hjo              Gnesta
##  [9] Trosa            Ludvika          Mullsj\366       Tran\345s
## [13] Haninge          S\366dert\344lje Arvidsjaur       Enk\366ping
```

```
## [17] Oskarshamn       Flen             Gullsp\345ng     Sval\366v
## 290 Levels: Ale Alings\345s Alvesta Aneby Arboga Arjeplog ... \326vertorne\345
```

The selected cities along with their respective size is shown below :

```
# Selected cities along with their respective sizes
city_size = data.frame(City = select$Municipality, Size = select$Population)

kable(city_size, booktabs = T) %>%
kable_styling(latex_option = "striped")
```

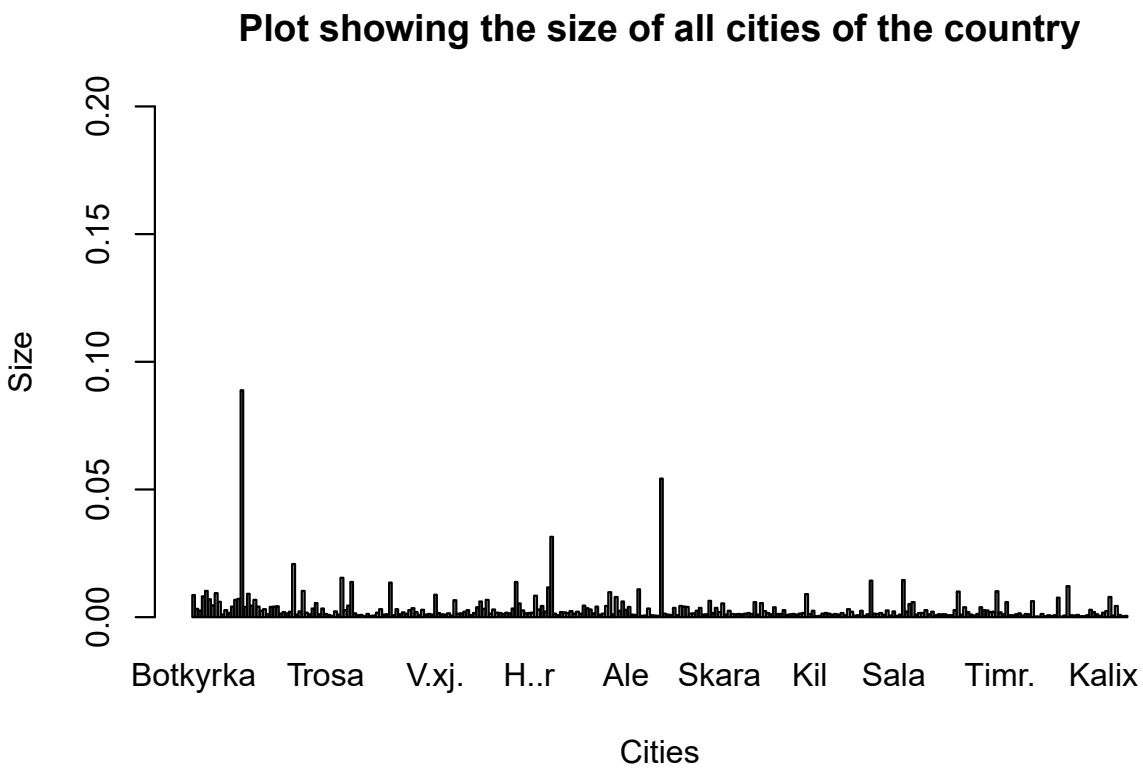
City	Size
Borgholm	10806
Sollefte<e5>	20442
Osby	12656
Torsby	12508
Sigtuna	39219
Bjurholm	2500
Hjo	8859
Gnesta	10318
Trosa	11446
Ludvika	25650
Mullsj<f6>	7027
Tran<e5>s	18043
Haninge	76237
S<f6>dert<e4>lje	85270
Arvidsjaur	6622
Enk<f6>ping	39360
Oskarshamn	26232
Flen	16139
Gullsp<e5>ng	5335
Sval<f6>v	13290

On observing the sizes of the cities that have been selected by the function, it can be seen that the function picked 20 cities which are of relatively large sizes. This is because the cities with huge populations have a much higher probability of being selected than the ones with small population.

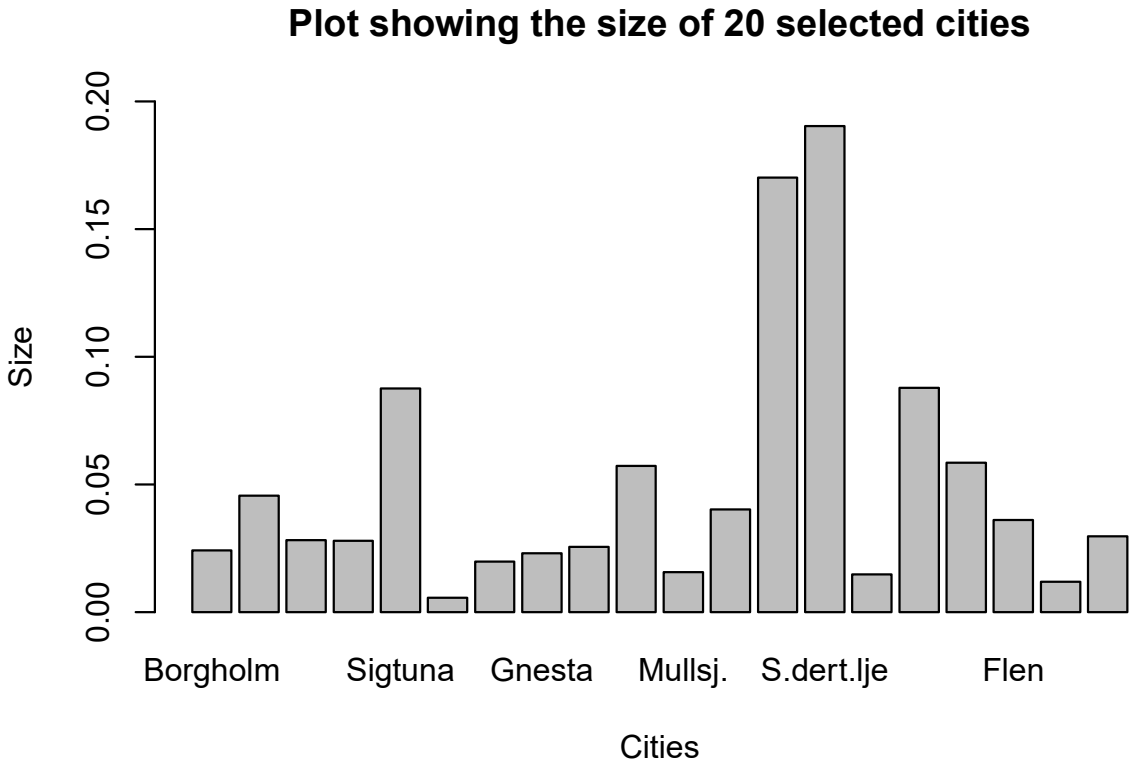
TASK 1.5

Plot one histogram showing the size of all cities of the country. Plot another histogram showing the size of the 20 selected cities. Conclusions?

```
# TASK 1.5

# Plot showing the size of all cities of the country
percentage<-data$Population/sum(data$Population)
barplot(percentage,names.arg = data$Municipality, xlab="Cities", ylab = "Size", ylim = c(0, 0.20))
title("Plot showing the size of all cities of the country")
```




```r
# Plot showing the size of 20 selected cities
percentage_select<-select$Population/sum(select$Population)
barplot(percentage_select, names.arg = select$Municipality, xlab="Cities", ylab = "Size", ylim = c(0, 0
title("Plot showing the size of 20 selected cities")
```



It can be seen that the two histograms follow a similar distribution. This shows that the function implemented

to select 20 cities based on the probabilities proportional to the number of inhabitants of the city performs a good job in the city selection.

QUESTION 2: Different distributions

TASK 2.1

Write a code generating double exponential distribution DE(0, 1) from Unif(0, 1) by using the inverse CDF method. Explain how you obtained that code step by step. Generate 10000 random numbers from this distribution, plot the histogram and comment whether the result looks reasonable.

The probability density for the double exponential (Laplace) function (PDF) is given by the formula:

$$DE(\mu, \alpha) = \frac{\alpha}{2} \exp\left(-\alpha|x - \mu|\right)$$

where the variable x is a real number as is the location parameter μ while the parameter α is a real positive number.

The cumulative density function (CDF) of a continuous random variable X is defined as:

$$F(x) = \int_{-\infty}^{x} f(x)dx, \qquad -\infty < x < \infty$$

Thus, the cumulative distribution for the double exponential distribution is given by:

For $x > \mu$:

$$F(x) = \int_{-\infty}^{x} \frac{\alpha}{2} e^{-\alpha(x-\mu)} dx, \qquad x > \mu$$

$$F(x) = 1 - \int_{x}^{\infty} \frac{\alpha}{2} e^{-\alpha(x-\mu)} dx$$

Integrating with respect to x, we have:

$$F(x) = 1 - \frac{1}{2} e^{-\alpha(x-\mu)}$$

Now, for $x \leq \mu$:

$$F(x) = \int_{-\infty}^{x} \frac{\alpha}{2} e^{\alpha(x-\mu)} dx, \qquad x \leq \mu$$

Integrating with respect to x, we have:

$$F(x) = \frac{1}{2} e^{\alpha(x-\mu)}$$

The inverse CDF technique:

Given a uniform random number between zero and one in U (i.e $U \sim U(0, 1)$), a random number from a Double-exponential distribution is given by solving the equation F(x)=U for x, as shown below:

For $U > 1/2$, we have:

$$U = 1 - \frac{1}{2} e^{-\alpha(x-\mu)}$$

Solving for x, we will get:

$$x = \mu - \frac{ln(2-2U)}{\alpha}$$

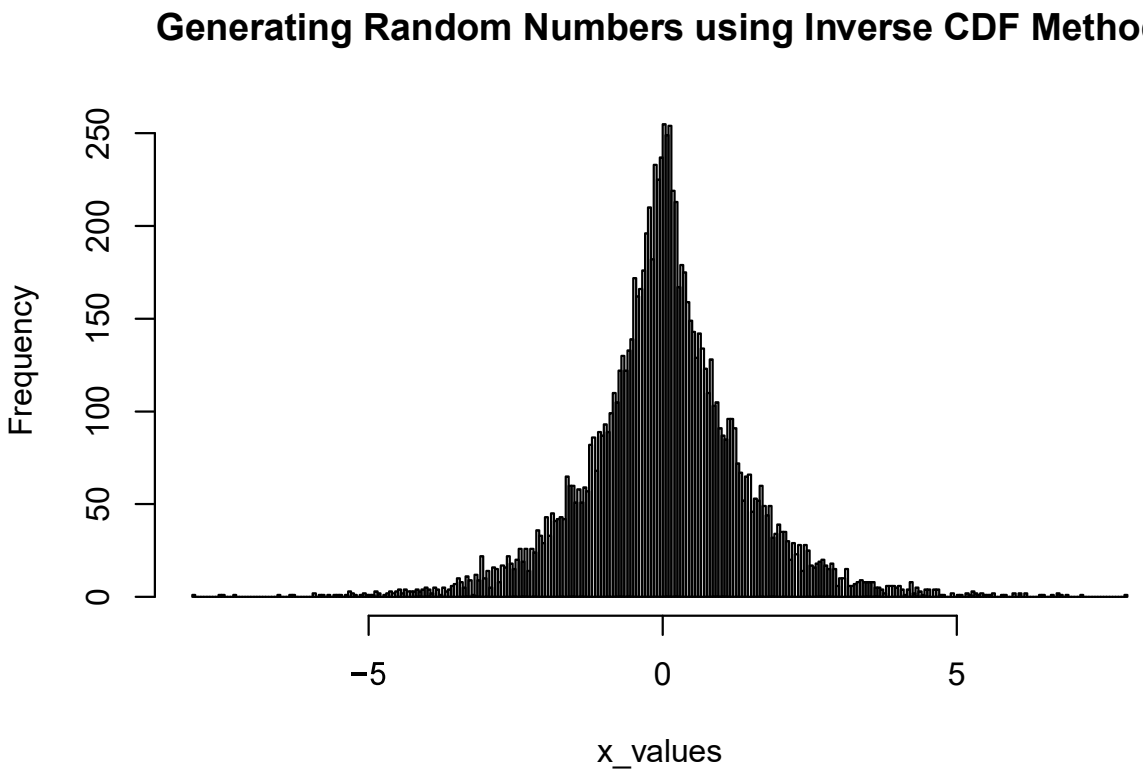
For $U \leq 1/2$, we have:

$$U = \frac{1}{2}e^{\alpha(x-\mu)}$$

Solving for x, we will get:

$$x = \mu + \frac{ln(2U)}{\alpha}$$

```r
# TASK 2.1

inverse_cdf = function(mu,alpha){
  U = runif(1,0,1)
  if (U <= 0.5){
    x=mu+(log(2*U))/alpha
  }else{
    x=mu-(log(2-2*U))/alpha
  }
  return(x)
}

x_values <- c()
for (i in 1:10000){
  x_values[i] <-inverse_cdf(0,1)
  x_values
}

hist(x_values, breaks = 300, main = "Generating Random Numbers using Inverse CDF Method")
```



We can conclude from the histogram, that the implementation of our Non Uniform Random Number generators using inverse CDF method appears to be correct, at least when compared empirically against standard histogram for the laplace distribution

TASK 2.2

Use the Acceptance/rejection method with DE(0; 1) as a majorizing density to generate N(0,1) variables.Explain step by step how this was done. How did you choose constant c in this method? Generate 2000 random numbers N(0,1) using your code and plot the histogram. Compute the average rejection rate R in theacceptance / rejection procedure. What is the expected rejection rate ER and how close is it to R? Generate 2000 numbers from N(0,1) using standard rnorm() procedure, plot the histogram and compare the obtained two histograms.

Acceptance/rejection methods for generating realizations of a random variable X make use of realizations of another random variable Y whose probability density $g_y(x)$ is similar to the probability density of X, $f_x(x)$. The random variable Y is chosen so that we can easily generate realizations of it and so that its density $g_y(x)$ can be scaled to majorize $f_x(x)$ using some constant c; that is,

$$c * g_y(x) \geq f_x(x) \qquad for \;\; all \;\; x$$

Where, the density $g_y(x)$ is called the *majorizing* density or the *proposal* density, and $c * g_y(x)$ is called the majorizing function. The density of interest, $f_x(x)$, is called the *target* density.

Thus, our majorizing density $f_y(x) \sim DE(0, 1)$ is,

$$g_y(x) = \frac{1}{2} e^{-|x|}$$

And the target density $f_y(x) \sim N(0, 1)$ is,

$$f_y(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

To evaluate the majorizing constant c, we have,

$$c \geq \frac{f_x(x)}{g_y(x)}$$

Which is,

$$c \geq \sqrt{\frac{2}{\pi}} e^{-\frac{x^2}{2}+|x|}$$

By differentiating the above formula and setting to zero we got the maximum at x=1, Hence the c value will be,

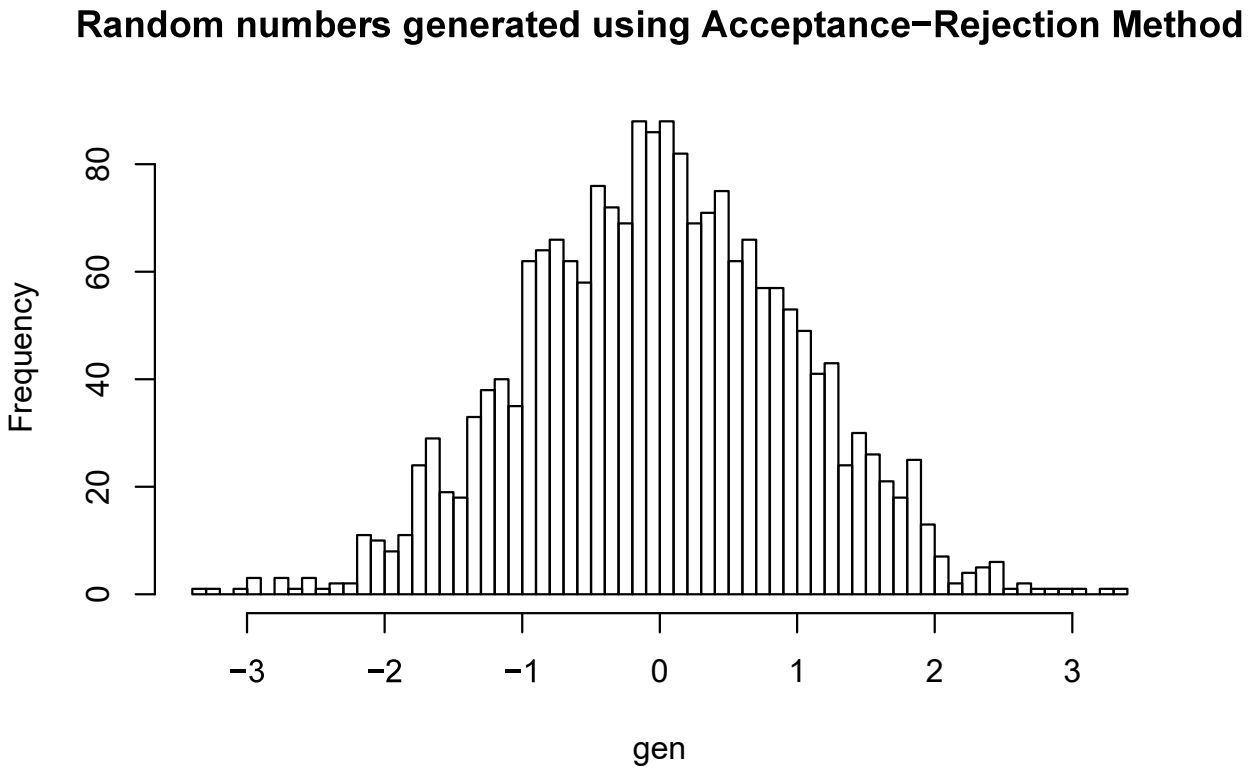
$$\sqrt{\frac{2}{\pi}} e^{\frac{1}{2}} = \sqrt{\frac{2e}{\pi}}$$

```
# TASK 2.2

accept_reject <- function(){

  X<-c()
```

```
  counter= 0  # counter for how many numbers been rejected

   while (length(X)==0) { # as long as our vector is empty >> do
    counter = counter+1

  Y<-inverse_cdf(0,1)
  fy <- (1/sqrt(2*pi))*exp(-(1/2)*Y^2)

  Gy <- 0.5*exp(-abs(Y))

  c <- sqrt(2*exp(1)/pi)

  U = runif(1,0,1)

  if (U <= fy/(c*Gy)){
    X=Y
  }
  }
  gen_itr <- cbind("gen" = X, "itr" = counter)
  return(gen_itr)
}
```

```
# generating 2000 random numbers using Acceptance\Rejection method
res <- list()
for (i in 1:2000){
  res[[i]]<- accept_reject()
}
```

```
# excluding the random numbers from the list for plot
gen <- c()
for (i in 1:2000){
  gen[i]<- res[[i]][1]
}

# plotting the generated random numbers using N(0,1)
hist(gen , breaks = 50, main = "Random numbers generated using Acceptance-Rejection Method")
```



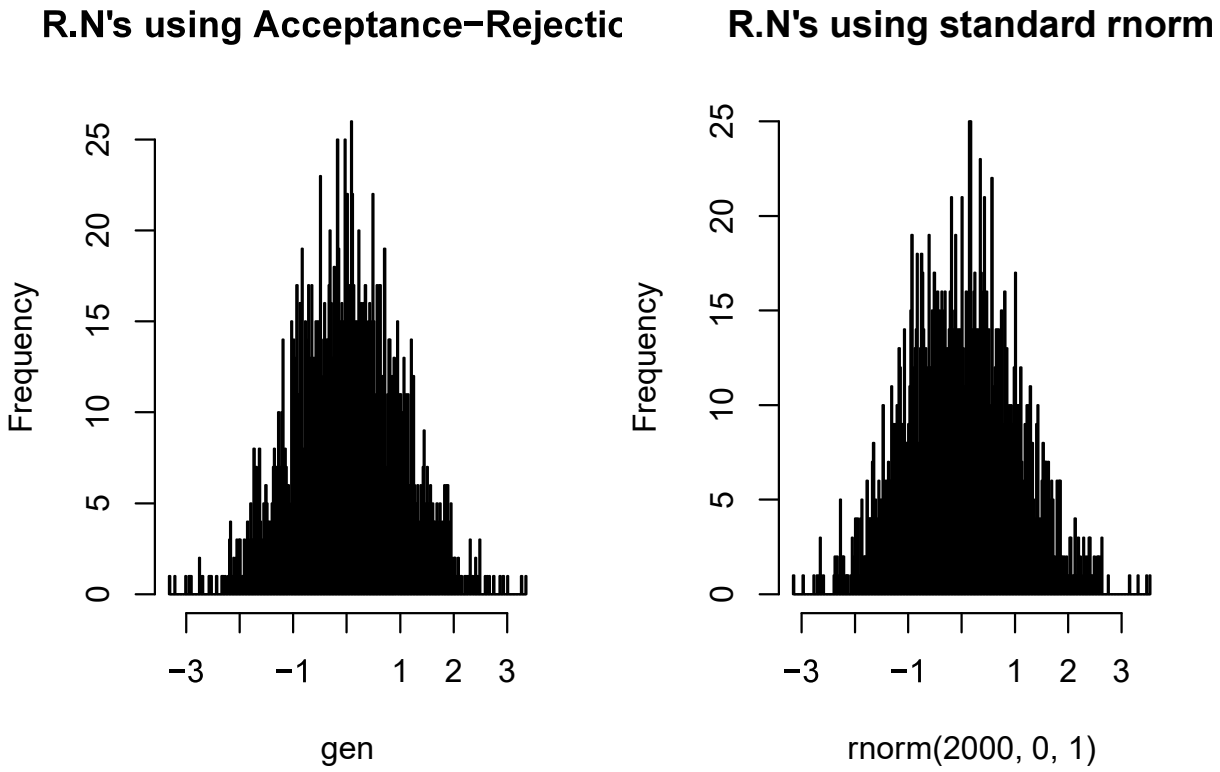

```
# getting the total number of iterations (accepted+rejected) to commpute the average
itr <- c()
for (i in 1:2000){
  itr[i]<- res[[i]][2]
}

# average rejection rate
average_rej <-1-(2000/sum(itr))
average_rej
```

```
## [1] 0.2351816
```

```
# expected rejection rate
c <- sqrt(2*exp(1)/pi)
expected_rej_rate<- 1-(1/c)
expected_rej_rate
```

```
## [1] 0.2398265
```

```
# Comparing Two plots for the generated random numbers

# plotting the generated random numbers using Acceptance-Rejection Method
par(mfrow=c(1,2))
hist(gen , breaks = 300, main = "R.N's using Acceptance-Rejection")
# histogram for 2000 R.N's generated by standard rnorm()
hist(rnorm(2000,0,1), breaks = 300,main = "R.N's using standard rnorm")
```



The above two histograms for the Random Numbers using Acceptance/Rejection method and the exact normal distribution show similar pattern, that means our method is quite acceptable.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
# Loading required R packages
library(ggplot2)
library(kableExtra)
library(gridExtra)


# QUESTION 1 : Cluster sampling


# TASK 1.1

# Importing the file
data <- read.csv2("population.csv")


# TASK 1.2

# Function which uses a uniform random number generator to select a city from a list
city_selection <- function(data){
data$cumsum <- cumsum(data$Population)/sum(data$Population)
selected <- which(runif(1) <= data$cumsum)[1]
```

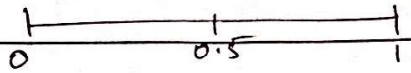
```r
city_selected <- data$Municipality[selected]
as.numeric(rownames(data[selected,]))
}


# TASK 1.3

select <- data.frame()

# List of all cities stored in data1
data1 <- data

# Selecting 20 cities
for(i in 1:20){
  selected <- which(city_selection(data1) == as.numeric(rownames(data)))
  select <- rbind(select, data1[selected,])
  # Removing the selected city from the list
  data1 <- data1[-c(selected),]
}


# TASK 1.4

# Dataframe of 20 selected cities
cat("Selected cities : \n")
select$Municipality


# Selected cities along with their respective sizes
city_size = data.frame(City = select$Municipality, Size = select$Population)

kable(city_size, booktabs = T) %>%
kable_styling(latex_option = "striped")


# TASK 1.5

# Plot showing the size of all cities of the country
percentage<-data$Population/sum(data$Population)
barplot(percentage,names.arg = data$Municipality, xlab="Cities", ylab = "Size", ylim = c(0, 0.20))
title("Plot showing the size of all cities of the country")

# Plot showing the size of 20 selected cities
percentage_select<-select$Population/sum(select$Population)
barplot(percentage_select, names.arg = select$Municipality, xlab="Cities", ylab = "Size", ylim = c(0, 0
title("Plot showing the size of 20 selected cities")


# QUESTION 2 : Different distributions


# TASK 2.1
```

```r
inverse_cdf = function(mu,alpha){
  U = runif(1,0,1)
  if (U <= 0.5){
    x=mu+(log(2*U))/alpha
  }else{
    x=mu-(log(2-2*U))/alpha
  }
  return(x)
}

x_values <- c()
for (i in 1:10000){
  x_values[i] <-inverse_cdf(0,1)
  x_values
}

hist(x_values, breaks = 300, main = "Generating Random Numbers using Inverse CDF Method")


# TASK 2.2

accept_reject <- function(){

  X<-c()
  counter= 0  # counter for how many numbers been rejected

   while (length(X)==0) { # as long as our vector is empty >> do
    counter = counter+1

  Y<-inverse_cdf(0,1)
  fy <- (1/sqrt(2*pi))*exp(-(1/2)*Y^2)

  Gy <- 0.5*exp(-abs(Y))

  c <- sqrt(2*exp(1)/pi)

  U = runif(1,0,1)

  if (U <= fy/(c*Gy)){
    X=Y
  }
  }
  gen_itr <- cbind("gen" = X, "itr" = counter)
  return(gen_itr)
}

# generating 2000 random numbers using Acceptance\Rejection method
res <- list()
for (i in 1:2000){
  res[[i]]<- accept_reject()
}
# excluding the random numbers from the list for plot
gen <- c()
```

```r
for (i in 1:2000){
  gen[i]<- res[[i]][1]
}

# plotting the generated random numbers using N(0,1)
hist(gen , breaks = 50, main = "Random numbers generated using Acceptance-Rejection Method")

# getting the total number of iterations (accepted+rejected) to commpute the average
itr <- c()
for (i in 1:2000){
  itr[i]<- res[[i]][2]
}

# average rejection rate
average_rej <-1-(2000/sum(itr))
average_rej
# expected rejection rate
c <- sqrt(2*exp(1)/pi)
expected_rej_rate<- 1-(1/c)
expected_rej_rate
# Comparing Two plots for the generated random numbers

# plotting the generated random numbers using Acceptance-Rejection Method
par(mfrow=c(1,2))
hist(gen , breaks = 300, main = "R.N's using Acceptance-Rejection")
# histogram for 2000 R.N's generated by standard rnorm()
hist(rnorm(2000,0,1), breaks = 300,main = "R.N's using standard rnorm")
```

$DE(\mu, \alpha) = \frac{\alpha}{2} \exp(-\alpha |x-\mu|)$

$u = unif(0,1)$



Inverse CDF -

$f(x) \leftarrow pdf$

$F(x) = \int_{-\infty}^{x} f(x)\,dx \quad , \quad -\infty < x < \infty$

need to find

$pdf \rightarrow cdf = u \longrightarrow (x) = (cdf)^{-1}$

DE

① $\frac{\alpha}{2} e^{-\alpha(x-\mu)}$, $x \geq \mu$, $x \geq 0$

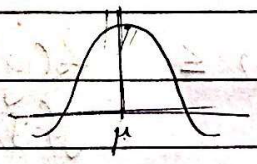
$-\infty < x < \infty$

② $\frac{\alpha}{2} e^{\alpha(x-\mu)}$, $x \leq \mu$, $x \leq 0$

① $u = \left(1 - \int_{x}^{\infty} \frac{\alpha}{2} e^{-\alpha(x-\mu)}\,dx\right)$

→ as $x > \mu$ ★ [cdf always sums up to 1]

↳ area under the area



$= 1 - \frac{\alpha}{2} \cdot \frac{e^{-\alpha(x-\mu)}}{-\alpha} \Big|_{x}^{\infty}$

$= 1 + \frac{1}{2}\left((0) - e^{-\alpha(x-\mu)}\right)$

↳ $\frac{1}{e^{\infty}} \Rightarrow \frac{1}{\infty} = 0$

$u = 1 - \frac{1}{2}\left(e^{-\alpha(x-\mu)}\right)$, when $u > 1/2$

$e^{-\alpha(x-\mu)} = 2 - u$

$-\alpha(x-\mu) = \ln(2-u)$

$x = \mu - \frac{1}{\alpha}\left(\ln(2-u)\right)$ when $u > \frac{1}{2}$

(2) $u = \int_{-\infty}^{x} \frac{\alpha}{2} e^{\alpha(x-\mu)} dx$

$= \frac{\alpha}{2} \cdot \frac{e^{\alpha(x-\mu)}}{\alpha} \Big|_{-\infty}^{x}$

$= \frac{1}{2}\left(e^{\alpha(x-\mu)} - 0\right)$

$u = \frac{1}{2}\left(e^{\alpha(x-\mu)}\right)$, when $u \le 1/2$

$e^{\alpha(x-\mu)} = 2u$

$\alpha(x-\mu) = \ln(2u)$

$\boxed{x = \mu + \frac{1}{\alpha}(\ln(2u))}$ when $u \le 1/2$

Acceptance / Rejection method –

$Y \sim g(y) \sim DE$ ← majorizing density

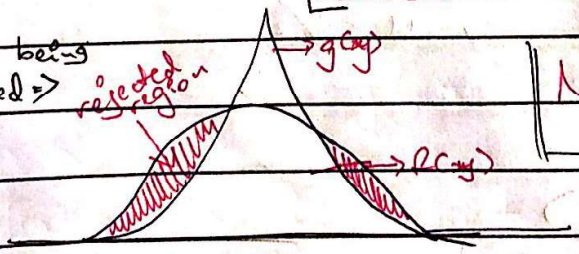
$X \sim f(y) \sim N(0,1)$ ← target density

$U \sim Unif(0,1)$

If $U \le \frac{f_x(y)}{c\,g_y(y)}$ ($f_x(y)$ ← target func., $g_y(y)$ ← proposal func.) $\Rightarrow \frac{N(0,1)}{c(DE(0,1))}$

then $X = Y$ → [we take realization y as desired] ★

Before being maximised ⇒



$N = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2}$

w.k.t.

there exists a constant c such that,

$\forall_y \; c\,g_y(y) \ge f_x(y)$

$$\therefore \quad c \geq \frac{f(y)}{g(y)}$$

$f(y) = N(0,1) \longrightarrow$ subst. $\mu = 0$ & $\sigma = 1$

$$\therefore = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2}$$

$$\Rightarrow c = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2}}{\frac{1}{2} e^{-|y|}}$$

$$c = \sqrt{\frac{2}{\pi}}\, e^{-\frac{1}{2}y^2 + |y|}$$

$$c' = \frac{\partial c}{\partial y} = \sqrt{\frac{2}{\pi}}\, e^{-\frac{1}{2}y^2 + |y|} \cdot \left(-y + \frac{y}{|y|}\right)$$

To get value of y,

→

Set der. to 0,

$$0 = \sqrt{\frac{2}{\pi}}\, \frac{e^{-\frac{1}{2}y^2 + |y|}}{e^{-\frac{1}{2}y^2 + |y|}} \left(-y + \frac{y}{|y|}\right)$$

~~$\ln \sqrt{\frac{\pi}{2}} = -\frac{1}{2}y^2 + |y|$~~ When $y = 1$, $c' = 0$.

Subst. $y = 1$ in c,

$$c = \sqrt{\frac{2}{\pi}}\, e^{-\frac{1}{2}} \simeq 1.3$$

Random sample obtained should be of a normal distribution. (y)

When you $\times y$ $g(y)$ with c, it will get maximised.

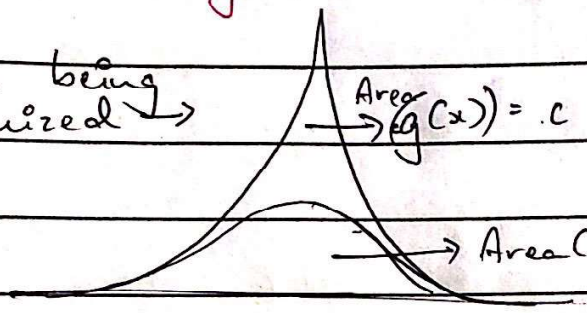
Area under target func. = 1

Because area of prob. = 1

Area under prob. dist. = prob.

Area of maj. func. =

Rej. rate is the prob. that a pt. is outside target distr.



Exp. Rej rate

$$= \frac{\text{Rej. region}}{\text{Total region}}$$

$$\Rightarrow \frac{c-1}{c}$$

Lab4 Computational Statistics

Andreas C Charitos(andch552) Omkar Bhutra (omkbh878)

18 Feb 2019

Question 1-Computations with Metropolis-Hastings

Subquestion 1-Sample using Lognormal

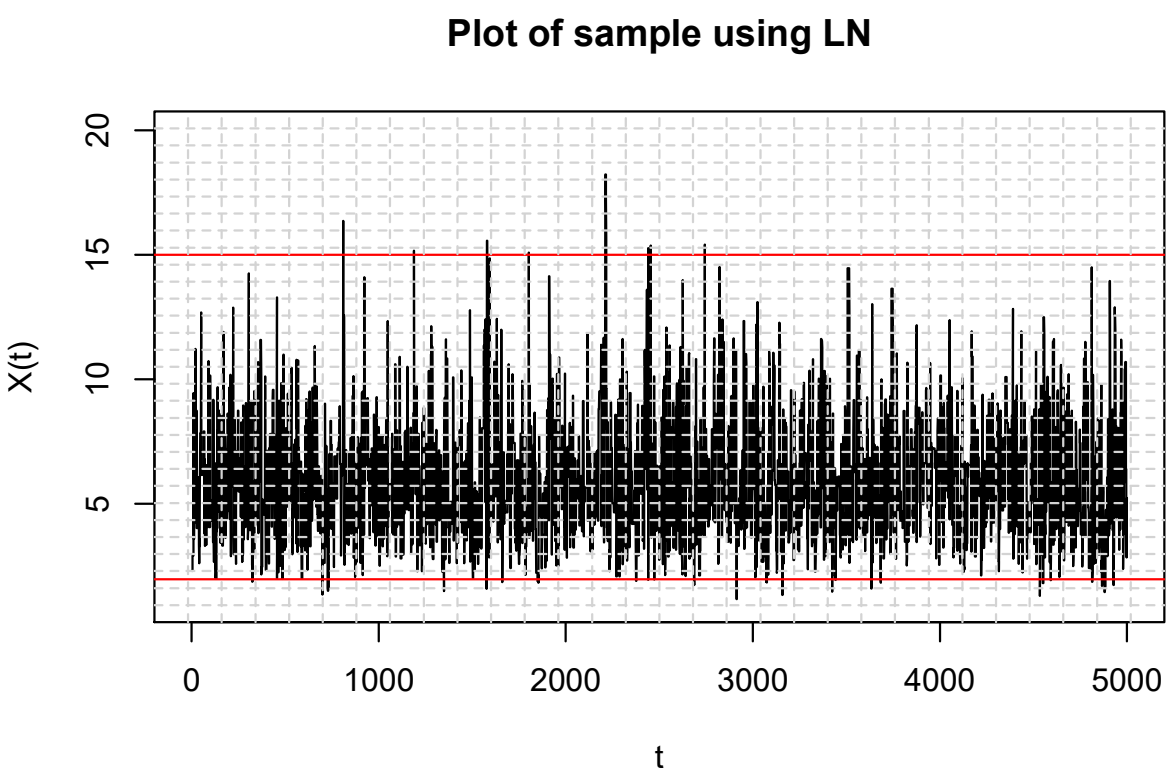
We have the target probability function $f(x) = x^5 e^{(-x)}, x > 0$ and we are going to generate samples using Metropolis-Hastings Algorithm by using $LN(X_t, 1)$ as proposal distibution.So sampling from our proposal we are going to use Metropolis-Hastings to approximate our target function.

```
#set seed
set.seed(123456)
#this is our target function
target_dist<-function(x){
  return(x^5*exp(-x))
  }

#initialize rejection counts
Rej<-0
#metropolis-hastings using lognormal
#input(n sample,starting point,sd given,T to return output ,T to return plot)
f.MCMC.MH<-function(nstep,X0,props,output=T,draw_plot=T){
  #initialize the first point
  X0<-X0
  #vector to use in the plot
  vN<-1:nstep
  #initialize our sample vector
  vecX<-rep(X0,nstep);
  #for loop for the algorithm
  for(i in 2:nstep){
    #take the previous point in our sample
    X<-vecX[i-1]
    #create candidate poiint from log-normal
    Xcand<-rlnorm(1,meanlog=log(X),sdlog=props)
    #random point
    u<-runif(1)
    #numerator of ratio
    num <- (target_dist(Xcand)*dlnorm(X,meanlog=log(Xcand),sdlog=props))
    #denumerator of ratio
    den <- (target_dist(X)*dlnorm(Xcand,meanlog=log(X),sdlog=props))

    #a<-min(c(1,(target_dist(Y)*dlnorm(X,meanlog=Y,sdlog=props)/(target_dist(X)*dlnorm(Y,meanlog=X,sdlo
    a<-min(1,num/den)
    #check condition if T accept the candidate point
    if (u <=a){
        vecX[i]<-Xcand
    }
    #reject candidate stay in the current point
    else{
```

```
      vecX[i]<-X
      #rejection count
      Rej<-Rej+1
      }

  }
  #draw plot
  if(draw_plot==T){

  plot(vN,vecX,pch=19,cex=0.3,col="black",xlab="t",ylab="X(t)",
       main="Plot of sample using LN",ylim=c(1,20),type="l")
  abline(h=0)
  abline(h=1.96,col="red")
  abline(h=15,col="red")
  grid(30,30,col="lightgray")
  }
  #return output of sample,rejections
 if(output==T){
  return(list("sample"=vecX,"rejections"=Rej))
 }

}
```

Plot of the chain using LN proposal

```
set.seed(123456)
#create a 5000 sample from our proposal
f<-f.MCMC.MH(5000,rlnorm(1,0,1),1)
```




```
cat("The rejection rate is :",f$rejections/5000)
```

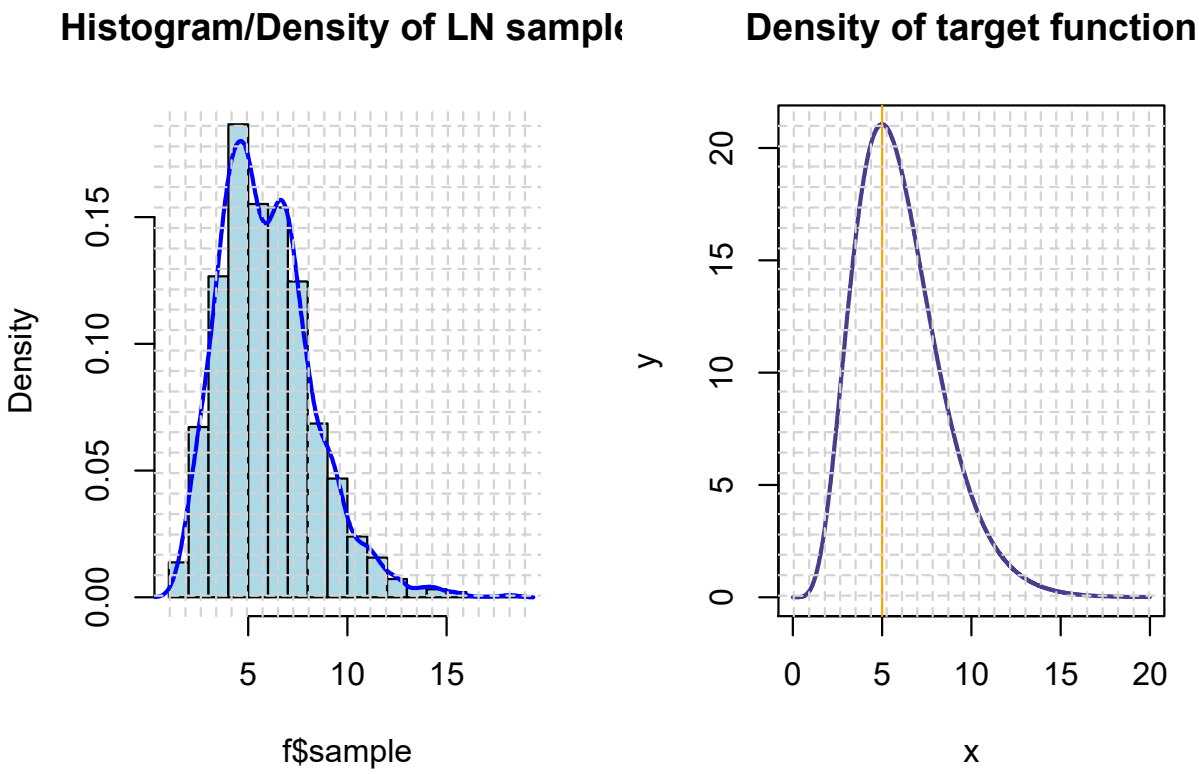
```
## The rejection rate is : 0.5582
```

The plot shows that we have well mixing chain and our algorithm creates candidate points that mostly get accepted and we move to other points but we have a big range of our points.When we examine the rejection rate which is 0.55 means that half of the candidate point rejected and we stay in the same point.

Plot of the densities

The above plots show the density from our sample and the target density we need to sample from.As we can see the lognormal is good approximation of our target denstity.

```
par(mfrow=c(1,2))
hist(f$sample,prob=T,col="lightblue",main="Histogram/Density of LN sample")
lines(density(f$sample),col="blue",lwd=2)
grid(25,25,col="lightgray")

#target
x=seq(0,20,0.01)
y=target_dist(x)
plot(x,y,type="l",main="Density of target function",
     col="darkslateblue",lwd=2)
abline(v=5,col="orange")
grid(25,25,col="lightgray")
```



Subquestion 2-Sample using Chi-square

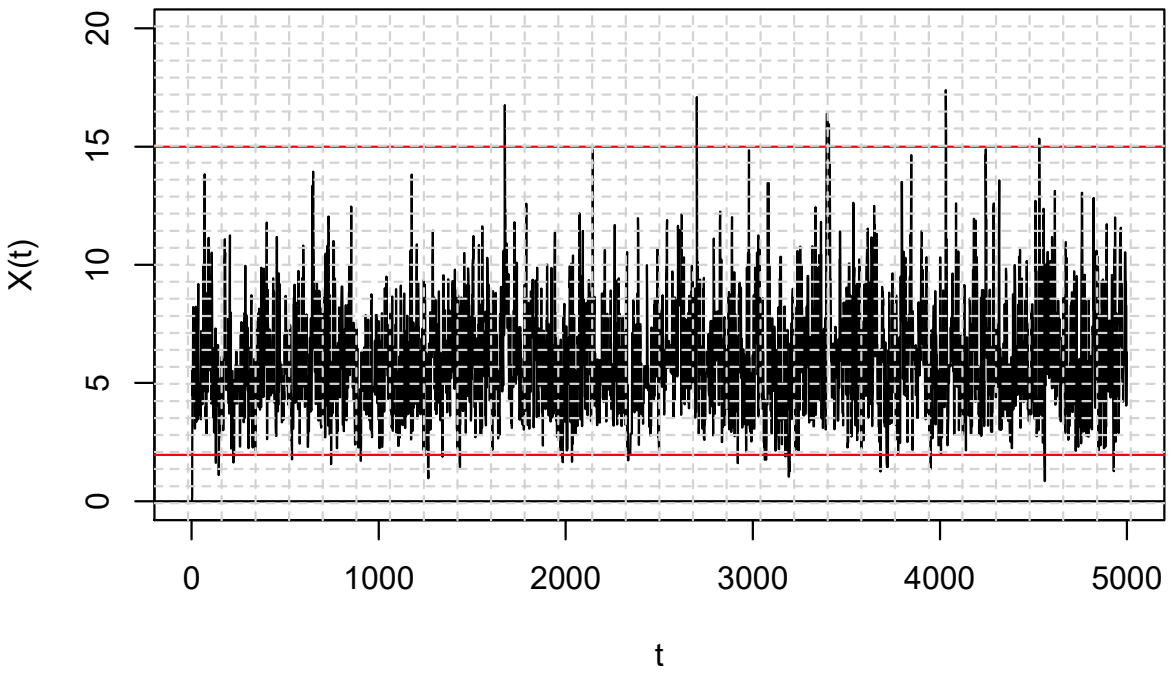
We are now going to use a new proposal distribution to sample from which is $x^2 = (fraction(X_t + 1)|)$ in order to see if we will get a better approimation for our target distribution.

```
#initialize rejection counts
Rej1<-0
#our finction to sample from chi-square
f.MCMC.MH1<-function(nstep,X0,output=T,draw_plot=T){
  X0<-X0
  #vector to use for the plot
  vN<-1:nstep
  #initialize our sample vector
  vecX<-rep(X0,nstep);
  #for lopp to create sample
  for (i in 2:nstep){
    #take the previous point in our sample
    Xt<-vecX[i-1]
    #create candidate using chi-square
    Xcand<-rchisq(1,df=floor(Xt+1))
    #
    u<-runif(1)
    #numerator of the ratio
    num <- target_dist(Xcand)*dchisq(Xt, df=floor(Xcand+1))
    #denumerator of the ratio
    den <- target_dist(Xt)*dchisq(Xcand, df=floor(Xt+1))
```

```
    #a<-min(c(1,(target_dist(Y)*dchisq(Xt,df=floor(Y+1))/(target_dist(Xt)*dchisq(Y,df=floor(Xt+1))))))
    a<-min(1,num/den)
    #check condition if T accept the candidate point
    if (u <=a){
      vecX[i]<-Xcand
    }
    #reject candidate stay on current point
    else{
      vecX[i]<-Xt
      #rejection count
      Rej1<-Rej1+1}
  }

  #return plot
  if(draw_plot==T){
  plot(vN,vecX,pch=19,cex=0.3,col="black",
       xlab="t",ylab="X(t)",main="",ylim=c(0,20),type="l")
  abline(h=0)
  abline(h=1.96,col="red")
  abline(h=15,col="red")
  grid(30,30,col="lightgray")
  }
  #return output of sample,rejections
  if(output==T){
    return(list("sample"=vecX,"rejections"=Rej1))
  }

}
```

Plot of the chain using chi-square proposal

```
f2<-f.MCMC.MH1(5000,rchisq(1,1),T,T)
```




```
cat("The rejection rate is :",f2$rejections/5000)#rejection rate
```

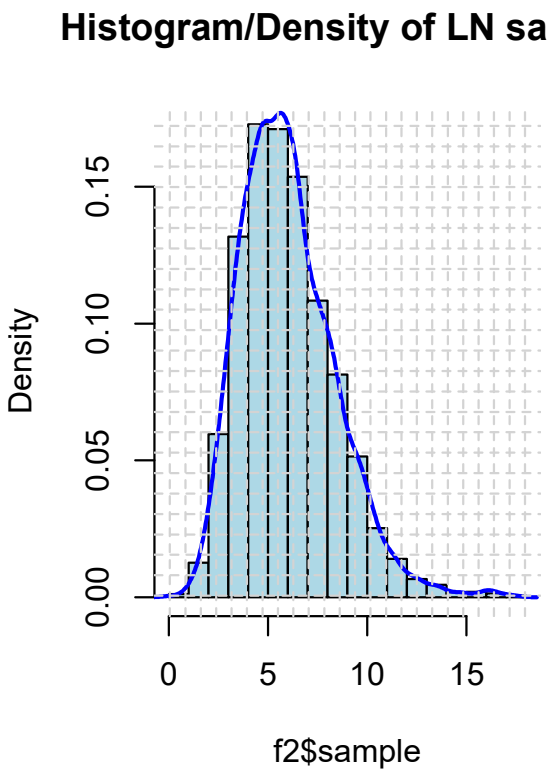
```
## The rejection rate is : 0.4062
```

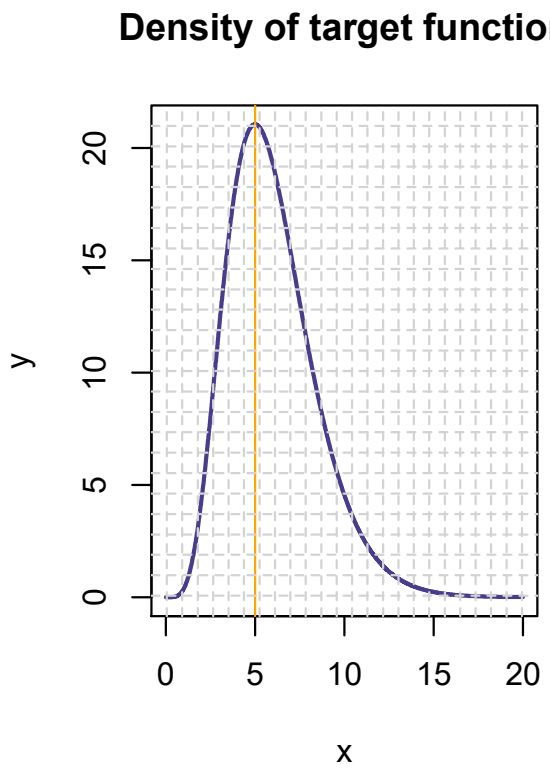
As we can see now from the plot of the chain we obtain we have similar with lognormal which means that also the function seems to approximate quite well the target function the points are chainging and we move to new points but now we have smaller range of the points.Looking at the rejection rate we can see that is smaller that the previous of lognormal.

Plot of the densities

The above plots show the density from our sample and the target density we need to sample from.As we can see the chi-square is very good approximation of our target denstity.

```
par(mfrow=c(1,2))
#histogram density of the sample
hist(f2$sample,prob=T,col="lightblue",main="Histogram/Density of LN sample")
lines(density(f2$sample),col="blue",lwd=2)
grid(25,25,col="lightgray")

# #target density
x=seq(0,20,0.01)
y=target_dist(x)
plot(x,y,type="l",main="Density of target function",
     col="darkslateblue",lwd=2)
abline(v=5,col="orange")
grid(25,25,col = "lightgray")
```





Subquestion 2-Sample using Chi-square

In conlusion comparing the 2 proposal distribution we can see that using Metropolis-Hastings Algorithm with lognormal we got well mixing chains meaning that the algorithm seems to explore a bigger span in the space and we dont stuck in the same point for long periods but the ranges of the points are diffrent with the range of the chi-square being smaller.Finally,we ploted the densities obtained by the 2 proposals against the target denstity and confirm that chi-square is a slightly better proposal than the lognormal with seems natural because its pdf looks more like the target function.

Subquestion 2-Analyze convergence with Gelman-Rubin method

In this section we are going to analyze the convergence of the 10 sequences using the chi-square proposal with starting points from $1, 2, ...10$ with the Gelman-Rubin method.

```
library(coda)
```

```
## Warning: package 'coda' was built under R version 3.5.2
```

```
set.seed(123456)
#number of sample
nsample<-5000
#number of starting points from 1...10
steps<-seq(1,10,1)
#initialize matrix(each row is a sample of 5000 for every starting point)
X1<-matrix(0,length(steps),nsample)
#for loop to fill the matrix
for(step in steps){
```

```
  X1[steps,]<-f.MCMC.MH1(nsample,step,output=T,draw_plot=F)$sample

}
#empty list
f1<-list()
#iterate every row and make it as mcmc object
for (i in 1:dim(X1)[1]){

  f1[[i]]<-as.mcmc(X1[i,])
}

#print gelman-rubin diagnostics
print(gelman.diag(f1))
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

The diagnostics from Gelman-Rubin method imply that an upper C.I. greater than 1 is an indicator that the method did not converge.In our case in seems that the method converges.

Subquestion 5-Integral estimation with sampling

we need to calculate the integral $\vartheta = \int_0^\infty xf(x)dx$.

Lets set $g(x) = x$ and $f(x) = x^5e^{-(x)}$ which is target function from before.The integral $\int_0^\infty f(x)dx = 1$ because its the CDF of the $f(x)$. So can now write that if X is a random variable that follows f(x) then:

$$\vartheta = int_0^\infty g(x)f(x)dx = E[g(X)]$$

and we know that expected value is the mean of the sample thus :

$$\hat{\vartheta} = \frac{1}{n}\sum_{i=1}^{n} g(xi) = \frac{1}{n}\sum_{i=1}^{n} xi, \quad \forall x_i \sim f(x)$$

What this formula tells us is that if we take a sample from our $f(x)$ and transform in using $g(x)$ and take the mean of the sample we can calculate the integral of $f(x)$ which in our case id the CDF of $f(x)$.

```
set.seed(123456)
#calculate mean sample using the lognormal sample
s1<-f.MCMC.MH(5000,rlnorm(1,0,1),1,output=T,draw_plot=F)
#calculate mean using the chi-square sample
s2<-f.MCMC.MH1(5000,rchisq(1,floor(1)),T,F)

cat("The integral estimation with a sample using log normal is :",mean(s1$sample))
```

```
## The integral estimation with a sample using log normal is : 5.952158
```

```
cat("\n-------------------------------------------------------------------")
```

```
##
## -------------------------------------------------------------------
```

```
cat("\nThe integral estimation with a sample using chi-square is :",mean(s2$sample))
```

```
##
## The integral estimation with a sample using chi-square is : 6.004897
```

Subquestion 6-Gamma distibution and its integral

The probability density for gamma distribution is given by the formula :

$$f(x;\alpha,\beta) = \frac{\beta^a x^{a-1e^{-\beta x}}}{\Gamma(a)} \quad for \quad x>0, \alpha,\beta>0$$

Where

$$\Gamma(a) = (a-1)!$$

if we set $\alpha = 1 \quad and \beta = 6$ we have:

$$f(x;\alpha=6,\beta=1) = \frac{x^5 e^{-x}}{120}(1)$$

we need to calculate

$$\int_{\infty}^{0} xf(x)dx(2)$$

if we assume that $f(x)$ is the gamma distribution (1) the requested interal (2) is the CDF Of the gamma distibution which is the expected value of X

$$\int_{\infty}^{0} xf(x)dx = E[X]$$

the expected value of the gamma distribution is then given by $E(X) = \frac{\alpha}{\beta} = \frac{6}{1} = 6$

Finally,the requested integral is 6.

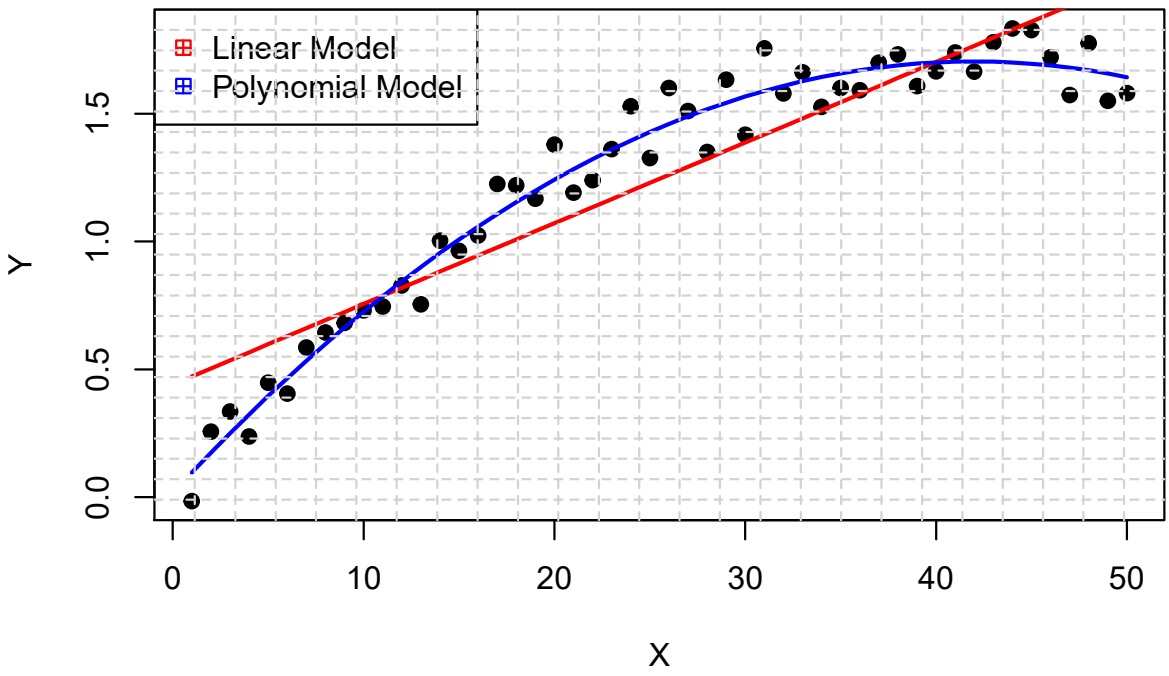
Question 2-Gibbs Sampling

Subquestion 1-Plot Dependence on chemical data

```
load("chemical.Rdata")
```

The above plot shows the dependance of the Y (measured concentration of the chemical) vs X(day of the measurement) for the chemical data given. We have plotted a liner model and a 2nd order polynomial model which seems to be better model for the given data.

```
#linear model
mod1<-lm(Y~X)
#second order polynomial model
mod2<-lm(Y~poly(X,2))
#preds for linear model
Y_lm=predict(mod1)
#preds for poly model2
Y_lm2=predict(mod2)
#plot
plot(X,Y,col="black",pch=19)
lines(X,Y_lm,col="red",lwd=2)
lines(X,Y_lm2,col="blue",lwd=2)
legend("topleft" ,pch=12, col=c("red", "blue"), c("Linear Model", "Polynomial Model"))
grid(25,25,col="lightgray")
```



Subquestion 2-Posterior Prior Bayesian Model

we are given that $Y_i \sim N(\mu_i, \sigma^2)$ $i = 1, 2, ..., n$ $\overrightarrow{\mu} = (\mu_1, \mu_2, ..., \mu_n)$ $p(\mu_1) = 1$ $p(\mu_{i+1}, \mu_i \sim N(\mu_i, \sigma^2) \quad i = 1, 2, 3, ..n-1$

The pdf of the Y_i is given by the formula :

$$f(Y_i/\mu_i, \sigma^2) = \frac{1}{\sqrt{2}\pi\sigma^2} e^{-(\frac{(Y_i-\mu_i)^2}{2\sigma^2})}$$

Then the Likekihood is given by :

$$L(Y_1, Y_2, ..., Y_n/\mu_i, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2}\pi\sigma^2} e^{-(\frac{(Y_i-\mu_i)^2}{\sigma^2})}$$

$$p(\overrightarrow{Y}/\overrightarrow{\mu}) \propto e^{-(\sum_{i=1}^{n} \frac{(Y_i-\mu_i)^2}{\sigma^2})} (1)$$

We now need to derive the prior which is given by

$$p(\overrightarrow{\mu}) = p(\mu_1)p(\mu_2/\mu_1)...p(\mu_n/\mu_{n-1}) =$$
$$\prod_{i=1}^{n-1} \frac{1}{\sqrt{2}\pi\sigma^2} e^{(-\frac{(\mu_{i+1}-\mu_i)^2}{\sigma^2})}$$

$$p(\overrightarrow{\mu}) \propto e^{-(\sum_{i=1}^{n-1} \frac{(\mu_{i+1}-\mu_i)^2}{\sigma^2})} (2)$$

we know that

$$Posterior \propto Prior \;\; Likehihood$$

or

$$p(\overrightarrow{\mu}/\overrightarrow{Y}) \propto p(\overrightarrow{Y}/\overrightarrow{\mu}) \;\; p(\overrightarrow{\mu})$$

so we have

$$(1) * (2) = e^{(-\sum_{i=1}^{n-1} \frac{(\mu_{i+1}-\mu_i)^2}{2\sigma^2} - \sum_{i=1}^{n} \frac{(Y_i-\mu_i)^2}{2\sigma^2})}$$

Subquestion 3-Posterior up to a constant proportionality and conditional probabilities

We know that

$$p(x_j/x_1, x_2, x_3, ..., x_n) \propto \frac{p(x_1, x_2, x_3..., x_n)}{p(x_1, x_2, ..., x_{j-1}, x_{j+1}, ...x_n)}$$

using the previous formula we are going to find :

$$p(\mu_n/\overrightarrow{\mu_{-n}}/\overrightarrow{Y}) \propto \frac{p(\overrightarrow{\mu}/\overrightarrow{Y})}{p(\overrightarrow{\mu_{-n}}/\overrightarrow{Y})}$$

so we have

$$\frac{e^{(-\sum_{i=1}^{n-1} \frac{(\mu_{i+1}-\mu_i)^2}{2\sigma^2} - \sum_{i=1}^{n} \frac{(Y_i-\mu_i)^2}{2\sigma^2})}}{e^{(-\sum_{i=1}^{n-2} \frac{(\mu_{i+1}-\mu_i)^2}{2\sigma^2} - \sum_{i=1}^{n-1} \frac{(Y_i-\mu_i)^2}{2\sigma^2})}} =$$

$$e^{-\frac{1}{2\sigma^2}[(\mu_n-\mu_{n-1})^2+(Y_n-\mu_n)^2]} =^{(HINTB)}$$

$$e - \frac{(\mu_n - (\mu_{n-1} + Y_n)/2)^2}{2\sigma^2/2} \sim N(\frac{\mu_{n-1+Y_n}}{2}, \frac{\sigma^2}{2})$$

Next we need to find

$$p(\mu_i/\overrightarrow{\mu_{-i}}/\overrightarrow{Y}) \propto \frac{p(\overrightarrow{\mu}/\overrightarrow{Y})}{p(\overrightarrow{\mu_{-i}}/\overrightarrow{Y})}$$

so now we have

$$\frac{e^{(-\sum_{i=1}^{n-1} \frac{(\mu_{i+1}-\mu_i)^2}{2\sigma^2} - \sum_{i=1}^{n} \frac{(Y_i-\mu_i)^2}{2\sigma^2})}}{e^{(-\sum_{j=i}^{i-2} \frac{(\mu_{j+1}-\mu_j)^2}{2\sigma^2} - \sum_{j=i+1}^{n-1} \frac{(\mu_{j+1}-\mu_j)^2}{2\sigma^2} \sum_{j=1}^{i-1} \frac{(Y_j-\mu_j)^2}{2\sigma^2}) - \sum_{j=i+1}^{n} \frac{(Y_j-\mu_j)^2}{2\sigma^2})}} =$$

$$e^{-\frac{1}{2\sigma^2}[(\mu_{i+1}-\mu_i)^2+(\mu_i-\mu_{i-1})^2+(Y_i-\mu_i)^2]} =^{(HINTC)}$$

$$e^{-\frac{(\mu_i-(\mu_{i-1}+Y_i+\mu_{i+1})/3)^2}{2\sigma^2/3}}$$

Using the same steps we obtain also that

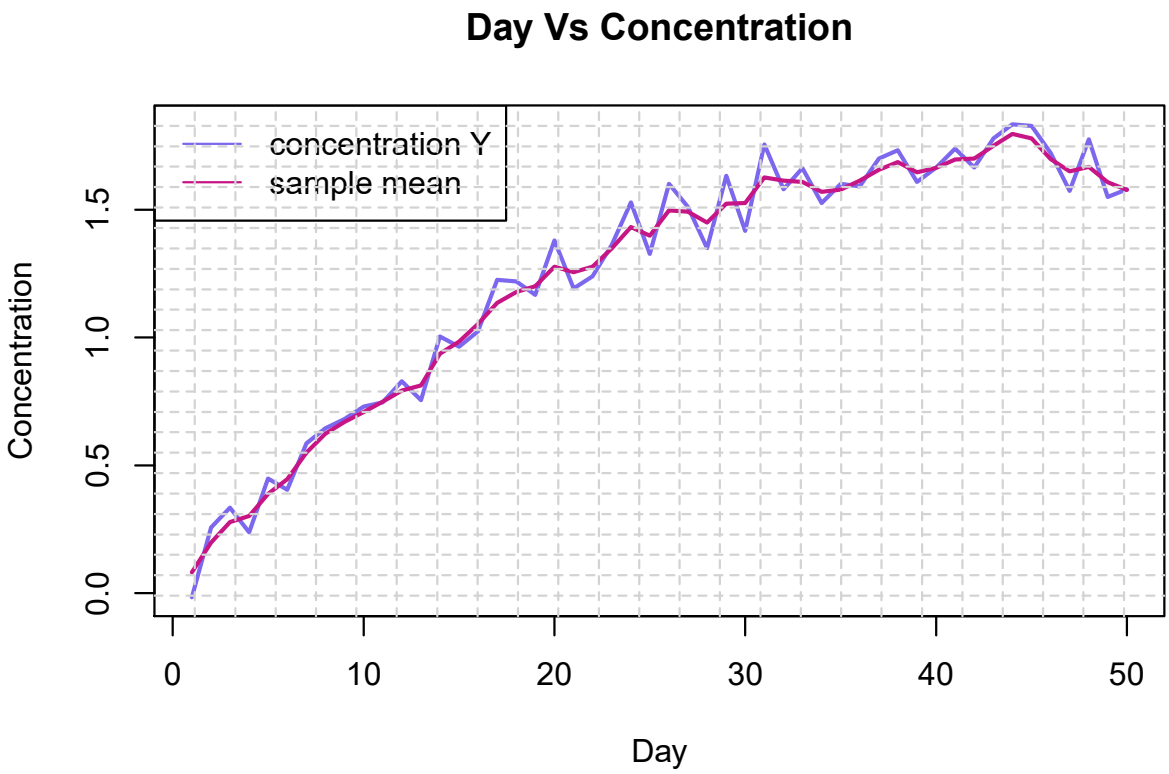
$$p(\mu_1/\overrightarrow{\mu_{-1}}/\overrightarrow{Y}) \propto e^{-\frac{1}{2\sigma^2}[(\mu_2-\mu_1)^2+(Y_1-\mu_1)^2]} \propto e^{-\frac{(\mu_1-(\mu_2+Y_1)/2)^2}{2\sigma^2/2}}$$

Subquestion 4-Sampling with Gibbs

```
#gibbs sampling function
f.MCMC.Gibbs <- function(nstep,X0,sd){

  mX<-matrix(0,nrow=length(X0),ncol=nstep)

  for (t in 1:(nstep-1)){
    for (i in 1:length(X0)){
      if (i==1){
        y <- (mX[i+1,t]+X0[i])/2
        mX[i,t+1] <- rnorm(1,y,sd/sqrt(2))
      }
      else if (i ==50){
        y <- (mX[i-1,t+1]+X0[i])/2
        mX[i,t+1] <- rnorm(1, y, sd/sqrt(2))
      }
      else{
        y <- (mX[i+1,t]+ mX[i-1,t+1] +X0[i])/3
        mX[i,t+1] <- rnorm(1,y,sd/sqrt(3))
      }
    }
  }
 return(mX)
}
```

Plot sampling with Gibbs

```
#create df to store the loaded data
chemical<- data.frame(X,Y)
X0<-chemical$Y
#call function to get sample
gib <- f.MCMC.Gibbs(1000, X0, sqrt(0.2))

#plot of expected value gib vs X and Y vs X
plot(chemical$X, chemical$Y, main="Day Vs Concentration", col="mediumslateblue",
     xlab="Day", ylab="Concentration", type="l",lwd=2)
points(chemical$X,rowMeans(gib), col="mediumvioletred", type="l",lwd=2)
legend("topleft", legend=c("concentration Y","sample mean"), col=c("mediumslateblue","mediumvioletred")
grid(25,25)
```

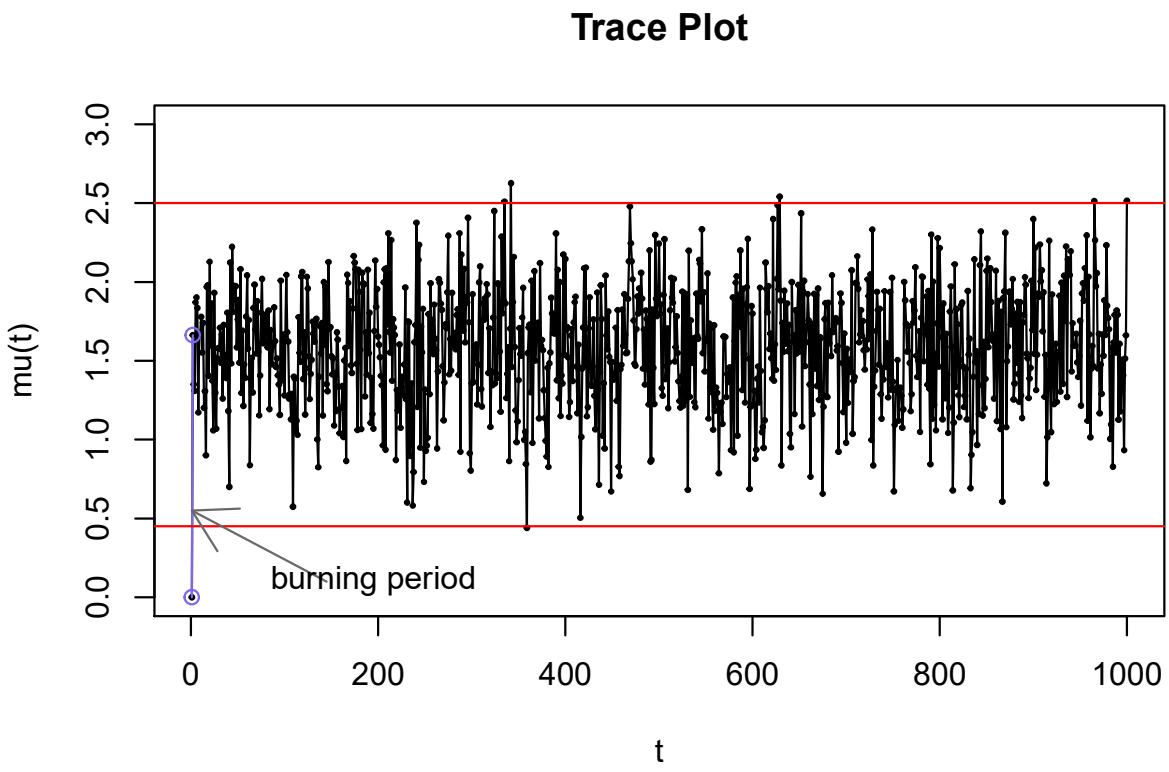


From the above plot we can conclude that the observed value of $\vec{\mu}$ seems to follow the patter of dependence between Y and X.

Subquestion 5-Trace Plot

Trace plot

```
vN<-1:length(gib[50,])
plot(vN,gib[50,],pch=19,cex=0.3,col="black",xlab="t",ylab="mu(t)",
     main="Trace Plot",ylim=c(0,3),type="o")
points(gib[50,1:2],col="mediumslateblue",type="o")
abline(h=0.45,col="red")
abline(h=2.5,col="red")

arrows(145,0.1,1,0.55,col="dimgrey")
text(195,0.1,c("burning period"))
```



The plot shows the clain for the last μ_{50} with blue we have 2 observations that seem not follow the overall pattern and we may consider them as burning period and we can discurd them from overall plot.

$\vec{\mu} = (\mu_1, \ldots, \mu_n)$

$Y_i \sim N(\mu_i, \sigma^2 = 0.2)$, $i = 1, \ldots, n$

$P(\mu_1 = 1)$

$P(\mu_{i+1} | \mu_i) = N(\mu_i, 0.2)$, $i = 1, \ldots, n-1$

Lab 4

(Q2)

Gibbs sampler:-

$$X_{t+1,i} \sim f(\cdot | X_{t+1,1}, \ldots, X_{t+1,i-1}, X_{t,i+1}, \ldots, X_{t,d})$$

We need to sample (to get a value for a particular X)

$f(\cdot | X_{t+1,1}, \ldots, X_{t,d})$ ← conditional distr

$\cdot$ → we are interested in

$X_{t+1,1}, \ldots, X_{t,d}$ → will be constant since we will integrate over this.

We are interested in the conditional distr., so from Baye's theorem,

$P(\theta|D) \propto P(D|\theta) P(\theta)$

Posterior, Likelihood, Prior

We are given,

$\vec{\mu} = (\mu_1, \ldots, \mu_n)$ ← unknown parameters

$Y_i \sim N(\mu_i, \text{variance} = 0.2)$, $i = 1, \ldots, n$

where prior is,

$P(\mu_1) = 1$

$P(\mu_{i+1}, \mu_i) = N(\mu_i, 0.2)$, $i = 1, \ldots, n-1$

$$\therefore P(\vec{\mu}|\vec{y}) = \frac{P(\vec{y}|\vec{\mu})\,P(\vec{\mu})}{\int P(\vec{y}|\vec{\mu})\,P(\vec{\mu})\,d\vec{\mu}}$$

$$P(\vec{\mu}|\vec{y}) \propto P(\vec{y}|\vec{\mu})\,P(\vec{\mu})$$

$$\therefore \text{w.k.t. } N(\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\frac{-1}{2\sigma^2}(x-\mu)^2}$$

$$\therefore \underline{\text{Likelihood}},\quad L(\vec{y}|\vec{\mu}) = \frac{1}{(\sqrt{2\pi\sigma^2})^n} \prod_{i=1}^{n} e^{\frac{-1}{2\sigma^2}(y_i-\mu_i)^2}$$

$$\underline{\text{Prior}},\quad P(\mu_{i+1}|\mu_i) = N(\mu_i, 0.2) = \underline{P(\mu_1)}\,P(\mu_2|\mu_1)\cdots P(\mu_n|\mu_{n-1})$$

$[i = 1, \ldots, n-1]$

$$\therefore P(\vec{\mu}) = \frac{1}{(\sqrt{2\pi\sigma^2})^{n-1}} \prod_{i=1}^{n-1} e^{\frac{1}{2\sigma^2}(\mu_{i+1}-\mu_i)^2}$$

$P(\vec{\mu}|\vec{y}) \propto L(\vec{y}|\vec{\mu}) \cdot P(\mu_{i+1}|\mu_i)$

↳ conditional dist.

Now, the posterior we are interested in:

$$P(\vec{\mu}|\vec{y}) \propto e^{\frac{-1}{2\sigma^2}\sum_{i=1}^{n}(\mu_{i+1}-\mu_i)^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu_i)^2}$$

$$\hookrightarrow \propto \left[e^{\frac{-1}{2\sigma^2}\left[\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2 + (y_i-\mu_i)^2\right] - \frac{1}{2\sigma^2}(y_n-\mu_n)^2}\right]$$

Now,

~~considering~~ $P(\mu_i|\vec{\mu}_{-i}, \vec{y})$ → ~~R(μ)~~

Considering, $P(\mu_1|\vec{\mu}_{-1}, \vec{y}) \rightarrow P(\mu_1|\mu_2, \mu_3, \ldots \mu_n, \vec{y})$

↳ we are interested only in μ_1

$$P(\mu_1|\vec{\mu}_{-1}, \vec{y}) \propto \exp\left[\frac{-1}{2\sigma^2}(\mu_2-\mu_1)^2 + (y_1-\mu_1)^2\right]$$

$$= \exp\left[-\frac{1}{2\sigma^2}(\mu_1-\mu_2)^2 + (\mu_1-y_1)^2\right]$$

On comparing with hint,

$$\exp\left[\frac{-1}{d}\left((x-a)^2+(x-b)^2\right)\right] \propto \exp\left[\frac{-(x-(a+b)/2)^2}{d/2}\right]$$

$$\Rightarrow \exp\left[\frac{-(\mu_1-(\mu_2+y_1)/2)^2}{2\sigma^2/2}\right]$$

$$\therefore P(\mu_1 \mid \vec{\mu}_{-1}, \vec{y}) \sim N\left(\frac{\mu_2+y_1}{2}, \frac{\sigma^2}{2}\right) \quad \text{——— (1)}$$

(2/4) Now,

Considering $P(\mu_n \mid \vec{\mu}_{-n}, \vec{y})$,

$$\propto \exp\left[\frac{-1}{2\sigma^2}\left((\mu_n-\mu_{n-1})^2+(\mu_n-y_n)^2\right)\right]$$

$$\propto \exp\left[\frac{-(\mu_n-(\mu_{n-1}+y_n)/2)^2}{\sigma^2}\right]$$

$$\sim N\left(\frac{\mu_{n-1}+y_n}{2}, \frac{\sigma^2}{2}\right) \quad \text{——— (2)}$$

(3) Now,

Taking $i=3$,

$$\sum_{i=2}^{3}(\mu_{i+1}-\mu_i)^2+(y_i-\mu_i)^2$$

$$=\underbrace{(\mu_3-\mu_2)^2+(y_2-\mu_2)^2}_{i=2}+\underbrace{(\mu_4-\mu_3)^2+(y_3-\mu_3)^2}_{i=3}$$

$$\propto \exp\left[\frac{-1}{2\sigma^2}\left((\mu_i-\mu_{i-1})^2+(\mu_{i+1}-\mu_i)^2+(y_i-\mu_i)^2\right)\right]$$

$$\propto \exp\left(\frac{-(\mu_i-(\mu_{i-1}+\mu_{i+1}+y_i)/3)^2}{2\sigma^2/3}\right)$$

$$\therefore P(\mu_i \mid \vec{\mu}_{-i}, \vec{y}) \sim N\left(\frac{\mu_{i-1}+\mu_{i+1}+y_i}{3}, \frac{2\sigma^2}{3}\right) \quad \text{——— (3)}$$

Lab5 Computational Statistics

Andreas C Charitos(andch552) Omkar Bhutra (omkbh878)

27 Feb 2019

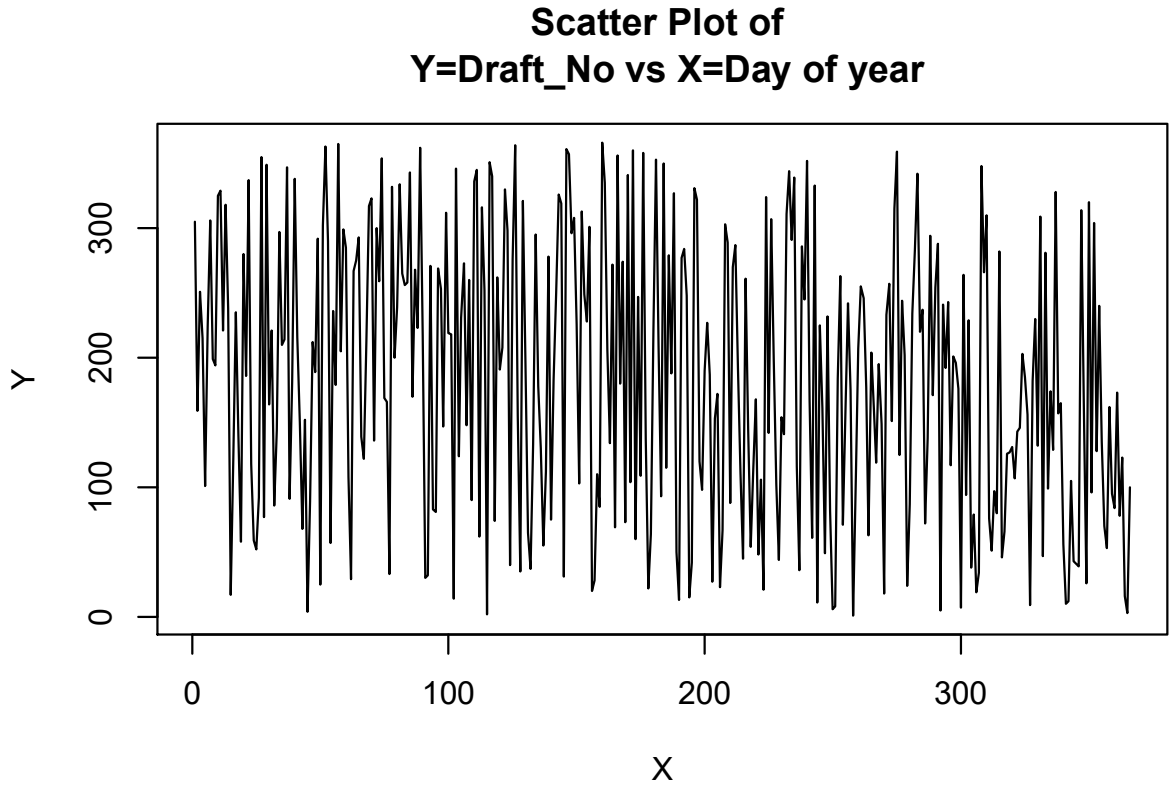
Question 1-Hypothesis testing

Subquestion 1-Scatterplot

```
#import the data
lottery<-readxl::read_excel("lottery.xls")
lottery<-as.data.frame(lottery)
```

The above plot shows the connection between Y=Draft_No (sorted by day of year) and X=Day of_year.As we conclude there doesn't seems to exists a specific patter so the relationship might considered as random.

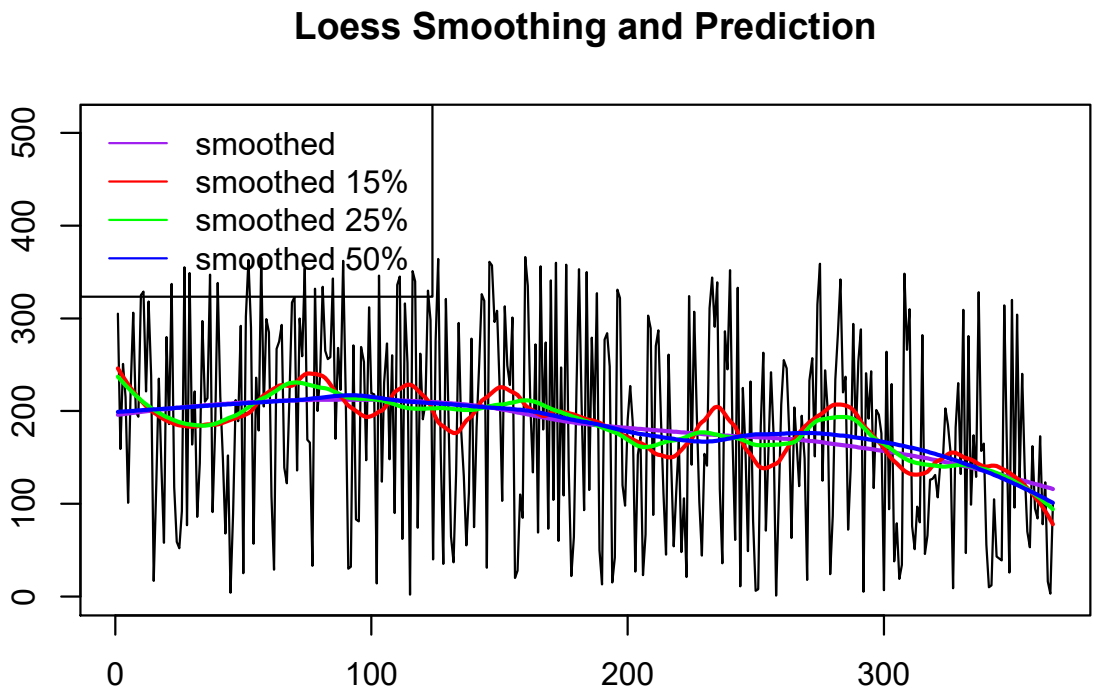
```
#assign the corresponding varaibles
Y=lottery$Draft_No
X=lottery$Day_of_year
#plot the data
plot(X,Y,type="l",main="Scatter Plot of \n Y=Draft_No vs X=Day of year")
```



Subquestion 2-Scatterplot with loess

Next we are compute $\hat{Y}$ using loess smoother with with 4 diffrent spans (default,15%,25%,50%).Again we can see that the curves are indicating that there seems to be a pattern in the original line so we can say that the relationship between Y and X is following a trend and doesn't seems to be random.

```
#calculate loess with diffrent spans
loessMod<-loess(Y~X)
loessMod15 <- loess(Y ~ X, span=0.15) # 15% smoothing span
loessMod25 <- loess(Y ~ X, span=0.25) # 25% smoothing span
loessMod50 <- loess(Y ~ X, span=0.50) #50% smoothing span

#make predictions with the loess models
smoothed<-predict(loessMod)
smoothed15 <- predict(loessMod15)
smoothed25 <- predict(loessMod25)
smoothed50 <- predict(loessMod50)

#scatterplot with loess models lines
plot(x=X,y=Y, type="l", main="Loess Smoothing and Prediction",
     xlab="", ylab="",ylim = c(0,510))
lines(smoothed,  x=X,col="purple",lwd=2)
lines(smoothed15, x=X, col="red",lwd=2)
lines(smoothed25, x=X, col="green",lwd=2)
lines(smoothed50, x=X, col="blue",lwd=2)
legend("topleft",legend = c("smoothed","smoothed 15% ",
                            "smoothed 25%","smoothed 50%"),col=c("purple","red","green","blue"),lty=1)
```



Subquestion 3-Randomness statistic evaluation

We are now going to check if the lottery is random using the following statistic:

$$T = \frac{\hat{Y}(X_b) - \hat{Y}(X_a)}{X_b - X_a} \quad , where \quad X_b = argmin_X Y(X), X_a = argmin_X Y(X)$$

by using by using a non parametric bootstrap for the previous statistic T with B = 2000

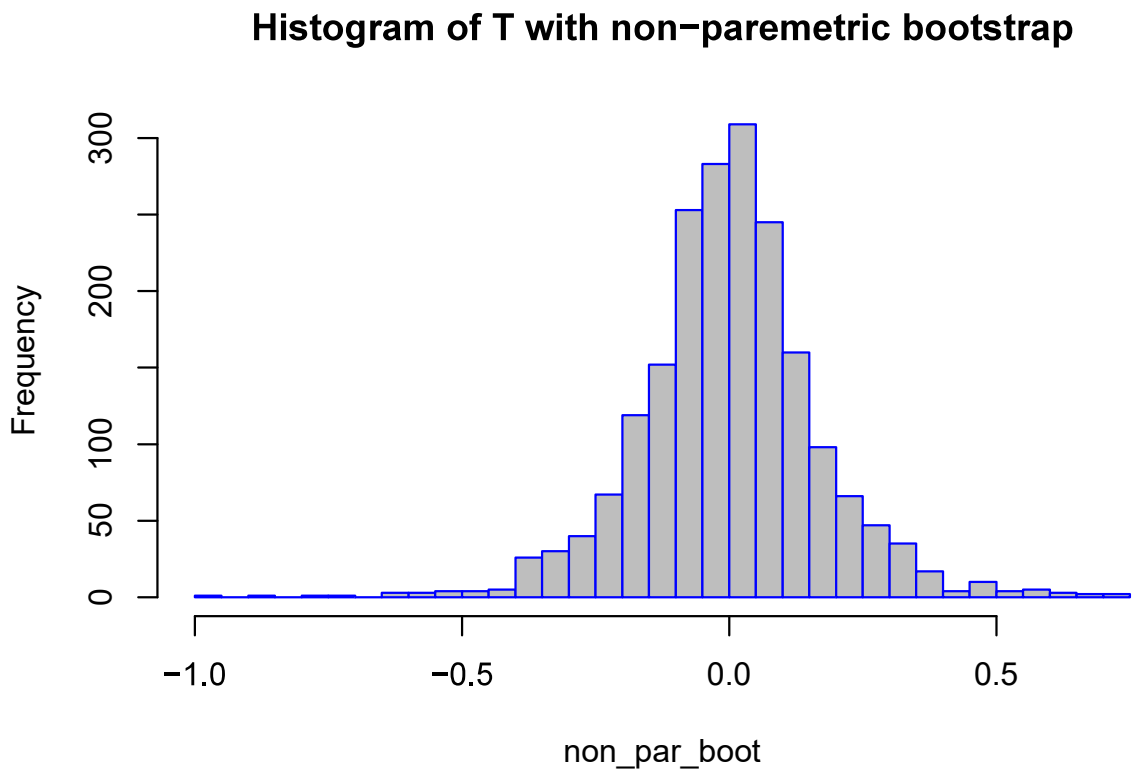
```
#set seed
set.seed(12345)

B=2000
#initialize vector for store the statistics
non_par_boot<-rep(0,B)

for(i in 1:B){
  #sample from the Y valus with replacement
  Y_samp<-sample(1:length(Y),length(Y),replace=T)
  #make a matrix
  dat<-cbind(X,Y_samp)
  #calculate the Xa,Xb
  Xb<-dat[which.max(dat[,2])] # the X that has max Y
  Xa<-dat[which.min(dat[,2])] # the Y that has min Y
  #predict model
  model<-loess(Y_samp ~ X,data=as.data.frame(dat),method="loess")
  #calculate Ya,Yb
  Y_xb<-model$fitted[Xb]
  Y_xa<-model$fitted[Xa]

  Tau<-(Y_xb-Y_xa)/(Xb-Xa)
  non_par_boot[i]<-Tau


}

#calculate the p value
p_val<-sum(non_par_boot>=0)/B

cat("The pvalue for the bootstrap sample is :",p_val)
```

```
## The pvalue for the bootstrap sample is : 0.5035
```

The p-value obtained is much larger that zero thus we can conclude that there seems to be a trend that the lottery is following and we conclude that is non random

Plot of the distribution of T with non parametric bootstrap

```
#histogram of the statistic distribution
hist(non_par_boot,breaks = 30,col="gray",include.lowest = TRUE,
     border="blue",main="Histogram of T with non-paremetric bootstrap")
```



The above plot shows the distribution of the statistic T obtained with non parametric bootstrap we can say that is following a normal curve but with long tails.

Subquestion 4-Hypothesis testing using permutation

In this part we are going to implement hypothesis tasting for our T statistic by using a permutation sampling.The null and alternative hypotheseis are :

$$H_o : Lottery \;\; is \;\; random$$

$$H_1 : Lottery \;\; is \;\; not \;\; random$$

```
set.seed(12345)

#sample size
B=2000
#permutation function
perm_func<-function(Y_value,X_value,B){

  call = match.call()
  ####################################################################
  #calculate the T statistic from the original data
  dat_origin<-cbind(X_value,Y_value)
  Xb_origin<-X_value[which.max(Y_value)]
  Xa_origin<-X_value[which.min(Y_value)]
```

```
  model_origin<-loess(Y_value ~ X_value,
                      data=as.data.frame(dat_origin),method="loess")

  Y_xb_origin<-model_origin$fitted[Xb_origin]
  Y_xa_origin<-model_origin$fitted[Xa_origin]

  Tau_origin<-(Y_xb_origin-Y_xa_origin)/(Xb_origin-Xa_origin)
  ################################################################

  #initialize vector to store perm statistics
  perm_Tau<-rep(0,B)

  for(i in 1:B){
    #take a sample from Y value without replacement
    Y_perm<-sample(1:length(Y_value),length(Y_value),replace=F)
    #make matrix
    dat_perm<-cbind(X_value,Y_perm)

    Xb<-dat_perm[which.max(dat_perm[,2])] #take X with max Y_perm
    Xa<-dat_perm[which.min(dat_perm[,2])]  #take X with min Y_prm
    #fit loess model
    model_perm<-loess(Y_perm ~ X_value,data=as.data.frame(dat_perm),method="loess")
    #calculate Ya,Yb for the predicted permutated data
    Y_xb<-model_perm$fitted[Xb]
    Y_xa<-model_perm$fitted[Xa]

    Tau_perm<-(Y_xb-Y_xa)/(Xb-Xa)  #calculate the T statistic
    perm_Tau[i]<-Tau_perm  #
  }

  perm_Tau<-perm_Tau
  p_value_perm<-sum(abs(perm_Tau)>=abs(Tau_origin))/B

  return(list(perm_Tau=perm_Tau,
              p_value_perm=p_value_perm,
         call=call))

}

#calculate B-permutated statistics
res=perm_func(lottery$Draft_No,lottery$Day_of_year,B)

cat("The pvalue from 2000 permutated sample is :",res$p_value_perm)
```

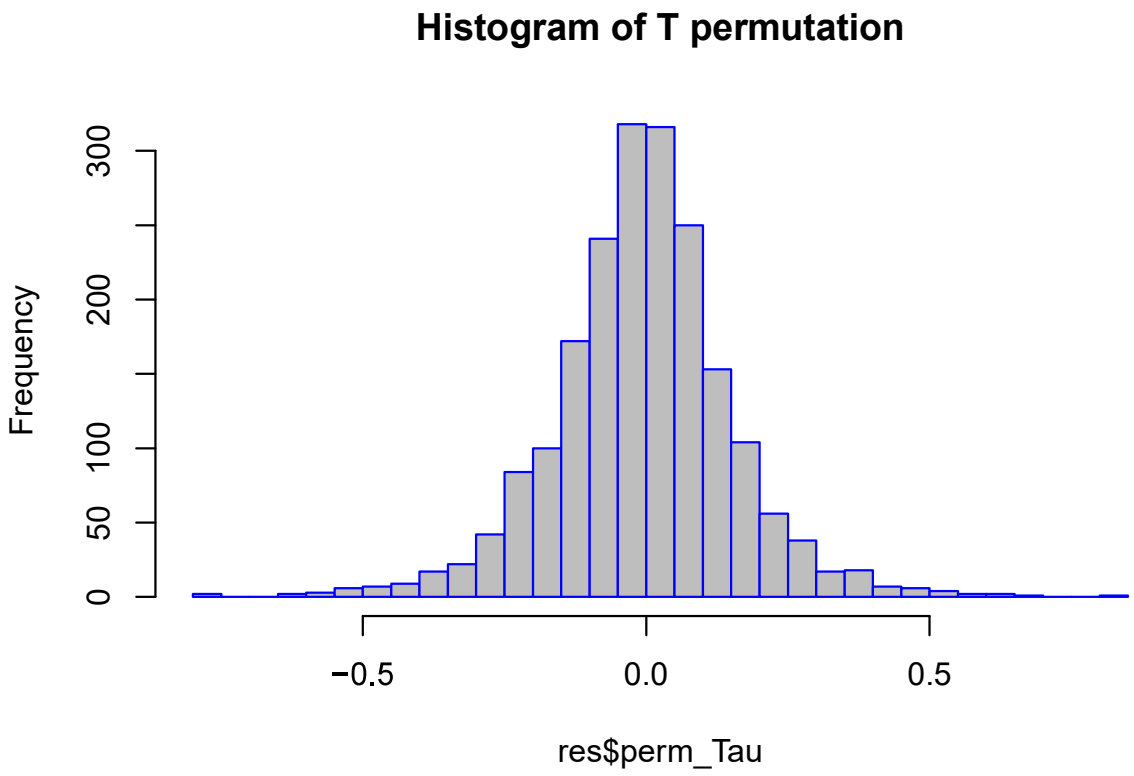
```
## The pvalue from 2000 permutated sample is : 0.086
```

From thhe obtained p-value which is 0.086 and using siginificant level $\alpha = 0.05$ since the p-value is greater we can't reject the null hypothesis, so according to our statistic usinf permutation the lottery is random.

Plot of the distribution of T with permuation

The above plot shows the distribution of the T with permutation sampling again the data seem to follow a normal curve but with long tails.

```r
#histogram of the permutated data
hist(res$perm_Tau,breaks = 30,col="gray",include.lowest = TRUE,
     border="blue",main = "Histogram of T permutation")
```



Subquestion 5-Hypothesis testing with permutation and generated data

Subquestion 5.a-Generate non-random data

In this part we are going to generate data according to the following pattern

$$Y(x) = max(0, min(\alpha x + \beta, 366)), \quad \alpha = 0.1, \quad \beta \sim N(183, sd = 10)$$

using X(Day of year) and we are going to implement the previous hypothesis testing with permutation to see if the generated data are random.

```r
set.seed(12345)
#function to generate data
gen_data<-function(n,data,alpha){
  #initialize vector
  Y_gen<-c()
  #loop to create sample of size n
  for (i in 1:n){
    beta<-rnorm(1,183,10)
    x<-data[i]
    Y_gen[i]<-max(0,min(alpha*x+beta,366))
  }
```

```
  return(Y_gen)
}
```

Subquestion 5.b-Hypothesis testing with permutation on generated data

```
#generate data
Y_gen_data<-gen_data(366,lottery$Day_of_year,0.1)

#calculate permutation statistics for generated data
res1<-perm_func(Y_gen_data,lottery$Day_of_year,200)

cat("The pvalue for generated data is :",res1$p_value_perm)
```

```
## The pvalue for generated data is : 0.48
```

We can see that the pvalue that is obtained from our artificially generated data is bigger than an $\alpha = 0.05$,$p - value(> 0.05)$,so we fail to reject null hypothesis and our generated data with 95% confidence are random.

Subquestion 5.c-Hypothesis testing with permutation on generated data with diffrent a

Finally,we are going to use diffrent values for α to generate the data and calculate the pvalue for each sample with diffrent α in the threshold

$\alpha = [0.2, 0.3, ..., 10]$

```
set.seed(12345)
#initialize the step
step<-seq(0.2,10,by=0.1)
#initialize vector to store permutated p values
tafs<-rep(0,length(step))
#for loop
for (j in 1:length(step)){
  new_y_data<-gen_data(366,lottery$Day_of_year,step[j])
  p<-perm_func(new_y_data,lottery$Day_of_year,200)
  tafs[j]<-p$p_value_perm
}

#tafs

signif_p <- sum(tafs<0.05)
power <- 1-sum(tafs>0.05)/length(tafs)
cat("The number of significant permutated p values are :",signif_p,"\n",
    "The value of the power is :",power)
```

```
## The number of significant permutated p values are : 97
##  The value of the power is : 0.979798
```

From the results above, we can see that 97 of 99 the cases reject the null hypothesis, which indicates that the lottery is not random.

Question 2-Bootstrap,jacknife and ci

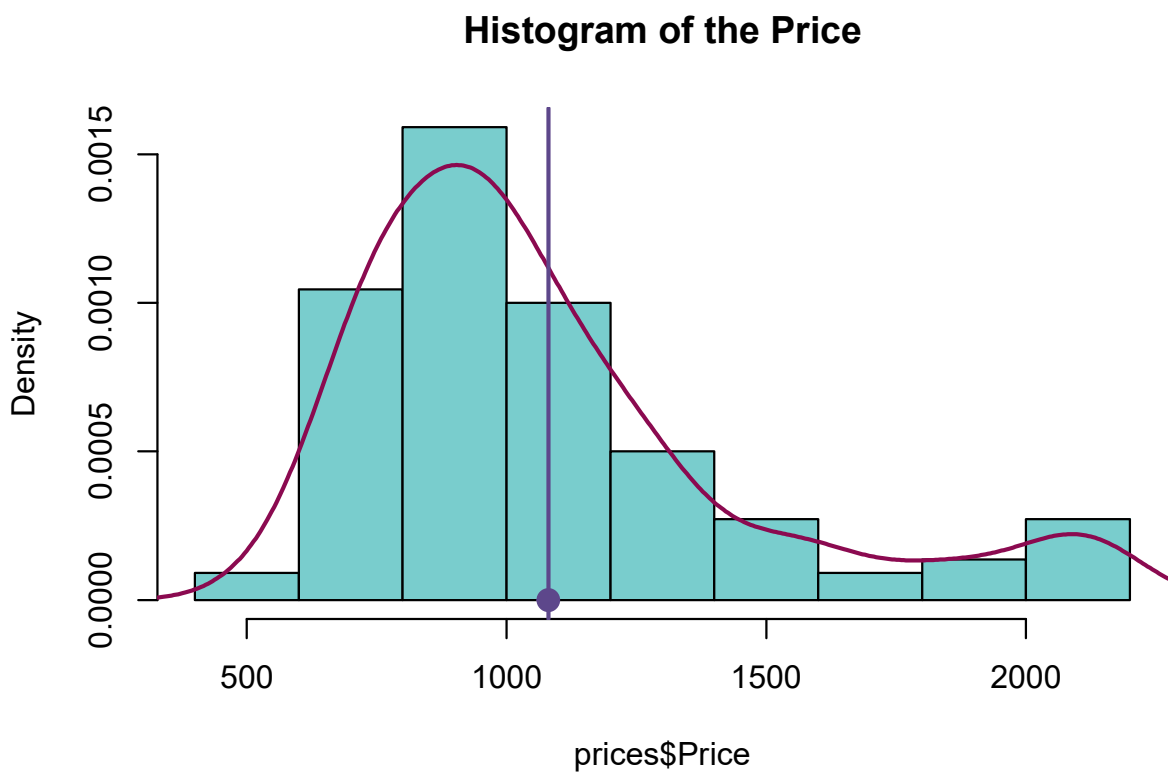
Subquestion 1-Histogram of Price

```
#import prices data
prices<-readxl::read_excel("prices1.xls")
prices<-as.data.frame(prices)

meanP<-mean(prices$Price)
cat("The mean of the Price is :",meanP)
```

```
## The mean of the Price is : 1080.473
```

Plot of the histogram of Price

```
hist(prices$Price,main="Histogram of the Price",
     col="darkslategray3",prob=T)
lines(density(prices$Price),col="deeppink4",lwd=2)
points(meanP,0,col="mediumpurple4",pch=19,cex=1.5)
abline(v=meanP,col="mediumpurple4",lwd=2)
```



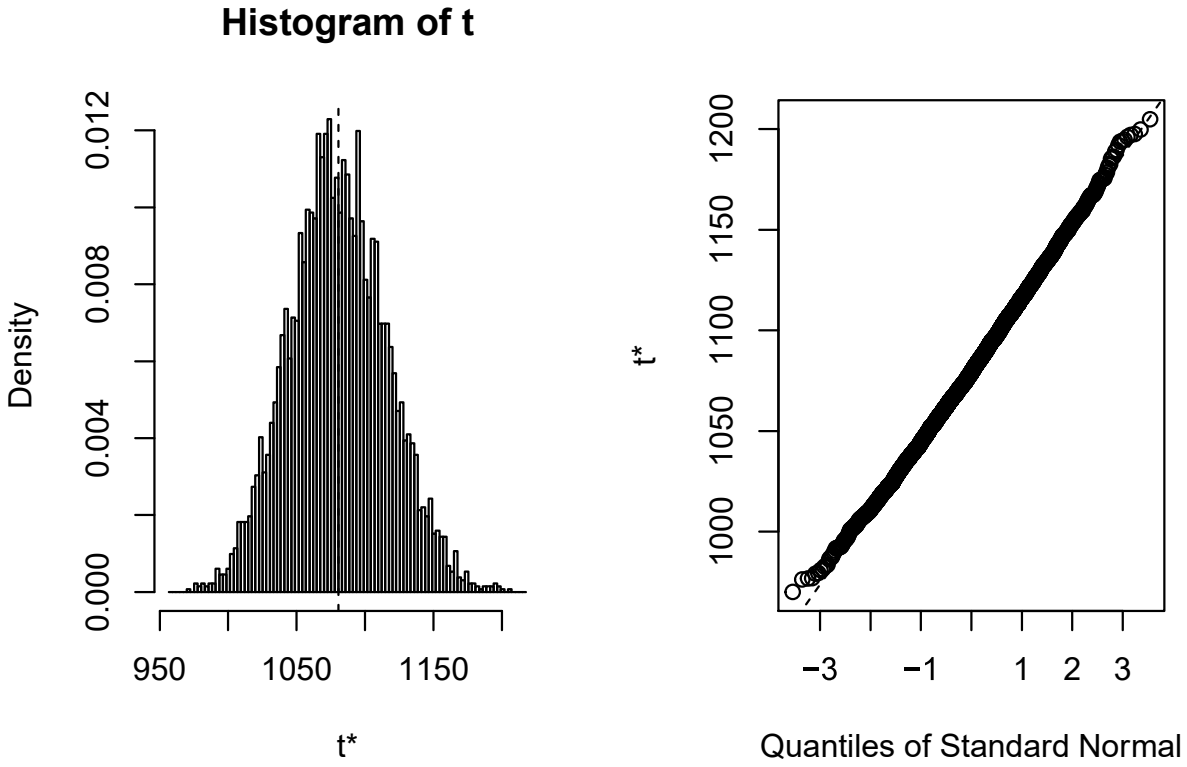
The above plot shows the distribution of the Price with the dot-vertical line to indicate the mean of price. The distribution seems to be gamma.

Subquestion 2-Bootstrap distribution estimation and CI

We are now going to estimate the distribution of the mean price of the house using bootstrap and Compute a 95% confidence interval for the mean price using bootstrap percentile, bootstrap BCa, and first order normal approximation.

The results from the bootstrap sampling are given below with the histogram of t and the quantiles showing a normal distribution

```
set.seed(12345)
library(boot)

#statistic function for boot function
meanfun <- function(dat, idx) mean(dat[idx], na.rm = TRUE)
#calculate bootstrap means
bot <- boot(prices$Price, statistic=meanfun, R=5000)
#plot boot object
plot(bot)
```




```
#
```

Next we calculate variance of the mean and the bias correction which is given by the formula :

$$T_1 := 2T(D) - \frac{1}{B}\sum_{i=1}^{B} T_i*$$

```
##Bias correction
bias_corr<-2*mean(prices$Price)-sum(bot$t)/5000
```

```
##variance of mean price
var_mean<-sum((bot$t-mean(bot$t))^2)/(5000-1)


cat("The bias correction is :",bias_corr)
```

```
## The bias correction is : 1080.654
```

```
cat("\n")
```

```
cat("The variance of the mean is :",var_mean)
```

```
## The variance of the mean is : 1261.819
```

Now we move to calculate the CI's with the BCa,percentile and first-order approximation

CI for mean Price with BCa

The output for the BCa CI is given below

```
#calculate bootstrap CI with BCa
bca_ci=boot.ci(bot, conf=0.95, type="bca",index=1)

print("The output of the Bca CI is :")
```

```
## [1] "The output of the Bca CI is :"
```

```
cat("\n")
```

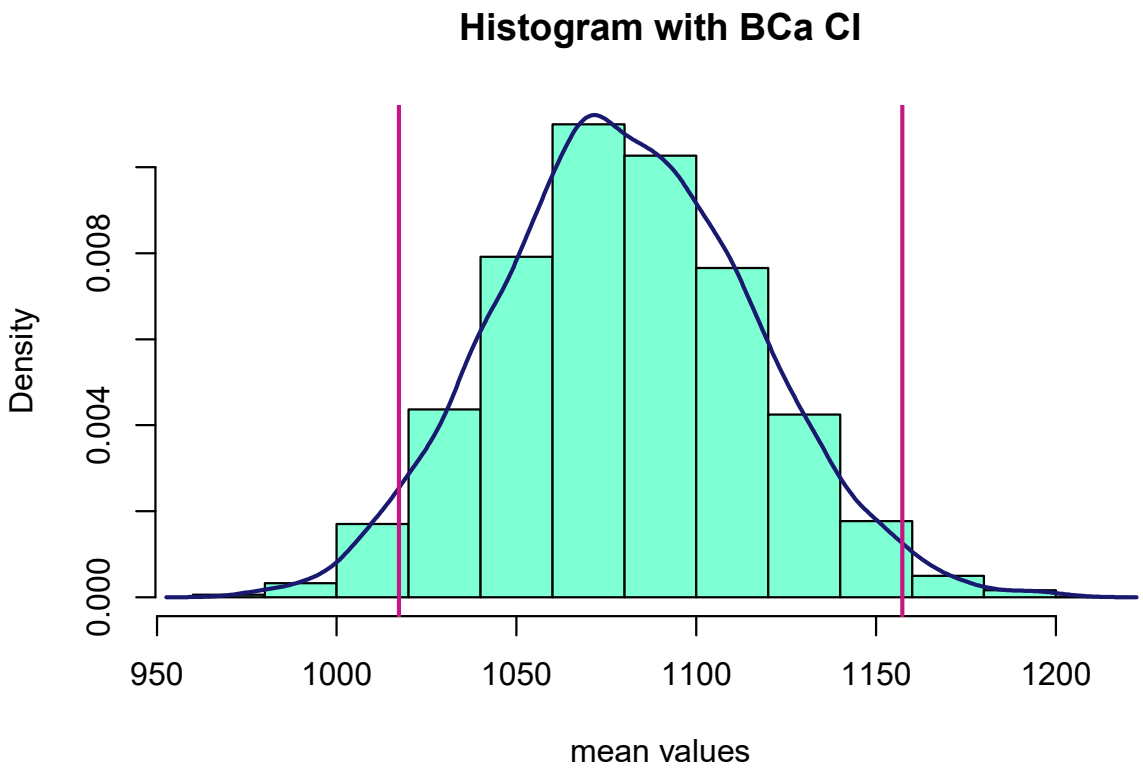
```
(bca_ci)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bot, conf = 0.95, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 95%   (1017, 1157 )
## Calculations and Intervals on Original Scale
```

Histogram for mean Price with BCa

The plot shows the histogram of the t obtained from bootstrap and the vertical lines are the confidence intervals obtained with BCa method.

```
#take the 4th and 5th objects from bca that correspond to CI values
CI_bca=bca_ci$bca[ , c(4, 5)]
#plot histogram with bca intervals
hist(bot$t[,1],main = 'Histogram with BCa CI',xlab = 'mean values', col = 'aquamarine', prob = T)
lines(density(bot$t[,1]), col = 'midnightblue',lwd=2)
abline(v = CI_bca, col = 'mediumvioletred',lwd=2)
```



CI for mean Price with percentile

The output for the percentile CI are given below

```
perc_ci=boot.ci(bot,conf=0.95,type="perc",index = 1)

print("The output of the percentile CI is :")
```

```
## [1] "The output of the percentile CI is :"
```

```
cat("\n")
```

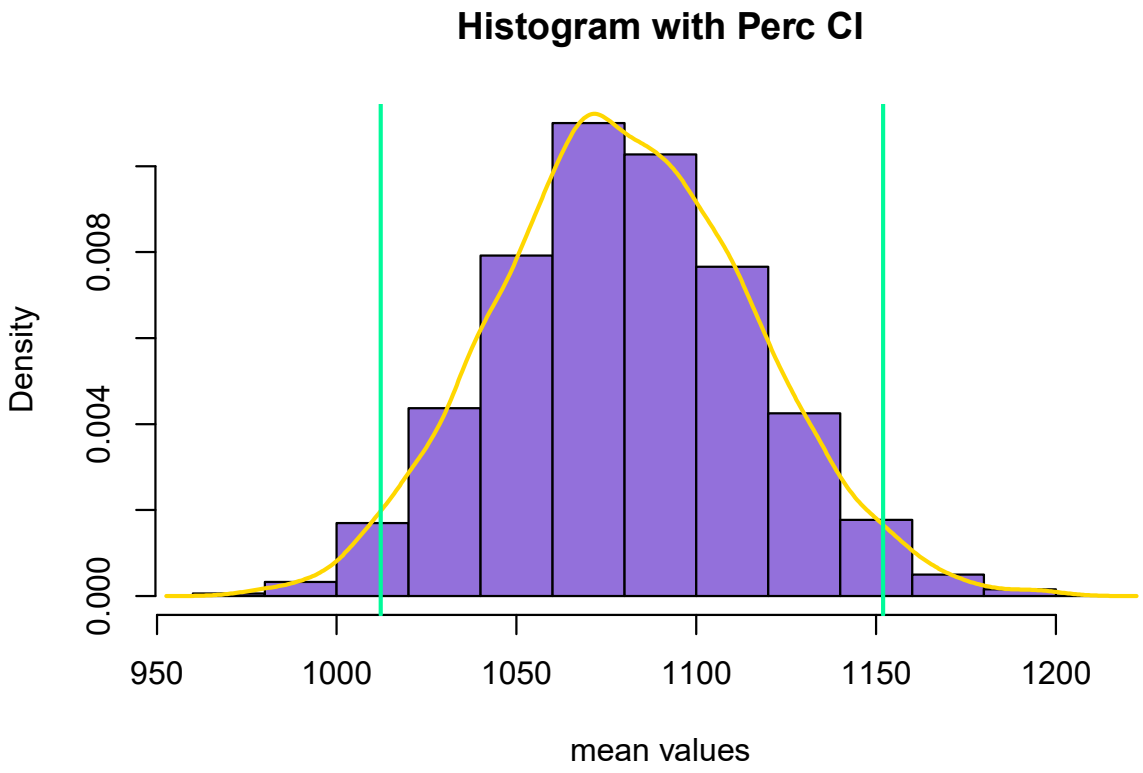
```
(perc_ci)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bot, conf = 0.95, type = "perc", index = 1)
##
## Intervals :
## Level     Percentile
## 95%   (1012, 1152 )
## Calculations and Intervals on Original Scale
```

Histogram for mean Price with percentile

The plot shows the histogram of the t obtained from bootstrap and the vertical lines are the confidence intervals obtained with percentile method.

```
CI_perc=perc_ci$percent[ , c(4, 5)]
#histogram with CI
hist(bot$t[,1],main = 'Histogram with Perc CI',xlab = 'mean values', col = 'mediumpurple', prob = T)
lines(density(bot$t[,1]), col = 'gold',lwd=2)
abline(v = CI_perc, col = 'mediumspringgreen',lwd=2)
```



CI for mean Price with first-order normalization

The output for the first-order normalization CI are given below

```
norm_ci=boot.ci(bot,conf=0.95,type = "norm",index=1)

print("The output of the first order normalization CI is :")
```

```
## [1] "The output of the first order normalization CI is :"
```

```
cat("\n")
```

```
(norm_ci)
```

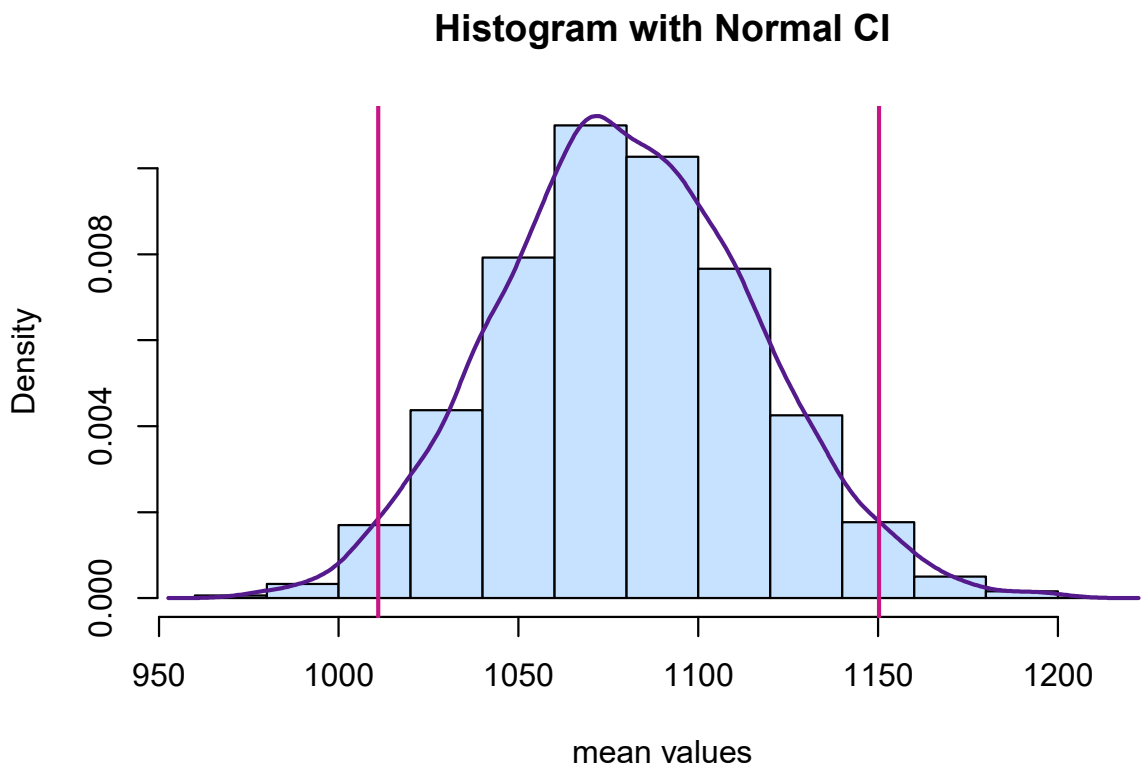
```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
```

```
## boot.ci(boot.out = bot, conf = 0.95, type = "norm", index = 1)
##
## Intervals :
## Level      Normal
## 95%   (1011, 1150 )
## Calculations and Intervals on Original Scale
```

Histogram for mean Price with first order normalization

The plot shows the histogram of the t obtained from bootstrap and the vertical lines are the confidence intervals obtained with first order normalization method.

```
CI_norm=norm_ci$normal # we dont need to subscript here normal!!

hist(bot$t[,1], main = 'Histogram with Normal CI',xlab = 'mean values', col = 'slategray1', prob = T)
lines(density(bot$t[,1]), col = 'purple4',lwd=2)
abline(v = CI_norm, col = 'mediumvioletred',lwd=2)
```



Subquestion 3-Jacknife estimation

```
#jacknife the reaper function
#function to calculate the jacknife sample



# jk_func = function (x, theta, ...)
```

```
# {
#   call = match.call()
#   n = length(x)
#   u = rep(0, n)
#   for (i in 1:n) {
#     u[i] = theta(x[-i], ...)
#   }
#   theta.hat = theta(x, ...)
#   pseudo.values = n*theta.hat - (n-1)*u
#   theta.jack = mean(pseudo.values)
#   jack.se = sqrt(sum((pseudo.values - theta.jack)^2)/(n*(n-1)))
#   # jack.bias = theta.jack - theta.hat
#   jack.bias = (n-1)*(theta.hat - mean(u))
#   #
#   return(list(theta.hat = theta.hat,
#               theta.jack = theta.jack,
#               jack.bias = jack.bias,
#               # jack.var = jack.se^2,
#               jack.se = jack.se,
#               leave.one.out.estimates = u,
#               pseudo.values = pseudo.values,
#               call = call))
# }
```

In this part we are going to implement jacknife method and calculate The variance of the mean which is given from the above formula

$$\hat{V}ar[T(.)] = \frac{1}{n(n-1)} \sum_{i=1}^{n} ((T_i*) - J(T))^2, \quad n = B$$

$$where \;\; T_i* = nT(D) - (n-1)T(D_i*), \quad J(T) = \frac{1}{n} \sum_{i=1}^{n} T_i*$$

The results are given below

```
n <- length(prices$Price)
#the statistic function
theta <- median(prices$Price)
#apply the statistic function to the Price but without the current point-i
jk <- sapply(1 : n,function(i) mean(prices$Price[-i]))
#mean of jacknife
mean_jk <- mean(jk)
#bias jacknife
bias_jk <- (n - 1) * (mean_jk - theta)
#variance jacknife
var_jk <- (n - 1) * mean((jk - mean_jk)^2)


jk_dt<-data.frame(c("jacknife mean"=mean_jk,"jacknife bias"=bias_jk,"variane jacknife"=var_jk))
colnames(jk_dt)<-"value"
library(knitr)
kable(jk_dt)
```

	value
jacknife mean	1080.473
jacknife bias	11496.527
variane jacknife	1320.911

The obtained variance using Jackknife method is 1320.911 while using bootstrapping, the obtained value was 1261.819.

Subquestion 4-Compare CI

Finally, we are going to compare the results for the diffrent CI methods used.The table below summarises the results.

```
intervals<-c("(1017.324 ,1157.234)" ,"(1012.239, 1151.954)" ,"(1011.032, 1150.276)"  )
lengths<-c( (CI_bca[2]-CI_bca[1]),(CI_perc[2]-CI_perc[1]),(CI_norm[3]-CI_norm[2]) )
centers<-c( sum(CI_bca)/2,sum(CI_perc)/2,sum(CI_norm[2:3])/2 )

ci_dataset<-data.frame(intervals,lengths,centers)

kable(ci_dataset)
```

intervals	lengths	centers
(1017.324 ,1157.234)	139.9098	1087.279
(1012.239, 1151.954)	139.7150	1082.097
(1011.032, 1150.276)	139.2441	1080.654

As we can see the lengths of the intervals are quite the same but the center obtained by the first-order normalization is closer to original mean data which is 1080.473 thus this method is preferable.

Computational Statistics (732A90) Lab 6

Omkar Bhutra (omkbh878), Andreas C Charitos (andch552)

8 March 2019

Question 1 -Genetic Algorithm

Subquestion 1-Define target function

We are going to perform one-dimensinal maximization with the help of genetic algorithm for the following function :

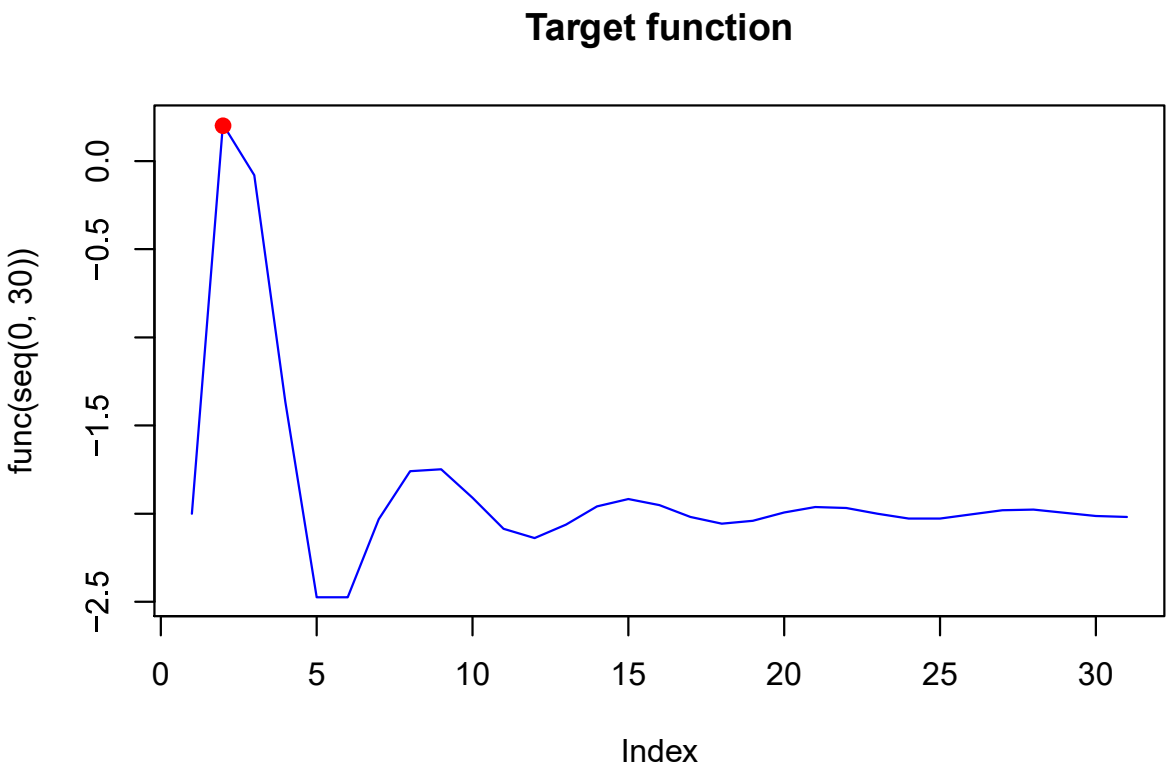
$$f(x) = \frac{x^2}{e^x} - 2e^{(-\frac{(9sin(x))}{(x^2+x+1)})}$$

Subquestion 2-Define crossover function

We are asked to perform the crossover function that for 2 scalars we get their "kid"

Subquestion 3-Define mutation function

The function mutate() takes a scalar x returns the result of the integer division $x^2 mod\ \ 30$ we will use this function to perform a mutation to the "kid" that will be produced in the crossover.



In the plot we can see that the function has some local maximum but with red circle is the global maximum we wish to approximate using genetic algorithm.

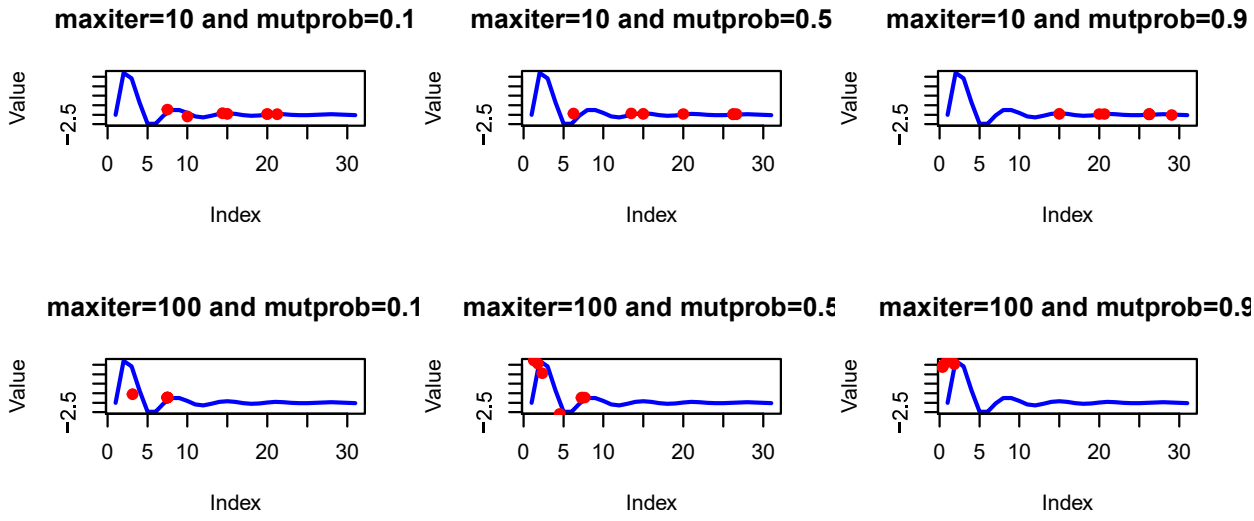
Subquestion 4-Define genetic function

we are going to implement the genetic function that has 2 arguments number of iterations and mutation probability

Subquestion 5-Run the function with diffrent settings

Last we are goning to try the algorithm for different combinations of maxiter= 10, 100 and mutprob= 0.1, 0.5, 0.9

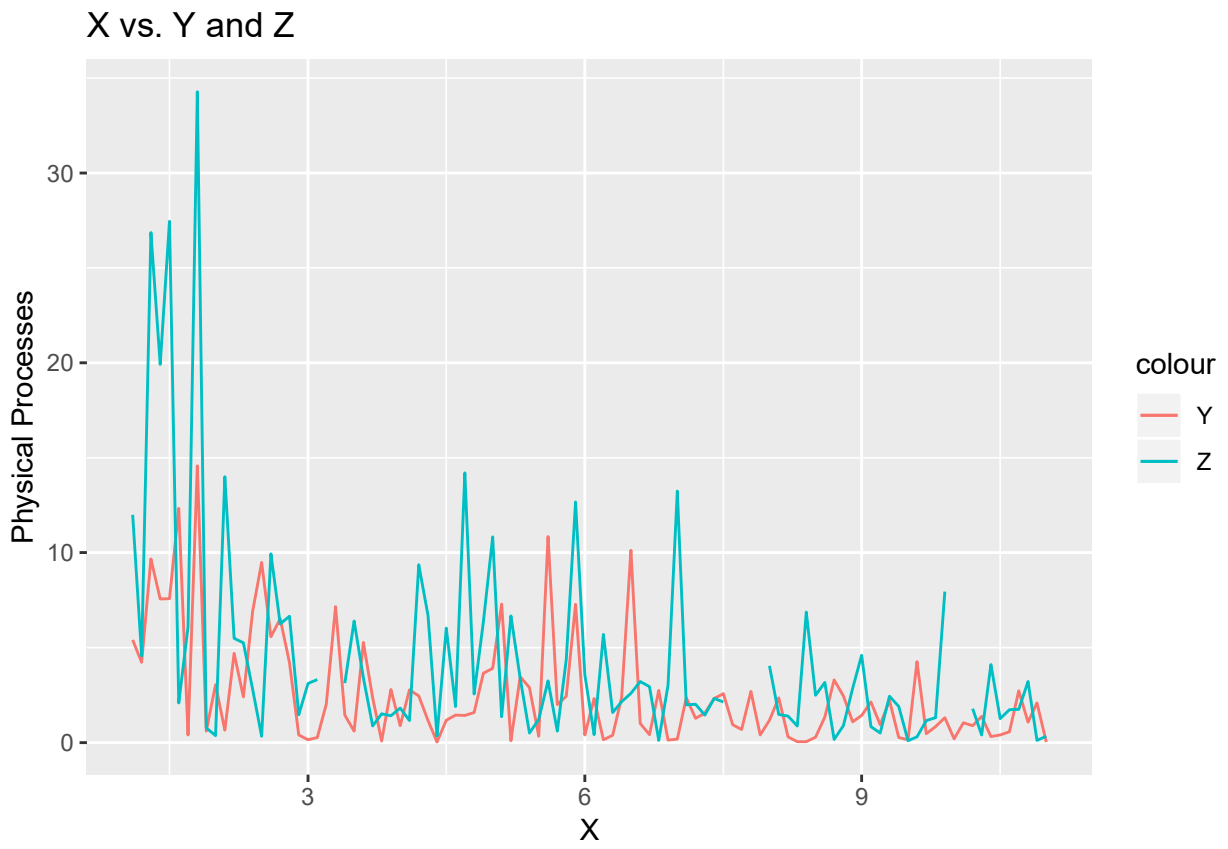
```
## NULL

## NULL

## NULL

## NULL

## NULL

## NULL
```



Finaly, the plot shows that with maxiter=100 have better performance than the one with maxiter=10 and the final population values are more closer to maximum. This is reasonable because the algorithm needs more iterations in order to make more crossovers and mutations.Furthermore,if the mutation probability is high the mutatation is very often for the "kids"-points and that makes them change points and explore new points which is the purpose of the mutation.

Question 2: EM algorithm

The data file physical.csv describes a behavior of two related physical processes Y = Y (X) and Z = Z(X).
1. Make a time series plot describing dependence of Z and Y versus X. Does it seem that two processes are related to each other? What can you say about the variation of the response values with respect to X?



Yes, it seems that the two responses are related to each other. It is observed that both responses Y and Z have decayed oscillations, this makes it seem that the responses are related to each other. The decay in response Z is lower than in Y.

2. Note that there are some missing values of Z in the data which implies problems in estimating models by maximum likelihood. Use the following model

$$Y_i \sim exp(X_i = \lambda); Z_i \sim exp(X_i = (2\lambda))$$

where λ is some unknown parameter. The goal is to derive an EM algorithm that estimates λ

$$X_i \sim exp(\lambda)$$

$$L(\lambda|Y,Z) = \prod_{i=1}^{n} f(Y).\prod_{i=1}^{n} f(Z)$$

$$= \prod_{i=1}^{n} \frac{X_i}{\lambda}.e^{\frac{-X_i.Y_i}{\lambda}}.\prod_{i=1}^{n} \frac{X_i}{2\lambda}.e^{\frac{-X_i.Y_i}{2\lambda}}$$

$$= \frac{1}{2\lambda}\prod_{i=1}^{n} X_i^2.exp(\frac{-X_i}{\lambda}.(Y_i + \frac{Z_i}{2}))$$

log-liklihood:

$$= \log(\frac{1}{2\lambda^{2n}}\prod_{i=1}^{n} X_i^2.exp(\frac{-X_i}{\lambda}.(Y_i + \frac{Z_i}{2})))$$

$$= 2\log\prod_{i=1}^{n} X_i - 2n\log(2\lambda) - \frac{1}{\lambda}\sum_{i=1}^{n} X_i(Y_i + \frac{Z_i}{2})$$

E-step:

$$E(X|\lambda) = \frac{1}{\lambda}$$

$$2\log\prod_{i=1}^{n} X_i - 2n\log(2n) - \frac{1}{\lambda}(\sum_{i=1}^{n} X_i Y_i + \sum_{i=1}^{r}\frac{X_i V_i}{2} + \sum_{i=r+1}^{n}\frac{X_i}{2}\frac{2\lambda_k}{X_i})$$

M-step: The maximum likelihood estimate of the parameters is obtained by taking the partial derivative with respect to λ. This is repeated till the solution converges.

$$\frac{\partial L(\lambda|Y,Z)}{\partial\lambda} = 0$$

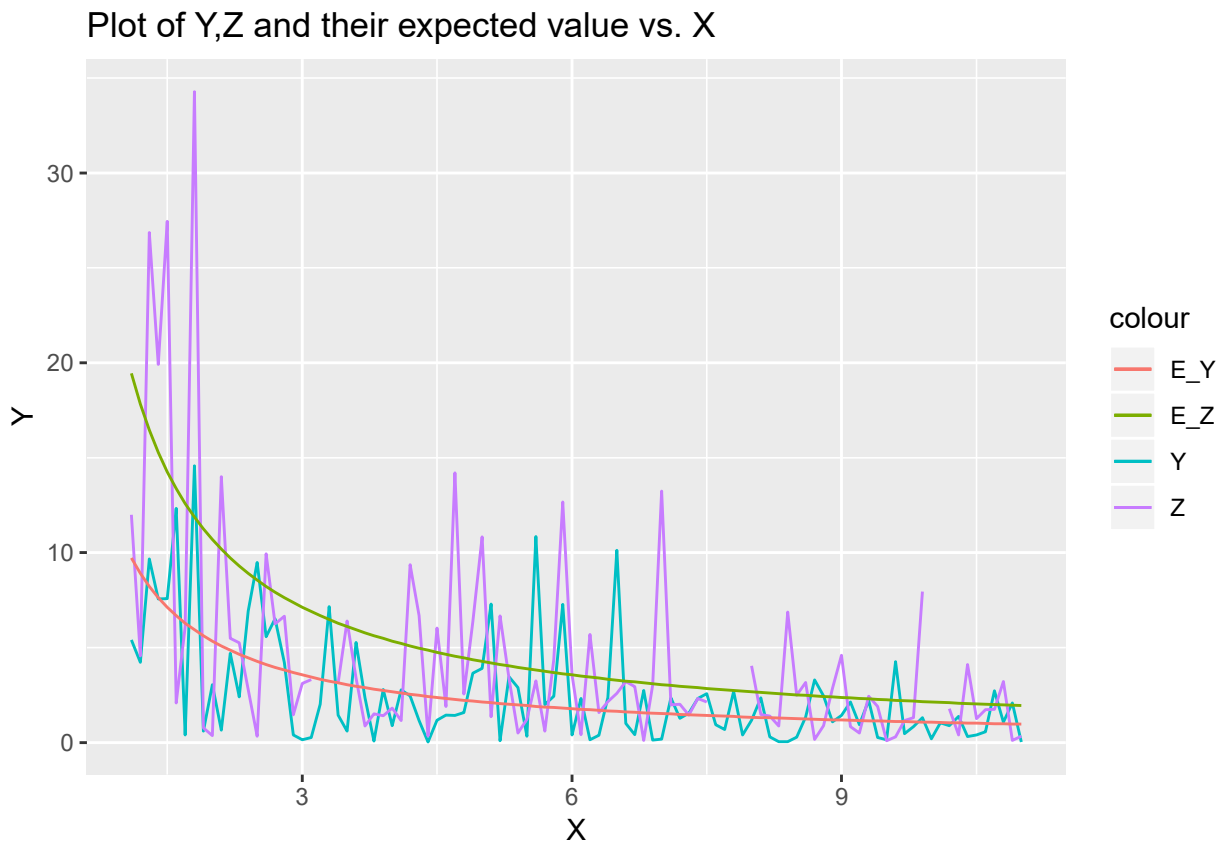
$$\lambda = \frac{1}{2n}(\sum_{i=1}^{n} X_i Y_i + \sum_{i=1}^{r}\frac{X_i V_i}{2} + (n-r)\lambda_k)$$

3. Implement this algorithm in R, use $\lambda_0 = 100$ and convergence criterion "stop if the change in λ is less than 0:001". What is the optimal λ and how many iterations were required to compute it?

```
## [1]   0 100   0
## [1] 10.69587 10.69566  5.00000
```

This result indicates that the lambda value is 10.69566 after 5 iterations that are needed for convergence.

4. Plot E [Y] and E[Z] versus X in the same plot as Y and Z versus X. Comment whether the computed λ seems to be reasonable.



From the graph , it is observed that both exponential's of Y and Z capture the flow of Y and Z against X. Hence, the lambda found from the above algorithm is optimal.

Apendix

```
knitr::opts_chunk$set(echo = TRUE)
library("boot")
library("ggplot2")
library("animation")

func<- function(x){
  result=x^2/exp(x)-2*exp(-(9*sin(x))/(x^2+x+1))
  return(result)
}

crossover<-function(x,y){
  if(is.atomic(x)==TRUE && is.atomic(y)==TRUE){
    kid=(x+y)/2
  }
  else("Inputs have to be scalar")
  return(kid)
}

mutate<-function(x){
  if(is.atomic(x)==TRUE){
    result=(x^2%%30)
  }
  return(result)
}


 targret_func<-function(x){

  result1<-x^2/exp(x)
  result2<-2*exp( -(9*sin(x)) / (x^2+x+1) )

  return(result1-result2)

}


plot(func(seq(0,30)),type = "l",col="blue",
     main="Target function")
points(2,0.2,col="red",pch=19)



genX<-function(maxiter,mutprob,animated=F){
  #plot funciton that plots the original
  #and the final population obtained
  plotf<-function(){
    plot(targret_func(seq(0,30)),type = "l",col="blue",lwd=2,
```

```
        main= paste0("maxiter=",maxiter," and mutprob=",mutprob),
        ylab="Value")
  points(initial_pop,targret_func(initial_pop),col="red",
         pch=19)
  return()

}


#initialize a population
initial_pop=seq(0,30,5)
#compute the value from target function
#for each population
Values<-sapply(initial_pop,targret_func)
#initialize vector to store
#max value for every iteration
max_Value<-rep(0,maxiter)
#if statement if we want gif animation
if (animated==F){
  for (i in 1:maxiter){
    #find 2 parents from the initial population
    parents<-initial_pop[c(sample(1:length(initial_pop),2,replace = F))]
    #get the victim that is going to be replaced
    victim<-order(Values)[1]
    #take "kid" from 2 parents with crossover
    new_kid<-crossover(parents[1],parents[2])
    #check the probability for mutation
    #if is higher than mutprob "kid"
    #will be mutated
    if (runif(1)<mutprob){
      new_kid<-mutate(new_kid)
    }
    #change the kid with the victim
    #in the initial population
    initial_pop[victim]<-new_kid
    #calculate the values of the new
    #population again
    Values<-sapply(initial_pop,targret_func)
    #get the max value
    max_Value[i]<-max(Values)



  }
  #plotf()
  return(list("plot"=plotf(),"final_pop"=max_Value))

}
#use this is you want animation gif
else{
  require(animation)
  saveGIF({
ani.options(interval=.3)
```

```
  col.range <- heat.colors(15)

  for (i in 1:maxiter){

    parents<-initial_pop[c(sample(1:length(initial_pop),2,replace = F))]
    victim<-order(Values)[1]
    new_kid<-crossover(parents[1],parents[2])

    if (runif(1)<mutprob){
      new_kid<-mutate(new_kid)
    }

    initial_pop[victim]<-new_kid
    Values<-sapply(initial_pop,targret_func)
    max_Value[i]<-max(Values)

    plot(targret_func(seq(0,30)),type = "l",col="blue",main=paste0("plot for",i,"th iteration"))
    points(initial_pop,targret_func(initial_pop),col="red")

    }#end of for loop

    })#end of animation
  }


}

set.seed(1234567)

par(mfrow=c(3,3))
#maxiter=10
genX(10,0.1,animated=F)$plotf
genX(10,0.5,animated=F)$plotf
genX(10,0.9,animated=F)$plotf
#maxiter=100
genX(100,0.1,animated=F)$plotf
genX(100,0.5,animated=F)$plotf
genX(100,0.9,animated=F)$plotf
physical <- read.csv("physical1.csv", sep=",")
ggplot(physical) + geom_line(aes(x=X, y = Y,color="Y")) +geom_line(aes(x=X, y = Z,color="Z"))+
ggtitle("X vs. Y and Z")+labs(x="X",y="Physical Processes")
EM<-function(physical,lambda0,eps,kmax){
    X<-physical$X
    Y<-physical$Y
    Z<-physical$Z
    Xobs <- X[!is.na(Z)]
    Zobs <- Z[!is.na(Z)]
    Zmiss <- Z[is.na(Z)]

    n <- length(X)
    r <- length(Xobs)

    k<<-0
```

```
    llvalprev<-0;
    llvalcurr<-lambda0
    print(c(llvalprev,llvalcurr,k))

    while ((abs(llvalprev-llvalcurr)>eps) && (k<(kmax+1))){
    llvalprev<-llvalcurr

    ## Compute log-likelihood
    llvalcurr<-(((sum(X*Y)+(sum((Xobs*Zobs)/2))+((n-r)*llvalprev))/(2*n)))
    k<<-k+1
  }
    print(c(llvalprev,llvalcurr,k))
}

EM(physical,100,0.001,50)
lambda <- 10.69566
new_data <- physical
new_data$E_Y <- lambda/physical$X
new_data$E_Z <- 2*lambda/physical$X
ggplot(data=new_data,aes(x=X, group=1)) +
geom_line(aes(y = Y, color = "Y")) +
geom_line(aes(y = Z, color = "Z")) +
geom_line(aes(y = E_Y, color = "E_Y")) +
geom_line(aes(y = E_Z, color = "E_Z")) +
ggtitle("Plot of Y,Z and their expected value vs. X")
```

$$Q(\theta, \theta^k) = E\left[\text{loglik}(\theta \mid y, z) \mid \theta^k, y\right]$$

$X \sim Exp(\lambda) \Rightarrow \lambda e^{-\lambda x}$

$\therefore y_i \sim Exp\left(\frac{x_i}{\lambda}\right) \rightarrow \frac{x_i}{\lambda} e^{-\frac{x_i}{\lambda} \cdot y_i}$

$z_i \sim Exp\left(\frac{x_i}{2\lambda}\right) \rightarrow \frac{x_i}{2\lambda} e^{-\frac{x_i}{2\lambda} \cdot z_i}$

$P(x, y, z \mid \lambda) = \frac{x_i}{\lambda} e^{-\frac{x_i}{\lambda} \cdot y_i} \cdot \frac{x_i}{2\lambda} e^{-\frac{x_i}{2\lambda} \cdot z_i}$

as dataset comprises of these λs

Likelihood:

$$L(\lambda) = \prod_{i=1}^{n} P(\lambda \mid x, y, z)$$

$$= \prod_{i=1}^{n} \frac{x_i^2}{2\lambda^2} e^{-\frac{x_i}{\lambda}\left(y_i + \frac{z_i}{2}\right)} \Rightarrow \frac{\prod_{i=1}^{n} x_i^2}{2\lambda^{2n}} e^{\frac{-1}{\lambda}\sum_{i=1}^{n} x_i\left(y_i + \frac{z_i}{2}\right)}$$

Log Likelihood:

$$\ln(L(\lambda)) = 2\ln\left(\prod_{i=1}^{n} x_i\right) - 2n\ln(2\lambda) - \frac{1}{\lambda}\sum_{i=1}^{n}\left[x_i\left(y_i + \frac{z_i}{2}\right)\right]$$

Converting z_i into, → due to missing values

$\sum_{i=1}^{m} v_i$ → observed $\times$ $\sum_{i=m+1}^{n} w_i$ → unobserved

$$\therefore \ln(L(\lambda)) = 2\ln\left(\prod_{i=1}^{n} x_i\right) - 2n\ln(2\lambda) - \frac{1}{\lambda}\left(\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{m} \frac{x_i v_i}{2} + \sum_{i=m+1}^{n} \frac{x_i w_i}{2}\right)$$

For $Q(\theta, \theta^k) = E\left[\text{loglik}(\theta \mid y, z) \mid \theta^k, y\right]$

$w \rightarrow$ missing

$\therefore E\left[\ln(\lambda \mid x, y, z) \mid \lambda^k, x, y, z\right]$

$$= 2\ln\left(\prod_{i=1}^{n} x_i\right) - 2n\ln(2\lambda) - \frac{1}{\lambda} E\left[\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \frac{x_i z_i}{2} \,\middle|\, \lambda^k, x, y\right]$$

as it is unobserved

$$= 2\ln\left(\prod_{i=1}^{n} x_i\right) - 2n\ln(2\lambda) - \frac{1}{\lambda}\left(\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{r} \frac{x_i v_i}{2} + \sum_{i=r+1}^{n} x_i \, E\left[\omega_i \mid \lambda^k, \ldots\right]\right.$$

$$= 2\ln\left(\prod_{i=1}^{n} x_i\right) - 2n\ln(2\lambda) - \frac{1}{\lambda}\left(\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{r} \frac{x_i v_i}{2} + \sum_{i=r+1}^{n} \frac{x_i}{x_i}\frac{2\lambda^k}{x_i}\right.$$

$\omega_i \sim \text{Exp}\left(\frac{x_i}{2\lambda^k}\right)$

$\therefore E[x_i] = \frac{1}{\lambda_i} \Rightarrow E[\omega_i \mid \ldots, \lambda^k] = \frac{1}{\frac{x_i}{2\lambda^k}} \Rightarrow \frac{2\lambda^k}{x_i}$

$$E[L(\lambda)] = 2\ln\left(\prod_{i=1}^{n} x_i\right) - 2n\ln(2\lambda) - \frac{1}{\lambda}\left(\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{r} \frac{x_i v_i}{2} + \sum_{i=r+1}^{n} \lambda^k\right)$$

To solve for λ,

$$\frac{\partial E[L(\lambda)]}{\partial \lambda} = -\frac{2n}{2\lambda}\cdot 2 + \frac{1}{\lambda^2}\left(\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{r} \frac{x_i v_i}{2} + (n-r)\lambda^k\right)$$

$$0 = -2n + \frac{1}{\lambda^2}\left(\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{r} \frac{x_i v_i}{2} + (n-r)\lambda^k\right)\lambda^2$$

$$0 = -2n\lambda + \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{r} \frac{x_i v_i}{2} + (n-r)\lambda^k$$

$$\lambda = \frac{1}{2n}\left(\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{r} \frac{x_i v_i}{2} + \underbrace{(n-r)}_{m}\lambda^k\right)$$

Introduction
Computer Arithmetics

732A90
Computational Statistics

Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

22 I 2019 (P42)
Department of Computer and Information Science
Linköping University

Computational Statistics — In brief

- Even simple data analysis (mean, variance) by hand is tedious
- Today: huge datasets, models capturing system complexity, interactions (between variables **and** observations)
- We will discuss:
  - Being careful with calculations—overflow
  - Generation of random variables including correlated ones
  - Numerically optimizing functions, esp. maximum likelihood
  - Computing confidence (credible) intervals for distributions when analytical ones are unobtainable

Lesson structure

- Lectures
- Computer Labs
- Seminars
- Examination: Reports, seminars, final exam
- Final exam: computer based
- Answer in English.
- Electronic reports as **.PDF**.
- Disclose **ALL** collaborations and sources.
- Provide source code (if used).
- E–mail contact: krzysztof.bartoszek@liu.se

Course materials, software

- Lecture slides
- 2016 lecture slides (732A38)
- Handouts, R code
- Various suggested www pages or articles
- **Googling**
- James E. Gentle "Computational Statistics", Springer, 2009
- Geof H. Givens, Jennifer A. Hoeting "Computational Statistics", Wiley, 2013

- R

Course contents

- Recap: R
- Recap: Basic Statistics
- Computer Arithmetics (JG pages 85–105)
- Optimization (JG pages 241–272, handouts)
- Random Number Generation (JG pages 305–312, 325–328, handouts)
- Monte Carlo Methods (JG pages 312–318, 328 417–429, handouts)
- Numerical Model Selection and Hypothesis Testing (JG pages 52–56, 424, 435–467, handouts)
- Expectation Maximization Algorithm and Stochastic Optimization (JG pages 275–284, 296–298, 480–483, handout)

Pages are recommended reading for each lecture, **NOT** exact lecture content. The lectures will build up on this material.

Examination

Computer labs (need to be passed)

Presentation or opposition and attendance at seminars (see 732A90_ComputationalStatisticsVT2019_CourseInformation.pdf).

Computer exam points
A: $[18, \infty)$, B: $[16, 18)$, C: $[14, 16)$, D: $[12, 14)$, E: $[10, 12)$, F: $[0, 10)$

Allowed aids for exam: printed books and own PDF document containing max 100 pages (see 732A90_ComputationalStatisticsVT2019_CourseInformation.pdf).

Computer Arithmetics

SHOULD YOU CARE?

For a detailed mathemstical treatment see Ch. 4.2 in
D. E. Knuth (1998). The Art of Computer Programming, Volume 2, Addison–Wesley.

Computer Arithmetics: Examples

Computations can be affected by magnitudes of numbers.

```
x<-0.5^10000;y<-0.4^10000;x/(x+y)+y/(x+y)
x<-0.5^1000;y<-0.4^1000;x/(x|y)|y/(x|y)
x<-0.1^1000;y<-0.2^1000;x/(x+y)+y/(x+y)
```

```
t<-rnorm(5,10^18,1);t[3]-t[4];t[1]-t[2]
```

```
x<-10^800;sd<-10^400;y<-x/sd;y
```

And your are doing estimation under a nice, fancy model ...

Data presentation

- Computers store information in binary form
  0 1 0 1 1 0 0 1
- 1Byte=8bits (typical counting unit)
- 1Word=32 or 64bits (depending on architecture)
- 1KB=1024bytes
- 1MB=1024KB
- and so on

QUESTION: Why binary form?

Character encoding

- ASCII (American Standard Code for Information Interchange)
  - 1 byte per character, 7 bits coding, 1 parity check or 0
  - $2^7 = 128$ characters can be encoded
  - "Usual" English letters, Arabic numerals, punctuation, i.e. "standard" keyboard
  - 1–31 control characters, 0: NULL **WHY?**
  - Design influenced by contemporary (1960) hardware
  - Extended ASCII: all 8 bits, 256 characters
- Unicode
  - 8, 16 or 32 bits encoding
  - "of more than 128,000 characters covering 135 modern and historic scripts, as well as multiple symbol sets" (Wikipedia)

- `read.csv()`, `read.table()` have `fileEncoding` argument

Fixed–point system (integers)

- We use the base–10 (decimal) system, e.g. $1234 = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$
- We could use base–m system for any m
- Computers: base–2 (binary) system
- Each integer represented as: $A = a_0 \cdot 2^0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + \ldots$

EXERCISE: 5, 16, 17, 31, 32, 33, 255, 256 in binary
QUESTION: What is the range of a byte, word, double word?

- Negative numbers
  - **Leading bit:** first bit 0 if positive, 1 if negative
  - **Twos–complement:** sign bit 1, remaining bits to **opposite** value and then +1
    e.g. 5 = 00000101, −5 = 11111011
    QUESTIONS $5 + (-5) = ?$, try 12 and −12
  - Range: $[-2^{k-1}, 2^{k-1} - 1]$ on k bits **WHY?**

Arithmetic operations

R operations on binary

- Addition, multiplication: base–2 instead of base–10
- Subtraction: $A - B = A + (-B)$ (*twos–complement*)
- Division: tedious, rounded towards 0 `as.integer(17/3)`

- **Overflow:** adding two large numbers the sign bit can be treated as a high order bit and on some architectures results in a negative number

Floating–point system (rational. "real")

- How can we represent fractions (rational numbers)?
- Sign
- Exponent (**signed**, read standards if interested)
- Mantissa or Significand
- on 64bits:

sign (1bit)	Exponent (11bits)	Mantissa (52 bits)

$$\pm 0.d_1 d_2 \ldots d_p \cdot b^e \qquad b = 2, \quad p = 52$$

- **Range:** $\approx [-10^{300}, 10^{300}] \approx [-b^{e_{max}}, b^{e_{max}}]$

Floating–point system

- Rationals rounded towards the nearest computer float

```
options(digits 22) #max possible
0.1
[1] 0.10000000000000000055511
```

- **EXAMPLE:** Assume base $b = 10$ and mantissa has 5 digits p 5:
  $1.2345 = +0.12345 \cdot 10^1$
  $4.0000567 = +0.40000 \cdot 10^1$
- Problem remains whatever base (b) is chosen
- **EXERCISE:** Try to convert some numbers

Floating–point system

- Distribution of computer floats

- Dense from -1 to 1
- Density decreases
- same number of points for each exponent:

$$\ldots, \ \cdot 10^{-3}, \ \cdot 10^{-2}, \ \cdot 10^{-1}, \ \cdot 10^{1}, \ \cdot 10^{2}, \ \cdot 10^{3}, \ \ldots$$

- What about integers?
  $5 = +0.50000 \cdot 10^1$

```
options(digits=22)
9007199254740992
9007199254740993
9007199254740994
```

Floating–point system, special "numbers"

- We do not discuss how the exponent is actually coded.
- Usually the maximum allowed number in the exponent is one unit less than possible.
- ±Inf: exponent is $\exp_{\max} + 1$, mantissa is 0
- NaN: exponent is $\exp_{\max} + 1$, mantissa is $\neq 0$
- 0 **WHY?**

Overflow: number larger than can be represented
Underflow: loss of significant digits

```
10^200*10^200 =  Inf
10^400/10^400 =  NaN
10^-200/10^200 =  0
10^-200*10^200 =
0*10^400 =
x<-10^300;while(1){x<-x+1}
```

Arithmetic operations

- Floats are rounded so usual mathematical laws do not hold — floating point arithmetic
- **Examples**

```
1/3+1/3 =  0.6666667
options(digits 22)
1/3+1/3 =  0.66666666666666296592
10^(-200)/(10^(-200)+10^(-200)) =  0.5
10^(-200)/(10^(-200)+20^(-200)) = 1
```

- Software is designed to make operations as correct as possible
- **Do we need to work with such extreme numbers?**

Arithmetic operations

- $X + Y$, $X \cdot Y$ can display overflow, underflow
- $A \neq B$ but $X + A = B + X$
- $A + X = X$ but $A + Y \neq Y$
- $A \mid X \quad X$ but $X - X \ / \ A$
- **COMPARING FLOATS IS TRICKY!**

```
options(digits=22)
x<-sqrt(2)
x*x
[1] 2.0000000000000004440 89
(x*x)==2
[1] FALSE
isTRUE(all.equal(x*x,2))
[1] TRUE
```

Summation

Underflow problems can occur with any summation (`x<-x+1`)

```
options(digits=22)
x<-1:1000000; sum(1/x); sum(1/rev(x))
[1] 14.392726722865722 52176
[1] 14.392726722865724 29812
```

- **WHICH ONE IS CORRECT?**
- **WHICH ONE IS MORE ACCURATE?**

Potential solutions

Solution A:

1. Sort the numbers ascending **CAN BE EXPENSIVE**
2. Sum in this order

Solution B:

1. Sum numbers pairwise, from n obtain $n/2$ numbers
   HOW TO CHOOSE PAIRS?
2. Continue until 1 number left

More on summing

Example

- Computing exponent using Taylor series

$$e^x = 1 + x + x^2/2 + x^3/6 + \ldots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

The exponential

```
options(digits=22)
fTaylor<-function(x,N){1+sum(sapply(1:N,
    function(i,x){x^i/prod(1:i)},x x,simplify
    TRUE))}
exp(20) #fine
[1] 485165195.40979027748 11
fTaylor(20,100)
[1] 485165195.40979027748 11
fTaylor(20,100)-exp(20)
[1] 0
exp(-20) #problem
[1] 2.0611536224385578699 42e-09
fTaylor(-20,100)
[1] -3.8538772173524193931 37e-10
fTaylor(-20,200)
[1] -3.8538772173524193931 37e-10
```

More on summing

Example

- Computing exponent using Taylor series

$$e^x \quad 1 \mid x \mid x^2/2 \mid x^3/6 \mid \ldots \quad \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

- **WHY?**
- Varying sign of terms
- **CANCELLATION** adding two numbers of almost equal magnitude but of opposite sign
- Effects of cancellations accumulate
- **SOLUTION:** Different algorithm ...

Can you explain why?

```
## Example due to Thomas Ericsson in his
    Numerical Analysis course at Chalmers
f1<-function(x){(x-1)^6}
f2<-function(x){1-6*x+15*x^2-20*x^3+15*x^4-6*x
    ^5+x^6}

x<-seq(from=0.995,to=1.005,by=0.0001)
y1<-f1(x);y2<-f2(x)

plot(x,y1,pch 19,cex 0.5,ylim c(-5*10^(-15),20
    *10^(-15)),main="Two ways to calculate (x
    -1)^6",xlab="x",ylab="y")
points(x,y2,pch=18,cex=0.8)
```

Matrix Calculations

- Many problems (in Statistics, Numerical methods, e.t.c) can be reduced to solving

$$\mathbf{A}\vec{x} = \vec{b}$$

$\mathbf{A}$ (design) matrix
$\vec{x}$ vector of unknowns
$\vec{b}$ vector of scalars (data)

- Algorithm should be **numerically stable**

(small changes in $\mathbf{A}$ or $\vec{b}$ imply in small changes in $\vec{x}$)

Example: Linear regression models

Minimize

$$RSS(\beta_0, \beta_1, \ldots, \beta_m) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_m x_{im})^2$$

The system of equations

$$\frac{\partial RSS}{\beta_0} = \frac{\partial RSS}{\beta_1} = \ldots \frac{\partial RSS}{\beta_m} = 0$$

can be written as

$$\mathbf{X}^T\mathbf{X}\vec{\beta} = \mathbf{X}\vec{y}$$

where

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \ldots & x_{1m} \\ 1 & x_{21} & \ldots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \ldots & x_{nm} \end{bmatrix}$$

Solving a linear system

- $\mathbf{A}\vec{x} = \vec{b}$ needs a stable numerical solution
- **Computer arithmetics**

- - Original system: $\mathbf{A}\vec{x} = \vec{b}$
  - Perturbed system: $\mathbf{A}\vec{x'} \quad \vec{b'}, \vec{x'} \quad x' \mid \delta x, \vec{b'} \quad \vec{b} \mid \delta\vec{b}$
- Stable: small perturbation in $\vec{b}$, small perturbations in $\vec{x}$
- $\|\vec{b}\| = \|\mathbf{A}\vec{x}\| \leq \|\mathbf{A}\|\|\vec{x}\|$ implies $\|\vec{x}\|^{-1} \leq \|\mathbf{A}\|\|\vec{b}\|^{-1}$
  $\|\delta\vec{x}\| = \|\mathbf{A}^{-1}(\delta\vec{b})\| \leq \|\mathbf{A}^{-1}\|\|\delta\vec{b}\|$
  together

$$\frac{\|\delta\vec{x}\|}{\|\vec{x}\|} \leq \|\mathbf{A}^{-1}\|\|\mathbf{A}\|\frac{\|\delta\vec{b}\|}{\|\vec{b}\|}$$

Solving a linear system

Condition number of a matrix

$$\kappa(\mathbf{A}) = \|\mathbf{A}^{-1}\|\|\mathbf{A}\|$$

- Large $\kappa(\mathbf{A})$ is a bad sign, but does not imply ill–conditioning
- L_2 norm: $\kappa(\mathbf{A})$ is the ratio of the maximum and minimum eigenvalues of $\mathbf{A}$
- Under L_2 norm

$$\kappa(\mathbf{A}^T\mathbf{A}) \geq \kappa(\mathbf{A})^2 \geq \kappa(\mathbf{A})$$

(regression setting)

Solving a linear system

Dealing with ill–conditioning

- Rescale the variables (columns)
- Use a different algorithm for solving e.g. QR, Cholesky, SVD

Cholesky: $\mathbf{A}$ symmetric–positive–definite
$\mathbf{A}\vec{x} = \vec{b}$ is equivalent to $\mathbf{L}\mathbf{L}^T\vec{x} = \vec{b}$ **WHY?**

1. Solve $\mathbf{L}\vec{y} = \vec{b}$
2. Solve $\mathbf{L}^T\vec{x} = \vec{y}$

Summary

- Computations can behave "differently" at different numerical ranges.
- Floating point system.
- Computer arithmetics is not the same as "usual" arithmetic.
- Summing series, solving linear systems (inversion?)

Optimization

732A90
Computational Statistics

Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

24 I 2019 (P42)
Department of Computer and Information Science
Linköping University

Plan for today

- Introduction
- Mathematical definition of problem
- 1D optimization
- kD optimization
- R code examples

Optimization

Nearly everything is optimization !

- Chemistry
- Physics
- Economics, **Industry**
- Engineering

BUT EVEN

- Your mobile price plan
- Course scheduling
- Your lunch choice

STATISTICS

- Fit parameters to data
- Propose optimal decision

Optimization: Example

ANY BIOLOGICAL ORGANISM

YOU

Optimization: Example

Industry

How to produce a cylindrical (**WHY?**) $0.5L$ beer can so it requires minimum material?

Given a certain product minimize e.g. material usage, production effort while still meeting consumer requirements.

Optimization: Example

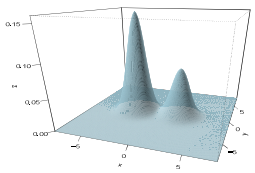
Economics/Logistics

- Travelling Salesman Problem
- Windmills
- Flight schedule (especially "cheap" airlines)

Optimization: Example

Statistics

Maximize likelihood, model fitting



Maximal likelihood

An i.i.d. sample $(X_1, \ldots, X_n)$ is drawn from a probability distribution $P(X\ \Theta)$, where Θ is an unknown parameter set.

The joint probability of all the observations is

$$P(X_1, \ldots, X_n|\Theta) = \prod_{i=1}^{n} P(X_i|\Theta).$$

Find Θ that maximizes $P(X_1, \ldots, X_n|\Theta)$.

Mathematical formulation

The goal is to minimize (maximize)

Objective function: $f(\theta)$
(reproduction, chances of survival, quality of life, cost, profit, likelihood, fit to data)

depending on

Parameters or **Unknowns** θ
(reproduction strategy, resource utilization, consumer choices, height & diameter, production, raw material choice, service times, route, iflight routes/times ,parameters)

Mathematical formulation

$$\min_{\theta \in \Theta} f(\theta) \quad \text{subject to} \quad \begin{array}{ll} c_i(\theta) = 0, & i \in E \\ c_i(\theta) \geq 0, & i \in I \end{array}$$

QUESTION: What should we do if we are interested in maximization instead of minimization?

QUESTION: What should we do if the constraints are $c_i(x) \leq 0,\ i \in I$?

Constraints examples

- Available environment
- Volume: 0.5l of can
- Production: Factories (F_1, F_2), retail outlets (R_1, R_2, R_3), cost of shipping $i \to j$: c_{ij}, production a_i per week, requirement b_j per week **to optimize:** x_{ij} amount shipped $i \to j$ per week

$$\begin{array}{ll} \min_{x_{ij} \in \mathbb{R}^3} \sum_{ij} c_{ij} x_{ij} & \text{minimize shipping costs} \\ \sum_{j=1}^{3} x_{ij} \leq a_i, i = 1, 2 & \text{production capacity} \\ \sum_{i=1}^{3} x_{ij} \geq b_j, j = 1, 2, 3 & \text{demand} \\ \forall_{i,j} x_{ij} \geq 0 & \end{array}$$

Question: What would happen if we drop demand constraint?

- ML: often no constraints

Exercise

- Split into pairs/triplets/quadruples
- Think of some human anatomy part/organ:
  - What is its function?
  - What could it have been optimized for over the course of time?
  - Is it still under selection?
  - What constraints was and is it under?
- Think of a situation where optimization is needed in your own student/professional/personal/financial situation.
- State the problem in terms of
  - Objective function
  - Parameters
  - Constraints
  - Does it have a trivial solution?
- **10** minutes

Optimization approaches

- Constrained optimization
  - Lagrange multipliers, linear programming
  - E.g. LASSO
  - **Not this lecture!**
- Unconstrained optimization
  - Steepest descent
  - Newton method
  - Quasi–Newton–Methods
  - Conjugate gradients

Why are there different methods?

1D Optimization

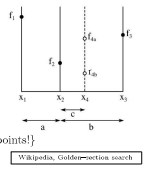
- Function of a single parameter, find minimum
- *What algorithm would you suggest?*
- **Golden–section search**
  local minimum on $[A, B]$ interval (constraint)
- Works by narrowing down the search interval with a constant reduction factor

$$1 \quad \alpha \quad \frac{\sqrt{5}-1}{2} \approx 0.62$$

Question: Does α remind you of something?

Golden section (minimization)

1: $x_1 \quad \Lambda,\ x_3 \quad B$,
2: **while** $x_1 - x_3 > \epsilon$ **do**
3: $\quad a = \alpha(x_3 - x_1)$
4: $\quad x_2 = x_1 + a,\ x_4 = x_3 - a$
5: $\quad$ **if** $f(x_4) > f(x_2)$ **then**
6: $\quad\quad x_1 \quad x_1,\ x_3 \quad x_4$
7: $\quad$ **else**
8: $\quad\quad x_1 = x_2,\ x_3 = x_3$
9: $\quad$ **end if**{We know the value at **3** points!}
10: **end while**



Wikipedia, Golden–section search

f has to be **UNIMODAL**

1D Optimization: Example

732A90_ComputationalStatisticsVT2019_Lecture02codeSlide16.R

Multi–dimensional optimization

Find

$$\min_{\vec{x}\in\mathbb{R}^n} f(\vec{x})$$

Using (known, or numerically evaluated)

Gradient $\nabla f(\vec{x}) = \left(\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)^T$

Hessian $\nabla^2 f(\vec{x}) = \left[\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j}\right]_{i,j=1}^{n}$

General strategy

1. Provide a **(good)** starting point $\vec{x}_0$, $\vec{x} = \vec{x}_0$
2. Choose a direction $\vec{p}$ ($\|p\|$ 1) and step size a
3. Move to $\vec{x} := \vec{x} + \alpha\vec{p}$
4. Repeat step 2 until convergence

How to choose the direction?

Taylor's theorem

$$f(\vec{x} + a\vec{p}) = f(\vec{x}) + \boxed{\alpha\vec{p}^T \cdot \nabla f(\vec{x})} + o(\alpha^2)$$

$\vec{p}$ s.t. $\vec{p}^T \cdot \nabla f(\vec{x}) < 0$ is a *descent* direction.

Steepest descent is

$$\vec{p} = -(\nabla f(\vec{x}))/\| \nabla f(\vec{x})\|$$

How to choose the step size?

- **Expensive way**: find the global minimum in direction $\vec{p}$
- **Trade–off way**: find a decrease which is *sufficient*

BACKTRACKING

Choose (large) $\alpha_0 > 0$, $\rho \in (0,1)$, $c \in (0,1)$,
$\alpha \quad \alpha_0$
repeat
  $\alpha = \rho\alpha$
until $f(\vec{x} + \alpha\vec{p}) \le f(\vec{x}) + c\alpha\vec{p}^T \nabla f(\vec{x})$

Newton's method

- Newton–Raphson method
- Hessian ignored in steepest descent
- If f is quadratic

$$f(p) \quad \frac{1}{2}p^T \mathbf{A} p \; b^T p \; c,$$

then minimum

$$p^* \quad \mathbf{A}^{-1}\vec{b}.$$

- Taylor expansion of f

$$f(\vec{x} + \alpha\vec{p}) = f(\vec{x}) - \alpha\vec{p}^T \cdot \nabla f(\vec{x}) + \frac{\alpha^2}{2}\vec{p}^T \nabla^2 f(\vec{x})\vec{p} + o(\alpha^3)$$

- $x := x + \alpha\vec{p}$ where $\boxed{\vec{p} = -\left(\nabla^2 f(\vec{x})\right)^{-1} \nabla f(\vec{x})}$

Newton's method

- $\left(\nabla^2 f(\vec{x})\right)^{-1}$ is expensive to compute, there are quicker approaches, e.g. Cholesky decomposition
- Hessian should be **positive definite** for $\vec{p}$ to be a descent direction (if not see book)
- Memory expensive — need to store $O(n^2)$ elements

BUT

- Method converges quickly esp. near optimum

Quasi–Newton methods

- k iteration number
- Compute an approximation to the Hessian, $\mathbf{B}$, that will allow for efficient choice of $\vec{p}$.
- **SECANT CONDITION:** (quasi–Newton condition)

$$\mathbf{B}_{k+1}(\vec{x}_{k-1} - \vec{x}_k) = \nabla f(\vec{x}_{k+1}) - \nabla f(\vec{x}_k)$$

BFGS Algorithm

Choose $\mathbf{B}_0 > 0$, $\vec{x}_0$, $k = 0$
repeat
  $\vec{p}_k$ is solution of $\mathbf{B}_k\vec{p}_k = \nabla f(\vec{x}_k)$
  find suitable α_k
  $x_{k+1} \quad x_k \; \alpha_k p_k$
  calculate $\mathbf{B}_{k-1}$ {next slide}
  $k = k + 1$
until convergence of $\vec{x}_k$ at minimum

How to compute $\mathbf{B}_{k+1}$?

- We want $\mathbf{B}_{k+1}$ and $\mathbf{B}_k$ to be close to each other
- $$\min_{\mathbf{B}} \|\mathbf{B} \quad \mathbf{B}_k\| \quad s.t.\ \mathbf{B} \quad \mathbf{B}^T, \text{ secant condition}$$
- $\vec{y}_k = \nabla f(\vec{x}_{k+1}) - \nabla f(\vec{x}_k)$, $\vec{s}_k = \vec{x}_{k+1} - \vec{x}_k$
- $$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \vec{y}_k \vec{y}_k^T \mathbf{B}_k}{\vec{y}_k^T \mathbf{B}_k \vec{y}_k} + \frac{\vec{s}_k \vec{s}_k^T}{\vec{y}_k \vec{s}_k^T}$$
- Closed form Sherman–Morrison formula for $\mathbf{B}_{k+1}^{-1}$
- We have to store $\mathbf{B}_k^{-1}$

BFGS

- BGFS: Broyden–Fletcher–Goldfarb–Shanno
- More iterations than Newton's method (uses approximation)
- Each iteration quicker, no numeric inversion
- Good for large scale problems
- Choice of $\mathbf{B}_0$?

Conjugate Gradient method—quadratic case

Minimize

$$f(\vec{x}) = \frac{1}{2}\vec{x}^T \mathbf{A}\vec{x} - \vec{b}^T\vec{x}$$

for $\mathbf{A}$ symmetric positive definite.
Gradient:

$$\vee f(x) \quad \mathbf{A}x \quad \vec{b} \quad r(x)$$

Two vectors $\vec{p}$ and $\vec{q}$ are **conjugate** with respect to $\mathbf{A}$ if

$$\vec{p}^T \mathbf{A}\vec{q} = 0.$$

IDEA: p and q are orthogonal w.r.t. to an inner product associated with $\mathbf{A}$. Use this to find a basis that will allow for easy finding of $\vec{x}$.

Conjugate Gradient method

- $\vec{p}_0 = \vec{r}_0$
- $\vec{p}_{k+1} = -\vec{r}_k + \beta_{k+1}\vec{p}_k$
- Conjugate condition has to be satisfied so

$$\beta_{k+1} = \frac{\vec{r}_k^T \mathbf{A}\vec{p}_{k-1}}{\vec{p}_k^T \mathbf{A}\vec{p}_k}$$

Exercise: check this

- Convergence in $\dim(\mathbf{A})$ steps (or unless cutoff for $\vec{r}_k$)

Nonlinear CG method

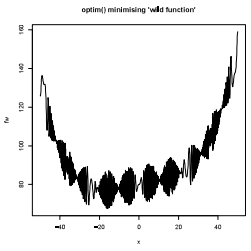
- If $f(\cdot)$ general, use $\nabla f(\cdot)$ instead of $r(\cdot)$

Choose $\vec{x}_0$, $\vec{p}_0 = -\nabla f(\vec{x}_0)$, $k = 0$
while $\nabla f(x_k) \neq \vec{0}$ **do**
  find suitable α_k
  $\vec{x}_{k+1} = \vec{x}_k + \alpha_k\vec{p}_k$ {and now update step}
  $\beta_{k+1} = \left(\nabla^T f(\vec{x}_{k+1}) \nabla f(\vec{x}_{k+1})\right) / \left(\nabla^T f(\vec{x}_k) \nabla f(\vec{x}_k)\right)$ {Fletcher–Reeves update, other possible}
  $\vec{p}_{k+1} = -\nabla f(\vec{x}_{k+1}) + \beta_{k+1}\vec{p}_k$
  $k = k + 1$
end while

Nonlinear CG method

- Local minimum convergence
- But this is true of all methods that cannot "jump out" of descent path
- Faster than steepest descent
- Slower than Newton and Quasi–Newton but significantly less memory

kD Optimization: Example



kD Optimization: Example

`732A90_ComputationalStatisticsVT2019_Lecture02codeSlide31.R`

Summary

- Optimization is everywhere
- Numerical methods for finding minimum
- 1D: Golden section (unimodal), `optimize()`
- kD: choose step size and direction (gradient), `optim()`

Random Number Generation

732A90
Computational Statistics

Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

30 I 2019 (P42)
Department of Computer and Information Science
Linköping University

Pseudorandom numbers

- **A computer is a deterministic machine**
- *Congruential generators*
- Functions of time
- Be careful with respect to application

First step: Generating Unif[0, 1]

Linear congruential generator

Define a sequence of integers according to

$$x_{k-1} = (a \cdot x_k + c) \mod m, \quad k \geq 0$$

x_0 is **seed**, e.g. based on time

mod m: remainder after division by m

- $x_k \in \{0, \ldots, m-1\}$ and integer
- $x_k/m \sim$ Unif[0, 1
- $a, c \in [0, m)$ need to be carefully selected

First step: Generating Unif[0, 1]

Generated numbers will get into a loop with a certain **period**

$$x_{k-1} = (a \cdot x_k + c) \mod m, \quad k \geq 0$$

$x_0 = a = c = 7$, $m = 10$

1. $x_1 = (7 \cdot 7 + 7) \mod 10 = 56 \mod 10 = 6$
2. $x_1 = (7 \cdot 6 + 7) \mod 10 = 49 \mod 10 = 9$
3. $x_1 = (7 \cdot 9 + 7) \mod 10 = 70 \mod 10 = 0$
4. $x_1 = (7 \cdot 0 + 7) \mod 10 = 7 \mod 10 = 7$
5. $x_1 = (7 \cdot 7 + 7) \mod 10 = 56 \mod 10 = 6$
6. ...

First step: Generating Unif[0, 1]

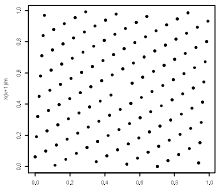
```
fthreebits<-function(k,s,L,X){
    X0<-4*s+1;a<-8*k+5;m<-2^L;X<-X0
    for (i in 1:N){
        print(c(X,rev(intToBits(X)[1:5])))
        X<-(a*X)%%m ##c=0
    }
}
```

```
> source("Congben.R");fthreebits(k=2,s=3,L=8,N=10)
[1] 13  0  1  1  0  1
[1] 17  1  0  0  0  1
[1] 101  0  0  1  0  1
[1] 73  0  1  0  0  1
[1] 253  1  1  1  0  1
[1] 193  0  0  0  0  1
[1] 233  1  0  1  0  1
[1] 121  1  1  0  0  1
[1] 237  0  1  1  0  1
[1] 113  1  0  0  0  1
```

Last three bits change between 001 and 101
Discard less significant bits

First step: Generating Unif[0, 1]

See also D. E. Knuth (1998). The Art of Computer Programming, Volume 2, Addison–Wesley. Ch. 3.3.4



732A90_ComputationalStatisticsVT2019_Lecture03codeSlide06.R

First step: Generating Unif[0, 1]

- Period is $\leq m$ by definition
- a, c, m (**large**) have to be chosen carefully
  1. c and m have to be relatively prime (no common divisors bar 1)
  2. $a = 1 \mod p$ for every prime divisor p of m
  3. $a = 1 \mod 4$ if 4 divides m
  4. Then full period m reached (**what about** $a = c = 1$?)
- Seed defines the random sequence — same seed, same sequence
  Be careful when re–opening an R workspace
- Other methods (not in this course)

Second step: Generating Unif[a, b]

- $U \sim$ Unif[0, 1] can be transformed into $X \sim$ Unif[a, b] as

$$X = a + U \cdot (b - a)$$

- U can also be transformed into **discrete** uniform distribution on integers $\in \{1, \ldots, n\}$ as ($\lfloor \cdot \rfloor$, integer part)

$$X = \lfloor nU \rfloor - 1$$

Questions

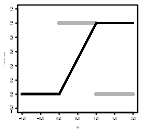
1. Why +1?
2. How can U be transformed into Y, where Y is discrete uniform on integers $(50, 55, 60)$?

Second step: Generating nonuniform random numbers

- $U \sim$ Unif$(0, 1)$
- Let F_U be the *cumulative distribution function* (CDF) of U

$$F_U(u) = P(U \leq u) = \begin{cases} 0 & u < 0 \\ u & 0 < u \leq 1 \\ 1 & 1 < u \end{cases}$$

- The *probability distribution function* (PDF) of U



$$f_U(u) \quad \begin{cases} 1 & 0 < u < 1 \\ 0 & u \notin (0, 1) \end{cases}$$

Inverse CDF method

Let X be a random variable with CDF $X \sim F_X$ (F_X strictly increasing)

Consider $Y = F_X^{-1}(U)$, where $U \sim$ Unif$(0, 1)$

$$\begin{aligned} F_Y(y) &= P(Y \leq y) = P(F_X^{-1}(U) \leq y) \\ &= P(F_X(F_X^{-1}(U)) \leq F_X(y)) \\ &= P(U \leq F_X(y)) = F_U(F_X(y)) = F_X(y) \end{aligned}$$

Y has same probability distribution as X

Inverse CDF method

If we can generate $U \sim$ Unif$(0, 1)$, then

we can generate $X \sim F_X$ as

$$X = F_X^{-1}(U)$$

Provided we can calculate $F_X^{-1} \ldots$

Inverse CDF method: Example

Let $X \sim \exp(\lambda)$, i.e. with pdf

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

implying (**SHOW THIS**)

$$F_X(x) \int_{-\infty}^{x} f_X(s) \mathrm{d}s \quad 1 \quad e^{-\lambda x}, \quad x > 0$$

QUESTIONS:
What is $F_X(x)$ for $x < 0$?
What is E$[X]$?

Inverse CDF method: Example

Find F_X^{-1}

$$\begin{aligned} y &= 1 - e^{-\lambda x} \\ e^{-\lambda x} &\quad 1 - y \\ x &= -\tfrac{1}{\lambda} \ln(1 - y) \\ F_X^{-1}(y) &= -\tfrac{1}{\lambda} \ln(1 - y) \end{aligned}$$

Hence, if $U \sim$ U$(0, 1)$, then

$$-\frac{1}{\lambda} \ln(1 - U) = X \sim \exp(\lambda)$$

Inverse CDF method

1. When F_X^{-1} can be derived: **EASY**
2. When **NOT**: numerical solution

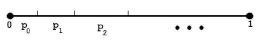
   time–consuming

   numerical errors ?

Situation 2 is common ... e.g. $\mathcal{N}(0, 1)$

Generating discrete RVs

1. Define distribution $P(X = x_i) = p_i$
2. Generate $U \sim$ Unif$(0, 1)$
3. If $U \leq p_0$, set $X = x_0$
4. Else if $U \leq p_0 + p_1$, set $X = x_1$
5. ...

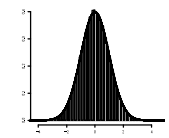


Generating $\mathcal{N}(0, 1)$

Assume

- $\theta \in$ Unif$(0, 2\pi)$
- $D \in$ Unif$(0, 1)$

1: Generate θ, D
2: Generate X_1 and X_2 as



$$X_1 = \sqrt{-2 \ln D} \cos\theta$$
$$X_2 = \sqrt{-2 \ln D} \sin\theta$$

X_1 and X_2 are independent and normally distributed

But finding such transformations is not easy

Acceptance/rejection methods

- IDEA: generate $Y \sim f_Y$ similar to some known PDF f_X
- IDEA: f_Y is easy to generate from
- REQUIREMENT: there exists a constant c

$$\forall_x c f_Y(x) \geq f_X(x)$$

- f_Y: majorizing density, proposal density
- f_X: target density
- c: majorizing constant

Acceptance/rejection methods

1: **while** X not generated **do**
2: Generate $Y \sim f_Y$
3: Generate $U \sim \text{Unif}(0,1)$
4: **if** $U \leq f_X(Y)/(cf_Y(Y))$ **then**
5: $X = Y$
6: Set X is generated
7: **end if**
8: **end while**

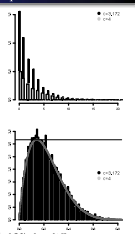
- $X \sim f_X$ **CHECK THIS**
- Larger c: larger rejection rates—c as small as possible
  number of draws $\sim$ Geometric(1/c) mean: c
- Can work in higher dimensions—**but** high rejection rate

Acceptance/rejection methods: Example

Generate beta(2,7)

```
y<-dbeta(seq(0,2,0.0001),2,7)
c<-max(y); c
[1] 3.172554
```

1: **while** X not generated **do**
2: Generate $Y \sim \text{Unif}(0,1)$
3: Generate $U \sim \text{Unif}(0,1)$
4: **if** $U \leq dbeta(Y,2,7)/(c \cdot 1)$ **then**
5: $X = Y$
6: Set X is generated
7: **end if**
8: **end while**



QUESTION:
Compare acceptance and rejection regions (of Y) for different c.

Acceptance/rejection methods:

- Acceptance/rejection is difficult to apply
- Difficult to find majorizing density
  - can always take $\sup(f_X) \cdot \text{Unif}(0,1)$
  - but what is the problem?

Generating multivariate normal

Generate $\mathcal{N}(\vec{\mu}, \mathbf{\Sigma}) \in \mathbb{R}^n$

1: Generate n i.i.d. $\mathcal{N}(0,1)$ r.vs. $\vec{X} = (X_1, \ldots, X_n)$
{We know how to do this, see slide 16}
2: Compute Cholesky decomposition (a.k.a. matrix square root) of $\mathbf{\Sigma}$, i.e. find $\mathbf{A}$, lower triangular s.t. $\mathbf{A}\mathbf{A}^T = \mathbf{\Sigma}$,
{in R: `chol()` }
3: $\vec{Y} = \mu + \mathbf{A}\vec{X}$

QUESTION:
what is the expectation and variance–covariance of $\vec{Y}$?

Random numbers in R

1. `d`distribution name`()`: density of distribution
2. `p`distribution name`()`: CDF of distribution
3. `q`distribution name`()`: quantiles of distribution
4. `r`distribution name`()`: simulate from distribution

Summary

- Computers generate pseudo–random numbers
- We draw from pseudo–uniform and transform to desired distribution
- Analytical methods for transforming exist but are distribution specific

Monte Carlo Methods

732A90
Computational Statistics

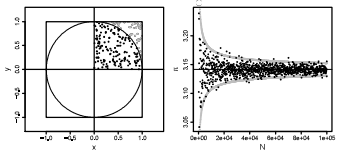
Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

5 II 2019 (P42)
Department of Computer and Information Science
Linköping University

What is the area of the unit circle?

```
f.circArea<-function(N){
  m.xy<-cbind(runif(N),runif(N))
  4*sum(apply(m.xy,1,function(xy){xy^2+xy
    [2]^2<1}))/N
}
```

$$3.141 \approx \pi = \int_{-1}^{1} 1 \mathrm{d}x$$



Monte Carlo methods: outline

- **Monte Carlo methods** are a class of computational algorithms that use repeated random sampling to compute their results.
- Monte Carlo methods for random number generation
  - Metropolis–Hastings algorithm
  - Gibbs sampler
- Monte Carlo methods for statistical inference
  - Estimate integrals (we already did!)
  - Variance estimation
  - Variance reduction: importance sampling, control variates

Markov Chain Monte Carlo

Previous lecture: Generate

- univariate distributions (inverse CDF, acceptance/rejection)
- multivariate normal

but general multivariate distribution?

MCMC

Bayesian inference: Recap

A dataset D is obtained by sampling from a distribution $f(\cdot|\theta)$. How to estimate θ?

- *Frequentists*: θ is an unknown but fixed parameter, compose likelihood $\mathcal{L}(D|\theta)$ and find θ that maximizes it.
- *Bayesians*: θ is a random variable with **prior** probability law $p(\theta)$ before observing D
- After observing D, Bayes' theorem gives

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)\mathrm{d}\theta}$$

Bayesian inference: Recap

$$p(\theta|D) \quad \frac{p(D|\theta)p(\theta)}{p(D)} \quad \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)\mathrm{d}\theta}$$

We know: $p(D|\theta)$ (the model), $p(\theta)$ (the prior)
We need: simulate from $p(\theta|D)$ (the posterior)

1. General (multivariate) type distribution
2. Integral can be impossible to compute

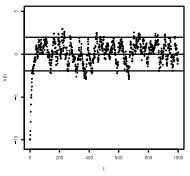
1. MCMC solves this
2. Not needed (given D it is constant)

Markov Chains: Recap

- A Markov chain is a sequence $X_0, X_1, \ldots$ of random variables such that the distribution of the next value depends only on the current one (and parameters).
- $P(X_{t+1}|X_t)$ is called a **transition kernel**. Assume it does not depend on t (**time homogeneous**).
- A Markov chain is **stationary**, with stationary distribution Φ, if $\forall_k\ X_k \sim \Phi$
- One shows (not trivial in general) that under *certain* conditions a Markov chain will converge to the stationary distribution in the limit.

Markov Chains: Example

$$X(t+1) = e^{-1}X(t) + \epsilon\,, \epsilon \sim \mathcal{N}(0, \tfrac{5}{2}\cdot(1-e^{-2}))$$



Discard first $K-1$ samples: **burn–in period**

MCMC: Example

Linear regression with residual normally/student/etc. distributed

$$Y \quad \beta X \mid c$$

How to find credible interval for β if we know $\mathrm{Var}[\epsilon] = \sigma^2$?

1. $$P(Y|X,\beta) = \prod_{i=1}^{N} f(Y_i|\mathrm{mean} = \beta X_i, \mathrm{var} = \sigma^2)$$
2. Obtain $P(\beta|Y,X)$ by drawing from $P(Y|X,\beta)P(\beta)$ **in a clever way**.
3. The prior ?
4. Use the MCMC sample to obtain quantiles.

Normal residual: analytical solution

Metropolis–Hastings algorithm

We have

- A PDF $\pi(x)$ that we want to sample from.
- A **proposal distribution** $q(\cdot|X_t)$ that has a **regular** form w.r.t. to $\pi(\cdot)$
  E.g. $q(\cdot|X_t)$ is normal with mean X_t and given variance
- *Regular* form: suffices that the proposal has the same support as π.

Metropolis–Hastings Sampler

$$\alpha(X_t, Y) = \min\left\{1, \frac{\pi(Y)q(X_t\,Y)}{\pi(X_t)q(Y|X_t)}\right\}$$

1: Initialize chain to X_0. $t = 0$
2: **while** $t < t_{\max}$ **do**
3: Generate a candidate point $Y \sim q(\cdot|X_t)$
4: Generate $U \sim Unif(0,1)$
5: **if** $U < \alpha(X_t, Y)$ **then**
6: $X_{t+1} = Y$
7: **else**
8: X_{t+1} X_t
9: **end if**
10: $t = t-1$
11: **end while**

Metropolis–Hastings Sampler: Properties

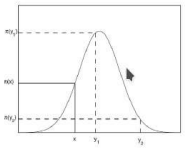
- Informally: "The chain $(X_t)_{t=0}^{\infty}$ will converge to $\pi(\cdot)$."
- The chain might not move sometimes.
- The values of the chain are dependent.
- If $q(X_t|Y) = q(Y|X_t)$ (i.e. symmetric proposal) we get **Random–walk Monte Carlo**:

$$\alpha(X_t, Y) = \min\left\{1, \frac{\pi(Y)}{\pi(X_t)}\right\}$$

Choice of proposal distribution

- In Random–Walk Monte Carlo

If $\pi(Y) \geq \pi(X)$, the chain moves to the next point, otherwise only with some probability.

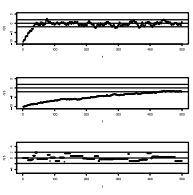


Choice of proposal dist.: target: $\pi(\cdot) = \mathcal{N}(0,1)$

`732A90_ComputationalStatistics_VT2019_Lecture04codeSlide14.R`

Choice of proposal distribution

q normal with sd: `props`= 0.5, 0.1 and 20



Gibbs sampler: alternative to Metropolis–Hastings

We want to generate from a distribution on $\mathbb{R}^d$.

1: Initialize chain to X_0 $(X_{0,1}, \ldots, X_{0,d})$, t 0
2: **while** $t < t_{\max}$ **do**
3: **for** $i = 1, \ldots, d$ **do**
4: Generate

$$X_{t+1,i} \sim f(\cdot|X_{t+1,1}, \ldots, \mathbf{X_{t+1,i-1}}, \mathbf{X_{t,i+1}}, \ldots, X_{t,d})$$

5: **end for**
6: $t = t-1$
7: **end while**

Gibbs sampler

- At each iteration inside the `for` loop univariate random numbers are generated.
- Only one element is updated.
- **WE NEED TO KNOW THE CONDITIONAL MARGINAL DISTRIBUTIONS**.
- Convergence may be slow.
- Can be useful in high dimensions (i.e. proposal density may be difficult to find in another way).

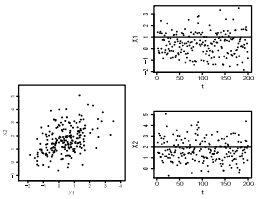
Gibbs sampler: target: d–dim $\mathcal{N}(\mu, \Sigma)$

`732A90_ComputationalStatisticsVT2019_Lecture04codeSlide18.R`

Gibbs sampler: Example (code: see R scripts)

Generate from

$$\mathcal{N}([1\ 2]^T, \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix})$$



Convergence monitoring

- When should we stop the chain? When are we (nearly) at the stationary distribution?
- Typically such a sample is generated to make further inference.

Convergence monitoring: Gelman–Rubin method

We want to estimate $v(\theta)$.

1. Generate k sequences of length n with different starting points.
2. Compute between– and within– sequence variances:

$$B \quad \frac{n}{k-1}\sum_{i=1}^{k}(\overline{v}_{i\cdot} \quad \overline{v}_{\cdot\cdot})^2 \quad W \quad \sum_{i=1}^{k}\frac{s_i^2}{k} \quad s_i^2 \quad \sum_{j=1}^{n}\frac{(\overline{v}_{ij}-\overline{v}_{i\cdot})^2}{n-1}$$

3. Overall variance estimate: $\hat{\mathrm{Var}}[v] \quad \frac{n-1}{n}W + \frac{1}{n}B$
4. Gelman–Rubin factor:

$$\sqrt{R} = \sqrt{\frac{\hat{\mathrm{Var}}[v]}{W}}$$

5. Values much larger than 1 indicate lack of convergence
6. See `?coda::gelman.diag`

Gibbs sampler

```
library(coda)
f1<-mcmc.list(); f2<-mcmc.list(); n<-100; k<-20
X1<-matrix(rnorm(n*k),ncol=k,nrow=n)
X2<-X1+(apply(X1,2,cumsum)*(matrix(rep(1:n,k),ncol=
    k)^2))
for (i in 1:k){f1[[i]]<-as.mcmc(X1[,i]); f2[[i]]<-as
    .mcmc(X2[,i])}
print(gelman.diag(f1))
# Potential scale reduction factors:
#      Point est. Upper C.I.
#[1,]       0.999       1.01

print(gelman.diag(f2))
# Potential scale reduction factors:
#      Point est. Upper C.I.
#[1,]       1.82        2.38
```

MC for inference

- Estimation of a definite integral

$$\theta = \int_D f(x)\mathrm{d}x \quad \left(\text{recall } \pi = \int_{\bigcirc} 1\mathrm{d}x\right)$$

- Decompose into:

$$f(x) = g(x)p(x) \quad \text{where} \quad \int_D p(x)\mathrm{d}x = 1$$

- Then, if $X \sim p(\cdot)$

$$\theta = \mathrm{E}[g(X)] = \int_D g(x)p(x)\mathrm{d}x$$

-

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} g(x_i), \quad \forall_i x_i \sim p(\cdot)$$

MC for inference

- Decomposition is not unique, some will be better (lower variance) others worse. $p(x) \propto |f(x)|$: minimal
- Can we easily generate from $p(\cdot)$?
- Bayesian inference: use MCMC samples from $p(\theta|D)$ to obtain a point estimator

$$\theta^* \quad \int \theta p(\theta|D) \approx \frac{1}{n}\sum_{i=1}^{m}\theta_i$$

- $\hat{\theta}$ depends on n and $g(X)$, how variable will it be?

$$\widehat{\mathrm{Var}\left[\hat{\theta}\right]} = \frac{1}{n(n-1)}\sum_{i=1}^{n}\left(g(x_i) - \overline{g(x)}\right)^2$$

- MCMC: estimator biases as chain correlated, use longer chain and batch mean instead of x_i.

Summary

1. Generating data from a general multivariate distribution
2. Markov Chain Monte Carlo: Metropolis–Hastings algorithm, Gibbs sampling
3. Convergence: Gelman–Rubin method
4. Estimation of integral

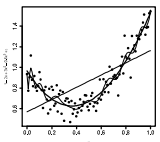
Model Selection and Hypothesis Testing

732A90
Computational Statistics

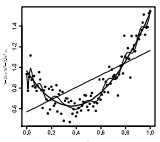
Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

II 2019 ()
Department of Computer and Information Science
Linköping University

Model selection



Model selection



Tools for model selection

- **Comparing different models**
- Information criteria (not this course)
- Cross–validation
- Hypothesis testing

- **Uncertainty estimation**
- Confidence intervals

Hypothesis testing: Recap

1. Assume a probabilistic model
   State a null hypothesis (H_0 e.g. no difference) and alternative (H_1 difference)
2. Observe data X
3. Calculate a test statistic e.g. $T(X) = (\overline{X})/(\widehat{\mathrm{sd}(X)})$ (different statistics will have different **efficiency** (power, ability to distinguish between hypotheses) associated with them)
4. Under H_0 $T(X)$ has "known" distribution
5. Decision: Is the value of $T(X)$ *surprising* (in the **critical region**)? If so reject H_0 in favour of H_1.

Hypothesis testing: Example

```
x<-rnorm(10,mean=4,sd=1)
```

Hypotheses:

$H_0 : \mu = 4, X \sim \mathcal{N}(\mu, \sigma^2)$
$H_1 : \mu \neq 4, X \sim \mathcal{N}(\mu, \sigma^2)$

Hypothesis testing: Example

```
x<-rnorm(10,mean=4,sd=1)
```

Hypotheses:

$H_0 : \mu = 4, X \sim \mathcal{N}(\mu, \sigma^2)$
$H_1 : \mu \neq 4, X \sim \mathcal{N}(\mu, \sigma^2)$

Test statistic

$T(x) = \frac{\overline{x}-\mu}{s/\sqrt{n}} \sim t(n-1)$

```
tx<-(mean(x)-4)/(sqrt(var(x)/length(x)))
t0<-qt(0.975,df=length(x)-1)
(tx>t0)||(tx<(-t0)) ## reject if TRUE
```

Hypothesis testing: Example

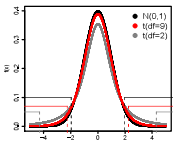
```
x<-rnorm(10,mean=4,sd=1)
```

Hypotheses:

$H_0 : \mu = 4, X \sim \mathcal{N}(\mu, \sigma^2)$
$H_1 : \mu \neq 4, X \sim \mathcal{N}(\mu, \sigma^2)$

Test statistic

$T(x) = \frac{\overline{x}-\mu}{s/\sqrt{n}} \sim t(n-1)$




```
tx<-(mean(x)-4)/(sqrt(var(x)/length(x)))
t0<-qt(0.975,df=length(x)-1)
(tx>t0)||(tx<(-t0)) ## reject if TRUE
```

Hypothesis testing: Power

How does one compares different statistics?
POWER
Power = 1 − Type II error

Ability to correctly identify *surprise*, i.e. indicate H_1.

How to compute power?

- Analytically (?)
- Generate data samples that satisfy H_1
  Compute percent of correct rejections

Monte Carlo Hypothesis testing

We may use "any" test statistic.
We do **not** need to know its distribution.

$H_0 : \mu = 4, X \sim \mathcal{N}(\mu, \sigma^2)$
$H_1 : \mu \neq 4, X \sim \mathcal{N}(\mu, \sigma^2)$

Monte Carlo Hypothesis testing

We may use "any" test statistic.
We do **not** need to know its distribution.

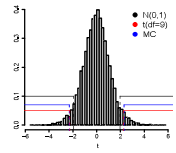
$H_0 : \mu = 4, X \sim \mathcal{N}(\mu, \sigma^2)$
$H_1 : \mu \neq 4, X \sim \mathcal{N}(\mu, \sigma^2)$

Test statistic

$T(x) = \frac{\overline{x}-\mu}{s/\sqrt{n}} \sim t(n-1)$

1: **for** $i = 1$ to B **do**
2: Generate $Y_1, \ldots, Y_n$ i.i.d. from H_0, i.e. $\mathcal{N}(\mathbf{4}, \sigma^2)$
3: Compute t_i from $Y_1, \ldots, Y_n$
4: **end for**
5: Use $t_1, \ldots, t_B$ to construct a histogram
6: Use the histogram as the distribution of $T(x)$ under H_0

Monte Carlo Hypothesis testing




```
x<-rnorm(10,4,1)
s<-var(x)
B<-10000
n<-length(x)
tsamp<-rep(NA,B)
for (i in 1:B){
    Y<-rnorm(n,4,s)
    tsamp[i]<-(mean(Y)-4)/(sd(Y)/sqrt(length(Y)))
}
hist(tsamp,breaks=50,col=gray(0.8),main="",xlab="t"
    ,ylab="",freq=FALSE,cex.axis=1.5,cex.lab=1.5)
```

Permutation tests

- A. k. a. randomization tests
- One solution if we do not know the distribution under H_0
- Computationally expensive
- Any sample size
- Two sample problem:
  - Population 1 distributed as F
  - Population 2 distributed as G
  - $H_0 : F = G$
    $H_1 : F \neq G$

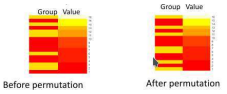
Permutation tests: mouse data

```
> t(mouse)
        [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9]  [,10] [,11] [,12]
Group   "y"   "z"   "z"   "y"   "y"   "z"   "z"   "y"   "y"   "y"   "z"   "z"
Value   " 10" " 16" " 23" " 27" " 31" " 38" " 46" " 46" " 50" " 52" " 94" " 99"
        [,13] [,14] [,15] [,16]
Group   "y"   "z"   "y"   "z"
Value   "104" "141" "146" "197"
```

Do the values differ significantly between control and treatment groups?

Permutation tests

IDEA: If $F = G$ then **group label does not matter**
We may permute labels and still have a sample from F (or G)



Test statistic:

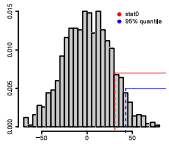
$T(X) = \text{mean(values|group} = z) - \text{mean(values|group} = y)$

Permutation test: scheme

1: $T(X)$ value of statistic from observed data
2: Create permutations $g_1^*, \ldots, g_B^*$ of group variable
{If the number of permutations is too large, sample B randomly **without** replacement. E.g. generate random permutations and keep only unique ones.}
3: Evaluate test statistic on each permutation
4: Estimate p–value: $\hat{p} = \#\{T(X_{g_k^*}) \geq T(X)\}/B$
5: If test is two–sided: $\hat{p} = \#\{|T(X_{g_k^*})| \geq |T(X)|\}/B$

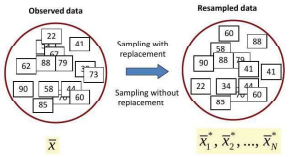
Permutation tests

Do we reject the null?




```
B=1000
stat=numeric(B)
n=dim(mouse)[1]
for(b in 1:B){
  Gb=sample(mouse$Group, n)
  stat[b]=mean(mouse$Value[Gb=='z'])-mean(mouse$
      Value[Gb=='y'])
}
stat0=mean(mouse$Value[mouse$Group=='z'])-mean(
    mouse$Value[mouse$Group=='y'])
print(c(stat0,mean(stat>stat0)))
## [1] 30.63492  0.12700
```

Resampling methods



$\overline{X}$ $\quad$ $\overline{X}_1^*, \overline{X}_2^*, \ldots, \overline{X}_N^*$

Jackknife and bootstrap

Theory **different**, coding **similar**

Data (**i.i.d.**) $X \sim F(\cdot, w)$

1: Observed data: $D \quad (X_1, \ldots, X_n)$, estimator $\widehat{w} \quad T(D)$
2: **for** $i = 1, \ldots, B$ { Jackknife $B \leq n$} **do**
3: Generate

$D_i^* = (X_1^*, \ldots, X_n^*)$ by sampling with replacement
{**Nonparametric Bootstrap**, F unknown}

$D_i^* = X[-i]$ {**Jackknife**, F unknown}

$D_i^* = (X_1^*, \ldots, X_n^*)$ by generating from $F(\cdot, \widehat{w})$
{**Parametric Bootstrap**. F known}
4: **end for**
5: Distribution of $\widehat{w}$ is estimated by $T(D_1^*), \ldots, T(D_B^*)$
{The histogram based on resampled values is used in place of the true density.}

Uncertainty estimation: confidence intervals

Estimate $100(1-\alpha)\%$ percentile confidence interval for w

$se(\cdot)$ is the square root of estimated variance (computationally heavy)

NOT by jackknife **TOO DEPENDENT!!**

1: Compute $T(D_1^*), \ldots, T(D_B^*)$
2: Sort in ascending order, obtaining $y_1, \ldots, y_B$
{**percentile** method} **OR**
Compute $y_i = (T(D_i^*) - T(D))/(se(T(D_i^*)))$ $i = 1, \ldots, B$
{**t** method}
3: Define $A_1 = \lfloor (B\alpha/2) \rfloor$, $A_2 = \lfloor (B - B\alpha/2) \rfloor$
4: Confidence interval is given by
(y_{A_1}, y_{A_2}) {**percentile** method} **OR**
$(T(D) - se(T(D^*)) \cdot y_{A_1}, T(D) + se(T(D^*)) \cdot y_{A_2})$
{**t** method}

Hypothesis testing: does statistic from observed data fall into CI (H_0) or not (H_1)

Uncertainty estimation: variance of estimator

Bootstrap

$$\widehat{\mathrm{Var}}[T(\cdot)] = \frac{1}{B-1} \sum_{i=1}^{B} \left(T(D_i^*) - \overline{T(D^*)} \right)^2$$

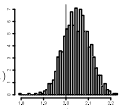
Jackknife ($n = B$)

$$\widehat{\mathrm{Var}}[T(\cdot)] = \frac{1}{n(n-1)} \sum_{i=1}^{n} (T_i^* - J(T))^2,$$

where

$$T_i^* = nT(D) - (n-1)T(D_i^*) \quad J(T) = \frac{1}{n} \sum_{i=1}^{n} T_i^*$$

Bootstrap in R



```
library("boot")
stat1<-function(data,vn){
    data<-as.data.frame(data[vn,])
    res<-lm(Response~Predictor,data)
    res$coefficients[2]
}
x<-rnorm(100);data<-cbind(Predictor=x,Response=3+2*
    x+rnorm(length(x),sd=0.5))
res<-boot(data,stat1,R=1000)
print(boot.ci(res))
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
##Based on 1000 bootstrap replicates
#Intervals :
#Level      Normal              Basic
#95%   ( 1.933,  2.164 )   ( 1.935,  2.162 )
# Level      Percentile            BCa
#95%   ( 1.934,  2.161 )   ( 1.936,  2.166 )
```

Bootstrap bias correction

1: Observed data: $D \quad (X_1, \ldots, X_n)$, estimator $\widehat{w} \quad T(D)$
2: **for** $i = 1, \ldots, B$ **do**
3: Generate

$D_i^* = (X_1^*, \ldots, X_n^*)$ by sampling with replacement.
4: Calculate $T_i^* = T(D_i^*)$.
5: **end for**
6: Bias corrected estimator is

$$T_1 := 2T(D) - \frac{1}{B} \sum_{i=1}^{B} T_i^*.$$

Jackknife also has a bias correction method (see 2016 slides).

Comments

- Jackknife overestimate variance
- Bootstrap–t method is more accurate than percentile
- Permutations: sampling **without** replacement, bootstrap **with**
- Permutation p–value exact if all permutations used, bootstrap always approximate
- Bootstrap may be used for a wider class of problems
- Nonparametric bootstrap works badly for small samples ($n < 40$)
- Parametric bootstrap can work for small samples
- Bias corrections
- Methods do not require distributional assumptions

Permutation tests for model selection

Data predictors: `X[,c(V1,V2)]`, response: `Y`

Model M relating Y and X

Competing models

H_0 variables `V1` should not be in M (smaller model)

H_1 all variables are significant

Test statistic: $T(M)$

Permutation test

1: **for** $i = 1 \ldots B$ **do**
2: Obtain `V1*` by permuting order of columns in `V1`, fit model `Y~M(X[,c(V1*,V2)])`
3: Compute test statistic T_i for this model
4: **end for**
5: Compute p–value using above distribution of T

Summary

- Why are some models better than others?
- Hypothesis testing
- Monte Carlo hypothesis testing
- Resampling methods (permutations, jackknife, bootstrap)
- Simulation methods (parametric bootstrap)

EM Algorithm, Stochastic Optimization

732A90
Computational Statistics

Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

28 II 2018 (A37)
Department of Computer and Information Science
Linköping University

Stochastic and combinatorial optimization

- So far: Unconstrained optimization
  - Predictor variables are continuous
  - Response function is differentiable
- We discussed Steepest descent, Newton, BFGS, CG
- But: predictors can be discrete
  (scheduling problems, travelling salesman)
- But: outcome can be discrete, noisy or multi–modal

Stochastic and combinatorial optimization

Given a (large) set of states S, find

$$\min_{s \in S} f(s)$$

- Exhaustive search (shortest path algorithm)
- Often exhaustive search is NP–hard (TSP)
- Alternative: stochastic methods
  random search

Simulated annealing

Motivation from physics: cooling of metal

- Parameters:
  Energy of metal
  (decreasing, but not strictly monotonic)
  Temperature (decreasing)
- Aim: find global minimum energy

Simulated annealing

0. Set $k = 1$ and initialize state s.
1. Compute the temperature $T(k)$.
2. Set $i = 0$ and $j = 0$.
3. Generate a new state r and compute $\delta f = f(r) - f(s)$.
4. Based on δf, decide whether to move from state s to state r.
   If $\delta f \leq 0$,
   accept state r;
   otherwise,
   accept state r with a probability $P(\delta f, T(k))$.
   If state r is accepted, set $s = r$ and $i = i + 1$.
5. If i is equal to the limit for the number of successes at a given temperature, go to step 1.
6. Set $j = j + 1$. If j is less than the limit for the number of iterations at given temperature, go to step 3.
7. If $i = 0$,
   deliver s as the optimum; otherwise,
   if $k < k_{max}$,
   set $k = k + 1$ and go to step 1;
   otherwise,
   issue message that
   'algorithm did not converge in k_{max} iterations'.

Simulated annealing

- `https://www.youtube.com/watch?v=iaq_Fpr4KZc`
- Generating new state:
  - Continuous: choose a new point a (random) distance from the current one
  - Discrete: similar or some rearrangement
- Selection probability: e.g $\exp(\ \delta f(x)/T)$: decreasing with $f(x)$, increasing with T
- Temperature function: constant, proportional to k, or

$$T(k+1) = b(k)T(k), \quad b(k) = (\log(k))^{-1}$$

Remember: A smaller value is better than one on the path to the global minimum! Always keep track of smallest found.

Simulated annealing: TSP example

Assume constant temperature

1: Choose initial configuration $(Town_1, \ldots, Town_n)$
2: $k = 1$
3: **while** $k < k_{max} + 1$ **do**
4: Generate new configuration by rearrangement.

$$(1,2,3,4,5,6,7,8,9) \rightarrow (1,6,5,4,3,2,7,8,9)$$
$$(1,2,3,4,5,6,7,8,9) \rightarrow (1,7,8,2,3,4,5,6,9)$$

5: Measure difference in path length (δf) between old and new configuration
6: **if** shorter path found **then**
7: accept it
8: **else**
9: accept it with probability $P(\delta f)$
10: **end if**
11: $k++$
12: **end while**

Genetic algorithm

- Inspiration from evolutionary theory: survival of the fittest
- Variables=genotypes
- Observation organism, characterized by genetic code
- State space=population of organisms
- Objective function=fitness of organism

New points are obtained from old points by crossover and mutation, the population only retains the fittest organisms (with better objective function).

`https://en.wikipedia.org/wiki/List_of_genetic_algorithm_applications`

Genetic algorithm

Encoding points

1. Enumerate each element of the state space, S
2. Code for observation i is binary representation of i (or something else)

Mutation and recombination rules

Generation k	Generation $k+1$
Crossover	
$x_i^{(k)}$ 1100|001	$\rightarrow x_i^{(k+1)}$ 11011010
$x_j^{(k)}$ 0011|010	
Inversion	
$x_i^{(k)}$ 1110|011	$\rightarrow x_i^{(k+1)}$ 11010111
Mutation	
$x_i^{(k)}$ 1110|011	$\rightarrow x_i^{(k+1)}$ 10111011
Clone	
$x_i^{(k)}$ 1110|011	$\rightarrow x_i^{(k+1)}$ 11101011

Genetic algorithm

0. Determine a representation of the problem, and define an initial population, $x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}$. Set $k = 0$.
1. Compute the objective function (the "fitness") for each member of the population, $f(x_i^{(k)})$ and assign probabilities p_i to each item in the population, perhaps proportional to its fitness.
2. Choose (with replacement) a probability sample of size $m \leq n$. This is the reproducing population.
3. Randomly form a new population $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}$ from the reproducing population, using various mutation and recombination rules (see Table 6.2). This may be done using random selection of the rule for each individual of pair of individuals.
4. If convergence criteria are met, stop, and deliver $\arg\min_{x_i^{(k+1)}} f(x_i^{(k+1)})$ as the optimum; otherwise, set $k = k + 1$ and go to step 1.

Genetic algorithm: TSP example

Encoding and crossover

- Encode tours as $A_1, \ldots, A_n$ but

Parent 1: FAB|ECGD Parent 2: DEA|CGBF
Child: FAB|CGBF Child: DEA|ECGD

Instead

1. Remove FAB from DEACGBF $\longrightarrow$ DECG.
   Child becomes FABDECG.
2. Second child will be by taking prefix from Parent 2:
   DEAFBCG

Genetic algorithm: Mutations

- If a population is small and only crossover: the input domain becomes limited and may converge to a local minimum.
- Large initial populations are computationally heavy.
- Mutations allow one to explore more of S: jump out of local minimum.
- In TSP: mutation move a city in the tour to another position.
- Reproduction: Among m tours selected at step 2, two best are selected for reproduction, two worst replaced by children.
- If m is large, some tours might never be parents, global solution may be missed. Random chance of reproduction?
- Mutation probability is usually small (unless you want to jump wildly)

EM algorithm

Fundamental algorithm of computational statistics!

Model depends on the data which are observed (known) $\mathbf{Y}$ and **latent** (unobserved) data $\mathbf{Z}$.

The data's (**both** $\mathbf{Y}$'s and $\mathbf{Z}$'s) distribution depends on some parameters θ.

AIM: Find MLE of θ.

- All data is known: Apply unconstrained optimization (discussed in Lecture 2)
- Unobserved data
  - **Sometimes** it is possible to look at the marginal distribution of the observed data.
  - Otherwise: **EM algorithm**

EM algorithm

Let

$$Q(\theta, \theta^k) = \int \log p(\mathbf{Y}, \mathbf{z}|\theta) p(\mathbf{z}|\mathbf{Y}, \theta^k) \mathrm{d}\mathbf{z} = \mathrm{E}\left[\mathrm{loglik}(\theta|\mathbf{Y}, \mathbf{Z})|\theta^k, \mathbf{Y}\right]$$

1: $k = 0$, $\theta^0 = \theta^0$
2: **while** Convergence not attained **and** $k < k_{max} + 1$ **do**
3: **E–step**: Derive $Q(\theta, \theta^k)$
4: **M–step**: $\theta^{k-1} = \mathrm{argmax}_\theta \ Q(\theta, \theta^k)$
5: $k++$
6: **end while**

Example: Normal data with missing values (but here analytical approach is also possible)

EM algorithm: R

732A90_ComputationalStatistics_VT2019_Lecture06codeSlide15.R

EM algorithm: R

```
> Y<-rnorm(100)
> Y[sample(1:length(Y),20,replace=FALSE)]<-NA
> EM.Norm(Y,0.0001,100)
[1]    1.0000    0.1000 -997.5705
[1]    0.1341894    1.3227095 -128.2789837
[1]   -0.03897274    1.38734070 -126.86036252
[1]   -0.07360517    1.39307050 -126.80801589
[1]   -0.08053165    1.39392861 -126.80593837
[1]   -0.08191695    1.39408871 -126.80585537
> mean(Y,na.rm=TRUE)
[1] -0.08226328
> var(Y,na.rm=TRUE)
[1] 1.411775
```

Notice: can be done by studying marginal distribution of observed data.

EM algorithm: Applications

Mixture models Z is a latent variable, $P(Z = k) = \pi_k$

- Mixed data comes from different sources (e.g. for regression, classification)
- Clustering
  1. Density in each cluster is normally distributed.
  2. Cluster label is latent (we do not know what are the chances an observation is from the given cluster)

$$p(x) = \sum_{k\ 1}^{K} \pi_k \mathcal{N}(x|\vec{\mu}_k, \mathbf{\Sigma}_k) \quad \text{(informally)}$$

Direct MLE leads to numerical problems.
Introduce latent class variables and use EM.

EM algorithm: Gaussian mixtures

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{9.23}$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{9.24}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\mathrm{T}} \tag{9.25}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{9.26}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{9.27}$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{9.28}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

Source: Pattern recognition by Bishop

$$Ez_{nk} = \gamma(z_{nk})$$

Summary

Random walk over the state space in search of minimum

1. Follow decreasing path
2. **BUT** with a certain probability go to higher values, to avoid local minima traps.
3. **Never forget** best found conformation!

4. Simulated annealing, Genetic algorithm, **EM algorithm**,
   Stochastic gradient descent (see 2016 slides)

Question 1: Computations with Metropolis–Hastings

Consider the following probability density function:

$$f(x) \propto x^5 e^{-x}, \quad x > 0.$$

You can see that the distribution is known up to some constant of proportionality. If you are interested (**NOT** part of the Lab) this constant can be found by applying integration by parts multiple times and equals 120.

1. Use Metropolis–Hastings algorithm to generate samples from this distribution by using proposal distribution as log–normal $LN(X_t, 1)$, take some starting point. Plot the chain you obtained as a time series plot. What can you guess about the convergence of the chain? If there is a burn–in period, what can be the size of this period?

2. . Perform Step 1 by using the chi–square distribution $\chi^2(\lfloor X_t + 1 \rfloor)$ as a proposal distribution, where $\lfloor x \rfloor$ is the floor function, meaning the integer part of x for positive x, i.e. $\lfloor 2.95 \rfloor = 2$

3. Compare the results of Steps 1 and 2 and make conclusions.

4. Generate 10 MCMC sequences using the generator from Step 2 and starting points $1, 2, \ldots,$ or 10. Use the Gelman–Rubin method to analyze convergence of these sequences.

5. Estimate
$$\int_0^\infty x f(x) \mathrm{d}x$$
using the samples from Steps 1 and 2.

6. The distribution generated is in fact a gamma distribution. Look in the literature and define the actual value of the integral. Compare it with the one you obtained.

Question 2: Gibbs sampling

A concentration of a certain chemical was measured in a water sample, and the result was stored in the data `chemical.RData` having the following variables:

- `X`: day of the measurement
- `Y`: measured concentration of the chemical.

The instrument used to measure the concentration had certain accuracy; this is why the measurements can be treated as noisy. Your purpose is to restore the expected concentration values.

1. Import the data to `R` and plot the dependence of `Y` on `X`. What kind of model is reasonable to use here?

2. A researcher has decided to use the following (random–walk) Bayesian model (n=number of observations, $\vec{\mu} = (\mu_1, \ldots, \mu_n)$ are unknown parameters):
$$Y_i \sim \mathcal{N}(\mu_i, \text{ variance} = 0.2), \quad i = 1. \ldots, n$$
where the prior is
$$p(\mu_1) = 1$$
$$p(\mu_{i+1}|\mu_i) = \mathcal{N}(\mu_i, 0.2), i = 1, \ldots, n1 \quad .$$
Present the formulae showing the likelihood $p(\vec{Y}|\vec{\mu})$ and the prior $p(\vec{\mu})$. **Hint**: a chain rule can be used here $p(\vec{\mu}) = p(\mu_1)p(\mu_2|\mu_1)p(\mu_3|\mu_2)\ldots p(\mu_n|\mu_{n1})$.

3. Use Bayes' Theorem to get the posterior up to a constant proportionality, and then find out the distributions of $(\mu_i|\vec{\mu}_{-i}, \vec{Y})$, where $\vec{\mu}_{-i}$ is a vector containing all μ values except of μ_i.

Hint A: consider for separate formulae for $(\mu_1|\vec{\mu}_{-1}, \vec{Y})$, $(\mu_n|\vec{\mu}_{-n}, \vec{Y})$ and then a formula for all remaining $(\mu_i|\vec{\mu}_{-i}, \vec{Y})$.

Hint B:
$$\exp\left(-\frac{1}{d}\left((x-a)^2 + (x-b)^2\right)\right) \propto \exp\left(-\frac{(x-(a+b)/2)^2}{d/2}\right)$$

Hint C:
$$\exp\left(-\frac{1}{d}\left((x-a)^2 + (x-b)^2 + (x-c)^2\right)\right) \propto \exp\left(-\frac{(x-(a+b+c)/3)^2}{d/3}\right)$$

4. Use the distributions derived in Step 3 to implement a Gibbs sampler that uses $\vec{\mu}^0 = (0, \ldots, 0)$ as a starting point. Run the Gibbs sampler to obtain 1000 values of $\vec{\mu}$ and then compute the expected value of $\vec{\mu}$ by using a Monte Carlo approach. Plot the expected value of $\vec{\mu}$ versus X and Y versus X in the same graph. Does it seem that you have managed to remove the noise? Does it seem that the expected value of $\vec{\mu}$ can catch the true underlying dependence between Y and X?

5. Make a trace plot for μ_n and comment on the burn–in period and convergence.

Question 1: Hypothesis testing

In 1970, the US Congress instituted a random selection process for the military draft. All 366 possible birth dates were placed in plastic capsules in a rotating drum and were selected one by one. The first date drawn from the drum received draft number one, the second date drawn received draft number two, etc. Then, eligible men were drafted in the order given by the draft number of their birth date. In a truly random lottery there should be no relationship between the date and the draft number. Your task is to investigate whether or not the draft numbers were randomly selected. The draft numbers (`Y=Draft_No`) sorted by day of year (`X=Day_of_year`) are given in the file `lottery.xls`.

1. Make a scatterplot of Y versus X and conclude whether the lottery looks random.

2. Compute an estimate $\hat{Y}$ of the expected response as a function of X by using a loess smoother (use `loess()`), put the curve $\hat{Y}$ versus X in the previous graph and state again whether the lottery looks random.

3. To check whether the lottery is random, it is reasonable to use test statistics

$$T = \frac{\hat{Y}(X_b) - \hat{Y}(X_a)}{X_b - X_a}, \quad \text{where } X_b = \operatorname{argmax}_X Y(X), X_a = \operatorname{argmin}_X Y(X)$$

   If this value is significantly greater than zero, then there should be a trend in the data and the lottery is not random. Estimate the distribution of T by using a non–parametric bootstrap with $B = 2000$ and comment whether the lottery is random or not. What is the p–value of the test?

4. Implement a function depending on *data* and B that tests the hypothesis
   H_0: Lottery is random
   versus
   H_1: Lottery is non–random
   by using a permutation test with statistics T. The function is to return the p–value of this test. Test this function on our data with $B = 2000$.

5. Make a crude estimate of the power of the test constructed in Step 4:

   (a) Generate (an obviously non–random) dataset with $n = 366$ observations by using same X as in the original data set and $Y(x) = \max(0, \min(\alpha x + \beta, 366))$, where $\alpha = 0.1$ and $\beta \sim \mathcal{N}(183, \text{sd} = 10)$.

   (b) Plug these data into the permutation test with $B = 200$ and note whether it was rejected.

   (c) Repeat Steps 5a–5b for $\alpha = 0.2, 0.3, \ldots, 10$.

   What can you say about the quality of your test statistics considering the value of the power?

Question 2: Bootstrap, jackknife and confidence intervals

The data you are going to continue analyzing is the database of home prices in Albuquerque, 1993. The variables present are `Price`; `SqFt`: the area of a house; `FEATS`: number of features such as dishwasher, refrigerator and so on; `Taxes`: annual taxes paid for the house. Explore the file `prices1.xls`.

1. Plot the histogram of `Price`. Does it remind any conventional distribution? Compute the mean price.

2. Estimate the distribution of the mean price of the house using bootstrap. Determine the bootstrap bias–correction and the variance of the mean price. Compute a 95% confidence interval for the mean price using bootstrap percentile, bootstrap BCa, and first–order normal approximation
   (**Hint**: use `boot()`,`boot.ci()`,`plot.boot()`,`print.bootci()`)

3. Estimate the variance of the mean price using the jackknife and compare it with the bootstrap estimate

4. Compare the confidence intervals obtained with respect to their length and the location of the estimated mean in these intervals.

Question 1: Genetic algorithm

In this assignment, you will try to perform one-dimensional maximization with the help of a genetic algorithm.

1. Define the function
$$f(x) := \frac{x^2}{e^x} - 2\exp(-(9\sin x)/(x^2+x+1))$$
2. Define the function `crossover()`: for two scalars x and y it returns their "kid" as $(x+y)/2$.
3. Define the function `mutate()` that for a scalar x returns the result of the integer division $x^2 \mod 30$. (Operation mod is denoted in `R` as `%%`).
4. Write a function that depends on the parameters `maxiter` and `mutprob` and:
   (a) Plots function f in the range from 0 to 30. Do you see any maximum value?
   (b) Defines an initial population for the genetic algorithm as $X = (0, 5, 10, 15, \ldots, 30)$.
   (c) Computes vector `Values` that contains the function values for each population point.
   (d) Performs `maxiter` iterations where at each iteration
      i. Two indexes are randomly sampled from the current population, they are further used as parents (use `sample()`).
      ii. One index with the smallest objective function is selected from the current population, the point is referred to as victim (use `order()`).
      iii. Parents are used to produce a new kid by crossover. Mutate this kid with probability `mutprob` (use `crossover()`, `mutate()`).
      iv. The victim is replaced by the kid in the population and the vector `Values` is updated.
      v. The current maximal value of the objective function is saved.
   (e) Add the final observations to the current plot in another colour.
5. Run your code with different combinations of `maxiter`$= 10, 100$ and `mutprob`$= 0.1, 0.5, 0.9$. Observe the initial population and final population. Conclusions?

Question 2: EM algorithm

The data file `physical.csv` describes a behavior of two related physical processes $Y = Y(X)$ and $Z = Z(X)$.

1. Make a time series plot describing dependence of Z and Y versus X. Does it seem that two processes are related to each other? What can you say about the variation of the response values with respect to X?
2. Note that there are some missing values of Z in the data which implies problems in estimating models by maximum likelihood. Use the following model
$$Y_i \sim \exp(X_i/\lambda), \qquad Z_i \sim \exp(X_i/(2\lambda))$$
where λ is some unknown parameter.
The goal is to derive an EM algorithm that estimates λ.
3. Implement this algorithm in `R`, use $\lambda_0 = 100$ and convergence criterion "stop if the change in λ is less than 0.001". What is the optimal λ and how many iterations were required to compute it?
4. Plot $\mathrm{E}[Y]$ and $\mathrm{E}[Z]$ versus X in the same plot as Y and Z versus X. Comment whether the computed λ seems to be reasonable.