STATS782 Assignment2

YujieZhang

August 24, 2019

Question 1 [10 marks] Write two versions of an R function: (a) Using a double for loop (for the sum and product, respectively). [5 marks]

```
lagrange = function(a,x,y)
{
    result = y[1]*Li(a,1,x)
    for (i in 2:length(x))
    {
      result = result + y[i]*Li(a,i,x)
    }
  signif(result,5)
}

Li = function(a,i,x)
{
  index = 1:length(x)
  p = 1
  for (k in index[index!=i])
  {
    p = p*((a-x[k])/(x[i]-x[k]))
  }
  p
}
alpha = 0.01
df = c(4^(0:3), Inf)
xi = 1 / df
yi = qt(alpha, df, lower.tail=FALSE)
lagrange(1/20, xi, yi)
```

```
## [1] 2.528
```

```
lagrange(1/80, xi, yi)
```

```
## [1] 2.3739
```

b. Using a single for loop (for the sum only). [5 marks]

```
lagrange = function(a,x,y)
{
  index = 1:length(x)
  r = 0
    for (i in 1:length(x))
    {
      subset = index[index!=i]
      c = cumprod((a-x[subset])/(x[i]-x[subset]))[length(subset)]
      r = r + y[i]*c
    }
  signif(r,5)
}

alpha = 0.01
df = c(4^(0:3), Inf)
xi = 1 / df
yi = qt(alpha, df, lower.tail=FALSE)
lagrange(1/20, xi, yi)
```

```
## [1] 2.528
```

```
lagrange(1/80, xi, yi)
```

```
## [1] 2.3739
```

Question 2 [15 marks] (a) Modify the nearest() function given in the coursebook (and rename it knn1()) so that it predicts the class labels of new observations, as given in a vector x. Use your function to predict the diagnostic outcomes for x = -4:13. [5 marks]

```
knn1 = function(test_x,train_dt)
{
  train_x = train_dt$x
  nearest_x = train_dt$x[apply(abs(outer(train_x,test_x,"-")),2,which.min)]
  nearest_y = train_dt[match(nearest_x,train_dt$x),]$diagnose
  return(nearest_y)
}

train_dataset <- read.csv(file = "cancer.csv", header = TRUE)
x = -4:13
knn1(x, train_dataset)
```

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

b. Modify knn1 (and rename it knn()) so that it works for any k ≥ 1 nearest neighbours,with k being an optional argument with a default value 1.Demonstrate that your function works properly for those x-values given in Part (a), with k = 1, 3, 5, 7, 51, 101, respectively.[10 marks]

```
near = function(v,k)
{
  sorted_v = sort(v, decreasing = FALSE)
  return(sorted_v[1:k])
}

knn = function(test_x,train_dt,k)
{
  train_x = train_dt$x
  distance = abs(outer(train_x,test_x,"-"))
  dataf = data.frame(train_x,distance)
  result = c()
  for (i in 1:length(test_x))
  {
    nearest = near(dataf[,i+1],k)
    nearest_x = dataf[is.element(dataf[,i+1],nearest),]$train_x
    nearest_y = train_dt[match(nearest_x,train_dt$x),]$diagnose
    label1 = sum(nearest_y == "1")
    label2 = sum(nearest_y == "2")
    if(label1 > label2){
      result[i] <- 1
    }else{
      result[i] <- 2
    }

  }
  result
}

train_dataset <- read.csv(file = "cancer.csv", header = TRUE)
x = -4:13
knn(x, train_dataset, 1)
```

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

```
knn(x, train_dataset, 3)
```

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

```
knn(x, train_dataset, 5)
```

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

```
knn(x, train_dataset, 7)
```

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

```
knn(x, train_dataset, 51)
```

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

```
knn(x, train_dataset, 101)
```

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

Question 3 [10 marks] Answer:

```
S1 = function(x, p, log) {
    n = sum(x)
    p0 = 1-sum(p)
    r1 = sum(log((n-1):1))
    r2 = sum(log((x[1]-1):1))
    r3 = x[1]*log(p0)
    r4 = sum(x[-1]*log(p))
    r5 = sum(sapply(x[-1],function(i) sum(log(i:1))))
    result = r1 - r2 +r3 + r4 - r5

    ifelse(log,result,exp(result))
}

x = 4:1*100
p = 3:1*0.1
S1(x, p, log = FALSE)
```

```
## [1] 1.636691e-05
```

```
x = 10:1*100
p = 9:1*0.02
S1(x, p, log = FALSE)
```

```
## [1] 7.706319e-90
```

```
x = 10:1*500
p = 9:1*0.02
S1(x, p, log = FALSE)
```

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

```
x = 4:1*100
p = 3:1*0.1
S1(x, p, log = TRUE)
```

```
## [1] -11.02025
```

```
x = 10:1*100
p = 9:1*0.02
S1(x, p, log = TRUE)
```

```
## [1] -205.1906
```

```
x = 10:1*500
p = 9:1*0.02
S1(x, p, log = TRUE)
```

```
## [1] -888.1959
```

Question 4 [15 marks] (a) Find the maximum likelihood estimates of the three parameters, α, β and δ, of the generalized Gamma distribution, for modelling the protein data, using optim() with methods “Nelder-Mead” and “BFGS”, respectively. [10 marks]

```
density_func = function(x, a, b, c)
{
   (b/a^c)*x^(c-1)*exp(1)^(-(x/a)^b)/gamma(c/b)
}

ll = function(x, theta)
{
  sum(log(density_func(x,theta[1],theta[2],theta[3])))
}
file = "protein.txt"
protein_data = read.delim(file, header = FALSE, sep = "\t", dec = ".")

#The initial value starts from 1,1,1
z = optim(c(1,1,1), ll, control = list(fnscale=-1), method = "Nelder-Mead", x = pr
otein_data, hessian = FALSE)
z
```

```
## $par
## [1] 6.122041 4.980432 8.345096
##
## $value
## [1] -625.0354
##
## $counts
## function gradient
##      502       NA
##
## $convergence
## [1] 1
##
## $message
## NULL
```

```
#The initial value starts from the value which are nearest to the actual value
z = optim(c(6,5,8), ll, control = list(fnscale=-1), method = "Nelder-Mead", x = pr
otein_data, hessian = FALSE)
z
```

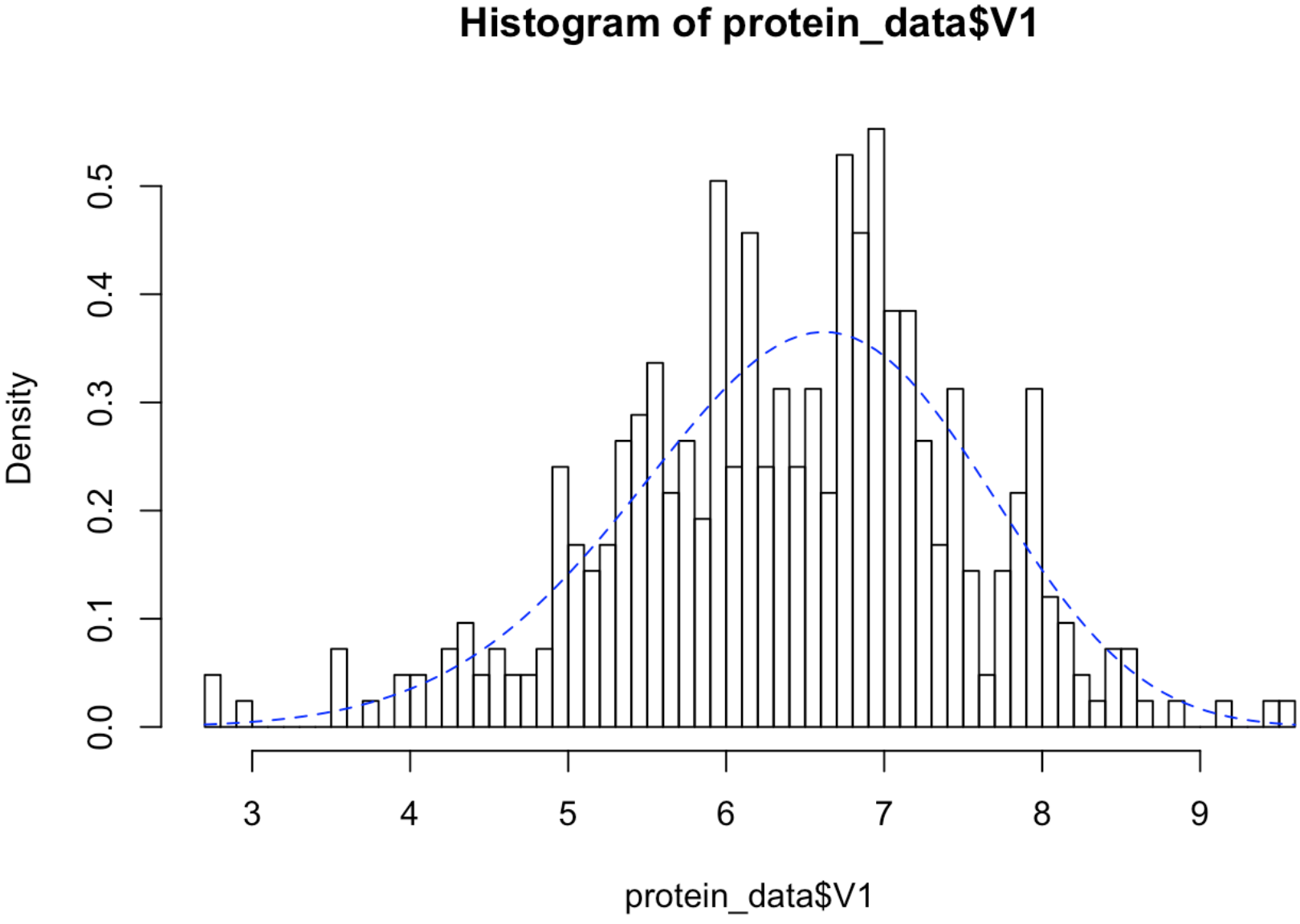
```
## $par
## [1] 6.128213 4.991586 8.333344
##
## $value
## [1] -625.0352
##
## $counts
## function gradient
##       66       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
z = optim(c(6,5,8), ll, control = list(fnscale=-1), method = "BFGS", x = protein_d
ata, hessian = FALSE)
z
```

```
## $par
## [1] 6.129261 4.993069 8.330837
##
## $value
## [1] -625.0352
##
## $counts
## function gradient
##       20        7
##
## $convergence
## [1] 0
##
## $message
## NULL
```

b. Produce a histogram as a density for the protein levels of all patients, with breaks=50, and superimpose it with the density curve of your estimated generalized Gamma distribution. The density curve should be shown in some non-solid line type and non-black colour. [5 marks]

```
hist(protein_data$V1,breaks=50,freq=FALSE)
d = seq(min(protein_data), max(protein_data), len=100)
lines(d,density_func(d,z$par[1],z$par[2],z$par[3]),col="blue",lty="dashed")
```



Bonus Question [5 marks] Write an efficient, vectorised R function for the Lagrange interpolation function L(x), which is described in Question 1. All three arguments, x, xi and yi, can be vectors.

```
lagrange = function(a,x,y)
{
  index = 1:length(x)
  result = c()
  r = 0
  for (j in 1:length(a))
  {
    for (i in 1:length(x))
    {
      subset = index[index!=i]
      c = cumprod((a[j]-x[subset])/(x[i]-x[subset]))[length(subset)]
      r = r + y[i]*c
    }
    result = c(result, signif(r,5))
  }
  return(result)
}

alpha = 0.01
df = c(2^(0:7), Inf)
xi = 1 / df
yi = qt(alpha, df, lower.tail=FALSE)
a = 1/c(1,5,10,16,50,Inf)
lagrange(a, xi, yi)
```

```
## [1] 31.821 35.185 37.949 40.533 42.936 45.262
```

```
system.time(lagrange(a, xi, yi))
```

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