Robust Statistics: Assignment-1

Ramit Nandi | MD2211

24th March 2024

Contents

Problem 1.

Simulate data from $N(0,1)$. Using the $N(\mu, \sigma^2)$ model, we want to estimate μ and σ^2. Use sample sizes $n = 20, 50, 100$ and a replication number $r = 1000$. Report the 'bias' and 'MSE' of your estimates. Use the following methods:

1. **Maximum likelihood.**

2. **Robust estimators of location and scale: $\hat{\mu}$ = median and $\hat{\sigma} = \frac{median_i|X_i - medianX|}{0.6743}$.**

3. **Minimum Hellinger Distance estimator with and without model smoothing (using Gaussian Kernel).**

Solution:

- We note that the maximum likelihood estimates of μ and σ^2 are as follows:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad \hat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{\mu})^2 \tag{1}$$

- We obtain the **Minimum Hellinger Distance estimator (without model smoothing)** by minimizing $\int(\sqrt{f_n^*} - \sqrt{f_\theta})^2$ which is equivalent to maximizing $\int \sqrt{f_n^*}\sqrt{f_\theta}$, where $\sqrt{f_n^*}$ is the **Gaussian kernel density estimate** obtained from the simulated data with optimal bandwidth of the order $n^{-1/5}$ (ideally bandwidth $h_n = s_n \times n^{-1/5}$, where s_n is a robust estimator of scale).

- In order to obtain the **model smoothed version**, we replace f_θ by f_θ^*, which is obtained by **smoothing the model with the same Gaussian Kernel** (with bandwidth h_n) as discussed above. Note that f_θ^* can be evaluated as:

$$f_\theta^*(x) = \frac{1}{\sqrt{2\pi}.\sqrt{\sigma^2 + {h_n}^2}} \exp[-\frac{1}{\sigma^2 + {h_n}^2}(x-\mu)^2] \tag{2}$$

- In both cases, we use robust estimators of location and scale, $\hat{\mu}$ = median and $\hat{\sigma} = \frac{median_i|X_i - medianX|}{0.6743}$ respectively as when required.

Our findings have been summarized in the tables below:

Table 1: Bias for μ over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	0.1758793	0.2153541	0.1782796	0.1782796
50	0.1108752	0.1434556	0.1114144	0.1114133
100	0.0767823	0.0969779	0.0771649	0.0771636

Table 2: MSE for μ over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	0.0498066	0.0739439	0.0509318	0.0509318
50	0.0188907	0.0308854	0.0190942	0.0190938
100	0.0091254	0.0151291	0.0092059	0.0092056

Table 3: Bias for σ^2 over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	0.2446011	0.3843490	0.3507830	0.2485934
50	0.1538576	0.2552180	0.2247819	0.1591988
100	0.1134510	0.1831245	0.1663088	0.1150031

Table 4: MSE for σ^2 over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	0.0918902	0.2324974	0.2042583	0.0940228
50	0.0370536	0.1023279	0.0813821	0.0394686
100	0.0198181	0.0520142	0.0429634	0.0202436

COMMENT: Since **no contamination is present**, the **Maximum Likelihood Estimatior** has the **least bias and MSE** among the four. The Minimum Hellinger Distance Estimator performs better than the Robust Estimator in this situation, but it is not as efficient in performance as the MLE. Error decreases with increasing sample sizes.

The R code for the simulation is displayed below:

```
n_samples = c(20,50,100)
n_itr = 1000
set.seed(101)
library(dplyr)
```

```r
Bias_mu = data.frame(matrix(nrow = 3,ncol = 4),row.names = n_samples)
MSE_mu = data.frame(matrix(nrow = 3,ncol = 4),row.names = n_samples)
Bias_sigma.square = data.frame(matrix(nrow = 3,ncol = 4),row.names = n_samples)
MSE_sigma.square = data.frame(matrix(nrow = 3,ncol = 4),row.names = n_samples)
  # 4columns for 4 type of estimators
names(Bias_mu) = names(MSE_mu) =
  names(Bias_sigma.square) = names(MSE_sigma.square) =
  c("MLE","Robust","MHD_unsmooth","MHD_smooth")


DATA = matrix(rnorm(100*n_itr),nrow=n_itr,byrow=T)
  # generating data all at once
  # each row will be used for one iteration
  # for sample size 20,50 only first 20,50 columns will be extracted


Helli.Dist_approx = function(theta,data,h,s=0)
# theta :(loc,var) , s :smoothness
# integration is approximated upto a multiplicative constant,
# by sum of area of rectangles
# over a large number of equally spaced points from the support of the density
{
  f_n = density(data, kernel = "gaussian", bw = h)
  f_theta = dnorm(f_n$x, mean = theta[1], sd = sqrt(theta[2]+s^2))
  return(sum((sqrt(f_n$y)-sqrt(f_theta))^2))
##NOTE: approximation can be improved using quadratic approximation of integration
 # via stats::integrate() function , but that takes more run-time
}


#### SIMULATION ========================
mu = 0
sigma = 1
One_iteration = function(data)
{
  mle_mu  = mean(data)
  rob_mu  = median(data)
  mle_square.sigma = mean((data - mle_mu)^2)
  rob_sigma = median(abs(data-rob_mu))/0.6743

  h_n =  rob_sigma*n^(-1/5)
  mhd_unsmooth = optim(c(mu,sigma^2),
                       function(theta) Helli.Dist_approx(theta,data,h_n))$par
  mhd_smooth = optim(c(mu,sigma^2),
                       function(theta) Helli.Dist_approx(theta,data,h_n,s=h_n))$par
```

```
  returnValue(c(mle_mu,rob_mu,mhd_unsmooth[1],mhd_smooth[1],
                mle_square.sigma,rob_sigma^2,mhd_unsmooth[2],mhd_smooth[2]))
}


for(idx in 1:length(n_samples))
{## THE MAIN COMPUTATION CHUNK .....
  n = n_samples[idx]
  Data = DATA[,1:n]
  Estimates = apply(Data,MARGIN=1,
                    FUN=One_iteration) %>% t()
  saveRDS(Estimates,paste0("Estimates_",n,".rds"))
    # each row corresponds to one iteration
    # first 4column are for mu, last 4columns are for sigma^2
  mu_hat = Estimates[,1:4]
  square.sigma_hat = Estimates[,5:8]

  Bias_mu[idx,] = abs(mu_hat-mu) %>% apply(MARGIN=2,mean)
  MSE_mu[idx,] = (mu_hat-mu)^2 %>% apply(MARGIN=2,mean)

  Bias_sigma.square[idx,] = abs(square.sigma_hat-sigma^2) %>% apply(MARGIN=2,mean)
  MSE_sigma.square[idx,] = (square.sigma_hat-sigma^2)^2 %>% apply(MARGIN=2,mean)

}

saveRDS(Bias_mu,"Bias_mu.rds")
saveRDS(MSE_mu,"MSE_mu.rds")
saveRDS(Bias_sigma.square,"Bias_sigma.square.rds")
saveRDS(MSE_sigma.square,"MSE_sigma.square.rds")
```

Problem 2.

Look at the following real life data sets.

(a) **Newcomb's Light Speed Data**

(b) **Short's Data**

(c) **Telephone Fault Data**

For each of these examples, use the model 'smoothed' and 'non-smoothed' versions to find estimates of μ and σ^2 under the Normal model. Also report the MLEs.

Solution:

(a): Newcomb's Light Speed Data

The dataset, which consists of 66 of **Newcomb's measurements** of the *passage of light* (in $\times 10^{-3} + 24.8$ millionths of a second) is given below:

Data Set 1 (n=20)				Data Set 2 (n=20)				Data Set 3 (n=26)				
28	-44	29	30	24	28	37	32	36	27	26	28	29
26	27	22	23	20	25	25	36	23	31	32	24	27
33	16	24	29	36	21	28	26	27	27	32	25	28
24	40	21	31	32	28	26	30	27	26	24	32	29
34	-2	25	19	36	29	30	22	28	33	39	25	16
												23

Assuming a $N(\mu, \sigma^2)$ model, we use the **model smoothed** and **unsmoothed** Minimum Hellinger Distance estimators to estimate μ and σ^2. Particulars and properties of the estimators have been expounded upon in the previous problem. Our findings have been presented in the table below.

Table 5: Findings for Newcomb's Light Speed Data

	MLE	MHD_unsmooth	MHD_smooth
mu	26.66667	27.69772	27.69760
square.sigma	130.58586	29.24131	25.53599

Comment: The Model smoothed and unsmoothed estimators seem to give more reliable estimates of location and scale than the MLE.

(b): Short's Data

The dataset, which consists of 17 of **Short's** determinations of the *parallax of the sun* (in seconds of a degree) based on transits of Venus, is given below:

8.65	7.8	9.87
8.35	7.71	8.86
8.71	8.3	5.76
8.31	9.71	8.44
8.36	8.5	8.23
8.58	8.28	-

Assuming a $N(\mu, \sigma^2)$ model, we use the **model smoothed** and **unsmoothed** Minimum Hellinger Distance estimators to estimate μ and σ^2. Particulars and properties of the estimators have been expounded upon in the previous problem. Our findings have been presented in the table below.

Table 7: Findings for Short's Parallax Data

	MLE	MHD_unsmooth	MHD_smooth
mu	8.3776471	8.4347339	8.4347939
square.sigma	0.7149356	0.2168283	0.1825466

Comment: The Model smoothed and unsmoothed estimators seem to give marginally more reliable estimates of location and scale than the MLE.

(c): Telephone Fault Data

The dataset, which consists of *Inverse Fault-Rate Differences for telephone lines* in 14 pairs of areas, is given below:

-988	-78	59	93	189	204	269
-135	3	83	110	197	229	310

Assuming a $N(\mu, \sigma^2)$ model, we use the **model smoothed** and **unsmoothed** Minimum Hellinger Distance estimators to estimate μ and σ^2. Particulars and properties of the estimators have been expounded upon in the previous problem. Our findings have been presented in the table below.

Table 9: Findings for Telephone Fault Data

	MLE	MHD_unsmooth	MHD_smooth
mu	38.92857	118.224	118.2282
square.sigma	96243.78061	24059.313	16326.8666

Comment: The Model smoothed and unsmoothed estimators seem to give more reliable estimates of location and scale than the MLE.

The R code for the simulation is as given below:

```
load("Qn2.RData")
  # it contains the readings from three mentioned datasets
DATA = list('newcomb'=newcomb,
            'short'=short,
            'tele_line'=tele_line)

for(d in names(DATA)){
  Data = DATA[[d]]
  n = length(Data)

  mle_loc = mean(Data)
  mle_sc  = mean((Data - mle_loc)^2)
  mle = c(mle_loc,mle_sc)

  h_n =  (median(abs(Data-median(Data)))/0.6743)*n^(-1/5)
   # Helli.Dist_approx function from Qn1
   # passing 'mle' as the initial guess for numerical optimization
  mhd_unsmooth = optim(mle,
                       function(theta) Helli.Dist_approx(theta,Data,h_n))$par
  mhd_smooth = optim(mle,
                       function(theta) Helli.Dist_approx(theta,Data,h_n,s=h_n))$par

  df.output = data.frame(MLE=mle,
                         MHD_unsmooth=mhd_unsmooth,
                         MHD_smooth=mhd_smooth,
                         row.names = c('mu','square.sigma'))
  saveRDS(df.output,paste0('output_',d,'.rds'))
}
```

Problem 3.

Generate data from $(1-\epsilon)N(0,1) + \epsilon N(8,1)$. Choose $\epsilon = 0.05, 0.1$. Repeat the exercise in Problem 1.

Solution:

We have to generate data from the **contaminated distribution** $(1-\epsilon)N(0,1) + \epsilon N(8,1)$. To do this:

- We randomly generate an observation $X \sim Ber(\epsilon)$ distribution (where $\mathbb{P}(X = 1) = \epsilon$).
- If $X = 1$, we sample from $N(8,1)$ distribution. Otherwise, we sample from $N(0,1)$ distribution.

The R code for the simulation is identical as that in Problem 1. Only the data generating process, being different, can be executed by the following code:

```
genetate.contamination = function(eps,n){
  id = rbinom(n,size=1,prob=eps)
    # indicator for contamination
    # TRUE with probability eps, FALSE otherwise
  population_0 = rnorm(n)
  ## Note: sampling x~N(8,1) is same as sampling x'+8 ,x'~N(0,1)
   # So in this special case we don't need to sample again from population_1
   # simply use 8+population_0 where needed
  ## It saves time
  return(population_0 + 8*id)
}


###  For contamination probability 0.05 =======

# DATA = matrix(genetate.contamination(0.05,100*n_itr),nrow=n_itr,byrow=T)
# other compuation same as Qn1 ...
# ...

###  For contamination probability 0.1 =======

# DATA = matrix(genetate.contamination(0.1,100*n_itr),nrow=n_itr,byrow=T)
# other compuation same as Qn1 ...
# ...
```

We now repeat the tasks in Problem 1. Our results have been summarized in the tables below:

For contamination probability 0.05

Table 10: Bias for μ over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	0.4574816	0.2331706	0.1855319	0.1855337
50	0.3966676	0.1520602	0.1128275	0.1128287
100	0.3903069	0.1166583	0.0829256	0.0829238

Table 11: MSE for μ over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	0.3515635	0.0846873	0.0545111	0.0545116
50	0.2290730	0.0368783	0.0198765	0.0198772
100	0.1891834	0.0218099	0.0108457	0.0108454

Table 12: Bias for σ^2 over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	2.832372	0.4410115	0.4016581	0.2745230
50	2.832145	0.3026520	0.2442091	0.1681620
100	2.906429	0.2328957	0.1794409	0.1198351

Table 13: MSE for σ^2 over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	15.217671	0.3567184	0.4514524	0.2468842
50	11.071989	0.1566518	0.0990363	0.0435759
100	9.977884	0.0898846	0.0511873	0.0215496

For contamination probability 0.1

Table 14: Bias for μ over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	0.8132457	0.2690468	0.2120775	0.2120800
50	0.7937241	0.1909160	0.1152286	0.1152306
100	0.7948044	0.1642267	0.0843968	0.0843979

Table 15: MSE for μ over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	0.9554290	0.1166115	0.1013220	0.1013209
50	0.7581997	0.0562175	0.0207293	0.0207296
100	0.6952120	0.0395675	0.0113146	0.0113150

Table 16: Bias for σ^2 over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	5.400246	0.6003985	0.6662464	0.4861944
50	5.532470	0.4129596	0.2832241	0.1782040
100	5.632920	0.3549553	0.2065512	0.1244038

Table 17: MSE for σ^2 over sample sizes

	MLE	Robust	MHD_unsmooth	MHD_smooth
20	40.86203	0.8570303	3.8719186	2.9548189
50	35.33087	0.3133951	0.1381755	0.0509681
100	34.14740	0.2034783	0.0677299	0.0236331

COMMENT: The bias and MSE increases with the increasing contamination. The presence of contamination shows that the **Maximum Likelihood Estimator fails to perform well** (from the standpoint of Bias and MSE), while the **Robust Estimator** and **Minimum Hellinger Distance Estimator perform satisfactorily** under such a situation. Error decreases with increasing sample sizes.

Problem 4.

(a): **Generate data from Poisson(λ=3) distribution, with sample sizes $n = 20, 50, 100$ and replication of 1000. Assume a Poisson(θ) model. Estimate the mean parameter by the following methods:**

1. **Maximum Likelihood**
2. **Minimum Hellinger Distance** $[2\sum(\sqrt{d} - \sqrt{f_\theta})^2]$
3. **Minimum Penalized Hellinger Distance.** $[2\sum_{d>0}(\sqrt{d} - \sqrt{f_\theta})^2 + \sum_{d=0} f_\theta]$
4. **Symmetric Chi-square.** $[\sum \frac{(d-f_\theta)^2}{(d+f_\theta)}]$

Report the bias and MSE in each case. Here d is the relative frequency distribution of the data generated.

(b):**Repeat** (a) **when the data is generated from** $(1-\epsilon)Poi(3) + \epsilon Poi(15)$**. Choose** $\epsilon = 0.05, 0.1$**.**

Solution:

- The **maximum likelihood estimator** of the mean parameter of the Poisson model is the **sample mean**.
- The **Minimum Penalized Hellinger Distance (P-MHD) estimator** can alternatively be expressed as:

$$\hat{\theta}_{PHD} = \arg\min_\theta [2\sum_{d>0}(\sqrt{d} - \sqrt{f_\theta})^2 + \sum_{d=0} f_\theta] = \arg\min_\theta [2\sum_{d>0}(\sqrt{d} - \sqrt{f_\theta})^2 + 1 - \sum_{d>0} f_\theta] \quad (3)$$

- The **Minimum Symmetric Chi-square (MCS) estimator** can be expressed as:

$$\hat{\theta}_{CS} = \arg\min_\theta [\sum \frac{(d-f_\theta)^2}{(d+f_\theta)}] = \arg\min_\theta [\sum_{d>0} \frac{(d-f_\theta)^2}{(d+f_\theta)} + 1 - \sum_{d>0} f_\theta] \quad (4)$$

Our findings have been summarized in the tables below:

(a): Uncontaminated Distribution

Table 18: Bias for λ over sample sizes

	MLE	MHD	P_MHD	MCS
20	0.30585	0.3598802	0.3171410	0.3267560
50	0.19366	0.2167703	0.1979320	0.2001876
100	0.13730	0.1499375	0.1413231	0.1424994

Table 19: MSE for λ over sample sizes

	MLE	MHD	P_MHD	MCS
20	0.1451125	0.2072053	0.1530119	0.1627154
50	0.0580988	0.0730971	0.0607308	0.0624604
100	0.0292692	0.0352377	0.0307407	0.0313238

COMMENT: We see that the **Maximum Likelihood estimator** performs the best when data is generated from a pure, uncontaminated distribution. Error decreases with increasing sample sizes.

The R code for the simulation is displayed below:

```
n_samples = c(20,50,100)
n_itr = 1000
set.seed(101)
library(dplyr)


Bias_lambda = data.frame(matrix(nrow = 3,ncol = 4),row.names = n_samples)
MSE_lambda = data.frame(matrix(nrow = 3,ncol = 4),row.names = n_samples)
  # 4columns for 4 type of estimators
names(Bias_lambda) = names(MSE_lambda) =
  c("MLE","MHD","P_MHD","MCS")


DATA = matrix(rpois(100*n_itr,lambda=3),
              nrow=n_itr,byrow=T)
  # generating data all at once
  # each row will be used for one iteration
  # for sample size 20,50 only first 20,50 columns will be extracted



# Relative Frequency Distribution .....
```

```r
d = function(x)
{
  counts = table(x) %>% data.frame(row.names = T)
  colnames(counts) = 'd'
  return(counts/sum(counts))
}

## Hellinger Distance .....
HD = function(theta,data)
{
  df = d(data)
  df$f = dpois(as.numeric(row.names(df)),theta)
  return((sqrt(df$d)-sqrt(df$f))^2 %>% sum())
  # multiplicative constant 2 does not affect optimization
}


## Penalized Hellinger Distance .....
P_HD = function(theta,data)
{
  df = d(data)
  df$f = dpois(as.numeric(row.names(df)),theta)
  return((2*(sqrt(df$d)-sqrt(df$f))^2 - df$f) %>% sum)
  # additive constant 1 does not affect optimization
}


## Symmetric Chi-Square .....
CS = function(theta,data)
{
  df = d(data)
  df$f = dpois(as.numeric(row.names(df)),theta)
  return(((((df$d-df$f)^2)/(df$d+df$f)) - df$f) %>% sum)
  # additive constant 1 does not affect optimization
}


#### SIMULATION ========================
lambda = 3
One_iteration = function(data)
{
  mle = mean(data)
  mhd = optim(lambda,function(theta) HD(theta,data))$par
  p.mhd = optim(lambda,function(theta) P_HD(theta,data))$par
  mcs = optim(lambda,function(theta) CS(theta,data))$par
  returnValue(c(mle,mhd,p.mhd,mcs))
}
```

```r
for(idx in 1:length(n_samples))
{## THE MAIN COMPUTATION CHUNK .....
  n = n_samples[idx]
  Data = DATA[,1:n]
  Estimates = apply(Data,MARGIN=1,
                    FUN=One_iteration) %>% t()
  saveRDS(Estimates,paste0("Estimates_",n,".rds"))
    # each row corresponds to one iteration

  Bias_lambda[idx,] = abs(Estimates-lambda) %>% apply(MARGIN=2,mean)
  MSE_lambda[idx,] = (Estimates-lambda)^2 %>% apply(MARGIN=2,mean)

}

saveRDS(Bias_lambda,"Bias_lambda.rds")
saveRDS(MSE_lambda,"MSE_lambda.rds")
```

(b): Contaminated Distributions

We have to generate data from the **contaminated distribution** $(1-\epsilon)Poi(3) + \epsilon Poi(15)$. To do this:

- We randomly generate an observation $X \sim Ber(\epsilon)$ distribution (where $\mathbb{P}(X = 1) = \epsilon$).
- If $X = 1$, we sample from $Poi(15)$ distribution. Otherwise, we sample from $Poi(3)$ distribution.

```r
genetate.contamination = function(eps,n){
  id = rbinom(n,size=1,prob=eps)
    # indicator for contamination
    # TRUE with probability eps, FALSE otherwise
  return(sapply(id,FUN = function(t) ifelse(t,
                                    rpois(1,lambda=15),
                                    rpois(1,lambda=3))))

}


###  For contamination probability 0.05 =======

# DATA = matrix(genetate.contamination(0.05,100*n_itr),nrow=n_itr,byrow=T)
# other compuation same as Qn4.a ...
# ...

###  For contamination probability 0.1 =======
```

```
# DATA = matrix(genetate.contamination(0.1,100*n_itr),nrow=n_itr,byrow=T)
# other compuation same as Qn4.a ...
# ...
```

We now repeat the tasks in Problem 4(a). Our results have been summarized in the tables below:

Contamination probability = 0.05

Table 20: Bias for λ over sample sizes

	MLE	MHD	P_MHD	MCS
20	0.69685	0.4112275	0.3260537	0.3280293
50	0.59390	0.2369454	0.2057736	0.2075437
100	0.58541	0.1662247	0.1459412	0.1421681

Table 21: MSE for λ over sample sizes

	MLE	MHD	P_MHD	MCS
20	0.8461625	0.2790042	0.1680657	0.1683973
50	0.5258580	0.0924363	0.0683191	0.0682215
100	0.4353387	0.0441298	0.0337631	0.0318456

Contamination probability = 0.1

Table 22: Bias for λ over sample sizes

	MLE	MHD	P_MHD	MCS
20	1.25820	0.4904386	0.3503340	0.3389924
50	1.19938	0.2839894	0.2219719	0.2121750
100	1.19701	0.1980500	0.1653847	0.1488560

Table 23: MSE for λ over sample sizes

	MLE	MHD	P_MHD	MCS
20	2.325415	0.4330995	0.1988726	0.1843743
50	1.760667	0.1296449	0.0805939	0.0721578
100	1.602261	0.0631294	0.0453733	0.0370421

COMMENT: The bias and MSE increases with the increasing contamination. The presence of contamination shows that the **Maximum Likelihood Estimator fails to perform**

well (from the standpoint of Bias and MSE), while the **Minimum Hellinger Distance Estimator perform satisfactorily** under such a situation. Error decreases with increasing sample sizes.