EMAT30007 Applied Statistics

Lab 2: Parameter estimation

This lab focusses on the material covered in Lecture 2: Methods to estimate the parameter values or quantities of a probability distribution.

1. Maximum Likelihood (ML)

ML estimate for the parameter of a Bernoulli distribution

Estimate the parameter p of a Bernoulli distribution from a sample of $n = 10$ observations using the ML method:

1. Generate a sample x of size $n = 10$ from a Bernoulli distribution with parameter $p = 0.5$ (like in Lab 1, you can use `makedist`).
2. Compute and plot the Likelihood function for the sample x, $L(p; x) = \prod_{i=1}^{n} P_X(x_i; p)$.
3. Compute and plot the log-Likelihood function for the sample x, $logL(p) = \log(L(p)) = \sum_{i=1}^{n} \log P_X(x_i; p)$.
4. Use Matlab's `fminsearch` to find p_L, the value of p that maximises $L(p)$, and p_{logL}, the value of p that maximises $logL(p)$.
5. Compute $\hat{p}_{MLE} = \frac{1}{n}\sum_i x_i$ the ML estimate for the sample x . Does $\hat{p}_{MLE} = p_L = p_{logL}$? In general, $\hat{p}_{MLE} \neq p$, the true parameter value. Why?
6. Increase the sample size $n = 100, 1000, \ldots$ and observe the likelihood functions and the estimates of p. What do you notice?
7. Check that Matlab's `mle` function gives the same result you found.

Solution

```
p_true = 0.75;
pd = makedist('Binomial', 'p', p_true);
x = random(pd, 1, 10);

% define the likelihood for the binomial distribution
likelihood = @(p) prod(p.^x .* (1-p).^(1-x));
% define the log-likelihood for the binomial distribution
loglikelihood = @(p) sum(x .* log(p) + (1-x) .* log(1-p));

% compute likelihood
p = 0: 0.001: 1;
likelihoods = zeros(1, length(p));
loglikelihoods = zeros(1, length(p));
for i = 1:length(p)
    likelihoods(i) = likelihood(p(i));
    loglikelihoods(i) = loglikelihood(p(i));
end

% plot
clf;
yyaxis left
plot(p, likelihoods, '-', 'Linewidth', 2)
ylabel('L') % y-axis label
yyaxis right
plot(p, loglikelihoods, '-', 'Linewidth', 2)
hold on

% Find p that maximises L (i.e., minimises -L)
negL = @(x) likelihood(x) * -1;
p_L = fminsearch(negL, 0)  % initial value of p is 0
```

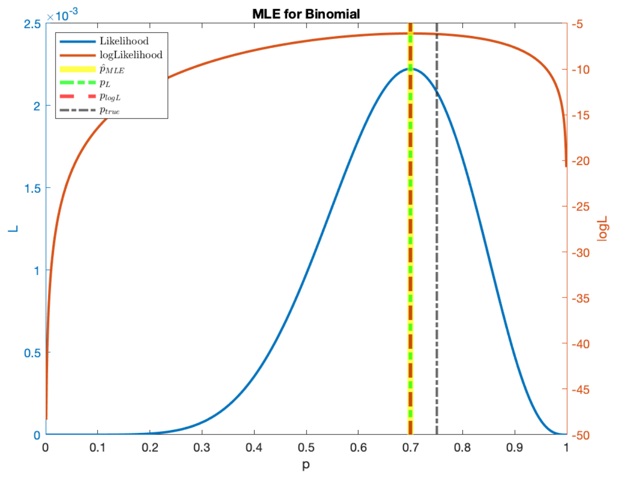
```
p_L = 0.7000
```

```
% Find p that maximises logL (i.e., minimises -logL)
neglogL = @(x) loglikelihood(x) * -1;
% fminsearch(neglogL, 0)
p_logL = fminsearch(neglogL, 0)
```

```
p_logL = 0.7000
```

Some times `fminsearch` may not converge and it returns wrong estimates. Starting from a different initial condition can solve the problem. In other cases this coul be due to numerical errors, for example when the function to be optimised has values beyond the Matlab's numerical precision -- this can happen for Likelihood functions for large samples (why?).

```
p_hat = mean(x);
xline(p_hat, '-y', 'Linewidth', 5);
hold on
xline(p_L, '-.g', 'Linewidth', 3);
hold on
xline(p_logL, '--r', 'Linewidth', 3);
hold on
xline(p_true, '-.', 'Linewidth', 2);


legend('Likelihood', 'logLikelihood', '$\hat{p}_{MLE}$', '$p_{L}$', '$p_{logL}$', '$p_{true}$', 'Location','northwest', 'Interpreter', '
title('MLE for Binomial') % title for plot
xlabel('p') % x-axis label
ylabel('logL') % y-axis label
```



Run the code a few times to see how likelihoods change for different data samples and for different sizes n.

$\hat{p}_{MLE} = p_L = p_{logL}$ because, as explained in the lecture, the likelihood and log-likelihood have the same (arg)maximum ($\log$ is strictly increasing) and $\hat{p}_{MLE}$ is the exact value. $\hat{p}_{MLE} \neq p$ because the sample x is subject to randomness and especially for small n it can have have much higher of fewer 1s (`heads') than expected. As n is increased the average number of 1s tend to the expected true value p (because of the law of large numbers).

Also, when n is large `fminsearch` has problems in finding the correct maximum of the Likelihood because of numerical errors arising from the fact that L , which is the product of many numbers smaller than 1, becames extremely small (e.g. 10^{-23}), beyond Matlab's numerical precision. The $\log L$ instead does not suffer from this problem, so it should be used to numerically estimate the maximim for large n.

```
% Matlab's MLE function
mle(x,'distribution','Bernoulli')
```

```
ans = 0.7000
```

ML estimate for the parameters of a Normal distribution

Follow the same steps as above to estimate the parameters μ and σ of a Normal distribution from a sample of $n = 10$ observations using the ML method:

1. Generate a sample x of size $n = 10$ from a Normal distribution with parameters $\mu = 0$ and $\sigma = 1$.
2. Define the Likelihood and log-Likelihood functions for the sample x as a function of parameters μ and σ (`normpdf(x, mu, sigma)` give the value of the Normal PDF at x).
3. Use Matlab's `fcontour(likelihood, [-1 1 0 2])` to plot the Likelihood function for the sample x, $L(p; x) = \prod_{i=1}^{n} P_X(x_i; p)$.
4. Use Matlab's `fminsearch` to find (μ_L, σ_L), the values of the parameters that maximise $L(\mu, \sigma)$,
5. Compute the ML estimate $(\hat{\mu}_{MLE}, \hat{\sigma}_{MLE})$. Does $(\hat{\mu}_{MLE}, \hat{\sigma}_{MLE}) = (\hat{\mu}_L, \hat{\sigma}_L)$? Why?
6. Increase the sample size $n = 100, 1000, ...$ and observe the likelihood functions and the estimates of p. What do you notice?
7. Check that Matlab's `mle` function gives the same result you found.

Solution

```
mu_true = 0;
sigma_true = 1;
pd = makedist('Normal', 'mu', mu_true, 'sigma', sigma_true);
x = random(pd, 1, 10);


% define the likelihood for the binomial distribution
likelihood = @(mu, sigma) prod(normpdf(x, mu, sigma));
% define the log-likelihood for the binomial distribution
loglikelihood = @(mu, sigma) sum(log(normpdf(x, mu, sigma)));

% plot
clf;
fcontour(likelihood, [-1 1 0 2],'Fill','on','MeshDensity',200);
```

```
Warning: Function behaves unexpectedly on array inputs. To improve performance, properly vectorize your function to return an output with the
same size and shape as the input arguments.
```

```

hold on
plot(mu_true, sigma_true, 'dr', 'MarkerSize', 10, 'MarkerFaceColor',"r")
hold on

% Find p that maximises L (i.e., minimises -L)
negL = @(ms) likelihood(ms(1), ms(2)) * -1;
msL = fminsearch(negL, [0.5, 0.5])
```

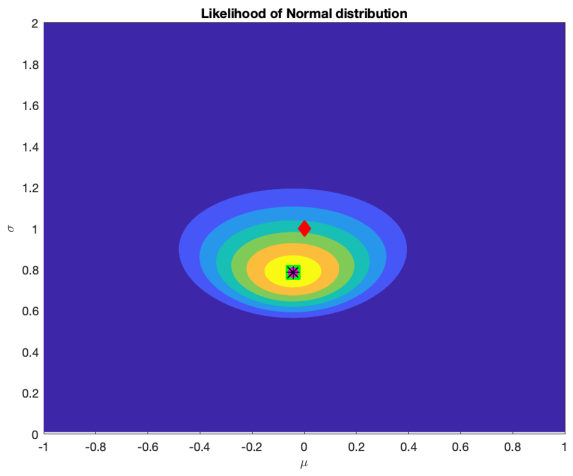
```
msL = 1×2
   -0.0436    0.7855
```

```
plot(msL(1), msL(2), 'sg', 'MarkerSize', 15, 'MarkerFaceColor',"g")
hold on

% Find p that maximises logL (i.e., minimises -logL)
neglogL = @(ms) loglikelihood(ms(1), ms(2)) * -1;
% fminsearch(neglogL, 0)
msllL = fminsearch(neglogL, [0.5, 0.5])
```

```
msllL = 1×2
   -0.0436    0.7855
```

```
plot(msllL(1), msllL(2), 'om', 'MarkerSize', 8, 'MarkerFaceColor',"m")
hold on
```

```
% MLE estimate
plot(mean(x), std(x, 1), '*k', 'MarkerSize', 10, 'MarkerFaceColor',"k")

title('Likelihood of Normal distribution') % title for plot
xlabel('$\mu$', 'Interpreter', 'latex') % x-axis label
ylabel('$\sigma$', 'Interpreter', 'latex') % y-axis label
```



Run the code a few times to see how likelihoods change for different data samples and for different sizes n.

```
% Matlab's MLE function
mle(x,'distribution','Normal')
```

```
ans = 1×2
   -0.0436    0.7855
Warning: Function behaves unexpectedly on array inputs. To improve performance, properly vectorize your function to return an output with the
same size and shape as the input arguments.
```

Compute MLE by hand

The PDF of the continuous RV X is $P_X(x) = \dfrac{\theta}{x^{\theta+1}}$ for $x > 1$ and 0 otherwise.

1. Find $\hat{\theta}_{MLE}$ the Maximum Likelihood estimator of θ analytically. Make sure to check that $\hat{\theta}_{MLE}$ is a maximum of the loglkelihood.
2. Use the code `x = rand(1, n) .^ (-1/theta);` to draw a sample `x` of size `n=10000` from the PDF of X with parameter `theta=3.5`. (Where does this code come from?)
3. Translate the estimator computed in 1. into Matlab code and use it to estimate `theta` for the sample `x` obtained in 2.

Solution

$L(\theta; X) = \prod_{i=1}^{n} \dfrac{\theta}{X_i^{\theta+1}}$ and $\log L(\theta; X) = n \log \theta - (\theta + 1) \sum_{i=1}^{n} \log X_i$

$\dfrac{\partial \log L}{\partial \theta} = \dfrac{n}{\theta} - \sum_{i=1}^{n} \ln X_i = 0$ whose solution is $\hat{\theta}_{MLE} = \dfrac{n}{\sum_{i=1}^{n} \ln X_i}$

Check that $\hat{\theta}_{MLE}$ is a maximum: $\dfrac{\partial^2 \log L}{\partial \theta^2} = -\dfrac{n}{\theta^2} < 0$ for all θ.

```
theta = 3.5;
```

```
Warning: Function behaves unexpectedly on array inputs. To improve performance, properly vectorize your function to return an output with the
same size and shape as the input arguments.
```

```
n = 10000;
x = rand(1, n) .^ (-1/theta);  % this comes from the IPIT (prove it)
n / sum(log(x))
```

```
ans = 3.4598
```

2. Method of Moments (MoM)

MoM estimate for the parameters of a Poisson distribution

1. Calculate $\hat{\lambda}_{MoM}$ the method of moments estimator for the parameter λ of a Poisson distribution with PMF $P_X(x; \lambda) = \dfrac{e^{-\lambda}\lambda^x}{x!}$, with $x = 0, 1, \ldots$ and $\lambda > 0$.
2. Generate a sample x of $n = 10$ observations from a Poisson distribution with parameter $\lambda = 10$ and estimate it using $\hat{\lambda}_{MoM}$, the estimator derived in 1.
3. Consider increasing sample sizes $n = 100, 1000, \ldots$ and for each n draw 1000 samples, compute their 1000 MoM estimates $\hat{\lambda}_{MoM}$, and calculate the estimates' standard deviation $\sigma_{\hat{\lambda}}(n) = \sqrt{E[(\hat{\lambda}_{MoM} - E(\hat{\lambda}_{MoM}))^2]}$. Plot n against $\sigma_{\hat{\lambda}}(n)$ and verify that $\sigma_{\hat{\lambda}}(n) \propto \dfrac{1}{\sqrt{n}}$ i.e. the "spread" of the estimates decreases as the square root of n , as the sample size increases.

Solution

The distribution has one parameter, so we have to compute the first theoretical moment (mean) and equate it to the sample mean $\bar{X}$:

$$E(X) = \sum_{x=0}^{\infty} P_X(x; \lambda) x = \sum_{x=0}^{\infty} \frac{e^{-\lambda}\lambda^x}{x!} x = \sum_{x=1}^{\infty} \frac{e^{-\lambda}\lambda^x}{x!} x = \sum_{x=1}^{\infty} \frac{e^{-\lambda}\lambda^{x-1}}{(x-1)!} \lambda = \sum_{x=0}^{\infty} \frac{e^{-\lambda}\lambda^x}{x!} \lambda = 1 \cdot \lambda = \lambda$$

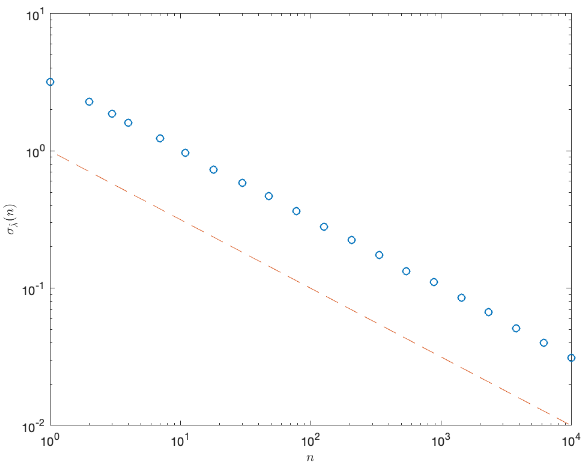
Solving the equation $E(X) = \bar{X}$ for λ we trivially obtain: $\hat{\lambda}_{MoM} = \bar{X} = \dfrac{\sum_{i=1}^{n} X_i}{n}$.

```
% generate Poisson random sample
lambda = 10;
n = 10000;
x = poissrnd(lambda, 1, n);
```

```
% MoM estimate of parameter lambda
mean(x)
```

```
ans = 9.9742
```

```
% increasing sample size
ns = round(logspace(0, 4, 20));

s = 1000;

% compute the standard deviation of the estimates for different sample sizes
stdevs = zeros(1, length(ns));
for i = 1:length(ns)
    n = ns(i);
    estimates = zeros(1, s);
    for j = 1:s
        x = poissrnd(lambda, 1, n);
        estimates(j) = mean(x);
    end
    stdevs(i) = std(estimates);
end

% plot standard deviation (spread) of estimates against the sample size
clf;
plot(ns, stdevs, 'o');
hold on
plot(ns, ns.^(-0.5), '--');
set(gca, 'XScale', 'log');
set(gca, 'YScale', 'log');
xlabel('$n$', 'Interpreter', 'latex') % x-axis label
ylabel('$\sigma_{\hat{\lambda}}(n)$', 'Interpreter', 'latex') % y-axis label
```



3. Unbiased estimators

Consider the PDF $P_X(x;\sigma) = \frac{1}{2\sigma}e^{-\frac{|x|}{\sigma}}$ for $x \in \mathbb{R}$.

1. Check that the PDF is normalised
2. Calculate $\hat{\sigma}_{MLE}$, the Maximum Likelihood estimator of σ.
3. Calculate $\hat{\sigma}_{MoM}$, the Method of Moments estimator of σ.
4. Do the two methods give the same estimator, i.e. $\hat{\sigma}_{MLE} = \hat{\sigma}_{MoM}$?
5. Check numerically if $\hat{\sigma}_{MLE}$ and $\hat{\sigma}_{MoM}$ look ubiased, asymptotically ubiased or biased: draw 20000 samples of various sizes, $n = [5, 10, 100, 1000]$, and plot the histograms and means of the distributions of the estimates for the different n. Is there a better estimator? To generate n random numbers from the PDF $P_X(x;\sigma)$, use the code `x = exprnd(sigma, 1, n) .* (2 * binornd(1, 0.5, 1, n) - 1)` (why?).

Solution

Normalisation

$$\int_{-\infty}^{\infty} \frac{1}{2\sigma}e^{-\frac{|x|}{\sigma}}dx = 2\int_0^{\infty}\frac{1}{2\sigma}e^{-\frac{x}{\sigma}}dx = \frac{2}{2\sigma}\sigma\int_0^{\infty}e^{-y}dy = 1$$

MLE

$$\log L(\theta; X) = \log \prod_i \frac{1}{2\sigma}e^{-\frac{|X_i|}{\sigma}} = \sum_{i=1}^{n}\log\left[\frac{1}{2\sigma}e^{-\frac{|X_i|}{\sigma}}\right] = n\log\frac{1}{2\sigma} - \sum_{i=1}^{n}\frac{|X_i|}{\sigma}$$

$\frac{\partial \log L}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^2}\sum_{i=1}^{n}|x_i| = 0$ whose solution is $\hat{\sigma}_{MLE} = \frac{\sum_{i=1}^{n}|X_i|}{n}$

MoM

The first theoretical moment is zero by symmetry of the PDF around zero, so we have to use the second moment:

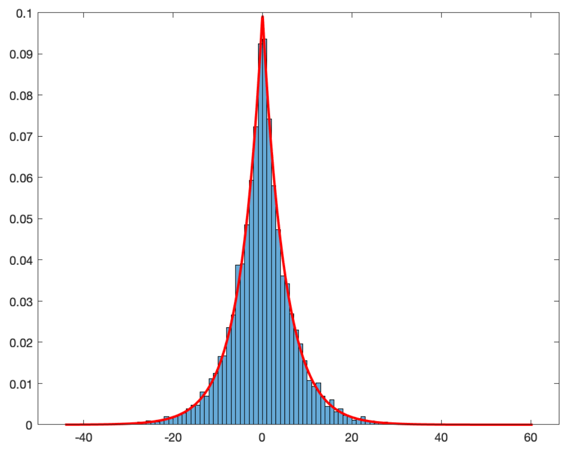
$E(X^2) = \int_{-\infty}^{\infty}\frac{1}{2\sigma}e^{-\frac{|x|}{\sigma}}x^2dx = \frac{1}{\sigma}\int_0^{\infty}e^{-\frac{x}{\sigma}}x^2dx = \frac{1}{\sigma}\sigma^3\int_0^{\infty}e^{-y}y^2dy = 4\sigma^2$, where the last integral is solved integrating twice by parts.

Hence, from the MoM equation $E(X^2) = \frac{\sum_{i=1}^{n}X_i^2}{n}$ we get $\hat{\sigma}_{MoM} = \sqrt{\frac{\sum_{i=1}^{n}X_i^2}{2n}}$.

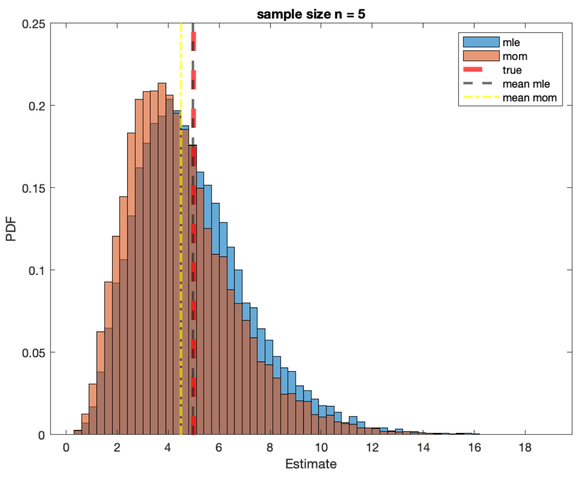
The two estimators are different.

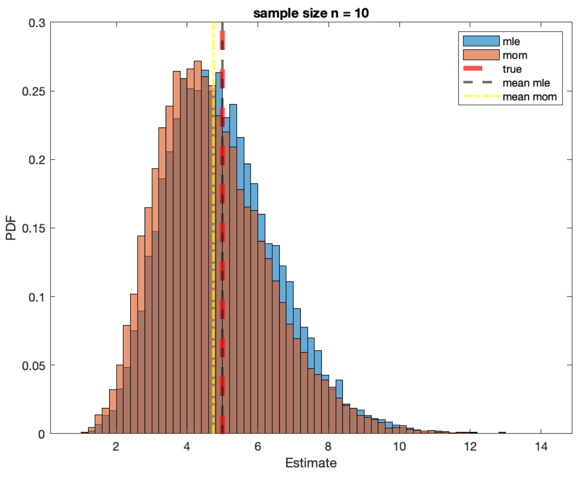
```
% true value
sigma = 5;
```

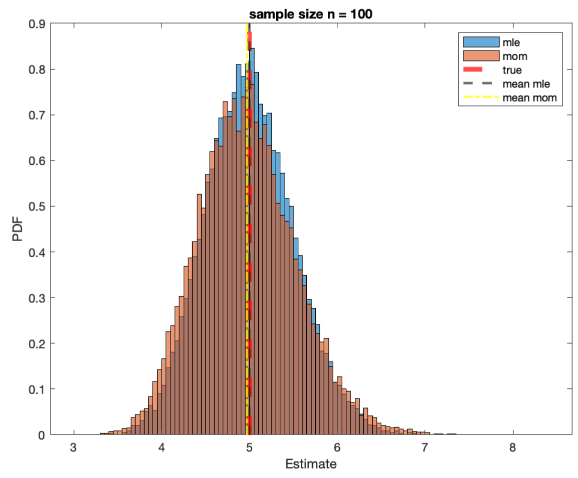
```
% sample
n = 10000;
x = exprnd(sigma, 1, n) .* (2 * binornd(1, 0.5, 1, n) - 1);


% pdf
clf;
histogram(x, 'Normalization',"pdf");
hold on

xx = min(x):0.1:max(x);
plot(xx, exp(- abs(xx) / sigma) / 2 / sigma, '-r', 'LineWidth', 2)
```



```
close all

% estimators
sigma_mle = @(x) sum(abs(x)) / length(x);
sigma_mom = @(x) (sum(x.^2) / 2 / length(x))^0.5;

% estimates distributions
S = 20000;
ns = [5 10, 100];

estimates_mle = zeros(length(ns), S);
estimates_mom = zeros(length(ns), S);

for i = 1:length(ns)
    n = ns(i);
    for j = 1:S
        x = exprnd(sigma, 1, n) .* (2 * binornd(1, 0.5, 1, n) - 1);
        estimates_mle(i, j) = sigma_mle(x);
        estimates_mom(i, j) = sigma_mom(x);
    end
end


% set(gcf, 'Position', [0 0 1200 300])
clf;
for i = 1:length(ns)
    figure(i)
    histogram(estimates_mle(i, :), 'Normalization',"pdf");
    hold on
    histogram(estimates_mom(i, :), 'Normalization',"pdf");
    hold on;
    xline(sigma, '--r', 'Linewidth', 4);
    hold on
    xline(mean(estimates_mle(i, :)), '--', 'Linewidth', 2);
    hold on
    xline(mean(estimates_mom(i, :)), '-.y', 'Linewidth', 2);
    legend('mle', 'mom', 'true', 'mean mle', 'mean mom');
    title(sprintf('sample size n = %d', ns(i)))%, 'Interpreter', 'latex')
    xlabel('Estimate');
    ylabel('PDF');
%     xlim([0 4]);  % uncomment this if you want all plots on the same scale
end
```







The mean of the MLE estimates seems equal to the true value even for small sample sizes, hence it seems an unbiased estimator.

The mean of the MoM estimates is shifted to the left of the true value for small sample sizes and for large sample sizes it tends to the true value, hence it seems an asymptotically unbiased estimator.

The MLE estimator is preferrable, especially for small sample sizes.

4. Comparison of estimators for the mean of non-skewed and skewed distributions

Use simulations to investigate whether the mean or median is a better estimator of the mean (parameter μ) of a Normal distribution. Compare the two estimators by analysing (1) bias and (2) standard deviation (spread) of the estimates.

Solution

Both mean and median are unbiased. The Median estimator has a larger spread (standard deviation of the estimates).

Use simulation to compare the use of the sample mean and median as estimators of the mean (parameter μ) of an Exponential distribution. Again, compare the two estimators by analysing (1) bias and (2) standard deviation (spread) of the estimates.

Solution

The Median is biased. Mean and median estimators have a similar spread (standard deviation of the estimates).

5. Estimate the height of the tallest students

You want to estimate h, the mean height of the top 10% tallest students at the University. In formulae: $h = 10\int_{F_X^{-1}(0.9)}^{\infty} P_X(x)\,x\,dx$ (why?), where P_X and F_X^{-1} are the PDF and quantile function of the hight distribution. To this end, you mesured the height of n students chosen at random. A Q-Q plot (see Lab 1) suggests that your data is compatible with a Normal distribution.

1. Can you think of (at least) two estimators for h?
2. What would you do to decide which estimator to use? Does you answer depend on n ?

(To compute integrals numerically, you can use Matlab's `vpaintegral`)

Solution

Two estimates for h are:

1. $\hat{h}_s = \frac{\sum_{i \in T_{10}(x)} x_i}{|T_{10}(x)|}$ where $T_{10}(x) = \{i | F_X(x_i) > 0.9\}$ is the set of top 10% tallest individuals in the sample (and $|T_{10}(x)|$ is its size). This is the mean height of the top 10% highest students in the data sample.
2. $\hat{h}_N = 10 \int_{F_X^{-1}(0.9;\hat{\mu},\hat{\sigma})}^{\infty} P_X(x;\hat{\mu},\hat{\sigma}) x \, dx$ where $P_X(x;\hat{\mu},\hat{\sigma})$ is the PDF and $F_X^{-1}(0.9;\hat{\mu},\hat{\sigma})$ is the quantile function evaluated at 0.9 of a Normal distribution with mean and standard deviation estimated from the data using the MLE.

To decide which one to use, you could check numerically if they are unbiased and how big is the "spread" of the estimates as a function of the sample size (like in questions 2.3 and 3.5 above):

- To see if estimators are biased we need to know the true value of what we want to estimate: define a true distribution of heights choosing a mean and standard deviation for the Normal distribution and compute h (numerically) using the definition.
- Estimate h numerically using both methods for a large number of samples (1000) and varying n and compute the mean and standard deviation of the samples, for all n
- Decide if the estimators are unbiased checking if the mean of the estimators is equal to the true value h for all n.
- Decide which estimator is more accurate as the one with the smaller standard deviation of the estimators. for the n of interest.

Note that for $n < 10$ the estimator $\hat{h}_s$ cannot be applied, so at least for small samples $\hat{h}_N$ is preferred.

```
true_mean = 167;
true_std = 10;

% True mean of the top 10% of the distribution of heights

m = true_mean;
v = true_std^2;

th_q09 = norminv(0.9, m, v^0.5)
```

```
th_q09 = 179.8155
```

```
syms y(h)
y(h) = 10 * h * exp(-(h - m)^2 / 2 / v) / (2 * pi * v)^0.5;

h_true = vpaintegral(y, th_q09, th_q09 * 10)  % the right extreme of integration should be infinity, but a sufficiently large number is
```

```
h_true = 184.54983319324933166427626929362
```

```
% sample the heights of n students
n = 10;

x = normrnd(true_mean, true_std, 1, n);

% h_s first estimator: the mean height of the top 10% highest student in the sample
% compute the empirical quantile 0.9
q09 = quantile(x, 0.9)
```

```
q09 = 182.5411
```

```
tallest = x(x > q09);
hs = mean(tallest)
```

```
hs = 183.9895
```

```
% h_N second estimator: the mean of the top 10% of the distribution of
% heights, whose parameters are estimated from the sample

m = mean(x)
```

```
m = 166.8365
```

```
v = std(x)^2
```

```
v = 120.1508
```

```
th_q09 = norminv(0.9, m, v^0.5)
```

```
th_q09 = 180.8840
```

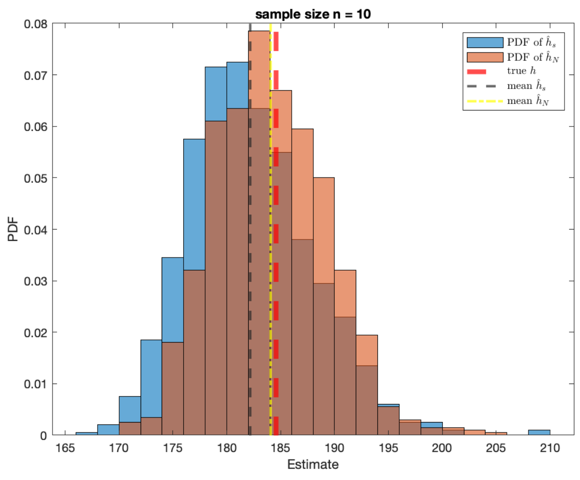
```
syms y(h)
y(h) = 10 * h * exp(-(h - m)^2 / 2 / v) / (2 * pi * v)^0.5;

hN = vpaintegral(y, th_q09, th_q09 * 10)
```

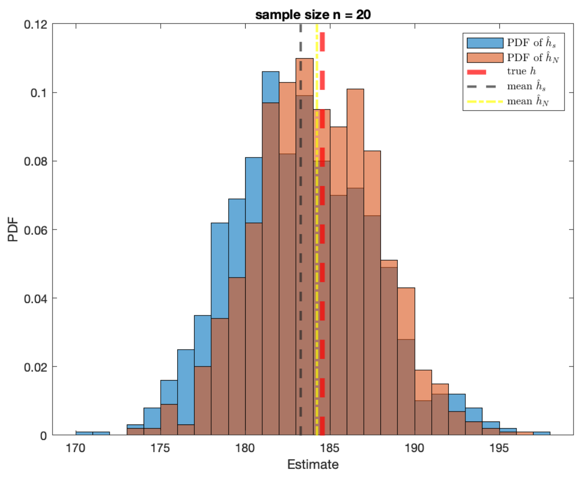
```
hN = 186.073
```

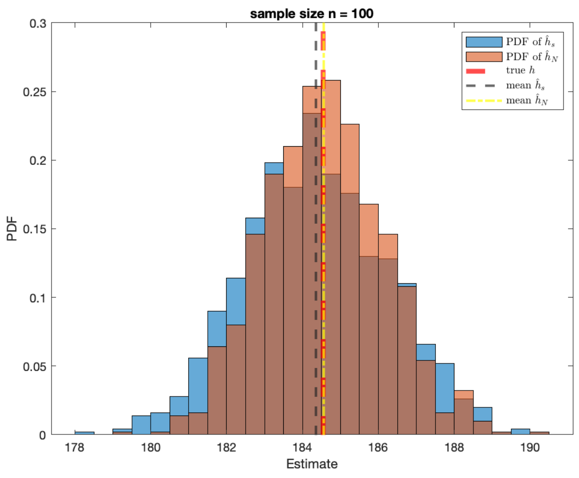
```
% estimates distributions

S = 100;  % fast, but not enough samples
% S = 1000;  % ...this is quite slow!
ns = [10 20, 100];

estimates_hs = zeros(length(ns), S);
estimates_hN = zeros(length(ns), S);

for i = 1:length(ns)
    n = ns(i);
    for j = 1:S
        x = normrnd(true_mean, true_std, 1, n);
        % hs
        q09 = quantile(x, 0.9);
        tallest = x(x > q09);
        hs = mean(tallest);

         % hN
        m = mean(x);
        v = std(x)^2;
        th_q09 = norminv(0.9, m, v^0.5);
        syms y(h)
        y(h) = 10 * h * exp(-(h - m)^2 / 2 / v) / (2 * pi * v)^0.5;
        hN = vpaintegral(y, th_q09, th_q09 * 10);

        estimates_hs(i, j) = hs;
        estimates_hN(i, j) = hN;
    end
end
```

The figures below are computed with `s = 1000` .

```
close all

% set(gcf, 'Position', [0 0 1200 300])
clf;
for i = 1:length(ns)
    figure(i)
    histogram(estimates_hs(i, :), 'Normalization',"pdf");
    hold on
    histogram(estimates_hN(i, :), 'Normalization',"pdf");
    hold on
    xline(184.55, '--r', 'Linewidth', 4);  % h_true doesn't work..?
    hold on
    xline(mean(estimates_hs(i, :)), '--', 'Linewidth', 2);
    hold on
    xline(mean(estimates_hN(i, :)), '-.y', 'Linewidth', 2);
    legend('PDF of $\hat{h}_s$', 'PDF of $\hat{h}_N$', 'true $h$', 'mean $\hat{h}_s$', 'mean $\hat{h}_N$', 'Interpreter', 'latex');
    title(sprintf('sample size n = %d', ns(i)))%, 'Interpreter', 'latex')
    xlabel('Estimate');
    ylabel('PDF');
end
```







Overall $\widehat{h}_N$ seems a better estimator:

- the mean of the estimates is closer to the true value (and seems unbiased)
- the PDF of the estimates is more symmetric around the mean and with less spread (smaller standard deviation)