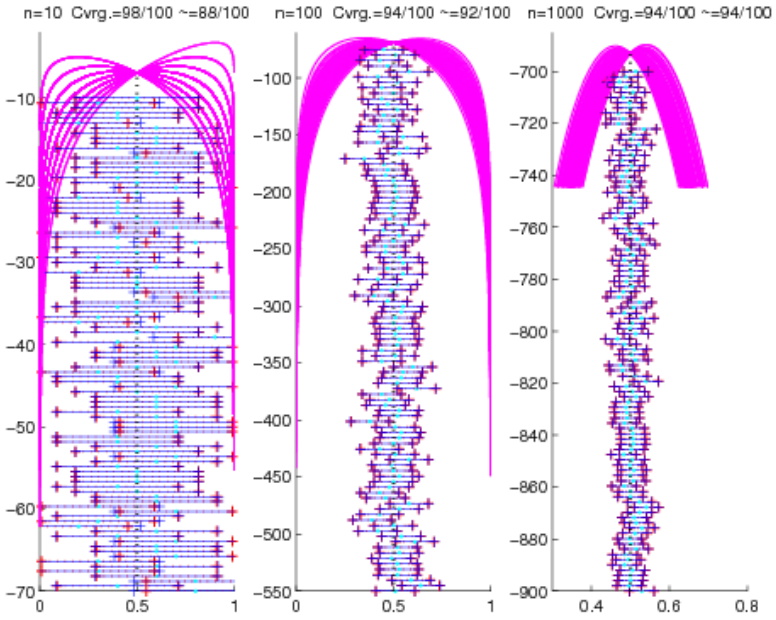
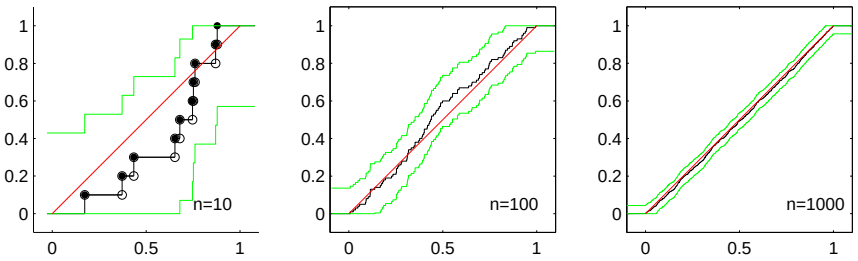
Elements of Simulation & Inference version 0.1

Raazesh Sainudiin

This work was partially supported by NSF grant DMS-03-06497 and NSF/NIGMS grant DMS-02-01037.





Contents

List of Figures

Chapter 1

Introduction and Preliminaries

Lecture Notes for ENCI 303 S1 2008: Engineering Decision Making
Raazesh Sainudiin, Dept. of Maths & Stats, University of Canterbury, Christchurch, NZ.

1.1 Computational Statistical Experiments

A *statistical experimenter* is someone who conducts a *statistical experiment* (for convenience, we will refer to an experimenter and an experiment from this point on). Roughly, an experiment is an action with an *empirically observable outcome* (data) that can not necessarily be predicted with certainty, in the sense that a *repetition* of the experiment may result in a different outcome. Most quantitative scientists are experimenters if they apply statistical principles to further their current understanding (their *theory*) of an *empirically observable real-world phenomenon*. If experimenters want to further their understanding or theory, they need to improve their mathematical models (a.k.a. "rigorous cartoons") of the phenomenon on the basis of the model's compatibility with the *observed data* or *outcome* of the experiment. In this sense, an experimenter attempts to learn about a phenomenon through the outcome of an experiment. An experimenter is often a scientist or engineer, and vice versa.

Technological advances mean that most experiments today involve computers. First, our instrumentational capacity of observing an empirical phenomenon by means of automated data gathering (sensing) and representation (storage and retrieval) is steadily increasing. Second, our computational capability to process statistical information or to make a statistical decision from these possibly massive data-sets is also steadily increasing. Thus, our recent technological advances make it possible for us to perform computationally intensive experiments with massive amounts of empirical observations, in a manner that was not viable a decade ago. Hence, a successful scientist or engineer in most specialisations today is a *computational statistical experimenter*, i.e. a statistical experimenter who understands the information structures used to represent data as well as the statistical algorithms used to process out ones scientific or engineering decisions. These notes are designed to help you take the first steps along this path.

Let us first demonstrate the need for a statistical experiment. Recall that statistical inference or learning is the process of using observations or data to infer the distribution function (DF) that generated it. A generic question is:

$$\text{Given realisations from } X_1, X_2, \ldots, X_n \sim \text{some unknown DF } F, \text{ how do we infer } F?$$

Some of the concrete problems involving experiments include:

- **Simulation:** Often, it is necessary to simulate a random variable (RV) with some specific distribution to gain insight into its features or simulate whole systems, such as the air-traffic queues at Heathrow Airport, to make better management decisions.
- **Estimation:**
  1. **Parametric Estimation:** Using samples from some unknown DF F that is parameterised by some unknown θ, we can estimate θ from a statistic $\widehat{\Theta}_n$ called the estimator of θ using one of several methods (maximum likelihood, moment estimation or parametric bootstrap).
  2. **Non-parametric Estimation of the DF:** Based on n independent and identically distributed (IID) observations from an unknown DF F, we can estimate it under the general assumption that $F \in \{\text{all DFs}\}$.
  3. **Confidence Sets:** We can obtain a $1-\alpha$ confidence set for the point estimates, of the unknown parameter $\theta \in \mathbb{\Theta}$ or the unknown DF $F \in \{\text{all DFs}\}$
- **Hypothesis Testing:** Based on observations from some DF F that is hypothesised as belonging to a subset $\mathbb{\Theta}_0$ of $\mathbb{\Theta}$ called the space of null hypotheses, we will learn to test (attempt to reject) the falsifiable null hypothesis that $F \in \mathbb{\Theta}_0 \subset \mathbb{\Theta}$.

1.2 Expectations and their Properties

We assume that the reader has taken a basic course in probability theory, such as chapters 1-4 of Ang & Tang. A working knowledge of set theory, sample spaces, random variables, probability, density and distribution functions are assumed. The following points should remind the reader of the basic probability theory needed before continuing.

Definition 1 (Expectation of an RV) *The* **expectation**, **expected value**, **mean** *or* **first moment** *of a random variable X, with distribution function F and density f, is defined as:*

$$E(X) := \int x\, dF(x) = \begin{cases} \sum_x x f(x) & \text{if } X \text{ is discrete} \\ \int x f(x)\, dx & \text{if } X \text{ is continuous} \end{cases}, \tag{1.1}$$

provided the sum or integral is well-defined. We say the expectation exists if:

$$\int |x|\ dF(x) < \infty\ . \tag{1.2}$$

Sometimes, we denote $E(X)$ by EX for brevity. Thus, the expectation is a single-number summary of the RV X and may be thought of as the average. We subscript E to specify the parameter $\theta \in \mathbb{\Theta}$ according to which integration is undertaken.

$$E_\theta X := \int x\, dF(x;\theta)$$

Definition 2 (Variance of an RV) *Let X be an RV with the mean or expectation $E(X)$. The* **variance** *of X (denoted by $V(X)$ or VX) is:*

$$V(X) := E\left((X - E(X))^2\right) = \int (x - E(X))^2\, dF(x)\ ,$$

provided this expectation exists. The **standard deviation** *is denoted by* $\mathrm{sd}(X) := \sqrt{V(X)}$. *Thus variance is a measure of the spread of a distribution. Once again, we subscript* V *to specify the parameter* $\theta \in \Theta$ *according to which integration is undertaken.*

$$V_\theta X := \int (x - E(X))^2 \, dF(x;\theta)$$

Definition 3 (k^{th} moment of an RV) *The* k^{th} *moment of the RV* X *is:*

$$E(X^k) = \int x^k \, dF(x) \ .$$

We say that the k^{th} *moment exists when* $E(|X|^k) < \infty$. *We call the following expectation as the* k^{th} *central moment:*

$$E\left((X - E(X))^k\right) \ .$$

Properties of Expectations

1. If the k^{th} moment exists and if $j < k$ then the j^{th} moment exists.
2. If $X_1, X_2, \ldots, X_n$ are RVs and $a_1, a_2, \ldots, a_n$ are constants, then:
$$E\left(\sum_{i=1}^{n} a_i X_i\right) = \sum_{i=1}^{n} a_i E(X_i) \ . \tag{1.3}$$
3. If $X_1, X_2, \ldots, X_n$ are independent RVs, then:
$$E\left(\prod_{i=1}^{n} X_i\right) = \prod_{i=1}^{n} E(X_i) \ . \tag{1.4}$$
4. $V(X) = E(X^2) - (E(X))^2$. [This can be proved by completing the square and applying (1.3)]
5. If a and b are constants then:
$$V(aX + b) = a^2 V(X) \ . \tag{1.5}$$
6. If $X_1, X_2, \ldots, X_n$ are independent and $a_1, a_2, \ldots, a_n$ are constants, then:
$$V\left(\sum_{i=1}^{n} a_i X_i\right) = \sum_{i=1}^{n} a_i^2 V(X_i) \ . \tag{1.6}$$

Chapter 2

Simulation

The Uniform$(0,1)$ RV of Model 1 forms the foundation for random variate generation and simulation. This is appropriately called the fundamental model or experiment, since every other experiment can be obtained from this one.

Model 1 (The Fundamental Model) *The probability density function (PDF) of the fundamental model or the* Uniform$(0,1)$ *RV is*

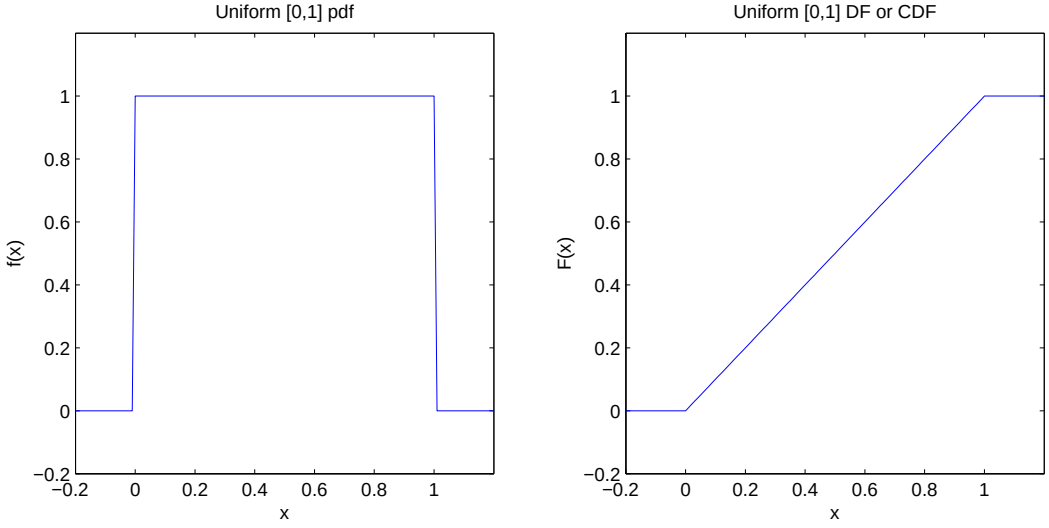
$$f(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1, \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

and its distribution function (DF) or cumulative distribution function (CDF) is:

$$F(x) := \int_{-\infty}^{x} f(y)\, dy = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } 0 \leq x \leq 1, \\ 1 & \text{if } x > 1 \end{cases} \tag{2.2}$$

Note that the DF is the identity map in $[0,1]$*. The PDF and DF are depicted in Figure 2.1.*

Figure 2.1: A plot of the PDF and DF or CDF of the Uniform$(0,1)$ continuous RV X.



Next, we simulate or generate samples from other RVs by making the following two assumptions:

1. independent samples from the Uniform$(0,1)$ RV can be generated, and
2. real arithmetic can be performed exactly in a computer.

Both these assumptions are, in fact, not true and require a more careful treatment of the subject. We may return to these careful treatments later on.

2.1 Inversion Sampler for Continuous Random Variables

Proposition 1 *Let* $F(x) := \int_{-\infty}^{x} f(y)\,dy : \mathbb{R} \to [0,1]$ *be a continuous DF with density* f*, and let its inverse* $F^{[-1]} : [0,1] \to \mathbb{R}$ *be:*

$$F^{[-1]}(u) := \inf\{x : F(x) = u\} \ .$$

Then, $F^{[-1]}(U)$ *has the distribution function* F*, provided* U *is a* Uniform$(0,1)$ *RV. Recall* $\inf(A)$ *or infimum of a set* A *of real numbers is the greatest lower bound of every element of* A*.*

Proof: The "one-line proof" of the proposition is due to the following equalities:

$$P(F^{[-1]}(U) \le x) = P(\inf\{y : F(y) = U)\} \le x) = P(U \le F(x)) = F(x), \quad \textit{for all } x \in \mathbb{R}.$$

■

This yields the inversion sampler or the inverse (C)DF sampler, where we (i) *generate* $u \sim$ Uniform$(0,1)$ and (ii) *return* $x = F^{[-1]}(u)$, as formalised by the following algorithm.

Algorithm 1 Inversion Sampler or Inverse (C)DF Sampler

1: *input:* (1) $F^{[-1]}(x)$, inverse of the DF of the target RV X, (2) the fundamental sampler
2: *initialise:* set the seed, if any, for the fundamental sampler
3: *output:* a sample from X distributed according to F
4: *draw* $u \sim$ Uniform$(0,1)$
5: *return:* $x = F^{[-1]}(u)$

This algorithm emphasises the fundamental sampler's availability in an *input* step, and its set-up needs in an *initialise* step. In the following sections, we will not mention these universal steps; they will be taken for granted. The direct applicability of Algorithm 1 is limited to univariate densities for which the inverse of the cumulative distribution function is explicitly known. The next section will consider some examples.

2.2 Some Simulations of Continuous Random Variables

Model 2 (Uniform(θ_1,θ_2)**)** *Given two real parameters* $\theta_1, \theta_2 \in \mathbb{R}$*, such that* $\theta_1 < \theta_2$*, the PDF of the* $Uniform(\theta_1,\theta_2)$ *RV* X *is:*

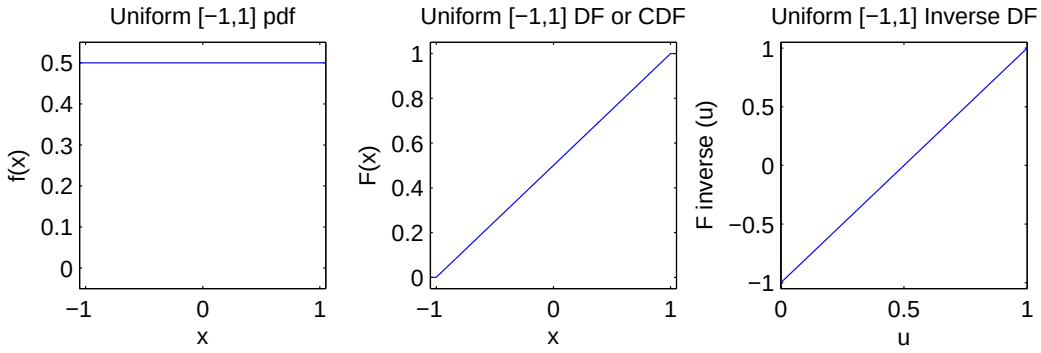
$$f(x;\theta_1,\theta_2) = \begin{cases} \frac{1}{\theta_2-\theta_1} & \textit{if } \theta_1 \le x \le \theta_2, \\ 0 & \textit{otherwise} \end{cases} \tag{2.3}$$

and its DF given by $F(x;\theta_1,\theta_2) = \int_{-\infty}^{x} f(y;\theta_1,\theta_2)\,dy$ *is:*

$$F(x;\theta_1,\theta_2) = \begin{cases} 0 & \text{if } x < \theta_1 \\ \frac{x-\theta_1}{\theta_2-\theta_1} & if\ \theta_1 \le x \le \theta_2, \\ 1 & \text{if } x > \theta_2 \end{cases} \tag{2.4}$$

Recall that we emphasise the dependence of the probabilities on the two parameters θ_1 *and* θ_2 *by specifying them following the semicolon in the argument for* f *and* F*.*

Figure 2.2: A plot of the PDF, DF or CDF and inverse DF of the Uniform$(-1,1)$ RV X.



Simulation 1 (Uniform(θ_1,θ_2)) *To simulate from* Uniform(θ_1,θ_2) *RV* X *using the Inversion Sampler, we first need to find* $F^{[-1]}(u)$ *by solving for* x *in terms of* $u = F(x;\theta_1,\theta_2)$*:*

$$u = \frac{x-\theta_1}{\theta_2-\theta_1} \quad \Longleftrightarrow \quad x = (\theta_2-\theta_1)u + \theta_1 \quad \Longleftrightarrow \quad F^{[-1]}(u;\theta_1,\theta_2) = \theta_1 + (\theta_2-\theta_1)u$$

Here is a simple implementation of the Inversion Sampler for the Uniform(θ_1,θ_2) *RV in* MATLAB*:*

```
>> rand('twister',786); % initialise the fundamental sampler for Uniform(0,1)
>> theta1=-1; theta2=1; % declare values for parameters theta1 and theta2
>> u=rand; % rand is the Fundamental Sampler and u is a sample from it
>> x=theta1+(theta2 - theta1)*u; % sample from Uniform(-1,1]) RV
>> disp(x); % display the sample from Uniform[-1,,1] RV
    0.5134
```

It is just as easy to draw n *IID samples from* Uniform(θ_1,θ_2) *RV* X *by transforming* n *IID samples from the* Uniform$(0,1)$ *RV as follows:*

```
>> rand('twister',786543); % initialise the fundamental sampler
>> theta1=-83; theta2=1004; % declare values for parameters a and b
>> u=rand(1,5); % now u is an array of 5 samples from Uniform(0,1)
>> x=theta1+(theta2 - theta1)*u; % x is an array of 5 samples from Uniform(-83,1004]) RV
>> disp(x); % display the 5 samples just drawn from Uniform(-83,1004) RV
  465.3065  111.4994   14.3535  724.8881  254.0168
```

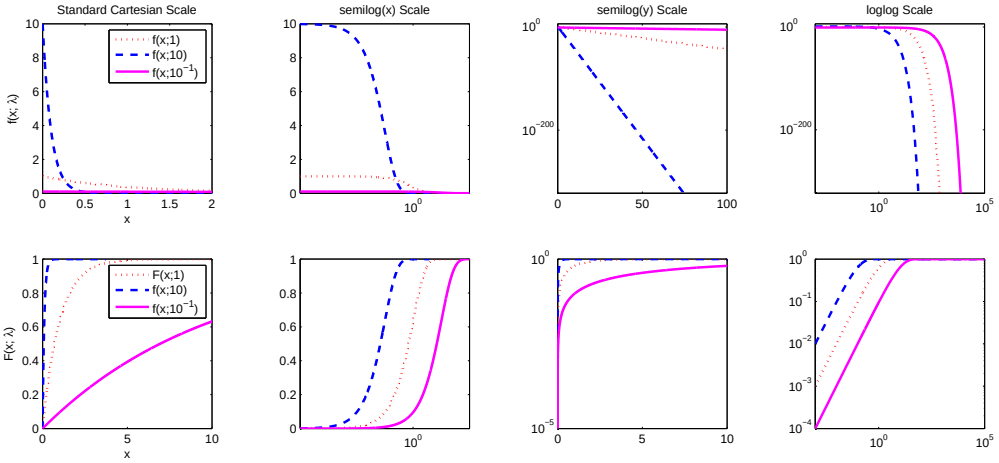
Model 3 (Exponential(λ)) *For a given* $\lambda > 0$*, an* Exponential(λ) *RV has the following PDF* f *and DF* F*:*

$$f(x;\lambda) = \lambda e^{-\lambda x} \qquad F(x;\lambda) = 1 - e^{-\lambda x}\ . \tag{2.5}$$

This distribution is fundamental because of its property of **memorylessness** *and plays a fundamental role in continuous time processes as we will see later.*

We encode the pdf and DF of the Exponential(λ) RV as MATLAB functions `ExponentialPdf` and `ExponentialCdf` and use them to produce Figure 2.3 in Labwork 5.0.10.

Figure 2.3: Density and distribution functions of *Exponential*(λ) RVs, for $\lambda = 1, 10, 10^{-1}$, in four different axes scales.



Example 2.2.1 (Mean and Variance of Exponential(λ)**)** *Show that the mean of an* Exponential(λ) *RV* X *is:*

$$E_\lambda(X) = \int_0^\infty x f(x;\lambda)\, dx = \int_0^\infty x\lambda e^{-\lambda x}\, dx = \frac{1}{\lambda}\ ,$$

and the variance is:

$$V_\lambda(X) = \left(\frac{1}{\lambda}\right)^2\ .$$

Let us consider the problem of simulating from an Exponential(λ) RV with realisations in $\mathbb{R}_+ := [0,\infty) := \{x : x \geq 0, x \in \mathbb{R}\}$ to model the waiting time for a bus at a bus stop.

Simulation 2 (Exponential(λ)) *For a given* $\lambda > 0$, *an* Exponential(λ) *RV has the following pdf* f, *DF* F *and inverse DF* $F^{[-1]}$*:*

$$f(x;\lambda) = \lambda e^{-\lambda x} \qquad F(x;\lambda) = 1 - e^{-\lambda x} \qquad F^{[-1]}(u;\lambda) = \frac{-1}{\lambda}\log_e(1-u) \tag{2.6}$$

We write the natural logarithm $\log_e$ *as* log *for notational simplicity. An implementation of the Inversion Sampler for* Exponential(λ) *as a function in the M-file:*

ExpInvCDF.m

```
function x = ExpInvCDF(u,lambda);
% Return the Inverse CDF of Exponential(lambda) RV X
% Call Syntax: x = ExpInvCDF(u,lambda);
%              ExpInvCDF(u,lambda);
% Input      : lambda = rate parameter,
%              u = array of numbers in [0,1]
% Output     : x
x=-(1/lambda) * log(1-u);
```

We can simply call the function to draw a sample from, say the Exponential($\lambda = 1.0$) *RV by:*

```
lambda=1.0;                          % some value for lambda
u=rand;                            % rand is the Fundamental Sampler
ExpInvCDF(u,lambda)         % sample from Exponential(1) RV via function in ExpInvCDF.m
```

Because of the following:

$$U \sim \text{Uniform}(0,1) \quad \Longrightarrow \quad -U \sim \text{Uniform}(-1,0) \quad \Longrightarrow \quad 1-U \sim \text{Uniform}(0,1) \ ,$$

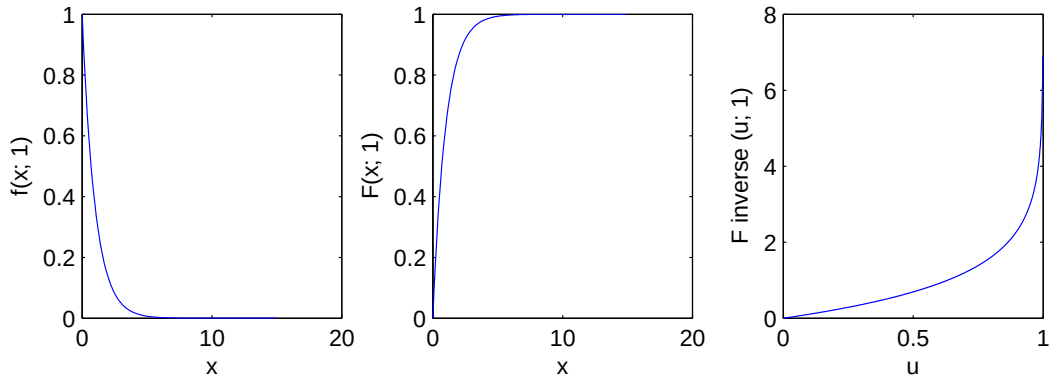
we could save a subtraction operation in the above algorithm by replacing `-(1/lambda) * log(1-u)` *by* `-(1/lambda) * log(u)`. *This is implemented as the following function.*

ExpInvSam.m

```
function x = ExpInvSam(u,lambda);
% Return the Inverse CDF based Sample from Exponential(lambda) RV X
% Call Syntax: x = ExpInvSam(u,lambda);
%              or ExpInvSam(u,lambda);
% Input      : lambda = rate parameter,
%              u = array of numbers in [0,1] from Uniform[0,1] RV
% Output     : x
x=-(1/lambda) * log(u);
```

```
>> rand('twister',46678); % initialise the fundamental sampler
>> Lambda=1.0;  % declare Lambda=1.0
>> x=ExpInvSam(rand(1,5),Lambda); % pass an array of 5 Uniform(0,1) samles from rand
>> disp(x); % display the Exponential(1.0) distributed samples
    0.5945    2.5956    0.9441    1.9015    1.3973
```

Figure 2.4: The pdf f, DF F, and inverse DF $F^{[-1]}$ of the the Exponential($\lambda = 1.0$) RV.



It is straightforward to do replicate experiments. Consider the experiment of drawing five independent samples from the Exponential($\lambda = 1.0$) RV. Suppose we want to repeat or replicate this experiment seven times and find the sum of the five outcomes of zeros and ones in each of these replicates. Then we may do the following:

```
>> rand('twister',1973); % initialise the fundamental sampler
>> % store 7 replications of 5 IID draws from Exponential(1.0) RV in array a
>> lambda=1.0; a= -1/lambda * log(rand(5,7)); disp(a);
    0.7267    0.3226    1.2649    0.4786    0.3774    0.0394    1.8210
    1.2698    0.4401    1.6745    1.4571    0.1786    0.4738    3.3690
    0.4204    0.1219    2.2182    3.6692    0.9654    0.0093    1.7126
```

```
    2.1427    0.1281    0.8500    1.4065    0.1160    0.1324    0.2635
    0.6620    1.1729    0.6301    0.6375    0.3793    0.6525    0.8330
>> %sum up the outcomes of the sequence of 5 draws in each replicate
>> s=sum(a); disp(s);
    5.2216    2.1856    6.6378    7.6490    2.0168    1.3073    7.9990
```

Labwork 2.2.2 *Consider the problem of modelling the arrival of buses at a bus stop. Suppose that the time between arrivals is an* Exponential($\lambda = 0.1$) *RV X with a mean inter-arrival time of $1/\lambda = 10$ minutes. Suppose you go to your bus stop and zero a stop-watch. Simulate the times of arrival for the next seven buses as indicated by your stop-watch. Seed the fundamental sampler by your Student ID (eg. if your ID is* `11424620` *then type* `rand('twister', 11424620);` *just before the simulation). Hand in the code with the arrival times of the next seven buses at your ID-seeded bus stop.*

For a continuous RV X with a closed-form expression for the inverse DF $F^{[-1]}$, we can employ Algorithm 1 to draw samples from X. Table 2.1 summarises some random variables that are amenable to Algorithm 1.

Table 2.1: Some continuous RVs that can be simulated from using Algorithm 1.

Random Variable X	$F(x)$	$X = F^{[-1]}(U), \quad U \sim \text{Uniform}(0,1)$	Simplified form
Uniform(a, b)	(2.4)	$a + (b-a)U$	–
Exponential(λ)	(2.5)	$\frac{-1}{\lambda}\log(1-U)$	$\frac{-1}{\lambda}\log(U)$
Cauchy	(2.9)	$\tan\left(\pi\left(U - \frac{1}{2}\right)\right)$	$\tan\left(\pi U\right)$

Next, we familiarise ourselves with the Gaussian or Normal RV.

Model 4 (Normal(μ, σ^2)**)** *X has a* Normal(μ, σ^2) *or* Gaussian(μ, σ^2) *distribution with the location parameter $\mu \in \mathbb{R}$ and the scale or variance parameter $\sigma^2 > 0$, if:*

$$f(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right), \qquad x \in \mathbb{R} \tag{2.7}$$

Normal$(0, 1)$ *distributed RV, which plays a fundamental role in asymptotic statistics, is conventionally denoted by Z. Z is said to have the* **Standard Normal** *distribution with pdf $f(z; 0, 1)$ and DF $F(z; 0, 1)$ conventionally denoted by $\phi(z)$ and $\Phi(z)$, respectively.*

There is no closed form expression for $\Phi(z)$ or $F(x; \mu, \sigma)$. The latter is simply defined as:

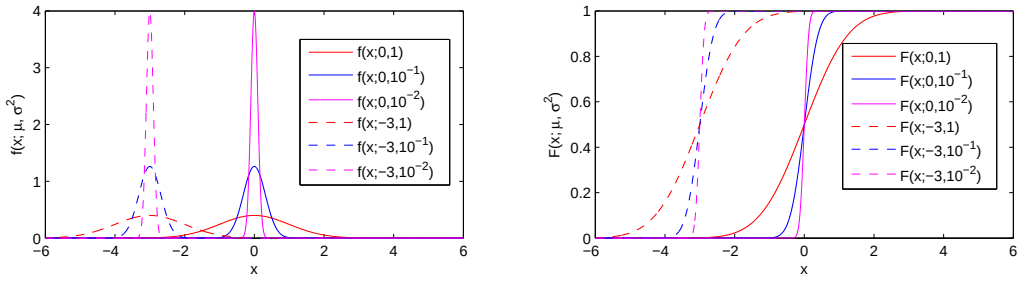
$$F(x; \mu, \sigma^2) = \int_{-\infty}^{x} f(y; \mu, \sigma)\, dy$$

*We can express $F(x; \mu, \sigma^2)$ in terms of the error function (*erf*) as follows:*

$$F(x; \mu, \sigma^2) = \frac{1}{2}\ \text{erf}\left(\frac{x-\mu}{\sqrt{2\sigma^2}}\right) + \frac{1}{2} \tag{2.8}$$

We implement the pdf (2.7) and DF (2.8) for a Normal(μ, σ^2) RV X as MATLAB functions `NormalPdf` and `NormalCdf`, respectively, in Labwork 5.0.9, and then produce their plots for various Normal(μ, σ^2) RVs, shown in Figure 2.5. Observe the concentration of probability mass, in terms of the pdf and DF plots, about the location parameter μ as the variance parameter σ^2 decreases.

Figure 2.5: Density and distribution function of several Normal(μ, σ^2) RVs.



Example 2.2.3 (Mean and Variance of Normal(μ, σ^2)**)** *The mean of a* Normal(μ, σ^2) *RV* X *is:*

$$E(X) = \int_{-\infty}^{\infty} x f(x; \mu, \sigma^2)\, dx = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} x \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) dx = \mu \ ,$$

and the variance is:

$$V(X) = \int_{-\infty}^{\infty} (x-\mu)^2 f(x; \mu, \sigma^2)\, dx = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} (x-\mu)^2 \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) dx = \sigma^2 \ .$$

Labwork 2.2.4 *Write a function to evaluate the* $P(X \in [a, b])$ *for the* Normal$(0, 1)$ *RV* X *for user-specified values of* a *and* b*. [Hint: one option is by making two calls to* `NormalCdf` *and doing one arithmetic operation.]*

Simulations 1 and 2 produce samples from a continuous RV X with a closed-form expression for the inverse DF $F^{[-1]}$ via Algorithm 1 (Table 2.1). But only a few RVs have an explicit $F^{[-1]}$. For example, Normal$(0, 1)$ RV does not have an explicit $F^{[-1]}$. Algorithm 2 is a more general but inexact method that relies on an approximate numerical solution of x, for a given u, that satisfies the equation $F(x) = u$.

Algorithm 2 Inversion Sampler by Numerical Solution of $F(X) = U$ via Newton-Raphson Method

input: $F(x)$, the DF of the target RV X
input: $f(x)$, the density of X
input: A reasonable `Stopping Rule`,
e.g. a specified tolerance $\epsilon > 0$ and a maximum number of iterations `MAX`
input: a careful mechanism to specify x_0
output: a sample from X distributed according to F
draw: $u \sim$ Uniform$(0, 1)$
initialise: $i \leftarrow 0$, $x_i \leftarrow x_0$, $x_{i+1} \leftarrow x_0 - \frac{F(x_0)-u}{f(x_0)}$
while `Stopping Rule` is not satisfied,
e.g. $|F(x_i) - F(x_{i-1})| > \epsilon$ AND $i <$ `MAX` **do**
  $x_i \leftarrow x_{i+1}$
  $x_{i+1} \leftarrow \left(x_i - \frac{F(x_i)-u}{f(x_i)}\right)$
  $i \leftarrow i + 1$
end while
return: $x \leftarrow x_i$

Simulation 3 (Normal(μ, σ^2)**)** *We may employ Algorithm 2 to sample from the Normal*(μ, σ^2) *RV* X *using the following function.*

Sample1NormalByNewRap.m

```
function x = Sample1NormalByNewRap(u,Mu,SigmaSq)
% Returns a sample from Normal(Mu, SigmaSq)
% Newton-Raphson numerical solution of F(x)=u
% Input: u = one random Uniform(0,1) sample
%        Mu = Mean of Normal(Mu, SigmaSq)
%        SigmaSq = Variance of Normal(Mu, SigmaSq)
% Usage: x = Sample1NormalByNewRap(u,Mu,SigmaSq)
% To transform an array Us of uniform samples to array Xs of Normal samples via arrayfun
%        Xs = arrayfun(@(u)(Sample1NormalByNewRap(u,-100.23,0.01)),Us);
Epsilon=1e-5; % Tolerance in stopping rule
MaxIter=10000; % Maximum allowed iterations in stopping rule
x=0; % initialize the output x as 0
% initialize i, xi, and xii
i=0;        % Mu is an ideal initial condition since F(x; Mu, SigmaSq)
xi = Mu;    % is convex when x < Mu and concave when x > Mu and the
% Newton-Raphson method started at Mu converges
xii = xi - (NormalCdf(xi,Mu,SigmaSq)-u)/NormalPdf(xi,Mu,SigmaSq);
% Newton-Raphson Iterations
while (abs(NormalCdf(xii,Mu,SigmaSq)-NormalCdf(xi,Mu,SigmaSq))...
        > Epsilon & i < MaxIter),
    xi = xii;
    xii = xii - (NormalCdf(xii,Mu,SigmaSq)-u)/NormalPdf(xii,Mu,SigmaSq);
    i=i+1;
end
x=xii; % record the simulated x from the j-th element of u
```

We draw five samples from the Normal$(0,1)$ *RV* Z *and store them in* z *as follows. The vector* z *can be obtained by a Newton-Raphson-based numerical transformation of the vector* u *of* 5 *IID samples from the* Uniform$(0,1)$ *RV. We simply need to apply the function* `Sample1NormalByNewRap` *to each element of an arra of* $Unifrm(0,1)$ *samples.* MATLAB*'s* `arrayfun` *command can be used to apply* `@(u)(Sample1NormalByNewRap(u,0,1))` *(i.e.,* `Sample1NormalByNewRap` *as a function of* u*) to every element of our array of* Uniform$(0,1)$ *samples, say* `Us`*. Note that* $F(z)$ *is the same as the drawn* u *from* U *at least up to four significant digits.*

```
>> rand('twister',563987);
>> Us=rand(1,5); % store 5 samples from Uniform(0,1) RV in array Us
>> disp(Us); % display Us
    0.8872    0.2569    0.5275    0.8650    0.8517
>> z=Sample1NormalByNewRap(u(1),0,1); %transform Us(1) to a Normal(0,1) sample z
>> disp(z); % display z
    1.2119
>> z = arrayfun(@(u)(Sample1NormalByNewRap(u,0,1)),Us); %transform array Us via arrayfun
>> % dislay array z obtained from applying Sample1NormalByNewRap to each element of Us
>> disp(z);
    1.2119   -0.6530    0.0691    1.1031    1.0439
>> % check that numerical inversion of F worked, i.e., is F(z)=u ?
>> disp(NormalCdf(z,0,1));
    0.8872    0.2569    0.5275    0.8650    0.8517
```

Next we draw five samples from the Normal$(-100.23, 0.01)$ *RV* X*, store it in an array* x *and observe that the numerical method is reasonably accurate by the equality of* u *and* $F(x)$*.*

```
>> rand('twister',563987);
>> disp(Us); % display Us
    0.8872    0.2569    0.5275    0.8650    0.8517
>> % transform array Us via arrayfun
>> x = arrayfun(@(u)(Sample1NormalByNewRap(u,-100.23,0.01)),Us);
>> disp(x);
```

```
 -100.1088 -100.2953 -100.2231 -100.1197 -100.1256
>> disp(NormalCdf(x,-100.23,0.01));
    0.8872    0.2569    0.5275    0.8650    0.8517
```

One has to be extremely careful with this approximate simulation algorithm implemented in floating-point arithmetic. More robust samplers for the Normal(μ, σ^2) *RV exist. However, Algorithm 2 is often the only choice when simulating from an arbitrary RV with an unknown closed-form expression for its* $F^{[-1]}$.

Next, we use our simulation capability to gain an informal and intuitive understanding of one of the most elementary theorems in probability and statistics, namely, the Central Limit Theorem (CLT). We will see a formal treatment of CLT later.

Informally, the CLT can be stated as follows:
"The sample mean of a large number of IID samples, none of which is dominant, tends to the Normal distribution as the number of samples increases."

Let us investigate the histograms from 10000 simulations of the sample mean of $n = 10, 100, 1000$ IID Exponential$(\lambda = 0.1)$ RVs as follows:

```
>> rand('twister',1973); % initialise the fundamental sampler
>> % a demonstration of Central Limit Theorem (CLT) -- Details of CLT are in the sequel
>> % the sample mean should be a Normal(1/lambda,lambda/n) RV
>> lambda=0.1; Reps=10000; n=10; hist(sum(-1/lambda * log(rand(n,Reps)))/n)
>> lambda=0.1; Reps=10000; n=100; hist(sum(-1/lambda * log(rand(n,Reps)))/n,20)
>> lambda=0.1; Reps=10000; n=1000; hist(sum(-1/lambda * log(rand(n,Reps)))/n,20)
```

Do you see a pattern in the histograms ?

Next, let us become familiar with an RV for which the expectation does not exist. This will help us appreciate the phrase "none of which is dominant" in the informal statement of the CLT earlier.

Model 5 (Cauchy**)** *The density of the* Cauchy *RV* X *is:*

$$f(x) = \frac{1}{\pi(1+x^2)}, \qquad -\infty < x < \infty \ ,$$

and its DF is:

$$F(x) = \frac{1}{\pi}\tan^{-1}(x) + \frac{1}{2} \ . \tag{2.9}$$

Example 2.2.5 (Mean of Cauchy **RV)** *The expectation of the* Cauchy *RV* X*, obtained via integration by parts (set* $u = x$ *and* $v = tan^{-1}(x)$*) does not exist , since:*

$$\int |x| \ dF(x) = \frac{2}{\pi}\int_0^\infty \frac{x}{1+x^2}\,dx = \left(x\tan^{-1}(x)\right]_0^\infty - \int_0^\infty \tan^{-1}(x)\,dx = \infty \ . \tag{2.10}$$

Variance and higher moments cannot be defined when the expectation itself is undefined.

Simulation 4 (Cauchy**)** *We can draw* n *IID samples from the* Cauchy *RV* X *by transforming* n *IID samples from* Uniform$(0, 1)$ *RV* U *using the inverse DF as follows:*

```
>> rand('twister',2435567);        % initialise the fundamental sampler
>> u=rand(1,5);                        % draw 5 IID samples from Uniform(0,1) RV
>> disp(u);                          % display the samples in u
    0.7176    0.6655    0.9405    0.9198    0.2598
>> x=tan(pi * u);     % draw 5 samples from Standard cauchy RV using inverse CDF
>> disp(x);  % display the samples in x
   -1.2272   -1.7470   -0.1892   -0.2575    1.0634
```

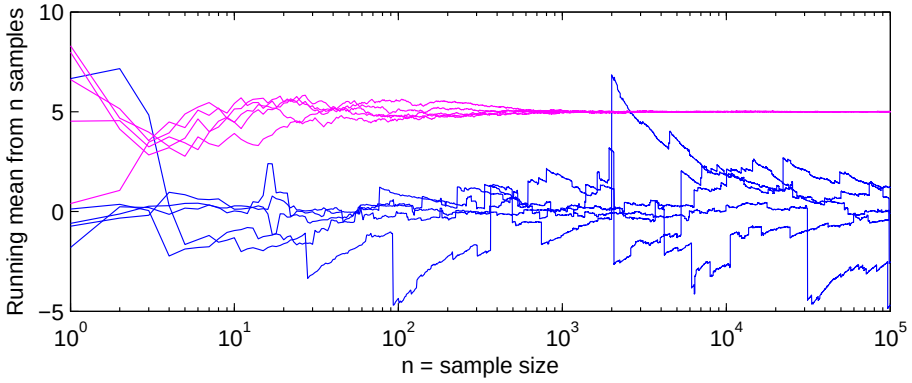
Recall that the mean of the Cauchy RV X does not exist since $\int |x| \, dF(x) = \infty$ (2.10). We will investigate this in Labwork 2.2.6.

Labwork 2.2.6 *Let us see what happens when we plot the running sample mean for an increasing sequence of IID samples from the Standard Cauchy RV X by implementing the following script file:*

PlotStandardCauchyRunningMean.m

```
% script to plot the oscillating running mean of Std Cauchy samples
% relative to those for the Uniform(0,10) samples
rand('twister',25567);        % initialize the fundamental sampler
for i=1:5
N = 10^5;                          % maximum sample size
u=rand(1,N);             % draw N IID samples from Uniform(0,1)
x=tan(pi * u);     % draw N IID samples from Standard cauchy RV using inverse CDF
n=1:N;               % make a vector n of current sample size [1 2 3 ... N-1 N]
CSx=cumsum(x); % CSx is the cumulative sum of the array x (type 'help cumsum')
% Runnign Means <- vector division of cumulative sum of samples by n
RunningMeanStdCauchy = CSx ./ n; % Running Mean for Standard Cauchy samples
RunningMeanUnif010 = cumsum(u*10.0) ./ n; % Running Mean for Uniform(0,10) samples
semilogx(n, RunningMeanStdCauchy) %
hold on;
semilogx(n, RunningMeanUnif010, 'm')
end
xlabel('n = sample size');
ylabel('Running mean from n samples')
```

Figure 2.6: Unending fluctuations of the running means based on n IID samples from the Standard Cauchy RV X in each of five replicate simulations (blue lines). The running means, based on n IID samples from the Uniform$(0, 10)$ RV, for each of five replicate simulations (magenta lines).



The resulting plot is shown in Figure 2.6. Notice that the running means or the sample mean of n samples as a function of n, for each of the five replicate simulations, never settles down to a particular value. This is because of the "thick tails" of the density function for this RV which produces extreme observations. Compare them with the running means, based on n IID samples

from the Uniform$(0, 10)$ *RV, for each of five replicate simulations (magenta lines). The latter sample means have settled down stably to the mean value of* 5 *after about* 700 *samples.*
See the histograms generated from the following code:

```
>> Reps=10000; n=1000; hist(sum(tan(pi * rand(n,Reps)))/n,20)
>> Reps=10000; n=1000; hist(sum(tan(pi * rand(n,Reps)))/n,20)
>> Reps=10000; n=1000; hist(sum(tan(pi * rand(n,Reps)))/n,20)
```

Classwork 2.2.7 *Explain in words why the mean of n IID samples from the* Cauchy *RV is* **not** *obeying the Central Limit Theorem.*

Model 6 (Lognormal(λ, ζ)**)** *X has a* Lognormal(λ, ζ) *distribution if* $\log(X)$ *has a* Normal(λ, ζ^2) *distribution. The location parameter* $\lambda = E(\log(X)) > 0$ *and the scale parameter* $\zeta > 0$*. The PDF is:*

$$f(x; \lambda, \zeta) = \frac{1}{\sqrt{2\pi}\zeta x} \exp\left(-\frac{1}{2\zeta^2}(\log(x) - \lambda)^2\right), \qquad x > 0 \tag{2.11}$$

No closed form expression for $F(x; \lambda, \zeta)$ *exists and it is simply defined as:*

$$F(x; \lambda, \zeta) = \int_0^x f(y; \lambda, \zeta)\, dy$$

We can express $F(x; \lambda, \zeta)$ *in terms of* Φ *(and, in turn, via the associated error function* erf*) as follows:*

$$F(x; \lambda, \zeta) = \Phi\left(\frac{\log(x) - \lambda}{\zeta}\right) = \frac{1}{2}\ \text{erf}\left(\frac{\log(x) - \lambda}{\sqrt{2}\zeta}\right) + \frac{1}{2} \tag{2.12}$$

Labwork 2.2.8 *Transform a sequence of samples obtained from the fundamental sampler to those from the* Lognormal(λ_C, ζ_C) *RV C by using only Algorithm 2 as an intermediate step. Do the following to demonstrate that you can produce* 1000 *samples from* $C = \frac{WF}{\sqrt{E}}$*, where:*

$W \sim$ Lognormal$(\log(2000), 0.20)$, $F \sim$ Lognormal$(\log(20), 0.15)$, $E \sim$ Lognormal$(\log(1.6), 0.125)$

1. *Clearly derive the distribution of* C*. In other words, what are the values of the parameters* λ_C *and* ζ_C*?*
2. *Seed the fundamental sampler by your Student ID. If your ID is* `11424620` *then type* `rand('twister', 11424620);`*.*
3. *Draw* 1000 *samples directly from* C *and report:*
   (a) *how many of them are larger than* 35000*,*
   (b) *the sample mean, and*
   (c) *the sample standard deviation.*
4. *Seed the fundamental sampler by your Student ID.*
5. *Draw* 1000 *samples indirectly from* C*, i.e. [(i) draw* 1000 *samples from* W*, (ii) draw* 1000 *samples from* F*, (iii) draw* 1000 *samples from* E*, and (iv) transform each of the* 1000 *triples into* 1000 *samples from* C *via* $WF/\sqrt{E}$*] and report:*
   (a) *how many of them are larger than* 35000*,*
   (b) *the sample mean, and*
   (c) *the sample standard deviation.*

[Hint: If Y *is a* Normal(λ, ζ) *RV, then* $Z = e^Y$ *is said to be a* Lognormal(λ, ζ) *RV.]*

2.3 Inversion Sampler for Discrete Random Variables

Next, consider the problem of **sampling from a random variable X with a discontinuous or discrete DF** using the inversion sampler. We need to define the inverse more carefully here.

Proposition 2 *Let the support of the RV X be over some real interval $[a, b]$ and let its inverse DF be defined as follows:*

$$F^{[-1]}(u) := \inf\{x \in [a, b] : F(x) \geq u,\ 0 \leq u \leq 1\} \ .$$

If $U \sim \text{Uniform}(0, 1)$ then $X = F^{[-1]}(U)$ has the DF F.

Proof: The proof is a consequence of the following equalities:

$$P(X \leq x) = P(F^{[-1]}(u) \leq x) = P(U \leq F(x)) = F(x)$$

■

2.4 Some Simulations of Discrete Random Variables

Model 7 (Bernoulli(θ)**)** *Given a parameter $\theta \in [0, 1]$, the probability mass function (PMF) for the* Bernoulli(θ) *RV X is:*

$$f(x;\theta) = \begin{cases} \theta & \text{if } x = 1, \\ 1-\theta & \text{if } x = 0, \\ 0 & \text{otherwise} \end{cases} \quad \text{or, equivalently,} \quad f(x;\theta) = \begin{cases} \theta^x(1-\theta)^{1-x} & \text{if } x \in \{0,1\}, \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

and its DF is:

$$F(x;\theta) = \begin{cases} 1 & \text{if } 1 \leq x, \\ 1-\theta & \text{if } 0 \leq x < 1, \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

We emphasise the dependence of the probabilities on the parameter θ by specifying it following the semicolon in the argument for f and F.

Simulation 5 (Bernoulli(θ)**)** *Consider the problem of simulating from a* Bernoulli(θ) *RV based on an input from a* Uniform$(0, 1)$ *RV. Recall that $\lfloor x \rfloor$ (called the 'floor of x') is the largest integer that is smaller than or equal to x, e.g. $\lfloor 3.8 \rfloor = 3$. Using the floor function, we can simulate a* Bernoulli(θ) *RV X as follows:*

```
>>  theta = 0.3;                % set theta = Prob(X=1)
  % return x  -- floor(y) is the largest integer less than or equal to y
>> x = floor(rand + theta)        % rand is the Fundamental Sampler
>> disp(x) % display the outcome of the simulation
     0
>> n=10;  % set the number of IID Bernoulli(theta=0.3) trials you want to simulate
>> x = floor(rand(1,10)+theta); % vectorize the operation
>> disp(x) % display the outcomes of the simulation
     0     0     1     0     0     0     0     0     1     1
```

Again, it is straightforward to do replicate experiments, e.g. to demonstrate the Central Limit Theorem for a sequence of n IID Bernoulli(θ) trials.

```
>> % a demonstration of Central Limit Theorem --
>> % the sample mean of a sequence of n IID Bernoulli(theta) RVs is Gaussian(theta,theta(1-theta)/n)
>> theta=0.5; Reps=10000; n=10; hist(sum(floor(rand(n,Reps)+theta))/n)
>> theta=0.5; Reps=10000; n=100; hist(sum(floor(rand(n,Reps)+theta))/n,20)
>> theta=0.5; Reps=10000; n=1000; hist(sum(floor(rand(n,Reps)+theta))/n,30)
```

Next let us consider a natural generalization of the Bernoulli(θ) RV with more than two outcomes.

Model 8 (de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$**)** *Given a specific point* $(\theta_1, \theta_2, \ldots, \theta_k)$ *in the* k*-Simplex:*

$$\triangle_k := \{ (\theta_1, \theta_2, \ldots, \theta_k) : \theta_1 \geq 0, \theta_2 \geq 0, \ldots, \theta_k \geq 0, \sum_{i=1}^{k} \theta_i = 1 \} ,$$

we say that an RV X *is* de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$ *distributed if its PMF is:*

$$f(x; \theta_1, \theta_2, \ldots, \theta_k) = \begin{cases} 0 & \text{if } x \notin [k] := \{1, 2, \ldots, k\}, \\ \theta_x & \text{if } x \in [k]. \end{cases}$$

The DF for de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$ *RV* X *is:*

$$F(x; \theta_1, \theta_2, \ldots, \theta_k) = \begin{cases} 0 & \text{if } -\infty < x < 1 \\ \theta_1 & \text{if } 1 \leq x < 2 \\ \theta_1 + \theta_2 & \text{if } 2 \leq x < 3 \\ \vdots & \\ \theta_1 + \theta_2 + \cdots + \theta_{k-1} & \text{if } k-1 \leq x < k \\ \theta_1 + \theta_2 + \cdots + \theta_{k-1} + \theta_k = 1 & \text{if } k \leq x < \infty \end{cases} \tag{2.15}$$

The de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$ *RV can be thought of as a probability model for "the outcome of rolling a polygonal cylindrical die with* k *rectangular faces that are marked with* $1, 2, \ldots, k$*". The parameters* $\theta_1, \theta_2, \ldots, \theta_k$ *specify how the die is loaded and may be idealised as specifying the cylinder's centre of mass with respect to the respective faces. Thus, when* $\theta_1 = \theta_2 = \cdots = \theta_k = 1/k$*, we have a probability model for the outcomes of a fair die.*

Next we simulate from de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$ RV X via its inverse DF

$$F^{[-1]} : [0, 1] \to [k] := \{1, 2, \ldots, k\} ,$$

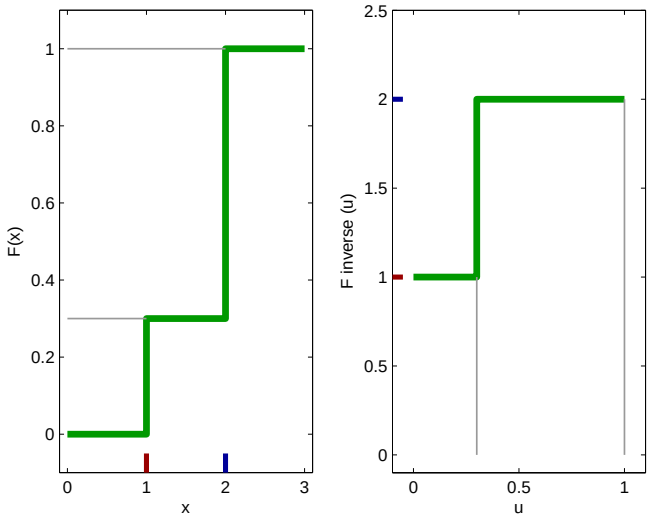
given by:

$$F^{[-1]}(u; \theta_1, \theta_2, \ldots, \theta_k) = \begin{cases} 1 & \text{if } 0 \leq u < \theta_1 \\ 2 & \text{if } \theta_1 \leq u < \theta_1 + \theta_2 \\ 3 & \text{if } \theta_1 + \theta_2 \leq u < \theta_1 + \theta_2 + \theta_3 \\ \vdots & \\ k & \text{if } \theta_1 + \theta_2 + \cdots + \theta_{k-1} \leq u < 1 \end{cases}$$

When $k = 2$ in the de Moivre(θ_1, θ_2) model, we have an RV that is similar to the Bernoulli$(p = \theta_1)$ RV. The DF F and its inverse $F^{[-1]}$ for a specific $\theta_1 = 0.3$ are depicted in Figure 2.7.

First we simulate from an equi-probable special case of the de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$ RV, with $\theta_1 = \theta_2 = \cdots = \theta_k = 1/k$.

Figure 2.7: The DF $F(x; 0.3, 0.7)$ of the de Moivre$(0.3, 0.7)$ RV and its inverse $F^{[-1]}(u; 0.3, 0.7)$.



Simulation 6 (de Moivre$(1/k, 1/k, \ldots, 1/k)$) *The equi-probable de Moivre$(1/k, 1/k, \ldots, 1/k)$ RV X with a discrete uniform distribution over $[k] = \{1, 2, \ldots k\}$ can be efficiently sampled using the ceiling function. Recall that $\lceil y \rceil$ is the smallest integer larger than or equal to y, eg. $\lceil 13.1 \rceil = 14$. Algorithm 3 produces samples from the de Moivre$(1/k, 1/k, \ldots, 1/k)$ RV X.*

Algorithm 3 Inversion Sampler for de Moivre$(1/k, 1/k, \ldots, 1/k)$ RV

1: *input:*

1. k in de Moivre$(1/k, 1/k, \ldots, 1/k)$ RV X
2. $u \sim$ Uniform$(0, 1)$

2: *output:* a sample from X

3: *return:* $x \leftarrow \lceil ku \rceil$

The M-file implementing Algorithm 3 is:

SimdeMoivreEqui.m

```
function x = SimdeMoivreEqui(u,k);
% return samples from de Moivre(1/k,1/k,...,1/k) RV X
% Call Syntax:  x = SimdeMoivreEqui(u,k);
% Input       : u = array of uniform random numbers eg. rand
%               k = number of equi-probabble outcomes of X
% Output      : x = samples from X
x = ceil(k * u) ; % ceil(y) is the smallest integer larger than y
%x = floor(k * u); if outcomes are in {0,1,...,k-1}
```

Let us use the function `SimdeMoivreEqui` *to draw five samples from a fair seven-faced cylindrical dice.*

```
>> k=7; % number of faces of the fair dice
>> n=5; % number of trials
>> rand('twister',78657); % initialise the fundamental sampler
>> u=rand(1,n); % draw n samples from Uniform(0,1)
```

```
>> % inverse transform samples from Uniform(0,1) to samples
>> % from de Moivre(1/7,1/7,1/7,1/7,1/7,1/7,1/7)
>> outcomes=SimdeMoivreEqui(u,k); % save the outcomes in an array
>> disp(outcomes);
     6     5     5     5     2
```

Now, let us consider the more general problem of implementing a sampler for an arbitrary but specified de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$ RV. That is, the values of θ_i need not be equal to $1/k$.

Simulation 7 (de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$) *We can generate samples from a* de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$ *RV* X *when* $(\theta_1, \theta_2, \ldots, \theta_k)$ *are specifiable as an input vector via the following algorithm.*

Algorithm 4 Inversion Sampler for de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$ RV X

1: *input:*

   1. parameter vector $(\theta_1, \theta_2, \ldots, \theta_k)$ of de Moivre$(\theta_1, \theta_2, \ldots, \theta_k)$ RV X.
   2. $u \sim$ Uniform$(0, 1)$

2: *output:* a sample from X
3: *initialise:* $F \leftarrow \theta_1$, $i \leftarrow 1$
4: **while** $u > F$ **do**
5: $\quad i \leftarrow i + 1$
6: $\quad F \leftarrow F + \theta_i$
7: **end while**
8: *return:* $x \leftarrow i$

The M-file implementing Algorithm 4 is:

SimdeMoivreOnce.m

```
function  x = SimdeMoivreOnce(u,f)
% Returns a sample from the de Moivre(f=(f_1,f_2,...,f_k)) RV X
% Call Syntax:  x = SimdeMoivreOnce(u,f);
%               deMoivreEqui(u,k);
% Input      : u = a uniform random number eg. rand
%              f = an array of probabilities f=[f1 f2...fk]
% Output     : x = sample from X
x=1; % initial index is 1
current_f=f(x);
while u>current_f;
    x=x+1;
    current_f = current_f + f(x);
end
```

Let us use the function `deMoivreEqui` *to draw five samples from a fair seven-faced dice.*

```
>> k=7; % number of faces of the fair dice
>> n=5; % number of trials
>> rand('twister',78657); % initialise the fundamental sampler
>> Us=rand(1,n); % draw n samples from Uniform(0,1)
>> disp(Us);
    0.8330    0.6819    0.6468    0.6674    0.2577
>> % inverse transform samples from Uniform(0,1) to samples
>> % from de Moivre(1/7,1/7,1/7,1/7,1/7,1/7,1/7)
>> f=[1/7 1/7 1/7 1/7 1/7 1/7 1/7];
>> disp(f);
```

```
    0.1429    0.1429    0.1429    0.1429    0.1429    0.1429    0.1429
>> % use funarray to apply function-handled SimdeMoivreOnce to
>> % each element of array Us and save it in array outcomes2
>> outcomes2=arrayfun(@(u)(SimdeMoivreOnce(u,f)),Us);
>> disp(outcomes2);
     6     5     5     5     2
>> disp(SimdeMoivreEqui(u,k)); % same result using the previous algorithm
     6     5     5     5     2
```

Clearly, Algorithm 4 may be used to sample from any de Moivre$(\theta_1, \ldots, \theta_k)$ RV X. We demostrate this by producing five samples from a randomly generated PMF `f2`.

```
>> rand('twister',1777); % initialise the fundamental sampler
>> f2=rand(1,10); % create an arbitrary array
>> f2=f2/sum(f2); % normalize to make a probability mass function
>> disp(f2); % display the weights of our 10-faced die
    0.0073    0.0188    0.1515    0.1311    0.1760    0.1121    ...
    0.1718    0.1213    0.0377    0.0723
>> disp(sum(f2)); % the weights sum to 1
    1.0000
>> disp(arrayfun(@(u)(SimdeMoivreOnce(u,f2)),rand(5,5))) % the samples from f2 are
     4     3     4     7     3
     6     7     4     5     3
     5     8     7    10     6
     2     3     5     7     7
     6     5     9     5     7
```

Note that the principal work here is the sequential search, in which the mean number of comparisons until success is:

$$1\theta_1 + 2\theta_2 + 3\theta_3 + \ldots + k\theta_k = \sum_{i=1}^{k} i\theta_i$$

For the de Moivre$(1/k, 1/k, \ldots, 1/k)$ RV, the right-hand side of the above expression is:

$$\sum_{i=1}^{k} i\frac{1}{k} = \frac{1}{k}\sum_{i=1}^{k} i = \frac{1}{k}\frac{k(k+1)}{2} = \frac{k+1}{2} \ ,$$

indicating that the average-case efficiency is linear in k. This linear dependence on k is denoted by $O(k)$. In other words, as the number of faces k increases, one has to work linearly harder to get samples from de Moivre$(1/k, 1/k, \ldots, 1/k)$ RV using Algorithm 4. Using the simpler Algorithm 3, which exploits the fact that all values of θ_i are equal, we generated samples in constant time, which is denoted by $O(1)$.

The above problem is a special case of simulating from the following more general RV.

Model 9 (*GD*$(\theta_0, \theta_1, \ldots)$) *We say X is a* GeneralDiscrete$(\theta_0, \theta_1, \ldots)$ *or* GD$(\theta_0, \theta_1, \ldots)$ *RV over the countable discrete state space $\mathbb{Z}_+ := \{0, 1, 2, \ldots\}$ with parameters $(\theta_0, \theta_1, \ldots)$ if the PMF of X is defined as follows:*

$$f(X = x; \theta_0, \theta_1, \ldots) = \begin{cases} 0, & if \quad x \notin \{0, 1, 2, \ldots\} \\ \theta_0, & if \quad x = 0 \\ \theta_1, & if \quad x = 1 \\ \vdots \end{cases}$$

Algorithm 5 allows us to simulate from any member of the class of non-negative discrete RVs as specified by the probabilities $(\theta_0, \theta_1, \ldots)$. *When an RV* X *takes values in another countable set* $\mathbb{X} \neq \mathbb{Z}_+$, *then we can still use the above algorithm provided we have a one-to-one and onto mapping* D *from* $\mathbb{Z}_+$ *to* $\mathbb{X}$ *that allows us to think of* $\{0, 1, 2, \ldots\}$ *as indices of an array* D.

Algorithm 5 Inversion Sampler for $GD(\theta_0, \theta_1, \ldots)$ RV X

1: *input:*

1. θ_0 and $\{C(i) = \theta_i/\theta_{i-1}\}$ for any $i \in \{1, 2, 3, \ldots\}$.
2. $u \sim \text{Uniform}(0, 1)$

2: *output:* a sample from X
3: *initialise:* $p \leftarrow \theta_0$, $q \leftarrow \theta_0$, $i \leftarrow 0$
4: **while** $u > q$ **do**
5: $\quad i \leftarrow i + 1$, $p \leftarrow p\, C(i)$, $q \leftarrow q + p$
6: **end while**
7: *return:* $x = i$

Model 10 (Binomial(n, θ) **RV)** *Let the RV* $X = \sum_{i=1}^{n} X_i$ *be the sum of* n *independent and identically distributed* Bernoulli(θ) *RVs, i.e.:*

$$X = \sum_{i=1}^{n} X_i, \qquad X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} \text{Bernoulli}(\theta) \ .$$

Given two parameters n *and* θ, *the pmf of the* Binomial(n, θ) *RV* X *is:*

$$f(x; n, \theta) = \begin{cases} \binom{n}{x} \theta^x (1-\theta)^{n-x} & \text{if } x \in \{0, 1, 2, 3, \ldots, n\} \ , \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

where, $\binom{n}{x}$ *is:*

$$\binom{n}{x} = \frac{n(n-1)(n-2)\ldots(n-x+1)}{x(x-1)(x-2)\cdots(2)(1)} = \frac{n!}{x!(n-x)!} \ .$$

$\binom{n}{x}$ *is read as "*n *choose* x*."*

Example 2.4.1 (Mean and variance of Binomial(n, θ) **RV)** *Let* $X \sim \text{Binomial}(n, \theta)$. *Based on the definition of expectation:*

$$E(X) = \int x\, dF(x; n, \theta) = \sum_x x f(x; n, \theta) = \sum_{x=0}^{n} x \binom{n}{x} \theta^x (1-\theta)^{n-x} \ .$$

However, this is a nontrivial sum to evaluate. Instead, we may use (1.3) *and* (1.6) *by noting that* $X = \sum_{i=1}^{n} X_i$, *where the* $\{X_1, X_2, \ldots, X_n\} \overset{\text{IID}}{\sim} \text{Bernoulli}(\theta)$, $E(X_i) = \theta$ *and* $V(X_i) = \theta(1-\theta)$:

$$E(X) = E(X_1 + X_2, \cdots, X_n) = E\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} E(X_i) = n\theta \ ,$$

$$V(X) = V\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} V(X_i) = \sum_{i=1}^{n} \theta(1-\theta) = n\theta(1-\theta) \ .$$

Simulation 8 (Binomial(n, θ)**)** *To simulate from a* Binomial(n, θ) *RV* X*, we can use Algorithm 5 with:*

$$\theta_0 = (1-\theta)^n, \qquad C(x+1) = \frac{\theta(n-x)}{(1-\theta)(x+1)}, \qquad \textit{Mean Efficiency: } O(1+n\theta)\ .$$

Similarly, with the appropriate θ_0 and $C(x+1)$, we can also simulate from the Geometric(θ) and Poisson(λ) RVs.

Labwork 2.4.2 *1. Implement Algorithm 5 via a function named* `MyGenDiscInvSampler` *in* `MATLAB`*. Hand in the* `M-file` *named* `MyGenDiscInvSampler.m` *giving detailed comments explaining your understanding of each step of the code. [Hint:* $C(i)$ *should be implemented as a function (use function handles via* `@`*) that can be passed as a parameter to the function* `MyGenDiscInvSampler`*].*

2. Show that your code works for drawing samples from a Binomial(n, p) *RV by doing the following:*

(a) Seed the fundamental sampler by your Student ID (if your ID is `11424620` *then type* `rand('twister', 11424620);`*)*

(b) Draw 100 samples from the Binomial$(n = 20, p = 0.5)$ *RV and report the results in an* 2×2 *table with column headings* `x` *and No. of observations. [Hint: the inputs* θ_0 *and* $C(i)$ *for the* Binomial(n, p) *RV is given above].*

3. Show that your code works for drawing samples from a Geometric(p) *RV by doing the following:*

(a) Seed the fundamental sampler by your Student ID.

(b) Set the variable `Mytheta=rand`*.*

(c) Draw 100 samples from the Geometric(`Mytheta`) *RV and report the sample mean. [Note: the inputs* θ_0 *and* $C(i)$ *for the* Geometric(θ) *RV should be derived and the workings shown].*

Chapter 3

Estimation

3.1 Introduction

The problem of estimation is of fundamental importance in statistical inference and learning. We will formalise the general estimation problem here after a brief introduction to statistics.

3.2 Statistics

Definition 4 (Data) *The function X measures the outcome ω of an experiment with sample space Ω [Often, the sample space is also denoted by S]. Formally, X is a random variable [or a random vector $X = (X_1, X_2, \ldots, X_n)$, i.e. a vector of random variables] taking values in the* **data space** *$\mathbb{X}$:*

$$X(\omega) : \Omega \mapsto \mathbb{X} \ .$$

The realisation of the RV X when an experiment is performed is the observation or data $x \in \mathbb{X}$. That is, when the experiment is performed once and it yields a specific $\omega \in \Omega$, the data $X(\omega) = x \in \mathbb{X}$ is the corresponding realisation of the RV X.

Figure 3.1: Sample Space, Random Variable, Realisation, Data, and Data Space.

Example 3.2.1 *For some given parameter* $\theta \in \Theta := [0,1]$*, consider* n *IID* Bernoulli(θ) *trials, i.e.* $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} \text{Bernoulli}(\theta)$*. Then the random vector* $X = (X_1, X_2, \ldots, X_n)$*, which takes values in the data space* $\mathbb{X} = \{0,1\}^n := \{(x_1, x_2, \ldots, x_n) : x_i \in \{0,1\},\ i = 1, 2, \ldots, n\}$*, made up of vertices of the* n*-dimensional hyper-cube, measures the outcomes of this experiment. A particular realisation of* X*, upon performance of this experiment, is the observation, data or data vector* $(x_1, x_2, \ldots, x_n)$*. For instance, if we observed* $n-1$ *tails and* 1 *heads, in that order, then our data vector* $(x_1, x_2, \ldots, x_{n-1}, x_n) = (0, 0, \ldots, 0, 1)$*.*

Definition 5 (Statistic) *A* **statistic** T *is any function of the data:*

$$T(x) : \mathbb{X} \mapsto \mathbb{T} \ .$$

Thus, a statistic T *is also an RV that takes values in the space* $\mathbb{T}$*. When* $x \in \mathbb{X}$ *is the realisation of an experiment, we let* $T(x) = t$ *denote the corresponding realisation of the statistic* T*. Sometimes we use* $T_n(X)$ *and* $\mathbb{T}_n$ *to emphasise that* X *is an* n*-dimensional random vector, i.e.* $\mathbb{X} \subset \mathbb{R}^n$

Classwork 3.2.2 *Is the RV* X*, for which the realisation is the observed data* $X(\omega) = x$*, a statistic? In other words, is the data a statistic? [Hint: consider the identity map* $T(x) = x : \mathbb{X} \mapsto \mathbb{T} = \mathbb{X}$*.]*

Next, we define two important statistics called the **sample mean** and **sample variance**. Since they are obtained from the sample data, they are called **sample moments**, as opposed to the **population moments**. The corresponding population moments are $E(X_1)$ and $V(X_1)$, respectively.

Definition 6 (Sample Mean) *From a given a sequence of RVs* $X_1, X_2, \ldots, X_n$*, we may obtain another RV called the* n*-samples mean or simply the sample mean:*

$$T_n(\ (X_1, X_2, \ldots, X_n)\ ) = \overline{X}_n(\ (X_1, X_2, \ldots, X_n)\ ) := \frac{1}{n}\sum_{i=1}^{n} X_i \ . \tag{3.1}$$

For brevity, we write

$$\overline{X}_n(\ (X_1, X_2, \ldots, X_n)\ ) \quad as \quad \overline{X}_n \ ,$$

and its realisation

$$\overline{X}_n(\ (x_1, x_2, \ldots, x_n)\ ) \quad as \quad \overline{x}_n \ .$$

Note that the expectation and variance of $\overline{X}_n$ are:

$$\begin{aligned} E(\overline{X}_n) &= E\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) && \text{[by definition (3.1)]} \\ &= \frac{1}{n}\sum_{i=1}^{n} E\left(X_i\right) && \text{[by property (1.3)]} \end{aligned}$$

Furthermore, if every X_i in the original sequence of RVs $X_1, X_2, \ldots$ is **identically** distributed with the same expectation, by convention $E(X_1)$, then:

$$E(\overline{X}_n) = \frac{1}{n}\sum_{i=1}^{n} E\left(X_i\right) = \frac{1}{n}\sum_{i=1}^{n} E\left(X_1\right) = \frac{1}{n}\ n\ E\left(X_1\right) = E\left(X_1\right) \ . \tag{3.2}$$

Similarly, we can show that:

$$\begin{aligned} V(\overline{X}_n) &= V\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) && \text{[by definition (3.1)]} \\ &= \left(\frac{1}{n}\right)^2 V\left(\sum_{i=1}^{n} X_i\right) && \text{[by property (1.5)]} \end{aligned}$$

Furthermore, if the original sequence of RVs $X_1, X_2, \ldots$ is **independently** distributed then:

$$V(\overline{X}_n) = \left(\frac{1}{n}\right)^2 V\left(\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\sum_{i=1}^{n} V(X_i) \qquad \text{[by property (1.6)]}$$

Finally, if the original sequence of RVs $X_1, X_2, \ldots$ is **independently and identically** distributed with the same variance ($V(X_1)$ by convention) then:

$$V(\overline{X}_n) = \frac{1}{n^2}\sum_{i=1}^{n} V(X_i) = \frac{1}{n^2}\sum_{i=1}^{n} V(X_1) = \frac{1}{n^2}\, n\, V(X_1) = \frac{1}{n}\, V(X_1) \ . \tag{3.3}$$

Labwork 3.2.3 *After initializing the fundamental sampler, we draw five samples and then obtain the sample mean using the* MATLAB *function* `mean`*. In the following, we will reuse the samples stored in the array* `XsFromUni01Twstr101`*.*

```
>> rand('twister',101); % initialise the fundamental Uniform(0,1) sampler
>> XsFromUni01Twstr101=rand(1,5); % simulate n=5 IID samples from Uniform(0,1) RV
>> SampleMean=mean(XsFromUni01Twstr101);% find sample mean
>> disp(XsFromUni01Twstr101); % The data-points x_1,x_2,x_3,x_4,x_5 are:
    0.5164    0.5707    0.0285    0.1715    0.6853
>> disp(SampleMean); % The Sample mean is :
    0.3945
```

We can thus use `mean` *to obtain the sample mean $\overline{x}_n$ of n sample points $x_1, x_2, \ldots, x_n$.*

Definition 7 (Sample Variance & Standard Deviation) *From a given a sequence of random variables $X_1, X_2, \ldots, X_n$, we may obtain another statistic called the n-samples variance or simply the sample variance :*

$$T_n(\ (X_1, X_2, \ldots, X_n)\ ) = S_n^2(\ (X_1, X_2, \ldots, X_n)\ ) := \frac{1}{n-1}\sum_{i=1}^{n} (X_i - \overline{X}_n)^2 \ . \tag{3.4}$$

For brevity, we write $S_n^2(\ (X_1, X_2, \ldots, X_n)\ )$ as S_n^2 and its realisation $S_n^2(\ (x_1, x_2, \ldots, x_n)\ )$ as s_n^2. Sample standard deviation is simply the square root of sample variance:

$$S_n(\ (X_1, X_2, \ldots, X_n)\ ) = \sqrt{S_n^2(\ (X_1, X_2, \ldots, X_n)\ )} \tag{3.5}$$

For brevity, we write $S_n(\ (X_1, X_2, \ldots, X_n)\ )$ as S_n and its realisation $S_n(\ (x_1, x_2, \ldots, x_n)\ )$ as s_n.

Once again, if $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} X_1$, the expectation of the sample variance is:

$$E(S_n^2) = V(X_1) \ .$$

Labwork 3.2.4 *We can compute the sample variance and sample standard deviation for the five samples stored in the array* `XsFromUni01Twstr101` *from Labwork 3.2.3 using* MATLAB*'s functions* `var` *and* `std`*, respectively.*

```
>> disp(XsFromUni01Twstr101); % The data-points x_1,x_2,x_3,x_4,x_5 are :
    0.5164    0.5707    0.0285    0.1715    0.6853
>> SampleVar=var(XsFromUni01Twstr101);% find sample variance
>> SampleStd=std(XsFromUni01Twstr101);% find sample standard deviation
>> disp(SampleVar) % The sample variance is:
    0.0785
>> disp(SampleStd) % The sample standard deviation is:
    0.2802
```

It is important to bear in mind that the statistics such as sample mean and sample variance are random variables and have an underlying distribution.

Definition 8 (Order Statistics) *Suppose $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} F$, where F is the DF from the set of all DFs over the real line. Then, the n-sample* **order statistics** *$X_{([n])}$ is:*

$$X_{([n])}(\ (X_1, X_2, \ldots, X_n)\ ) := \left(X_{(1)}, X_{(2)}, \ldots X_{(n)}\right), \text{ such that, } X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(n)}\ . \quad (3.6)$$

For brevity, we write $X_{([n])}(\ (X_1, X_2, \ldots, X_n)\ )$ as $X_{([n])}$ and its realisation $X_{([n])}(\ (x_1, x_2, \ldots, x_n)\ )$ as $x_{([n])} = (x_{(1)}, x_{(2)}, \ldots x_{(n)})$.

Without going into the details of how to sort the data in ascending order to obtain the order statistics (an elementary topic of an Introductory Computer Science course), we simply use MATLAB's function `sort` to obtain the order statistics, as illustrated in the following example.

Labwork 3.2.5 *The order statistics for the five samples stored in* `XsFromUni01Twstr101` *from Labwork 3.2.3 can be computed using* `sort` *as follows:*

```
>> disp(XsFromUni01Twstr101); % display the sample points
    0.5164    0.5707    0.0285    0.1715    0.6853
>> SortedXsFromUni01Twstr101=sort(XsFromUni01Twstr101); % sort data
>> disp(SortedXsFromUni01Twstr101); % display the order statistics
    0.0285    0.1715    0.5164    0.5707    0.6853
```

Therefore, we can use `sort` *to obtain our order statistics $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ from n sample points $x_1, x_2, \ldots, x_n$.*

Next, we will introduce a family of common statistics, called the q^{th} quantile, by first defining the function:

Definition 9 (Inverse DF or Inverse CDF or Quantile Function) *Let X be an RV with DF F. The* **inverse DF** *or* **inverse CDF** *or* **quantile function** *is:*

$$F^{[-1]}(q) := \inf\{x : F(x) > q\}, \quad \text{for some } q \in [0, 1]\ . \quad (3.7)$$

If F is strictly increasing and continuous then $F^{[-1]}(q)$ is the unique $x \in \mathbb{R}$ such that $F(x) = q$.

A **functional** is merely a function of another function. Thus, $T(F) : \{\text{All DFs }\} \mapsto \mathbb{T}$, being a map or function from the space of DFs to its range $\mathbb{T}$, is a functional. Some specific examples of functionals we have already seen include:

1. The **mean** of RV $X \sim F$ is a function of the DF F:
$$T(F) = E(X) = \int x \, dF(x) \ .$$
2. The **variance** of RV $X \sim F$ is a function of the DF F:
$$T(F) = E(X - E(X))^2 = \int (x - E(X))^2 \, dF(x) \ .$$
3. The **value of DF at a given** $x \in \mathbb{R}$ of RV $X \sim F$ is also a function of DF F:
$$T(F) = F(x) \ .$$

Other functionals of F that depend on the quantile function $F^{[-1]}$ are:

1. The q^{th} **quantile** of RV $X \sim F$:
$$T(F) = F^{[-1]}(q) \ \text{ where } q \in [0, 1] \ .$$
2. The **first quartile** or the 0.25^{th} **quantile** of the RV $X \sim F$:
$$T(F) = F^{[-1]}(0.25) \ .$$
3. The **median** or the **second quartile** or the 0.50^{th} **quantile** of the RV $X \sim F$:
$$T(F) = F^{[-1]}(0.50) \ .$$
4. The **third quartile** or the 0.75^{th} **quantile** of the RV $X \sim F$:
$$T(F) = F^{[-1]}(0.75) \ .$$

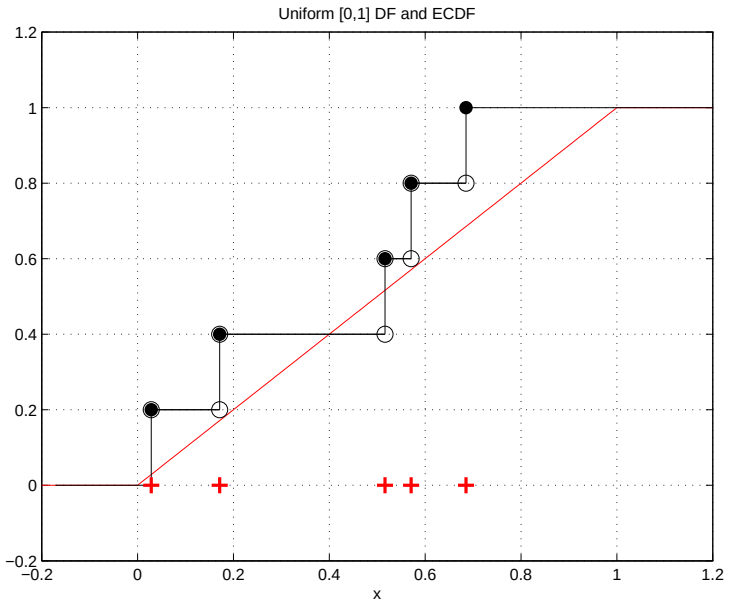
Definition 10 (Empirical Distribution Function (EDF or ECDF)) *Suppose we have n IID RVs,* $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} F$*, where* F *is a DF from the set of all DFs over the real line. Then, the n-sample empirical distribution function (EDF or ECDF) is the discrete distribution function* $\widehat{F}_n$ *that puts a probability mass of* $1/n$ *at each sample or data point* x_i*:*

$$\widehat{F}_n(x) = \frac{\sum_{i=1}^{n} \mathbb{1}(X_i \leq x)}{n} \ , \qquad \textit{where} \qquad \mathbb{1}(X_i \leq x) := \begin{cases} 1 & \textit{if } x_i \leq x \\ 0 & \textit{if } x_i > x \end{cases} \tag{3.8}$$

Labwork 3.2.6 *Let us plot the ECDF for the five samples drawn from the* $Uniform(0, 1)$ *RV in Labwork 3.2.3 using the* MATLAB *function* `ECDF` *(given in Labwork 5.0.11). Let us super-impose the samples and the true DF as depicted in Figure 3.2 with the following script:*

plotunifecdf.m
```
xs = -1:0.01:2; % vector xs from -1 to 2 with increment .05 for x values
% get the [0,1] uniform DF or cdf of xs in vector cdf
cdf=zeros(size(xs));% initialise cdf as zero
indices = find(xs>=1); cdf(indices) = 1; % set cdf as 1 when xs >= 1
indices = find(xs>=0 & xs<=1); cdf(indices)=xs(indices); % cdf=xs when 0 <= xs <= 1
plot(xs,cdf,'r') % plot the DF
hold on; title('Uniform [0,1] DF and ECDF'); xlabel('x'); axis([-0.2 1.2 -0.2 1.2])
x=[0.5164, 0.5707, 0.0285, 0.1715, 0.6853]; % five samples
plot(x,zeros(1,5),'r+','LineWidth',2,'MarkerSize',10)% plot the data as red + marks
hold on; grid on; % turn on grid
ECDF(x,1,.2,.6);% ECDF (type help ECDF) plot is extended to left and right by .2 and .4, respectively.
```

Figure 3.2: Plot of the DF of Uniform$(0,1)$, five IID samples from it, and the ECDF based on the five samples. Note that the ECDF $\widehat{F}_5$ for data points $x = (x_1, x_2, x_3, x_4, x_5) = (0.5164, 0.5707, 0.0285, 0.1715, 0.6853)$ jumps by $1/5 = 0.20$ at each of the five samples.



Definition 11 (q^{th} Sample Quantile) *For some $q \in [0,1]$ and n IID RVs $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} F$, we can obtain the ECDF $\widehat{F}_n$ using* (3.8)*. The* q^{th} **sample quantile** *is defined as the statistic (statistical functional):*

$$T(\widehat{F}_n) = \widehat{F}_n^{[-1]}(q) := \inf\left\{x : \widehat{F}_n^{[-1]}(x) \geq q\right\} . \tag{3.9}$$

By replacing q in this definition of the q^{th} sample quantile by 0.25*,* 0.5 *or* 0.75*, we obtain the first, second (***sample median***) or third* **sample quartile***, respectively.*

The following algorithm can be used to obtain the q^{th} sample quantile of n IID samples $(x_1, x_2, \ldots, x_n)$ on the basis of their order statistics $(x_{(1)}, x_{(2)}, \ldots, x_{(n)})$.

The q^{th} sample quantile, $\widehat{F}_n^{[-1]}(q)$, is found by interpolation from the order statistics $(x_{(1)}, x_{(2)}, \ldots, x_{(n)})$ of the n data points $(x_1, x_2, \ldots, x_n)$, using the formula:

$$\widehat{F}_n^{[-1]}(q) = (1-\delta)x_{(i+1)} + \delta x_{(i+2)}, \quad \text{where,} \quad i = \lfloor (n-1)q \rfloor \quad \text{and} \quad \delta = (n-1)q - \lfloor (n-1)q \rfloor .$$

Thus, the **sample minimum** of the data points $(x_1, x_2, \ldots, x_n)$ is given by $\widehat{F}_n^{[-1]}(0)$, the **sample maximum** is given by $\widehat{F}_n^{[-1]}(1)$ and the **sample median** is given by $\widehat{F}_n^{[-1]}(0.5)$, etc.

Labwork 3.2.7 *Use the implementation of Algorithm 6 in Labwork 5.0.12 as the* MATLAB *function* `qthSampleQuantile` *to find the q^{th} sample quantile of two simulated data arrays:*

1. `SortedXsFromUni01Twstr101`*, the order statistics that was constructed in Labwork 3.2.5 and*
2. *Another sorted array of* 7 *samples called* `SortedXs`

Algorithm 6 q^{th} Sample Quantile of Order Statistics

input:

1. q in the q^{th} sample quantile, i.e. the argument q of $\widehat{F}_n^{[-1]}(q)$,
2. order statistic $(x_{(1)}, x_{(2)}, \ldots, x_{(n)})$, i.e. the sorted $(x_1, x_2, \ldots, x_n)$, where $n > 0$.

output: $\widehat{F}_n^{[-1]}(q)$, the q^{th} sample quantile
$i \leftarrow \lfloor (n-1)q \rfloor$
$\delta \leftarrow (n-1)q - i$
if $i = n - 1$ **then**
  $\widehat{F}_n^{[-1]}(q) \leftarrow x_{(i+1)}$
else
  $\widehat{F}_n^{[-1]}(q) \leftarrow (1-\delta)x_{(i+1)} + \delta x_{(i+2)}$
end if
return: $\widehat{F}_n^{[-1]}(q)$

```
>> disp(SortedXsFromUni01Twstr101)
    0.0285    0.1715    0.5164    0.5707    0.6853
>> rand('twister',420);
>> SortedXs=sort(rand(1,7));
>> disp(SortedXs)
    0.1089    0.2670    0.3156    0.3525    0.4530    0.6297    0.8682
>> for q=[0, 0.25, 0.5, 0.75, 1.0]
       disp([q, qthSampleQuantile(q,SortedXsFromUni01Twstr101) ...
                qthSampleQuantile(q,SortedXs)])
   end
         0    0.0285    0.1089
    0.2500    0.1715    0.2913
    0.5000    0.5164    0.3525
    0.7500    0.5707    0.5414
    1.0000    0.6853    0.8682
```

3.3 Convergence of Random Variables

This important topic is concerned with the limiting behavior of sequences of RVs

$$\{X_i\}_{i=1}^n := X_1, X_2, X_3, \ldots, X_{n-1}, X_n \qquad \text{as } n \to \infty \ .$$

From a statistical viewpoint, $n \to \infty$ is associated with the amount of data or information tending to ∞. Readers should ensure that they are familiar with the notions of convergence, limits and continuity in the real line before proceeding further.

Consider the class of discrete RVs with distributions that place all probability mass on a single real number. This is the probability model for the deterministic real variable.

Model 11 (Point Mass(θ)) *Given a specific point $\theta \in \mathbb{R}$, we say an RV X has point mass at θ or is* Point Mass(θ) *distributed if the DF is:*

$$F(x;\theta) = \begin{cases} 0 & \text{if } x < \theta \\ 1 & \text{if } x \geq \theta \end{cases} \tag{3.10}$$

and the PMF is:

$$f(x;\theta) = \begin{cases} 0 & \text{if } x \neq \theta \\ 1 & \text{if } x = \theta \end{cases} \tag{3.11}$$

Thus, Point Mass(θ) RV X is deterministic in the sense that every realisation of X is exactly equal to $\theta \in \mathbb{R}$. We will see that this distribution plays a central limiting role in asymptotic statistics.

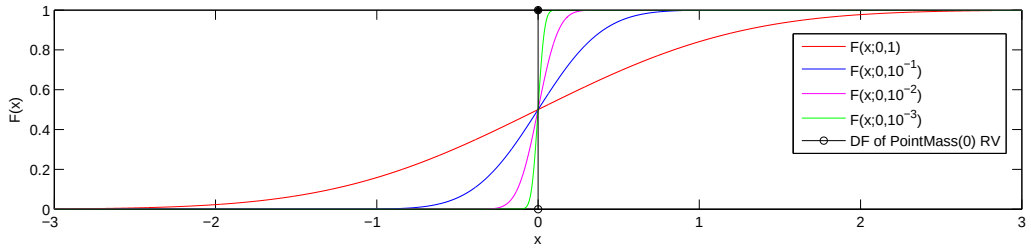
Example 3.3.1 (Mean and variance of Point Mass(θ) **RV)** *Let* $X \sim$ Point Mass(θ). *Then:*

$$E(X) = \sum_x x f(x) = \theta \times 1 = \theta \ , \qquad V(X) = E(X^2) - (E(X))^2 = \theta^2 - \theta^2 = 0 \ .$$

Classwork 3.3.2 *Suppose you are given an independent sequence of RVs* $\{X_i\}_{i=1}^n$, *where* $X_i \sim$ Normal$(0, 1/i)$. *How would you talk about the convergence of* $X_n \sim$ Normal$(0, 1/n)$ *as* n *approaches* ∞*? Figure 3.3 shows that the probability mass of* X_n *increasingly concentrates about* 0 *as* n *approaches* ∞ *and the variance* $1/n$ *approaches* 0. *Based on this observation, can we expect* $\lim_{n\to\infty} X_n = X$, *where the limiting RV* $X \sim$ Point Mass(0) *?*

The answer is **no**. *This is because* $P(X_n = X) = 0$ *for any* n, *since* $X \sim$ Point Mass(0) *is a discrete RV with exactly one outcome* 0, *and* $X_n \sim$ Normal$(0, 1/n)$ *is a continuous RV for every* n, *however large. In other words, a continuous RV, such as* X_n, *has* 0 *probability of realising any single real number in its support, such as* 0.

Figure 3.3: Distribution functions of several Normal(μ, σ^2) RVs for $\sigma^2 = 1, \frac{1}{10}, \frac{1}{100}, \frac{1}{1000}$.



Thus, we need more sophisticated notions of convergence for sequences of RVs. Two such notions are formalised next as they are necessary for a clear understanding of three basic propositions in Statistics :

1. Weak Law of Large Numbers,
2. Central Limit Theorem,
3. Gilvenko-Cantelli Theorem.

Definition 12 (Convergence in Distribution) *Let* $X_1, X_2, \ldots,$ *be a sequence of RVs and let* X *be another RV. Let* F_n *denote the DF of* X_n *and* F *denote the DF of* X. *We say that* X_n *converges to* X *in distribution, and write:*

$$X_n \rightsquigarrow X$$

if for any real number t *at which* F *is continuous,*

$$\lim_{n\to\infty} F_n(t) = F(t) \qquad \textit{[in the sense of limits in the real line } \mathbb{R}\textit{]}$$

Definition 13 (Convergence in Probability) *Let $X_1, X_2, \ldots,$ be a sequence of RVs and let X be another RV. Let F_n denote the DF of X_n and F denote the DF of X. We say that X_n converges to X in probability, and write:*

$$X_n \xrightarrow{P} X$$

if for every real number $\epsilon > 0$,

$$\lim_{n\to\infty} P(|X_n - X| > \epsilon) = 0 \qquad \textit{[in the sense of limits in the real line } \mathbb{R}\textit{]}$$

Let us revisit the problem of convergence in Classwork 3.3.2 with these new concepts.

Example 3.3.3 *Suppose you are given an independent sequence of RVs $\{X_i\}_{i=1}^n$, where $X_i \sim$ Normal$(0, 1/i)$ with DF F_n, and let $X \sim$ Point Mass(0) with DF F. We can formalise our observation in Classwork 3.3.2 that X_n is concentrating about 0 as $n \to \infty$ by the statement:*

$$X_n \textit{ is converging in distribution to } X\textit{, i.e. } X_n \rightsquigarrow X.$$

Regarding the same sequence of RVs in Classwork 3.3.2 and Example 3.3.3, we are tempted to ask whether $X_n \sim$ Normal$(0, 1/n)$ converges in probability to $X \sim$ Point Mass(0), i.e. whether $X_n \xrightarrow{P} X$. We need some elementary inequalities in probability to help us answer this question.

Example 3.3.4 *Does the sequence of RVs $X_n \sim$ Normal$(0, 1/n)$ converge in probability to $X \sim$ Point Mass(0), i.e. does $X_n \xrightarrow{P} X$?*

To find out if $X_n \xrightarrow{P} X$, we need to show that for any $\epsilon > 0$, $\lim_{n\to\infty} P(|X_n - X| > \epsilon) = 0$.

Let ϵ be any real number greater than 0, then:

$$\begin{aligned} P(|X_n| > \epsilon) &= P(|X_n|^2 > \epsilon^2) \\ &= \frac{E(X_n^2)}{\epsilon^2} \qquad \textit{[by Markov's Inequality: } P(X \geq \epsilon) \leq \frac{E(X)}{\epsilon}, \quad \textit{for any} \quad \epsilon > 0 \textit{]} \\ &= \frac{\frac{1}{n}}{\epsilon^2} \to 0, \quad \textit{as} \quad n \to \infty \end{aligned}$$

Hence, we have shown that for any $\epsilon > 0$, $\lim_{n\to\infty} P(|X_n - X| > \epsilon) = 0$ and therefore by Definition 13, $X_n \xrightarrow{P} X$ or $X_n \xrightarrow{P} 0$.

Convention: *When X has a* Point Mass(θ) *distribution and $X_n \xrightarrow{P} X$, we simply write $X_n \xrightarrow{P} \theta$.*

Now that we have been introduced to two notions of convergence for RV sequences, we can begin to appreciate the basic limit theorems used in statistical inference.

3.4 Point Estimation

Point estimation is any statistical methodology that provides one with a "**single best guess**" of some specific quantity of interest. Traditionally, we denote this **quantity of interest as θ^*** and **its point estimate as $\widehat{\theta}$ or $\widehat{\theta}_n$**. The subscript n in the point estimate $\widehat{\theta}_n$ emphasises that our estimate is based on n observations or data points from a given statistical experiment to estimate θ^*. This quantity of interest, which is usually unknown, can be:

- a **parameter** θ^* which is an element of the **parameter space** $\mathbf{\Theta}$, denoted $\theta \in \mathbf{\Theta}$,
- a **distribution function (DF)** $F^* \in \mathbb{F} :=$ the set of all DFs
- a **regression function** $g^* \in \mathbb{G}$, where $\mathbb{G}$ is a class of regression functions.

Recall that a statistic is an RV $T(X)$ that maps every data point x in the data space $\mathbb{X}$ with $T(x) = t$ in its range $\mathbb{T}$, i.e. $T(x) : \mathbb{X} \mapsto \mathbb{T}$ (Definition 5). Next, we look at a specific class of statistics whose range is the parameter space $\mathbf{\Theta}$.

Definition 14 *A* **point estimator** $\widehat{\Theta}$ *of some* **fixed and possibly unknown** $\theta^* \in \mathbf{\Theta}$ *is a statistic that associates each data point* $x \in \mathbb{X}$ *with an estimate* $\widehat{\Theta}(x) = \widehat{\theta} \in \mathbf{\Theta}$*,*

$$\widehat{\Theta} := \widehat{\Theta}(x) = \widehat{\theta} : \mathbb{X} \mapsto \mathbf{\Theta} \ .$$

If our data point $x := (x_1, x_2, \ldots, x_n)$ *is an* n*-vector or a point in the* n*-dimensional real space, i.e.* $x := (x_1, x_2, \ldots, x_n) \in \mathbb{X}_n \subset \mathbb{R}^n$*, then we emphasise the dimension* n *in our point estimator* $\widehat{\Theta}_n$ *of* $\theta^* \in \mathbf{\Theta}$*.*

$$\widehat{\Theta}_n := \widehat{\Theta}_n(x := (x_1, x_2, \ldots, x_n)) = \widehat{\theta}_n : \mathbb{X}_n \mapsto \mathbf{\Theta}, \quad \mathbb{X}_n \subset \mathbb{R}^n \ .$$

The typical situation for us involves point estimation of $\theta^* \in \mathbf{\Theta}$ *on the basis of one realisation* $x \in \mathbb{X}_n \subset \mathbb{R}^n$ *of an independent and identically distributed (IID) random vector* $X = (X_1, X_2, \ldots, X_n)$*, such that* $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} X_1$ *and the DF of* X_1 *is* $F(x_1; \theta^*)$*, i.e. the distribution of the IID RVs,* $X_1, X_2, \ldots, X_n$*, is parameterised by* $\theta^* \in \mathbf{\Theta}$*.*

3.4.1 Some Properties of Point Estimators

Given that an estimator is merely a function from the data space to the parameter space, we need choose only the best estimators available. Recall that a point estimator $\widehat{\Theta}_n$, being a statistic or an RV of the data has a probability distribution over its range $\mathbf{\Theta}$. This distribution over $\mathbf{\Theta}$ is called the **sampling distribution** of $\widehat{\Theta}_n$. Note that the sampling distribution not only depends on the statistic $\widehat{\Theta}_n := \widehat{\Theta}_n(X_1, X_2, \ldots, X_n)$ but also on θ^* which in turn determines the distribution of the IID data vector $(X_1, X_2, \ldots, X_n)$. The following definitions are useful for selecting better estimators.

Definition 15 (Bias of a Point Estimator) *The* bias_n *of an estimator* $\widehat{\Theta}_n$ *of* $\theta^* \in \mathbf{\Theta}$ *is:*

$$\mathsf{bias}_n = \mathsf{bias}_n(\widehat{\Theta}_n) := E_{\theta^*}(\widehat{\Theta}_n) - \theta^* \ . \tag{3.12}$$

We say that the estimator $\widehat{\Theta}_n$ *is* **unbiased** *if* $\mathsf{bias}_n(\widehat{\Theta}_n) = 0$ *or if* $E_{\theta^*}(\widehat{\Theta}_n) = \theta^*$ *for every* n*. If* $\lim_{n\to\infty} \mathsf{bias}_n(\widehat{\Theta}_n) = 0$*, we say that the estimator is* **asymptotically unbiased**.

Since the expectation of the sampling distribution of the point estimator $\widehat{\Theta}_n$ depends on the unknown θ^*, we emphasise the θ^*-dependence by $E_{\theta^*}(\widehat{\Theta}_n)$.

Definition 16 (Standard Error of a Point Estimator) *The standard deviation of the point estimator* $\widehat{\Theta}_n$ *of* $\theta^* \in \mathbf{\Theta}$ *is called the* **standard error***:*

$$\mathsf{se}_n = \mathsf{se}_n(\widehat{\Theta}_n) = \sqrt{V_{\theta^*}(\widehat{\Theta}_n)} \ . \tag{3.13}$$

Since the variance of the sampling distribution of the point estimator $\widehat{\Theta}_n$ depends on the fixed and possibly unknown θ^*, as emphasised by V_{θ^*} in (3.13), the se_n is also a possibly unknown quantity and may itself be estimated from the data. The estimated standard error, denoted by $\widehat{\mathsf{se}}_n$, is calculated by replacing $V_{\theta^*}(\widehat{\Theta}_n)$ in (3.13) with its appropriate estimate.

Another reasonable property of an estimator is that it converge to the "true" parameter θ^*, as we gather more and more IID data from a θ^*-specified DF $F(x;\theta^*)$. This property is stated precisely next.

Definition 17 (Asymptotic Consistency of a Point Estimator) *A point estimator $\widehat{\Theta}_n$ of $\theta^* \in \Theta$ is said to be* **consistent** *if $\widehat{\Theta}_n \xrightarrow{P} \theta^*$.*

Definition 18 (Mean Squared Error (MSE) of a Point Estimator) *Often, the quality of a point estimator $\widehat{\Theta}_n$ of $\theta^* \in \Theta$ is assessed by the* **mean squared error** *or MSE_n defined by:*

$$\mathsf{MSE}_n = \mathsf{MSE}_n(\widehat{\Theta}_n) := E_{\theta^*}\left((\widehat{\Theta}_n - \theta^*)^2\right) . \tag{3.14}$$

The following proposition shows a simple relationship between the mean square error, bias and variance of an estimator $\widehat{\Theta}_n$ of θ^*.

Proposition 3 (The $\sqrt{\mathsf{MSE}_n} : \mathsf{se}_n : \mathsf{bias}_n$–Sided Right Triangle of an Estimator) *Let $\widehat{\Theta}_n$ be an estimator of $\theta^* \in \Theta$. Then:*

$$\mathsf{MSE}_n(\widehat{\Theta}_n) = (\mathsf{se}_n(\widehat{\Theta}_n))^2 + (\mathsf{bias}_n(\widehat{\Theta}_n))^2 . \tag{3.15}$$

Proposition 4 *Let $\widehat{\Theta}_n$ be an estimator of $\theta^* \in \Theta$. Then, if $\mathsf{bias}_n(\widehat{\Theta}_n) \to 0$ and $\mathsf{se}_n(\widehat{\Theta}_n) \to 0$ as $n \to \infty$, the estimator $\widehat{\Theta}_n$ is asymptotically consistent:*

$$\widehat{\Theta}_n \xrightarrow{P} \theta^* .$$

We want our estimator to be unbiased with small standard errors as the sample size n gets large. The **point estimator** $\widehat{\Theta}_n$ will then produce a **point estimate**:

$$\widehat{\Theta}_n((x_1, x_2, \ldots, x_n)) = \widehat{\theta} \in \Theta,$$

on the basis of the **observed data** $(x_1, x_2, \ldots, x_n)$ that is close to the **true parameter** $\theta^* \in \Theta$.

Next we look at two specific point estimators, namely, moment estimators (MMEs) and maximum likelihood estimators (MLEs).

3.4.2 Moment Estimator (MME)

The way in which you have been making probability statements in problems from Chapters 3 and 4 of Ang & Tang involved the solving of **moment equations** for parameters by substituting the **sample moments** for the **population moments**. Let us formalise this general estimation procedure.

See notes scribed on the board from class and tutorials (Use Table 6.1 of Ang & Tang to solve for MME of parameters in common distributions).

3.4.3 Maximum Likelihood Estimator (MLE)

See notes from class.

Example 3.4.4 (Coin Tossing Experiment ($X_1, \ldots, X_n \overset{IID}{\sim} \text{Bernoulli}(\theta^*)$)) *I tossed a coin that has an unknown probability θ^* of landing Heads independently and identically 10 times in a row. Four of my outcomes were Heads and the remaining six were Tails, with the actual sequence of Bernoulli outcomes (Heads $\mapsto 1$ and Tails $\mapsto 0$) being $(1,0,0,0,1,1,0,0,1,0)$. I would like to estimate the probability $\theta^* \in \Theta = [0,1]$ of observing Heads using the maximum likelihood estimator or MLE $\widehat{\Theta}_n((X_1, X_2, \ldots, X_n))$ of θ. We derive the MLE next.*

First, the likelihood function is:

$$\begin{aligned} L(\theta) &:= L(x_1, x_2, \ldots, x_n; \theta) = P(x_1, x_2, \ldots, x_n|\theta) \\ &= P(x_1|\theta)P(x_2|\theta)\cdots P(x_n|\theta) := \prod_{i=1}^{n} P(x_i|\theta) \\ &= \theta^{\sum_{i=1}^n x_i}(1-\theta)^{n-\sum_{i=1}^n x_i} := \theta^{t_n}(1-\theta)^{n-t_n} \end{aligned}$$

In the last step, we have formally defined the following statistic of the data:

$$T_n(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} X_i : \mathbb{X}_n \to \mathbb{T}_n$$

with the corresponding realisation $t_n := T_n(x_1, x_2, \ldots, x_n) = \sum_{i=1}^n x_i \in \mathbb{T}_n$. Let us now take the natural logarithm of both sides:

$$\log(L(\theta)) := \log(L(x_1, x_2, \ldots, x_n; \theta)) = \log\left(\theta^{t_n}(1-\theta)^{n-t_n}\right) = t_n \log(\theta) + (n - t_n)\log(1-\theta)$$

Next, we take the derivative with respect to the parameter θ:

$$\begin{aligned} \frac{\partial}{\partial\theta}\log(L(\theta)) &= \frac{\partial}{\partial\theta} t_n \log(\theta) + \frac{\partial}{\partial\theta}(n - t_n)\log(1-\theta) \\ &= \frac{t_n}{\theta} - \frac{n - t_n}{1-\theta} \end{aligned}$$

Now, set $\frac{\partial}{\partial\theta}\log(L(\theta)) = 0$ and solve for θ to obtain the maximum likelihood estimate $\widehat{\theta}_n$:

$$\frac{\partial}{\partial\theta}\log(L(\theta)) = 0 \iff \frac{t_n}{\theta} = \frac{n-t_n}{1-\theta} \iff \frac{1-\theta}{\theta} = \frac{n-t_n}{t_n} \iff \frac{1}{\theta} - 1 = \frac{n}{t_n} - 1 \iff \widehat{\theta}_n = \frac{t_n}{n}$$

Therefore the MLE is:

$$\widehat{\Theta}_n(X_1, X_2, \ldots, X_n) = \frac{1}{n} T_n(X_1, X_2, \ldots, X_n) = \frac{1}{n}\sum_{i=1}^{n} X_i = \overline{X}_n$$

For the coin tossing experiment I just performed ($n = 10$ times), the point estimate of θ is:

$$\begin{aligned} \widehat{\theta}_{10} = \widehat{\Theta}_{10}((x_1, x_2, \ldots, x_{10})) &= \widehat{\Theta}_{10}((1,0,0,0,1,1,0,0,1,0)) \\ &= \frac{1+0+0+0+1+1+0+0+1+0}{10} = \frac{4}{10} = 0.40 \ . \end{aligned}$$

Practical Excursion in One-dimensional Optimisation

Numerically maximising a log-likelihood function of one parameter is a useful technique. This can be used for models with no analytically known MLE. A fairly large field of maths, called optimisation, exists for this sole purpose. Conventionally, in optimisation, one is interested in minimisation. Therefore, the basic algorithms are cast in the "find the minimiser and the minimum" of a target function $f : \mathbb{R} \mapsto \mathbb{R}$. Since we are interested in maximising our target, which is the likelihood or log-likelihood function, say $\log(L(x_1, \ldots, x_n; \theta)) : \Theta \mapsto \mathbb{R}$, we will simply apply the standard optimisation algorithms directly to $-\log(L(x_1, \ldots, x_n; \theta)) : \Theta \mapsto \mathbb{R}$.

The algorithm implemented in `fminbnd` is based on the golden section search and an inverse parabolic interpolation, and attempts to find the minimum of a function of one variable within a given fixed interval. Briefly, the golden section search proceeds by successively **bracketing** the minimum of the target function within an acceptably small interval inside the given starting interval [see Section 8.2 of Forsythe, G. E., M. A. Malcolm, and C. B. Moler, 1977, *Computer Methods for Mathematical Computations*, Prentice-Hall]. MATLAB's `fminbnd` also relies on Brent's inverse parabolic interpolation [see Chapter 5 of Brent, Richard. P., 1973, *Algorithms for Minimization without Derivatives*, Prentice-Hall, Englewood Cliffs, New Jersey]. Briefly, additional smoothness conditions are assumed for the target function to aid in a faster bracketing strategy through polynomial interpolations of past function evaluations. MATLAB's `fminbnd` has several limitations, including:

- The likelihood function must be continuous.
- Only local MLE solutions, i.e. those inside the starting interval, are given.
- One needs to know or carefully guess the starting interval that contains the MLE.
- MATLAB's `fminbnd` exhibits slow convergence when the solution is on a boundary of the starting interval.

Labwork 3.4.5 (Coin-tossing experiment) *The following script was used to study the coin-tossing experiment in* MATLAB. *The plot of the log-likelihood function and the numerical optimisation of MLE are carried out using* MATLAB*'s built-in function* `fminbnd` *(See Figure 3.4).*

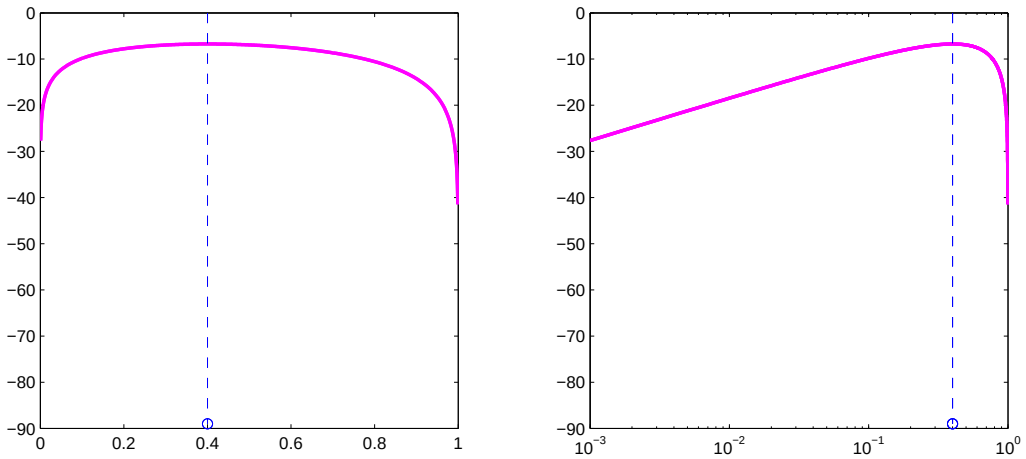
BernoulliMLE.m

```
% To simulate n coin tosses, set theta=probability of heads and n
% Then draw n IID samples from Bernoulli(theta) RV
% theta=0.5; n=20; x=floor(rand(1,n) + theta);
% enter data from a real coin tossing experiment
x=[1 0 0 0 1 1 0 0 1 0]; n=length(x);
t = sum(x); % statistic t is the sum of the x_i values
% display the outcomes and their sum
display(x)
display(t)

% Analyticaly MLE is t/n
MLE=t/n
% l is the log-likelihood of data x as a function of parameter theta
l=@(theta)sum(log(theta ^ t * (1-theta)^(n-t)));
ThetaS=[0:0.001:1]; % sample some values for theta

% plot the log-likelihood function and MLE in two scales
subplot(1,2,1);
plot(ThetaS,arrayfun(l,ThetaS),'m','LineWidth',2);
hold on; stem([MLE],[-89],'b--'); % plot MLE as a stem plot
subplot(1,2,2);
```

```
semilogx(ThetaS,arrayfun(l,ThetaS),'m','LineWidth',2);
hold on; stem([MLE],[-89],'b--'); % plot MLE as a stem plot

% Now we will find the MLE by finding the minimiser or argmin of -l
% negative log-likelihood function of parameter theta
negl=@(theta)-sum(log(theta ^ t * (1-theta)^(n-t)));
% read help fminbnd
% you need to supply the function to be minimised and its search interval
% NumericalMLE = fminbnd(negl,0,1)
% to see the iteration in the numerical minimisation
 NumericalMLE = fminbnd(negl,0,1,optimset('Display','iter'))
```

```
>> BernoulliMLE
x =     1     0     0     0     1     1     0     0     1     0
t =     4
MLE =    0.4000
Func-count     x          f(x)         Procedure
    1       0.381966      6.73697        initial
    2       0.618034      7.69939        golden
    3       0.236068       7.3902        golden
    4       0.408979      6.73179        parabolic
    5       0.399339      6.73013        parabolic
    6       0.400045      6.73012        parabolic
    7       0.400001      6.73012        parabolic
    8       0.399968      6.73012        parabolic
Optimisation terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04
NumericalMLE =   0.4000
```

Figure 3.4: Plot of $\log(L(1,0,0,0,1,1,0,0,1,0;\theta))$ as a function of the parameter θ over the parameter space $\Theta = [0,1]$ and the MLE $\widehat{\theta}_{10}$ of 0.4 for the coin-tossing experiment.



Labwork 3.4.6 *Recall Labwork 2.2.2 where you modeled the arrival of buses at a bus stop using the IID* Exponential($\lambda^* = 0.1$) *distributed inter-arrival times with a mean of* $1/\lambda^* = 10$ *minutes. Once again, seed the fundamental sampler by your Student ID (e.g. if your ID is* `11424620` *then type* `rand('twister', 11424620);`*), just before simulating the inter-arrival times of the next seven buses. Hand in the following six items:*

1. *Waiting times* $x_1, x_2, \ldots, x_7$ *between arrivals of the next seven buses at your ID-seeded bus stop;*

2. *A plot of the empirical DF $\widehat{F}_n$ from your (simulated) data $x_1, x_2, \ldots, x_7$. [You may use the* MATLAB *function* `ECDF` *of Labwork 5.0.11)];*

3. *The first, second and third sample quartiles as well as the 0.20^{th} sample quantile for your data $x_1, x_2, \ldots, x_7$. [You may use the* MATLAB *function* `qthSampleQuantile` *of Labwork 5.0.12];*

4. *Pretending that you did not know the true parameter ($\lambda^* = 0.1$) used in the simulation, produce the maximum likelihood estimate (ML estimate) $\widehat{\lambda}_7$ from your seven observations $x_1, x_2, \ldots, x_7$;*

5. *Plot the log-likelihood function for your data $x_1, x_2, \ldots, x_7$ as a function of the parameter λ; and*

6. *Show that you have verified that the numerical optimisation routine* `fminbnd` *returns the correct ML estimate $\widehat{\lambda}_7$.*

Summarizing Table of Point Estimators

Using the sample mean $\overline{X}_n$ and sample standard deviation S_n defined in (3.1) and (3.5), respectively, we summarise the two point estimators of the parameters of some common distributions below. For some cases, the MLE is the same as the MME and can be solved analytically.

Table 3.1: Summary of the Method of Moment Estimator (MME) and the Maximum Likelihood Estimator (MLE) for some IID Experiments.

Statistical Experiment	MLE	MME
$X_1, X_2, \ldots, X_n \overset{IID}{\sim}$ Bernoulli(θ)	$\widehat{\theta} = \overline{X}_n$	same as MLE
$X_1, X_2, \ldots, X_n \overset{IID}{\sim}$ Exponential(λ)	$\widehat{\lambda} = 1/\overline{X}_n$	same as MLE
$X_1, X_2, \ldots, X_n \overset{IID}{\sim}$ Normal(μ, σ^2)	$\widehat{\mu} = \overline{X}_n, \widehat{\sigma} = \sqrt{\frac{n-1}{n} S_n^2}$	$\widehat{\mu} = \overline{X}_n, \widehat{\sigma} = S_n$
$X_1, X_2, \ldots, X_n \overset{IID}{\sim}$ Lognormal(λ, ζ)	$\widehat{\lambda} = \frac{1}{n}\sum_{i=1}^n \log(X_i)$ $\widehat{\zeta} = \sqrt{\frac{1}{n}\sum_{i=1}^n (\log(X_i) - \widehat{\lambda})^2}$	$\widehat{\lambda} = \log(\overline{X}_n) - \frac{1}{2}\widehat{\zeta}^{\,2}$ $\widehat{\zeta} = \sqrt{\log\left(S_n^2/\overline{X}_n^2 + 1\right)}$

Example 3.4.7 (Homework 3.4.7 of Assignment 9 from 6.7 on page 275 of Ang & Tang) *The distribution of ocean wave heights, H, may be modeled with the* Raleigh(α) *RV with parameter α and probability density function,*

$$f(h;\alpha) = \frac{h}{\alpha^2} \exp\left(-\frac{1}{2}(h/\alpha)^2\right), \qquad h \in \mathbb{H} := [0, \infty) \ .$$

The parameter space for alpha is $\mathbb{A} = (0, \infty)$. Suppose that the following measurements $h_1, h_2, \ldots, h_{10}$ of wave heights in meters were observed to be

$$1.50,\ 2.80,\ 2.50,\ 3.20,\ 1.90,\ 4.10,\ 3.60,\ 2.60,\ 2.90,\ 2.30 \ ,$$

respectively. Under the assumption that the 10 samples are IID realisations from a Raleigh(α^*) *RV with a fixed and unknown parameter α^*, find the ML estimate $\widehat{\alpha}_{10}$ of α^*.*

We first obtain the log-likelihood function of α for the data $h_1, h_2, \ldots, h_n \overset{IID}{\sim} \text{Raleigh}(\alpha)$.

$$\begin{aligned}\ell(\alpha) := \log(L(h_1, h_2, \ldots, h_n; \alpha)) &= \log\left(\prod_{i=1}^{n} f(h_i; \alpha)\right) = \sum_{i=1}^{n} \log(f(h_i; \alpha)) \\ &= \sum_{i=1}^{n} \log\left(\frac{h_i}{\alpha^2} e^{-\frac{1}{2}(h_i/\alpha)^2}\right) = \sum_{i=1}^{n} \left(\log(h_i) - 2\log(\alpha) - \frac{1}{2}(h_i/\alpha)^2\right) \\ &= \sum_{i=1}^{n} (\log(h_i)) - 2n\log(\alpha) - \sum_{i=1}^{n} \left(\frac{1}{2} h_i^2 \alpha^{-2}\right)\end{aligned}$$

Now, let us take the derivative with respect to α,

$$\begin{aligned}\frac{\partial}{\partial \alpha}(\ell(\alpha)) &:= \frac{\partial}{\partial \alpha}\left(\sum_{i=1}^{n} (\log(h_i)) - 2n\log(\alpha) - \sum_{i=1}^{n} \left(\frac{1}{2} h_i^2 \alpha^{-2}\right)\right) \\ &= \frac{\partial}{\partial \alpha}\left(\sum_{i=1}^{n} (\log(h_i))\right) - \frac{\partial}{\partial \alpha}(2n\log(\alpha)) - \frac{\partial}{\partial \alpha}\left(\sum_{i=1}^{n} \left(\frac{1}{2} h_i^2 \alpha^{-2}\right)\right) \\ &= 0 - 2n\frac{1}{\alpha} - \sum_{i=1}^{n} \left(\frac{1}{2} h_i^2 (-2\alpha^{-3})\right) = -2n\alpha^{-1} + \alpha^{-3} \sum_{i=1}^{n} \left(h_i^2\right)\end{aligned}$$

Next, we set the derivative to 0, solve for α, and set the solution equal to the ML estimate $\widehat{\alpha}_n$.

$$\begin{aligned}0 = \frac{\partial}{\partial \alpha}(\ell(\alpha)) &\iff 0 = -2n\alpha^{-1} + \alpha^{-3} \sum_{i=1}^{n} h_i^2 \iff 2n\alpha^{-1} = \alpha^{-3} \sum_{i=1}^{n} h_i^2 \\ &\iff 2n\alpha^{-1}\alpha^3 = \sum_{i=1}^{n} h_i^2 \iff \alpha^2 = \frac{1}{2n} \sum_{i=1}^{n} h_i^2 \iff \widehat{\alpha}_n = \sqrt{\frac{1}{2n} \sum_{i=1}^{n} h_i^2}\end{aligned}$$

Therefore, the ML estimate of the unknown $\alpha^* \in \mathbb{A}$ on the basis of our 10 observations $h_1, h_2, \ldots, h_{10}$ of wave heights is

$$\begin{aligned}\widehat{\alpha}_{10} &= \sqrt{\frac{1}{2 * 10} \sum_{i=1}^{10} h_i^2} \\ &= \sqrt{\frac{1}{20}(1.50^2 + 2.80^2 + 2.50^2 + 3.20^2 + 1.90^2 + 4.10^2 + 3.60^2 + 2.60^2 + 2.90^2 + 2.30^2)} \cong 2\end{aligned}$$

Labwork 3.4.8 *Recall labwork 2.2.8 where you simulated 1000 samples directly from the RV C (in part 3.). Pretend that you do not know the true parameters used in this particular simulation from RV C and do the following:*

1. *Store the first 10, the first 100 and all 1000 samples in data arrays named* `x10`, `x100` *and* `x1000` *[Please don't do this manually!].*
2. *Report the MLE of the two parameters for each of the three sub-arrays of data, namely* `x10`, `x100` *and* `x1000`, *i.e. report the six point estimates: $\widehat{\lambda}_{10}, \widehat{\zeta}_{10}, \widehat{\lambda}_{100}, \widehat{\zeta}_{100}, \widehat{\lambda}_{1000}, \widehat{\zeta}_{1000}$.*
3. *Report the MME of the two parameters for each of the three sub-arrays of data, namely* `x10`, `x100` *and* `x1000`, *i.e. report the six point estimates: $\widehat{\lambda}_{10}, \widehat{\zeta}_{10}, \widehat{\lambda}_{100}, \widehat{\zeta}_{100}, \widehat{\lambda}_{1000}, \widehat{\zeta}_{1000}$.*

4. *Discuss in a short paragraph what you can deduce from the two sets of point estimates. Explain how the maximum likelihood (ML) and method of moments (MM) estimates of the parameters are related to the true parameters used in the simulation as the sample size increases in powers of* 10.

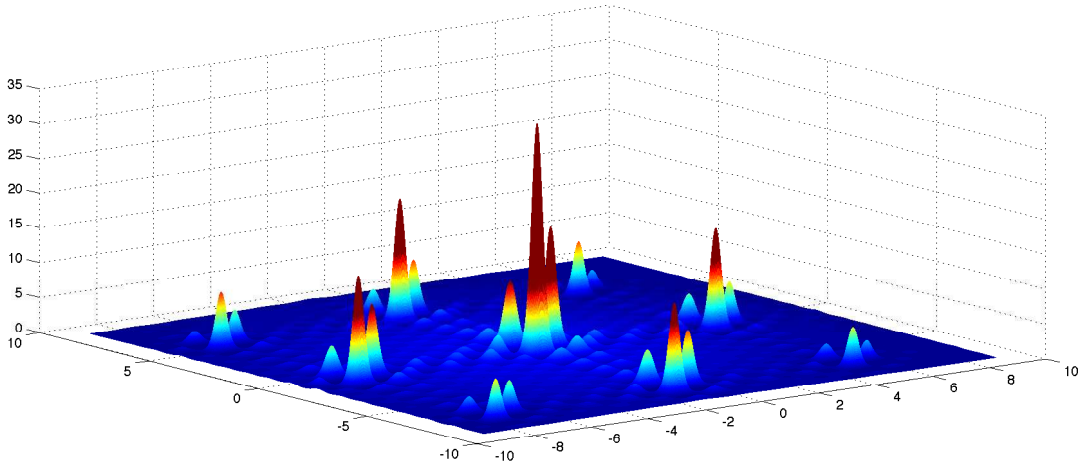
5. **There is no credit for this part***: Try to use* `fminsearch` *to numerically find the MLE for your data and make a 2D-plot of the log-likelihood function being maximised.*

Practical Excursion in Multi-dimensional Optimisation

The basic idea involves multi-dimensional iterations that attempt to converge on a local maximum close to the starting vector $\theta^{(0)} \in \Theta$ (our initial guess). We can employ MATLAB's built-in function `fminsearch` to find the MLE of vector-valued parameters such as in the Lognormal model with two parameters, i.e. $\theta = (\lambda, \zeta) \in \Theta \subset \mathbb{R}^2$. The function `fminsearch` is similar to `fminbnd` except that it handles a given function of many variables, and the user specifies a starting vector $\theta^{(0)}$ rather than a starting interval. We illustrate the use of `fminsearch` on a more challenging target called the Levy density:

$$f(x,y) = \exp\left(-\frac{1}{50}\left(\left(\sum_{i=1}^{5} i\cos\left((i-1)x+i\right)\right)\left(\sum_{j=1}^{5} j\cos\left((j+1)y+j\right)\right) + (x+1.42513)^2 + (y+0.80032)^2\right)\right)$$

Figure 3.5: Plot of Levy density as a function of the parameter $(x, y) \in [-10, 10]^2$ scripted in Labwork 5.0.13.



`fminsearch` uses the simplex search method [Nelder, J.A., and Mead, R. 1965, Computer Journal, vol. 7, p. 308-313]. For an animation of the method and more details, please visit `http://en.wikipedia.org/wiki/Nelder-Mead_method`. An advantage of the method is that it does not use numerical (finite differencing) or analytical (closed-form expressions) gradients but relies on a direct search method. Briefly, the simplex algorithm tries to "tumble and shrink" a simplex towards the local valley of the function to be minimised. If k is the dimension of the parameter space or domain of the function to be optimised, a k-dimensional simplex is specified by its $k+1$ distinct vertices

each of dimension k. Thus, a simplex is a triangle in a two-dimensional space and a pyramid in a three-dimensional space. At each iteration of the algorithm:

1. A new point inside or nearby the current simplex is proposed.
2. The function's value at the newly proposed point is compared with its values at the vertices of the simplex.
3. One of the vertices is typically replaced by the proposed point, giving rise to a new simplex.
4. The first three steps are repeated until the diameter of the simplex is less than the specified tolerance.

A major limitation of `fminsearch`, as demonstrated with the Levy target (encoded in Labwork 5.0.14) is that it can only give local solutions. The **global maximiser** of the Levy function $f(x, y)$ is $(-1.3069, -1.4249)$ and the **global maximum** is $f(-1.3069, -1.4249) = 33.8775$ For instance, if we start the search close to, say $(x^{(0)}, y^{(0)}) = (-1.3, -1.4)$, as shown below, then the simplex algorithm converges as desired to the solution $(-1.3068, -1.4249)$.

```
>> [params, fvalue, exitflag, output] = fminsearch('NegLevyDensity',[-1.3 -1.4],options)
params =   -1.3068   -1.4249
fvalue =  -33.8775
exitflag =     1
output =
    iterations: 24
     funcCount: 46
     algorithm: 'Nelder-Mead simplex direct search'
       message: [1x194 char]
```

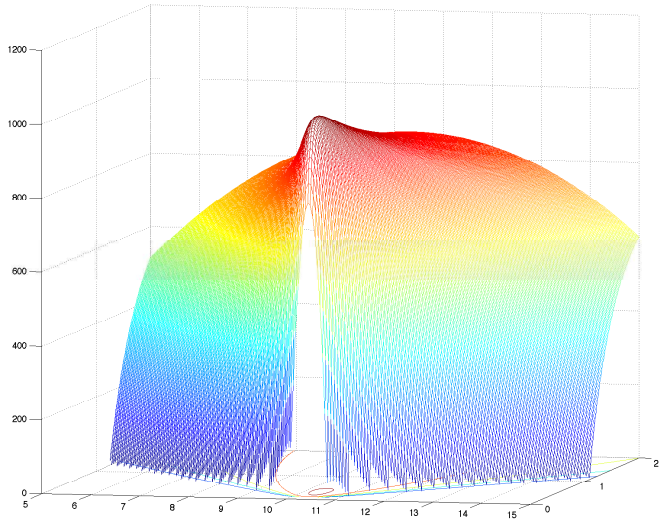
However, if we start the search further away, say $(x^{(0)}, y^{(0)}) = (1.3, 1.4)$, as shown below, then the algorithm converges to the **local maximiser** $(1.1627, 1.3093)$ with a **local maximum** value of $f(1.1627, 1.3093) = 0.9632$, which is clearly smaller than the global maximum of 33.8775.

```
>> [params, fvalue, exitflag, output] = fminsearch('NegLevyDensity',[1.3 1.4],options)
params = 1.1627    1.3093
fvalue = -0.9632
exitflag = 1
output =
   iterations: 29
     funcCount: 57
     algorithm: 'Nelder-Mead simplex direct search'
       message: [1x194 char]
```

Therefore, we have to be extremely careful when using point-valued, iterative, local optimisation algorithms, implemented in floating-point arithmetic to find the global maximum. Other examples of such algorithms include:

- **Conjugate Gradient Method**:
  `http://en.wikipedia.org/wiki/Conjugate_gradient_method`
- **Broyden-Fletcher-Goldfarb-Shanno (BFGS) method**:
  `http://en.wikipedia.org/wiki/BFGS_method`
- **Simulated Annealing**:
  `http://en.wikipedia.org/wiki/Simulated_annealing`

Figure 3.6: Plot of the "well-behaved" (uni-modal and non-spiky) $\log(L((x_1, x_2, \ldots, x_{100}); \lambda, \zeta))$, based on 100 samples $(x_1, x_2, \ldots, x_{100})$ drawn from the Lognormal$(\lambda^* = 10.36, \zeta^* = 0.26)$ as per Labwork 5.0.15.



In general, we have no guarantee that the output of such local optimisation routines will indeed be the global optimum. In practice, you can start the search at several distinct starting points and choose the best local maximum from the lot.

When the target function is "well-behaved," i.e. uni-modal or single-peaked and not too spiky, the optimisation routine can be expected to perform well. Log-likelihood functions are often well-behaved. Let us generate 100 samples from an RV $C \sim$ Lognormal$(\lambda^* = 10.36, \zeta^* = 0.26)$ by exponentiating the samples from the Normal$(10.36, 0.26^2)$ RV, and then compute the corresponding MMEs and MLEs for parameters (λ, ζ) using the formulae in Table 3.1.

```
>> rand('twister',001); % set the fundamental sampler
>> % draw 100 samples from the Lognormal(10.36,0.26) RV
>> Cs = exp(arrayfun(@(u)(Sample1NormalByNewRap(u,10.36,0.26^2)),rand(1,100)));
>> MLElambdahat = mean(log(Cs)) % maximum likelihood estimate of lambda
MLElambdahat =   10.3397
>> MLEzetahat = sqrt(mean( (log(Cs)-MLElambdahat) .^ 2)) % max. lkl. estimate of zeta
MLEzetahat =    0.2744
>> MMEzetaahat = sqrt(log(var(Cs)/(mean(Cs)^2) + 1)) % moment estimate of zeta
MMEzetaahat =    0.2624
>> MMElambdahat = log(mean(Cs))-(0.5*MMEzetaahat^2) % moment estimate of lambda
MMElambdahat =   10.3417
```

Let us try to apply the simplex algorithm to find the MLE numerically. We first encode the negative log-likelihood function of the parameters $(\lambda, \zeta) \in (0, \infty)^2$ for the given data x, as follows:

NegLogNormalLogLkl.m

```
function l = NegLogNormalLogLkl(x,params)
% Returns the -log likelihood of [lambda zeta]=exp(params)
% for observed data vector x=(x_1,...,x_n) ~IID LogNormal(lambda, zeta).
% We define lambda and zeta as exp(params) to allow for unconstrained
% minimisation by fminsearch and respect the positive domain constraints
```

```
% for Lambda and zeta. So in the end we re-transform, i.e. [lambda zeta]=exp(params)
% lambda=params(1); zeta=params(1);
lambda=exp(params(1)); zeta=exp(params(2));
% minus Log-likelihood function
l = -sum(log((1 ./ (sqrt(2*pi)*zeta) .* x) .* exp((-1/(2*zeta^2))*(log(x)-lambda).^2)));
```

Here is how we can call `fminsearch` and find the MLE after the re-transformation.

```
>> [params, fvalue, exitflag, output] = ...
fminsearch(@(params)(NegLogNormalLogLkl(Cs,params)),[log(5), log(1)])
params =    2.3360   -1.2931
fvalue =  -1.0214e+03
exitflag =     1
output =
    iterations: 74
     funcCount: 131
     algorithm: 'Nelder-Mead simplex direct search'
       message: [1x194 char]
>> % But we want exp(params) since we defined lambda and zeta as exp(params)
exp(params)
ans =   10.3397    0.2744
```

Note that the MLEs $(\widehat{\lambda}_{100}, \widehat{\zeta}_{100}) = (10.3397, 0.2744)$ from 74 iterations or "tumbles" of the 'Nelder-Mead simplex (triangle)' and the MLEs agree well with the direct evaluations `MLElambdahat` and `MLEzetahat` based on the formulae in Table 3.1.

3.5 Confidence Sets

As we saw in Section 3.4, the point estimate $\widehat{\theta}_n$ is a "single best guess" of the fixed and possibly unknown parameter $\theta^* \in \mathbf{\Theta}$. However, if we wanted to make a statement about our confidence in an estimation procedure, then one possibility is to produce subsets from the parameter space $\mathbf{\Theta}$ called **confidence sets** that "engulf" θ^* with a probability of at least $1 - \alpha$.

Formally, an $\mathbf{1 - \alpha}$ **confidence interval** for the parameter $\theta^* \in \mathbf{\Theta} \subset \mathbb{R}$, based on n observations or data points $X_1, X_2, \ldots, X_n$, is an interval C_n that is a function of the data:

$$C_n := [\underline{C}_n, \overline{C}_n] = [\underline{C}_n(X_1, X_2, \ldots, X_n), \overline{C}_n(X_1, X_2, \ldots, X_n)] \ ,$$

such that:

$$P_{\theta^*}\left(\theta^* \in C_n := [\underline{C}_n, \overline{C}_n]\right) \geq 1 - \alpha \ .$$

Note that the confidence interval $C_n := [\underline{C}_n, \overline{C}_n]$ is a two-dimensional RV or a random vector in $\mathbb{R}^2$ that depends on the two statistics $\underline{C}_n(X_1, X_2, \ldots, X_n)$ and $\overline{C}_n(X_1, X_2, \ldots, X_n)$, as well as θ^*, which in turn determines the distribution of the data $(X_1, X_2, \ldots, X_n)$. In words, C_n engulfs the true parameter $\theta^* \in \mathbf{\Theta}$ with a probability of at least$1 - \alpha$. We call $1 - \alpha$ as the **coverage** of the confidence interval C_n.

Formally, a $1-\alpha$ **confidence set** C_n for a vector-valued $\theta^* \in \mathbf{\Theta} \subset \mathbb{R}^k$ is any subset of $\mathbf{\Theta}$ such that $P_{\theta^*}(\theta^* \in C_n) \geq 1 - \alpha$. The typical forms taken by C_n are k-dimensional boxes or hyper-cuboids, hyper-ellipsoids and subsets defined by inequalities involving level sets of some estimator of θ^*.

Typically, we take $\alpha = 0.05$ because we are interested in the $1 - \alpha = 0.95$ or 95% confidence interval/set $C_n \subset \mathbf{\Theta}$ of $\theta^* \in \mathbf{\Theta}$ from an estimator $\widehat{\Theta}_n$ of θ^*.

Proposition 5 (Central Limit Theorem (CLT)) *Let* $X_1, X_2, \ldots \overset{\text{IID}}{\sim} X_1$ *and suppose* $E(X_1)$ *and* $V(X_1)$ *exists. Then:*

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i \quad \rightsquigarrow \quad X \sim \text{Normal}\left(E(X_1), \frac{V(X_1)}{n}\right) , \tag{3.16}$$

$$\overline{X}_n - E(X_1) \quad \rightsquigarrow \quad X - E(X_1) \sim \text{Normal}\left(0, \frac{V(X_1)}{n}\right) , \tag{3.17}$$

$$\sqrt{n}\left(\overline{X}_n - E(X_1)\right) \quad \rightsquigarrow \quad \sqrt{n}\left(X - E(X_1)\right) \sim \text{Normal}\left(0, V(X_1)\right) , \tag{3.18}$$

$$Z_n := \frac{\overline{X}_n - E(\overline{X}_n)}{\sqrt{V(\overline{X}_n)}} = \frac{\sqrt{n}\left(\overline{X}_n - E(X_1)\right)}{\sqrt{V(X_1)}} \quad \rightsquigarrow \quad Z \sim \text{Normal}\left(0, 1\right) , \tag{3.19}$$

$$\lim_{n\to\infty} P\left(\frac{\overline{X}_n - E(\overline{X}_n)}{\sqrt{V(\overline{X}_n)}} \leq z\right) = \lim_{n\to\infty} P(Z_n \leq z) \quad = \quad \Phi(z) := \int_{-\infty}^{z} \left(\frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right)\right) dx . \tag{3.20}$$

Thus, for sufficiently large sample size n $(n > 30)$ *we can make the following approximation:*

$$P\left(\frac{\overline{X}_n - E(\overline{X}_n)}{\sqrt{V(\overline{X}_n)}} \leq z\right) \approxeq P(Z \leq z) = \Phi(z) := \int_{-\infty}^{z} \left(\frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right)\right) dx . \tag{3.21}$$

Proof: *See any intermediate to advanced undergraduate text in probability.* ■

Heuristic Interpretation of CLT: *Probability statements about the sample mean RV* $\overline{X}_n$ *can be approximated using a* Normal *distribution.*

Let us look at an example that makes use of the CLT next.

Example 3.5.1 (Errors in computer code (Wasserman03, p. 78)) *Suppose the collection of RVs* $X_1, X_2, \ldots, X_n$ *model the number of errors in* n *computer programs named* $1, 2, \ldots, n$*, respectively. Suppose that the RV* X_i *modelling the number of errors in the* i^{th} *program is the* $Poisson(\lambda^* = 5)$ *for any* $i = 1, 2, \ldots, n$*. Also suppose that they are independently distributed. In short, we suppose that:*

$$X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} \text{Poisson}(\lambda^* = 5) .$$

Suppose we have $n = 125$ *programs and want to make a probability statement about* $\overline{X}_n$ *which is the average number of errors per program out of these* 125 *programs. Since* $E(X_i) = \lambda^* = 5$ *and* $V(X_i) = \lambda^* = 5$*, we may want to know how often our sample mean* $\overline{X}_{125}$ *differs from the expectation of* 5 *errors per program. Using the CLT, we can approximate* $P(\overline{X}_n < 5.5)$*, for instance, as follows:*

$$\begin{aligned}
P(\overline{X}_n < 5.5) &= P\left(\frac{\sqrt{n}(\overline{X}_n - E(X_1))}{\sqrt{V(X_1)}} < \frac{\sqrt{n}(5.5 - E(X_1))}{\sqrt{V(X_1)}}\right) \\
&\approxeq P\left(Z < \frac{\sqrt{n}(5.5 - \lambda^*)}{\sqrt{\lambda^*}}\right) \qquad \textit{[by (3.21), and } E(X_1) = V(X_1) = \lambda^* \textit{]} \\
&= P\left(Z < \frac{\sqrt{125}(5.5 - 5)}{\sqrt{5}}\right) \qquad \textit{[Since, } \lambda^* = 5 \textit{ and } n = 125 \textit{ in this Example]} \\
&= P(Z \leq 2.5) = \Phi(2.5) = \int_{-\infty}^{2.5} \left(\frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right)\right) dx \approxeq 0.993790334674224 .
\end{aligned}$$

To obtain the final number in this approximation, we need the following:

Labwork 3.5.2 *The numerical approximation of* $\Phi(2.5)$ *was obtained via the call shown below to our* erf*-based* `NormalCdf` *function from 5.0.9. We could have also found it from a pre-computed Table for* $\Phi(x)$.

```
>> format long
>> disp(NormalCdf(2.5,0,1))
   0.993790334674224
```

The CLT says that if $X_1, X_2, \ldots \overset{\text{IID}}{\sim} X_1$ then $Z_n := \sqrt{n}(\overline{X}_n - E(X_1))/\sqrt{V(X_1)}$ is approximately distributed as Normal$(0,1)$. In Example 3.5.1, we knew $\sqrt{V(X_1)}$ since we assumed knowledge of $\lambda^* = 5$. However, in general, we may not know $\sqrt{V(X_1)}$. The next proposition says that we may estimate $\sqrt{V(X_1)}$ using the sample standard deviation S_n of $X_1, X_2, \ldots, X_n$, according to (3.5), and still make probability statements about the sample mean $\overline{X}_n$ using a Normal distribution, **provided n is not too small**, for e.g. $n \geq 30$.

Proposition 6 (CLT based on Sample Variance) *Let* $X_1, X_2, \ldots \overset{\text{IID}}{\sim} X_1$ *and suppose* $E(X_1)$ *and* $V(X_1)$ *exists, then:*

$$\frac{\sqrt{n}\left(\overline{X}_n - E(X_1)\right)}{S_n} \rightsquigarrow \text{Normal}(0,1) \ . \tag{3.22}$$

The following property of an estimator makes the task of producing confidence intervals straightforward.

Definition 19 (Asymptotic Normality of Estimators) *An estimator* $\widehat{\Theta}_n$ *of a fixed and possibly unknown parameter* $\theta^* \in \Theta$ *is* **asymptotically normal** *if:*

$$\frac{\widehat{\Theta}_n - \theta^*}{\mathsf{se}_n} \rightsquigarrow \text{Normal}(0,1) \ . \tag{3.23}$$

That is, $\widehat{\Theta}_n \rightsquigarrow \text{Normal}(\theta^*, \mathsf{se}_n^2)$. *By a further estimation of* $\mathsf{se}_n := \sqrt{V_\theta(\widehat{\Theta}_n)}$ *by* $\widehat{\mathsf{se}}_n$, *we can see that* $\widehat{\Theta}_n \rightsquigarrow \text{Normal}(\theta^*, \widehat{\mathsf{se}}_n^2)$ *on the basis of* (3.22).

Proposition 7 (Normal-based Asymptotic Confidence Interval) *Suppose an estimator* $\widehat{\Theta}_n$ *of parameter* $\theta^* \in \Theta \subset \mathbb{R}$ *is asymptotically normal:*

$$\widehat{\Theta}_n \rightsquigarrow \text{Normal}(\theta^*, \widehat{\mathsf{se}}_n^2) \ .$$

Let the RV $Z \sim \text{Normal}(0,1)$ *have DF* Φ *and inverse DF* Φ^{-1}. *Let:*

$$z_{\alpha/2} = \Phi^{-1}(1-(\alpha/2)), \quad \textit{that is,} \quad P(Z > z_{\alpha/2}) = \alpha/2 \ \textit{ and } \ P(-z_{\alpha/2} < Z < z_{\alpha/2}) = 1-\alpha \ .$$

Then:

$$P_{\theta^*}(\theta^* \in C_n) = P\left(\theta^* \in [\widehat{\Theta}_n - z_{\alpha/2}\widehat{\mathsf{se}}_n, \widehat{\Theta}_n + z_{\alpha/2}\widehat{\mathsf{se}}_n]\right) \to 1-\alpha \ .$$

Therefore:

$$C_n := [\underline{C}_n, \overline{C}_n] = [\widehat{\Theta}_n - z_{\alpha/2}\widehat{\mathsf{se}}_n, \widehat{\Theta}_n + z_{\alpha/2}\widehat{\mathsf{se}}_n]$$

is the $1-\alpha$ *Normal-based asymptotic confidence interval that relies on the asymptotic normality of the estimator* $\widehat{\Theta}_n$ *of* $\theta^* \in \Theta \subset \mathbb{R}$.

Proof: *Define the centralised and scaled estimator as* $Z_n := (\widehat{\Theta}_n - \theta^*)/\widehat{\mathsf{se}}_n$. *By assumption,* $Z_n \rightsquigarrow Z \sim \text{Normal}(0,1)$. *Therefore,*

$$\begin{aligned} P_{\theta^*}(\theta^* \in C_n) &= P_{\theta^*}\left(\theta^* \in [\widehat{\Theta}_n - z_{\alpha/2}\widehat{\mathsf{se}}_n, \widehat{\Theta}_n + z_{\alpha/2}\widehat{\mathsf{se}}_n]\right) \\ &= P_{\theta^*}\left(\widehat{\Theta}_n - z_{\alpha/2}\widehat{\mathsf{se}}_n \leq \theta^* \leq \widehat{\Theta}_n + z_{\alpha/2}\widehat{\mathsf{se}}_n\right) \\ &= P_{\theta^*}\left(-z_{\alpha/2}\widehat{\mathsf{se}}_n \leq \widehat{\Theta}_n - \theta^* \leq z_{\alpha/2}\widehat{\mathsf{se}}_n\right) \\ &= P_{\theta^*}\left(-z_{\alpha/2} \leq \frac{\widehat{\Theta}_n - \theta^*}{\widehat{\mathsf{se}}_n} \leq z_{\alpha/2}\right) \\ &\to P_{\theta^*}\left(-z_{\alpha/2} \leq Z \leq z_{\alpha/2}\right) \\ &= 1 - \alpha \end{aligned}$$

■

Figure 3.7: Density and Confidence Interval of the Asymptotically Normal Point Estimator

For 95% *confidence intervals,* $\alpha = 0.05$ *and* $z_{\alpha/2} = z_{0.025} = 1.96 \approxeq 2$. *This leads to the* **approximate 95% confidence interval** *of* $\widehat{\theta}_n \pm 2\widehat{\mathsf{se}}_n$, *where* $\widehat{\theta}_n = \widehat{\Theta}_n(x_1, x_2, \ldots, x_n)$ *and* $x_1, x_2, \ldots, x_n$ *are the data or observations of the RVs* $X_1, X_2, \ldots, X_n$.

Example 3.5.3 (Confidence interval for θ^* from n Bernoulli(θ^*) trials) *Let* $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} \text{Bernoulli}(\theta^*)$ *for some fixed but unknown parameter* $\theta^* \in \mathbb{\Theta} = [0,1]$. *Consider the following point estimator of* θ^**:*

$$\widehat{\Theta}_n((X_1, X_2, \ldots, X_n)) = n^{-1}\sum_{i=1}^{n} X_i \ .$$

That is, we take the sample mean of the n *IID* Bernoulli(θ^*) *trials to be our point estimator of* $\theta^* \in [0,1]$. *Then,* **this estimator is unbiased** *since:*

$$E_{\theta^*}(\widehat{\Theta}_n) = E_{\theta^*}\left(n^{-1}\sum_{i=1}^{n} X_i\right) = n^{-1}E_{\theta^*}\left(\sum_{i=1}^{n} X_i\right) = n^{-1}\sum_{i=1}^{n} E_{\theta^*}(X_i) = n^{-1}n\theta^* = \theta^* \ .$$

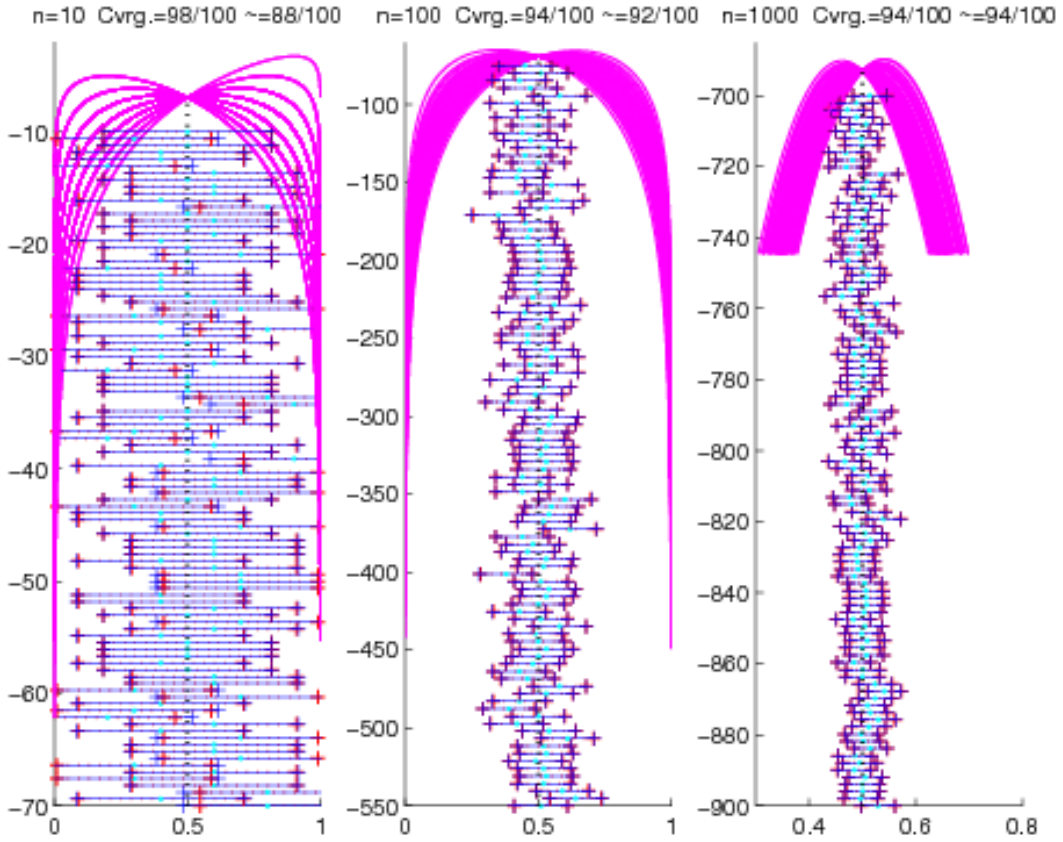
The above computations should remind you that the statistic:

$$T_n((X_1, X_2, \ldots, X_n)) := n\,\widehat{\Theta}_n((X_1, X_2, \ldots, X_n)) = \sum_{i=1}^{n} X_i$$

is the Binomial(n, θ^*) *RV. The standard error* se_n *of this estimator is:*

$$\mathsf{se}_n = \sqrt{V_{\theta^*}(\widehat{\Theta}_n)} = \sqrt{V_{\theta^*}\left(\sum_{i=1}^{n}\frac{X_i}{n}\right)} = \sqrt{\left(\sum_{i=1}^{n}\frac{1}{n^2}V_{\theta^*}(X_i)\right)} = \sqrt{\frac{n}{n^2}V_{\theta^*}(X_i)} = \sqrt{\theta^*(1-\theta^*)/n} \ .$$

Figure 3.8: 100 realisations of $C_{10}, C_{100}, C_{1000}$ based on samples of size $n = 10$, 100 and 1000 drawn from the Bernoulli($\theta^* = 0.5$) RV as per Labwork 5.0.16. The MLE $\widehat{\theta}_n$ (cyan dot) and the log-likelihood function (magenta curve) for each of the 100 replications of the experiment for each sample size n are depicted. The approximate normal-based 95% confidence intervals with blue boundaries are based on the exact $\mathsf{se}_n = \sqrt{\theta^*(1-\theta^*)/n} = \sqrt{1/4}$, while those with red boundaries are based on the estimated $\widehat{\mathsf{se}}_n = \sqrt{\widehat{\theta}_n(1-\widehat{\theta}_n)/n}$. The fraction of times the true parameter $\theta^* = 0.5$ was engulfed by the exact and approximate confidence interval (empirical coverage) over the 100 replications of the experiment for each of the three sample sizes are given by the numbers after `Cvrg.=` and `~=`, above each sub-plot, respectively.



Since θ^* is unknown, we obtain the estimated standard error $\widehat{\mathsf{se}}_n$ from the point estimate $\widehat{\theta}_n$ of θ^* on the basis of n observed data points $x = (x_1, x_2, \ldots, x_n)$ of the experiment:

$$\widehat{\mathsf{se}}_n = \sqrt{\widehat{\theta}_n(1-\widehat{\theta}_n)/n}, \quad \textit{where,} \quad \widehat{\theta}_n = \widehat{\Theta}_n((x_1, x_2, \ldots, x_n)) = n^{-1}\sum_{i=1}^{n} x_i \ .$$

By the central limit theorem, $\widehat{\Theta}_n \rightsquigarrow \text{Normal}(\theta^*, \widehat{\mathsf{se}}_n)$, i.e. $\widehat{\Theta}_n$ is asymptotically normal. Therefore, an asymptotically (for large sample size n) approximate $1-\alpha$ normal-based confidence interval is:

$$\widehat{\theta}_n \pm z_{\alpha/2}\widehat{\mathsf{se}}_n = \widehat{\theta}_n \pm z_{\alpha/2}\sqrt{\frac{\widehat{\theta}_n(1-\widehat{\theta}_n)}{n}} := \left[\widehat{\theta}_n - z_{\alpha/2}\sqrt{\frac{\widehat{\theta}_n(1-\widehat{\theta}_n)}{n}} \ , \ \widehat{\theta}_n + z_{\alpha/2}\sqrt{\frac{\widehat{\theta}_n(1-\widehat{\theta}_n)}{n}}\right]$$

Finally, since $\mathsf{bias}_n(\widehat{\Theta}_n) = 0$ *for any* n *and* $\mathsf{se}_n = \sqrt{\theta^*(1-\theta^*)/n} \to 0$, *as* $n \to \infty$, *by Proposition 4,* $\widehat{\Theta}_n \xrightarrow{P} \theta^*$. *That is* $\widehat{\Theta}_n$ *is an* **asymptotically consistent estimator** *of* θ^*. *Thus, we can make the width of the confidence interval as small as we want by making the number of observations or sample size* n *as large as we can.*

The confidence Interval for the coin tossing experiment in Example 3.4.4 with the observed sequence of Bernoulli outcomes (Heads $\mapsto 1$ *and Tails* $\mapsto 0$*) being* $(1,0,0,0,1,1,0,0,1,0)$. *We estimated the probability* θ^* *of observing Heads with the* **unbiased, asymptotically consistent estimator** $\widehat{\Theta}_n((X_1, X_2, \ldots, X_n)) = n^{-1}\sum_{i=1}^n X_i$ *of* θ^*. *The point estimate of* θ^* *was:*

$$\begin{aligned}\widehat{\theta}_{10} = \widehat{\Theta}_{10}((x_1, x_2, \ldots, x_{10})) &= \widehat{\Theta}_{10}((1,0,0,0,1,1,0,0,1,0)) \\ &= \frac{1+0+0+0+1+1+0+0+1+0}{10} = \frac{4}{10} = 0.40\ .\end{aligned}$$

The normal-based confidence interval for θ^* *may not be a valid approximation here with just* $n = 10$ *samples. Nevertheless, we will compute a* 95% *normal-based confidence interval:*

$$C_{10} = 0.40 \pm 1.96\sqrt{\frac{0.40(1-0.40)}{10}} = 0.40 \pm 0.3036 = [0.0964, 0.7036]$$

with a width of 0.6072. *When I increased the sample size* n *of the experiment from* 10 *to* 100 *by tossing the same coin another* 90 *times, I discovered that a total of* 57 *trials landed as Heads. Thus my point estimate and confidence interval for* θ^* *are:*

$$\widehat{\theta}_{100} = \frac{57}{100} = 0.57 \qquad \textit{and} \qquad C_{100} = 0.57 \pm 1.96\sqrt{\frac{0.57(1-0.57)}{100}} = 0.57 \pm 0.0495 = [0.5205, 0.6195]$$

with a much smaller width of 0.0990. *Thus our confidence interval shrank considerably from a width of* 0.6072 *after an additional* 90 *Bernoulli trials.*

3.5.4 Properties of the Maximum Likelihood Estimator

Next, we list some nice properties of the ML Estimator $\widehat{\Theta}_n$ for the fixed and possibly unknown $\theta^* \in \mathbf{\Theta}$.

1. The ML Estimator is asymptotically consistent, i.e. $\widehat{\Theta}_n \xrightarrow{P} \theta^*$.
2. The ML Estimator is asymptotically normal, i.e. $(\widehat{\Theta}_n - \theta^*)/\widehat{se}_n \rightsquigarrow \text{Normal}(0,1)$.
3. The estimated standard error of the ML Estimator, $\widehat{\mathsf{se}}_n$, can usually be computed analytically using the **Fisher Information**.
4. Because of the previous two properties, the $1-\alpha$ confidence interval can also be computed analytically as $\widehat{\Theta}_n \pm z_{\alpha/2}\widehat{\mathsf{se}}_n$.
5. The ML Estimator is **equivariant**, i.e. $\widehat{\psi}_n = g(\widehat{\theta}_n)$ is the ML Estimate of $\psi^* = g(\theta^*)$, for some smooth function $g(\theta) = \psi : \mathbf{\Theta} \mapsto \mathbf{\Psi}$.
6. We can also obtain the estimated standard error of the estimator $\widehat{\Psi}_n$ of $\psi^* \in \mathbf{\Psi}$ via the **Delta Method**.
7. The ML Estimator is **asymptotically optimal** or **efficient**. This means that the MLE has the smallest variance among the well-behaved class of estimators as the sample size gets larger.
8. ML Estimator is close to the Bayes estimator (obtained in the Bayesian inferential paradigm).

3.5.5 Fisher Information

Let $X_1, X_2, \ldots, X_n \overset{IID}{\sim} f(X_1;\theta)$. Here, $f(X_1;\theta)$ is the probability density function (pdf) or the probability mass function (pmf) of the RV X_1. Since all RVs are identically distributed, we simply focus on X_1 without loss of generality.

Definition 20 (Fisher Information) *The* **score function** *of an RV X for which the density is parameterised by θ is defined as:*

$$\mathcal{S}(X;\theta) := \frac{\partial log f(X;\theta)}{\partial \theta}, \qquad and \quad E_\theta(\mathcal{S}(X;\theta)) = 0 \ .$$

The **Fisher Information** *is*

$$I_n := V_\theta\left(\sum_{i=1}^{n} \mathcal{S}(X_i;\theta)\right) = \sum_{i=1}^{n} V_\theta\left(\mathcal{S}(X_i;\theta)\right) = nI_1(\theta), \tag{3.24}$$

where I_1 is the Fisher Information of just one of the RVs X_i,e.g. X:

$$\begin{aligned} I_1(\theta) &:= V_\theta\left(\mathcal{S}(X;\theta)\right) = E_\theta\left(\mathcal{S}^2(X,\theta)\right) \\ &= -E_\theta\left(\frac{\partial^2 log f(X;\theta)}{\partial^2\theta}\right) = \begin{cases} -\sum_{x\in\mathbb{X}}\left(\frac{\partial^2 \log f(x;\theta)}{\partial^2\theta}\right) f(x;\theta) & \text{for discrete } X \\ -\int_{x\in\mathbb{X}}\left(\frac{\partial^2 \log f(x;\theta)}{\partial^2\theta}\right) f(x;\theta)dx & \text{for continuous } X \end{cases} \end{aligned} \tag{3.25}$$

Next, we give a **general method** for obtaining:

1. The standard error $\mathsf{se}_n(\widehat{\Theta}_n)$ of **any** maximum likelihood estimator $\widehat{\Theta}_n$ of the possibly unknown and fixed parameter of interest $\theta^* \in \Theta$, and
2. The $1-\alpha$ confidence interval for θ^*.

Proposition 8 (Asymptotic Normality of the ML Estimator & Confidence Intervals) *Let $\widehat{\Theta}_n$ be the maximum likelihood estimator of $\theta^* \in \Theta$ with standard error $\mathsf{se}_n := \sqrt{V_{\theta^*}(\widehat{\Theta}_n)}$. Under appropriate regularity conditions, the following propositions are true:*

1. *The standard error se_n can be approximated by the side of a square whose area is the inverse Fisher Information at θ^*, and the distribution of $\widehat{\Theta}_n$ approaches that of the* $\text{Normal}(\theta^*, \mathsf{se}_n^2)$ *distribution as the samples size n gets larger. In other terms:*

$$\mathsf{se}_n \approxeq \sqrt{1/I_n(\theta^*)} \qquad and \quad \frac{\widehat{\Theta}_n - \theta^*}{\mathsf{se}_n} \rightsquigarrow \text{Normal}(0,1)$$

2. *The approximation holds even if we substitute the ML Estimate $\widehat{\theta}_n$ for θ^* and use the estimated standard error $\widehat{\mathsf{se}}_n$ instead of se_n. Let $\widehat{\mathsf{se}}_n = \sqrt{1/I_n(\widehat{\theta}_n)}$. Then:*

$$\frac{\widehat{\Theta}_n - \theta^*}{\widehat{\mathsf{se}}_n} \rightsquigarrow \text{Normal}(0,1)$$

3. *Using the fact that $\widehat{\Theta}_n \rightsquigarrow \text{Normal}(\theta^*, \widehat{se}_n^2)$, we can construct the estimate of an approximate Normal-based $1-\alpha$ confidence interval as:*

$$C_n = [\underline{C}_n, \overline{C}_n] = [\widehat{\theta}_n - z_{\alpha/2}\widehat{se}_n, \widehat{\theta}_n + z_{\alpha/2}\widehat{\mathsf{se}}_n] = \widehat{\theta}_n \pm z_{\alpha/2}\widehat{\mathsf{se}}_n$$

Now, let us do an example.

Example 3.5.6 (MLE and Confidence Interval for the IID Poisson(λ) experiment) *Suppose the fixed parameter $\lambda^* \in \mathbb{\Lambda} = (0, \infty)$ is unknown. Let $X_1, X_2, \ldots, X_n \overset{IID}{\sim} \text{Poisson}(\lambda^*)$. We want to find the ML Estimate $\widehat{\lambda}_n$ of λ^* and produce a $1-\alpha$ confidence interval for λ^*.*

The MLE can be obtained as follows:

The likelihood function is:

$$L(\lambda) := L(x_1, x_2, \ldots, x_n; \lambda) = \prod_{i=1}^{n} f(x_i; \lambda) = \prod_{i=1}^{n} e^{-\lambda} \frac{\lambda_i^x}{x_i!}$$

Hence, the log-likelihood function is:

$$\begin{aligned} \ell(\theta) := \log(L(\lambda)) &= \log\left(\prod_{i=1}^{n} e^{-\lambda} \frac{\lambda^{x_i}}{x_i!}\right) = \sum_{i=1}^{n} \log\left(e^{-\lambda} \frac{\lambda^{x_i}}{x_i!}\right) = \sum_{i=1}^{n} \Big(\log(e^{-\lambda}) + \log(\lambda^{x_i}) - \log(x_i!)\Big) \\ &= \sum_{i=1}^{n} (-\lambda + x_i \log(\lambda) - \log(x_i!)) = \sum_{i=1}^{n} -\lambda + \sum_{i=1}^{n} x_i \log(\lambda) - \sum_{i=1}^{n} \log(x_i!) \\ &= n(-\lambda) + \log(\lambda) \left(\sum_{i=1}^{n} x_i\right) - \sum_{i=1}^{n} \log(x_i!) \end{aligned}$$

Next, take the derivative of $\ell(\lambda)$:

$$\frac{\partial}{\partial \lambda} \ell(\lambda) = \frac{\partial}{\partial \lambda} \left(n(-\lambda) + \log(\lambda) \left(\sum_{i=1}^{n} x_i\right) - \sum_{i=1}^{n} \log(x_i!)\right) = n(-1) + \frac{1}{\lambda} \left(\sum_{i=1}^{n} x_i\right) + 0$$

and set it equal to 0 to solve for λ, as follows:

$$0 = n(-1) + \frac{1}{\lambda} \left(\sum_{i=1}^{n} x_i\right) + 0 \iff n = \frac{1}{\lambda} \left(\sum_{i=1}^{n} x_i\right) \iff \lambda = \frac{1}{n} \left(\sum_{i=1}^{n} x_i\right) = \overline{x}_n$$

Finally, the ML Estimator of λ^* is $\widehat{\Lambda}_n = \overline{X}_n$ and the ML estimate is $\widehat{\lambda}_n = \overline{x}_n$.

Now, we want an $1-\alpha$ confidence interval for λ^* using the $\widehat{\mathsf{se}}_n \approxeq \sqrt{1/I_n(\widehat{\lambda}_n)}$ that is based on the Fisher Information $I_n(\lambda) = nI_1(\lambda)$ given in (3.24). We need I_1 given in (3.25). Since $X_1, X_2, \ldots, X_n \sim \text{Poisson}(\lambda)$, we have discrete RVs:

$$I_1 = -\sum_{x \in \mathbb{X}} \left(\frac{\partial^2 \log(f(x;\lambda))}{\partial^2 \lambda}\right) f(x;\lambda) = -\sum_{x=0}^{\infty} \left(\frac{\partial^2 \log(f(x;\lambda))}{\partial^2 \lambda}\right) f(x;\lambda)$$

First find

$$\begin{aligned} \frac{\partial^2 \log(f(x;\lambda))}{\partial^2 \lambda} &= \frac{\partial}{\partial \lambda} \left(\frac{\partial}{\partial \lambda} \log\left(f(x;\lambda)\right)\right) = \frac{\partial}{\partial \lambda} \left(\frac{\partial}{\partial \lambda} \log\left(e^{-\lambda} \frac{\lambda^x}{x!}\right)\right) \\ &= \frac{\partial}{\partial \lambda} \left(\frac{\partial}{\partial \lambda} \left(-\lambda + x \log(\lambda) - \log(x!)\right)\right) = \frac{\partial}{\partial \lambda} \left(-1 + \frac{x}{\lambda} - 0\right) = -\frac{x}{\lambda^2} \end{aligned}$$

Now, substitute the above expression into the right-hand side of I_1 to obtain:

$$I_1 = -\sum_{x=0}^{\infty} \left(-\frac{x}{\lambda^2}\right) f(x;\lambda) = \frac{1}{\lambda^2} \sum_{x=0}^{\infty} (x)\, f(x;\lambda) = \frac{1}{\lambda^2} \sum_{x=0}^{\infty} (x)\, e^{-\lambda} \frac{\lambda^x}{x!} = \frac{1}{\lambda^2} E_\lambda(X) = \frac{1}{\lambda^2} \lambda = \frac{1}{\lambda}$$

In the third-to-last step above, we recognise the sum as the expectation of the Poisson(λ) RV X, namely $E_\lambda(X) = \lambda$. Therefore, the estimated standard error is:

$$\widehat{\mathsf{se}}_n \approxeq \sqrt{1/I_n(\widehat{\lambda}_n)} = \sqrt{1/(nI_1(\widehat{\lambda}_n))} = \sqrt{1/(n(1/\widehat{\lambda}_n))} = \sqrt{\widehat{\lambda}_n/n}$$

and the approximate $1-\alpha$ confidence interval is

$$\widehat{\lambda}_n \pm z_{\alpha/2}\widehat{\mathsf{se}}_n = \widehat{\lambda}_n \pm z_{\alpha/2}\sqrt{\widehat{\lambda}_n/n}$$

Thus, using the MLE and the estimated standard error via the Fisher Information, we can carry out point estimation and confidence interval construction in **most** parametric families of RVs encountered in typical engineering applications.

Example 3.5.7 (This was Homework 3.5.5 of Assignment 10) *Suppose* $X_1, X_2, \ldots, X_n \overset{IID}{\sim}$ Bernoulli(θ^*). *Also, suppose that* $\theta^* \in \Theta = [0,1]$ *is unknown. We have already shown in Example 3.4.4 that the ML estimator of* θ^* *is* $\widehat{\theta}_n = \overline{X}_n$. *Using the identity:*

$$\widehat{\mathsf{se}}_n = \frac{1}{\sqrt{I_n(\widehat{\theta}_n)}}$$

(1) we can compute $\widehat{\mathsf{se}}_n(\widehat{\Theta}_n)$, *the estimated standard error of the unknown parameter* θ^* *as follows:*

$$\widehat{\mathsf{se}}_n(\widehat{\Theta}_n) = \frac{1}{\sqrt{I_n(\widehat{\theta}_n)}} = \frac{1}{\sqrt{nI_1(\widehat{\theta}_n)}} \ .$$

So, we need to first compute $I_1(\theta)$, *the Fisher Information of one sample. Due to* (3.25) *and the fact that the* Bernoulli(θ^*) *distributed RV* X *is discrete with probability mass function* $f(x;\theta) = \theta^x(1-\theta)^{1-x}$, *for* $x \in \mathbb{X} := \{0,1\}$, *we have,*

$$I_1(\theta) = -E_\theta\left(\frac{\partial^2 log f(X;\theta)}{\partial^2\theta}\right) = -\sum_{x\in\mathbb{X}=\{0,1\}}\left(\frac{\partial^2 \log\left(\theta^x(1-\theta)^{1-x}\right)}{\partial^2\theta}\right)\theta^x(1-\theta)^{1-x}$$

Next, let us compute,

$$\begin{aligned}\frac{\partial^2 \log\left(\theta^x(1-\theta)^{1-x}\right)}{\partial^2\theta} &:= \frac{\partial}{\partial\theta}\left(\frac{\partial}{\partial\theta}\left(\log\left(\theta^x(1-\theta)^{1-x}\right)\right)\right) = \frac{\partial}{\partial\theta}\left(\frac{\partial}{\partial\theta}\left(x\log(\theta) + (1-x)\log(1-\theta)\right)\right)\\ &= \frac{\partial}{\partial\theta}\left(x\theta^{-1} + (1-x)(1-\theta)^{-1}(-1)\right) = \frac{\partial}{\partial\theta}\left(x\theta^{-1} - (1-x)(1-\theta)^{-1}\right)\\ &= x(-1)\theta^{-1-1} - (1-x)(-1)(1-\theta)^{-1-1}(-1) = -x\theta^{-2} - (1-x)(1-\theta)^{-2}\end{aligned}$$

Now, we compute the expectation I_1, *i.e. the sum over the two possible values of* $x \in \{0,1\}$,

$$\begin{aligned}I_1(\theta) &= -\sum_{x\in\mathbb{X}=\{0,1\}}\left(\frac{\partial^2 \log\left(\theta^x(1-\theta)^{1-x}\right)}{\partial^2\theta}\right)\theta^x(1-\theta)^{1-x}\\ &= -\left(\left(-0\ \theta^{-2} - (1-0)(1-\theta)^{-2}\right)\theta^0(1-\theta)^{1-0} + \left(-1\ \theta^{-2} - (1-1)(1-\theta)^{-2}\right)\theta^1(1-\theta)^{1-1}\right)\\ &= -\left(\left(0 - 1(1-\theta)^{-2}\right)\ 1\ (1-\theta)^1 + \left(-\theta^{-2} - 0\right)\theta^1\ 1\right) = (1-\theta)^{-2}(1-\theta)^1 + \theta^{-2}\theta^1\\ &= (1-\theta)^{-1} + \theta^{-1} = \frac{1}{1-\theta} + \frac{1}{\theta} = \frac{\theta}{\theta(1-\theta)} + \frac{1-\theta}{\theta(1-\theta)} = \frac{\theta + (1-\theta)}{\theta(1-\theta)} = \frac{1}{\theta(1-\theta)}\end{aligned}$$

Therefore, the desired estimated standard error of our estimator, can be obtained by substituting the ML estimate $\widehat{\theta}_n = \overline{x}_n := n^{-1}\sum_{i=1}^n x_i$ *of the unknown* θ^* *as follows:*

$$\widehat{\mathsf{se}}_n(\widehat{\theta}_n) = \frac{1}{\sqrt{I_n(\widehat{\theta}_n)}} = \frac{1}{\sqrt{nI_1(\widehat{\theta}_n)}} = \sqrt{\frac{1}{n\frac{1}{\widehat{\theta}_n(1-\widehat{\theta}_n)}}} = \sqrt{\frac{\widehat{\theta}_n(1-\widehat{\theta}_n)}{n}} = \sqrt{\frac{\overline{x}_n(1-\overline{x}_n)}{n}} \ .$$

(2) Using $\widehat{\mathsf{se}}_n(\widehat{\theta}_n)$ *we can construct an approximate* 95% *confidence interval* C_n *for* θ^**, due to the asymptotic normality of the ML estimator of* θ^**, as follows:*

$$C_n = \widehat{\theta}_n \pm 1.96\sqrt{\frac{\widehat{\theta}_n(1-\widehat{\theta}_n)}{n}} = \overline{x}_n \pm 1.96\sqrt{\frac{\overline{x}_n(1-\overline{x}_n)}{n}}$$

Recall that C_n *is the realisation of a random set based on your observed samples or data* $x_1, x_2, \ldots, x_n$*. Furthermore,* C_n*'s construction procedure ensures the engulfing of the unknown* θ^* *with probability approaching* 0.95 *as the sample size* n *gets large.*

(3) Flip any New Zealand coin as identically and independently as possible exactly 30 *times and record the outcomes (*1 *for heads and* 0 *for tails). Report the ML point estimate and the* 95% *confidence interval from your data. Do you think that the way you have flipped your coin and the outcomes you have witnessed can hint at the fairness (*$\theta^* = 0.5$*) or unfairness (*$\theta^* \neq 0.5$*) of the coin. Write a couple of sentences to make your case. [Take the time to flip coins this many times in a row, if you have not done so already. Be honest and really do it. I flipped an American quarter* 100 *times to produce the data in Example 3.5.3].*

Example 3.5.8 (This was Homework 3.5.6 of Assignment 10) *Let us get our hands dirty with a continuous RV next. Let* $X_1, X_2, \ldots, X_n \overset{IID}{\sim} \text{Exponential}(\lambda^*)$*. We saw that the ML estimator of* $\lambda^* \in \mathbb{\Lambda} = (0, \infty)$ *is* $\widehat{\Lambda}_n = 1/\overline{X}_n$ *and its ML estimate is* $\widehat{\lambda}_n = 1/\overline{x}_n$*, where* $x_1, x_2, \ldots, x_n$ *are our observed data.*

(1) Let us obtain the Fisher Information I_n *for this experiment to find the standard error:*

$$\widehat{\mathsf{se}}_n(\widehat{\Lambda}_n) = \frac{1}{\sqrt{I_n(\widehat{\lambda}_n)}} = \frac{1}{\sqrt{nI_1(\widehat{\lambda}_n)}}$$

and construct an approximate 95% *confidence interval for* λ^* *using the asymptotic normality of its ML estimator* $\widehat{\Lambda}_n$*.*

So, we need to first compute $I_1(\theta)$*, the Fisher Information of one sample. Due to* (3.25) *and the fact that the* Exponential(λ^*) *distributed RV* X *is continuous with probability density function* $f(x;\lambda) = \lambda e^{-\lambda x}$*, for* $x \in \mathbb{X} := [0, \infty)$*, we have,*

$$I_1(\theta) = -E_\theta\left(\frac{\partial^2 log f(X;\theta)}{\partial^2\theta}\right) = -\int_{x\in\mathbb{X}=[0,\infty)}\left(\frac{\partial^2 \log\left(\lambda e^{-\lambda x}\right)}{\partial^2\lambda}\right)\lambda e^{-\lambda x}\, dx$$

Let us compute the above integrand next.

$$\begin{aligned}\frac{\partial^2 \log\left(\lambda e^{-\lambda x}\right)}{\partial^2\lambda} &:= \frac{\partial}{\partial\lambda}\left(\frac{\partial}{\partial\lambda}\left(\log\left(\lambda e^{-\lambda x}\right)\right)\right) = \frac{\partial}{\partial\lambda}\left(\frac{\partial}{\partial\lambda}\left(\log(\lambda) + \log(e^{-\lambda x}\right)\right)\\ &= \frac{\partial}{\partial\lambda}\left(\frac{\partial}{\partial\lambda}\left(\log(\lambda) - \lambda x\right)\right) = \frac{\partial}{\partial\lambda}\left(\lambda^{-1} - x\right) = -\lambda^{-2} - 0 = -\frac{1}{\lambda^2}\end{aligned}$$

Now, let us evaluate the integral by recalling that the expectation of the constant 1 *is 1 for any RV* X *governed by some parameter, say* θ*. For instance when* X *is a continuous RV,* $E_\theta(1) = \int_{x\in\mathbb{X}} 1\ f(x;\theta) = \int_{x\in\mathbb{X}}\ f(x;\theta) = 1$*. Therefore, the Fisher Information of one sample is*

$$I_1(\theta) = -\int_{x\in\mathbb{X}=[0,\infty)} \left(\frac{\partial^2 \log\left(\lambda e^{-\lambda x}\right)}{\partial^2 \lambda}\right) \lambda e^{-\lambda x}\ dx = -\int_0^\infty \left(-\frac{1}{\lambda^2}\right) \lambda e^{-\lambda x}\ dx$$
$$= -\left(-\frac{1}{\lambda^2}\right)\int_0^\infty \lambda e^{-\lambda x}\ dx = \frac{1}{\lambda^2}\ 1 = \frac{1}{\lambda^2}$$

Now, we can compute the desired estimated standard error, by substituting in the ML estimate $\widehat{\lambda}_n = 1/(\overline{x}_n) := 1/\left(\sum_{i=1}^n x_i\right)$ *of* λ^**, as follows:*

$$\widehat{\mathsf{se}}_n(\widehat{\Lambda}_n) = \frac{1}{\sqrt{I_n(\widehat{\lambda}_n)}} = \frac{1}{\sqrt{nI_1(\widehat{\lambda}_n)}} = \frac{1}{\sqrt{n\frac{1}{\widehat{\lambda}_n^2}}} = \frac{\widehat{\lambda}_n}{\sqrt{n}} = \frac{1}{\sqrt{n}\ \overline{x}_n}$$

Using $\widehat{\mathsf{se}}_n(\widehat{\lambda}_n)$ *we can construct an approximate* 95% *confidence interval* C_n *for* λ^**, due to the asymptotic normality of the ML estimator of* λ^**, as follows:*

$$C_n = \widehat{\lambda}_n \pm 1.96\frac{\widehat{\lambda}_n}{\sqrt{n}} = \frac{1}{\overline{x}_n} \pm 1.96\frac{1}{\sqrt{n}\ \overline{x}_n}$$

(2) Recall labwork 2.2.2 where you modeled the arrival of buses using Exponential($\lambda^* = 0.1$) *distributed inter-arrival time with a mean of* $1/\lambda^* = 10$ *minutes. Using the data of these seven inter-arrival times at your ID-seeded bus stop and pretending that you do not know the true* λ^**, report (2.a) the ML estimate of* λ^**, (2.b)* 95% *confidence interval for it and (2.c) whether the true value* $\lambda^* = 1/10$ *is engulfed by your confidence interval.*

3.5.9 Delta Method

A more general estimation problem of interest concerns some function of the parameter $\theta \in \mathbf{\Theta}$, say $g(\theta) = \psi : \mathbf{\Theta} \mapsto \mathbf{\Psi}$. So, $g(\theta) = \psi$ is a function from the parameter space $\mathbf{\Theta}$ to $\mathbf{\Psi}$. Thus, we are not only interested in estimating the fixed and possibly unknown $\theta^* \in \mathbf{\Theta}$ using the ML estimator $\widehat{\Theta}_n$ and its ML estimate $\widehat{\theta}_n$, but also in estimating $\psi^* = g(\theta^*) \in \mathbf{\Psi}$ via an estimator $\widehat{\Psi}_n$ and its estimate $\widehat{\psi}_n$. We exploit the equivariance property of the ML estimator $\widehat{\Theta}_n$ of θ^* and use the Delta method to find the following analytically:

1. The ML estimator of $\psi^* = g(\theta^*) \in \mathbf{\Psi}$ is

$$\boxed{\widehat{\Psi}_n = g(\widehat{\Theta}_n)}$$

and its point estimate is

$$\boxed{\widehat{\psi}_n = g(\widehat{\theta}_n)}$$

2. Suppose $g(\theta) = \psi : \mathbf{\Theta} \mapsto \mathbf{\Psi}$ is **any** smooth function of θ, i.e. g is differentiable, and $g'(\theta) := \frac{\partial}{\partial\theta} g(\theta) \neq 0$. Then, the distribution of the ML estimator $\widehat{\Psi}_n$ is asymptotically Normal$(\psi^*, \widehat{\mathsf{se}}_n(\widehat{\Psi}_n)^2)$, i.e.:

$$\boxed{\frac{\widehat{\Psi}_n - \psi^*}{\widehat{\mathsf{se}}_n(\widehat{\Psi}_n)} \rightsquigarrow \text{Normal}(0,1)}$$

where the standard error $\widehat{\mathsf{se}}_n(\widehat{\Psi}_n)$ of the ML estimator $\widehat{\Psi}_n$ of the unknown quantity $\psi^* \in \mathbf{\Psi}$ can be obtained from the standard error $\widehat{\mathsf{se}}_n(\widehat{\Theta}_n)$ of the ML estimator $\widehat{\Theta}_n$ of the parameter $\theta^* \in \mathbf{\Theta}$, as follows:

$$\boxed{\widehat{\mathsf{se}}_n(\widehat{\Psi}_n) = |g'(\widehat{\theta}_n)|\widehat{\mathsf{se}}_n(\widehat{\Theta}_n)}$$

3. Using Normal$(\psi^*, \widehat{\mathsf{se}}_n(\widehat{\Psi}_n)^2)$, we can construct the estimate of an approximate Normal-based $1-\alpha$ confidence interval for $\psi^* \in \mathbf{\Psi}$:

$$\boxed{C_n = [\underline{C}_n, \overline{C}_n] = \widehat{\psi}_n \pm z_{\alpha/2}\widehat{\mathsf{se}}_n(\widehat{\psi}_n)}$$

Let us do an example next.

Example 3.5.10 *Let $X_1, X_2, \ldots, X_n \overset{IID}{\sim}$ Bernoulli(θ^*). Let $\psi = g(\theta) = \log(\theta/(1-\theta))$. Suppose we are interested in producing a point estimate and confidence interval for $\psi^* = g(\theta^*)$. We can use the Delta method as follows:*

First, the estimated standard error of the ML estimator of θ^*, as shown in Example 3.5.7, is

$$\widehat{\mathsf{se}}_n(\widehat{\Theta}_n) = \sqrt{\frac{\widehat{\theta}_n(1-\widehat{\theta}_n)}{n}} \ .$$

The ML estimator of ψ^* is:

$$\widehat{\Psi}_n = \log(\widehat{\Theta}_n/(1-\widehat{\Theta}_n))$$

and the ML estimate of ψ^* is:

$$\widehat{\psi}_n = \log(\widehat{\theta}_n/(1-\widehat{\theta}_n)) \ .$$

Since, $g'(\theta) = 1/(\theta(1-\theta))$, by the Delta method, the estimated standard error of the ML estimator of ψ^* is:

$$\widehat{\mathsf{se}}_n(\widehat{\Psi}_n) = |g'(\widehat{\theta}_n)|(\widehat{\mathsf{se}}_n(\widehat{\Theta}_n)) = \frac{1}{\widehat{\theta}_n(1-\widehat{\theta}_n)}\sqrt{\frac{\widehat{\theta}_n(1-\widehat{\theta}_n)}{n}} = \frac{1}{\sqrt{n\widehat{\theta}_n(1-\widehat{\theta}_n)}} = \frac{1}{\sqrt{n\overline{x}_n(1-\overline{x}_n)}} \ .$$

An approximate 95% confidence interval for $\psi^* = \log(\theta^*/(1-\theta^*))$ is:

$$\widehat{\psi}_n \pm \frac{1.96}{\sqrt{n\widehat{\theta}_n(1-\widehat{\theta}_n)}} = \log(\widehat{\theta}_n/(1-\widehat{\theta}_n)) \pm \frac{1.96}{\sqrt{n\widehat{\theta}_n(1-\widehat{\theta}_n)}} = \log(\overline{x}_n/(1-\overline{x}_n)) \pm \frac{1.96}{\sqrt{n\overline{x}_n(1-\overline{x}_n)}} \ .$$

Example 3.5.11 (This was Homework 3.5.8 of Assignment 10) *Let us try the Delta method on a continuous RV. Let $X_1, X_2, \ldots, X_n \overset{IID}{\sim}$ Normal(μ^*, σ^{*2}). Suppose that μ^* is known and σ^* is unknown. Let us derive the ML estimate $\widehat{\psi}_n$ of $\psi^* = \log(\sigma^*)$ and a 95% confidence interval for it in 6 steps.*

(1) First let us find the log-likelihood function $\ell(\sigma)$

$$\ell(\sigma) := \log(L(\sigma)) := \log(L(x_1, x_2, \ldots, x_n; \sigma)) = \log\left(\prod_{i=1}^{n} f(x_i;\sigma)\right) = \sum_{i=1}^{n} \log\left(f(x_i;\sigma)\right)$$

$$= \sum_{i=1}^{n} \log\left(\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right)\right) \quad \because f(x_i;\sigma) \text{ in (2.7) is pdf of Normal}(\mu,\sigma^2) \text{ RV with known } \mu$$

$$= \sum_{i=1}^{n} \left(\log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \log\left(\exp\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right)\right)\right)$$

$$= \sum_{i=1}^{n} \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \sum_{i=1}^{n}\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right) = n\log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \left(-\frac{1}{2\sigma^2}\right)\sum_{i=1}^{n}(x_i-\mu)^2$$

$$= n\left(\log\left(\frac{1}{\sqrt{2\pi}}\right) + \log\left(\frac{1}{\sigma}\right)\right) - \left(\frac{1}{2\sigma^2}\right)\sum_{i=1}^{n}(x_i-\mu)^2$$

$$= n\log\left(\sqrt{2\pi}^{-1}\right) + n\log\left(\sigma^{-1}\right) - \left(\frac{1}{2\sigma^2}\right)\sum_{i=1}^{n}(x_i-\mu)^2$$

$$= -n\log\left(\sqrt{2\pi}\right) - n\log\left(\sigma\right) - \left(\frac{1}{2\sigma^2}\right)\sum_{i=1}^{n}(x_i-\mu)^2$$

(2) Let us find its derivative with respect to the unknown parameter σ next.

$$\frac{\partial}{\partial\sigma}\ell(\sigma) := \frac{\partial}{\partial\sigma}\left(-n\log\left(\sqrt{2\pi}\right) - n\log\left(\sigma\right) - \left(\frac{1}{2\sigma^2}\right)\sum_{i=1}^{n}(x_i-\mu)^2\right)$$

$$= \frac{\partial}{\partial\sigma}\left(-n\log\left(\sqrt{2\pi}\right)\right) - \frac{\partial}{\partial\sigma}\left(n\log\left(\sigma\right)\right) - \frac{\partial}{\partial\sigma}\left(\left(\frac{1}{2\sigma^2}\right)\sum_{i=1}^{n}(x_i-\mu)^2\right)$$

$$= 0 - n\frac{\partial}{\partial\sigma}\left(\log(\sigma)\right) - \left(\frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^2\right)\frac{\partial}{\partial\sigma}\left(\sigma^{-2}\right)$$

$$= -n\sigma^{-1} - \left(\frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^2\right)\left(-2\sigma^{-3}\right) = -n\sigma^{-1} + \sigma^{-3}\sum_{i=1}^{n}(x_i-\mu)^2$$

(3) Now, let us set the derivative equal to 0 and solve for σ.

$$0 = -n\sigma^{-1} + \sigma^{-3}\sum_{i=1}^{n}(x_i-\mu)^2 \iff n\sigma^{-1} = \sigma^{-3}\sum_{i=1}^{n}(x_i-\mu)^2 \iff n\sigma^{-1}\sigma^{+3} = \sum_{i=1}^{n}(x_i-\mu)^2$$

$$\iff n\sigma^{-1+3} = \sum_{i=1}^{n}(x_i-\mu)^2 \iff n\sigma^2 = \sum_{i=1}^{n}(x_i-\mu)^2$$

$$\iff \sigma^2 = \left(\sum_{i=1}^{n}(x_i-\mu)^2\right)/n \iff \sigma = \sqrt{\sum_{i=1}^{n}(x_i-\mu)^2/n}$$

Finally, we set the solution, i.e. the maximiser of the concave-down log-likelihood function of σ with a known and fixed μ^* as our ML estimate $\widehat{\sigma}_n = \sqrt{\sum_{i=1}^{n}(x_i-\mu^*)^2/n}$. Analogously, the ML estimator of σ^* is $\widehat{\Sigma}_n = \sqrt{\sum_{i=1}^{n}(X_i-\mu^*)^2/n}$. Don't confuse Σ, the upper-case sigma, with $\sum_{i=1}^{n}\bigcirc_i$, the summation over some $\bigcirc_i$'s. This is usually clear from the context.

(4) Next, let us get the estimated standard error $\widehat{se}_n$ for the estimator of σ^* via Fisher Information. The Log-likelihood function of σ, based on one sample from the Normal(μ, σ^2) *RV with known μ is,*

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

Therefore, in much the same way as in part (2) earlier,

$$\begin{aligned}\frac{\partial^2 \log f(x;\sigma)}{\partial^2\sigma} &:= \frac{\partial}{\partial\sigma}\left(\frac{\partial}{\partial\sigma}\left(-\log\left(\sqrt{2\pi}\right) - \log(\sigma) - \left(\frac{1}{2\sigma^2}\right)(x-\mu)^2\right)\right) \\ &= \frac{\partial}{\partial\sigma}\left(-\sigma^{-1} + \sigma^{-3}(x-\mu)^2\right) = \sigma^{-2} - 3\sigma^{-4}(x-\mu)^2\end{aligned}$$

Now, we compute the Fisher Information of one sample as an expectation of the continuous RV X over $\mathbb{X} = (-\infty, \infty)$ with density $f(x;\sigma)$,

$$\begin{aligned}I_1(\sigma) &= -\int_{x\in\mathbb{X}=(-\infty,\infty)}\left(\frac{\partial^2 \log f(x;\sigma)}{\partial^2\lambda}\right) f(x;\sigma)\ dx = -\int_{-\infty}^{\infty}\left(\sigma^{-2} - 3\sigma^{-4}(x-\mu)^2\right) f(x;\sigma)\ dx \\ &= \int_{-\infty}^{\infty} -\sigma^{-2} f(x;\sigma)\ dx + \int_{-\infty}^{\infty} 3\sigma^{-4}(x-\mu)^2 f(x;\sigma)\ dx \\ &= -\sigma^{-2}\int_{-\infty}^{\infty} f(x;\sigma)\ dx + 3\sigma^{-4}\int_{-\infty}^{\infty}(x-\mu)^2 f(x;\sigma)\ dx \\ &= -\sigma^{-2} + 3\sigma^{-4}\sigma^2 \qquad\qquad \because \sigma^2 = V(X) = E(X-E(X))^2 = \int_{-\infty}^{\infty}(x-\mu)^2 f(x;\sigma)\ dx \\ &= -\sigma^{-2} + 3\sigma^{-4+2} = -\sigma^{-2} + 3\sigma^{-2} = 2\sigma^{-2}\end{aligned}$$

Therefore, the estimated standard error of the estimator of the unknown σ^* is

$$\widehat{\mathsf{se}}_n(\widehat{\Sigma}_n) = \frac{1}{\sqrt{I_n(\widehat{\sigma}_n)}} = \frac{1}{\sqrt{nI_1(\widehat{\sigma}_n)}} = \frac{1}{\sqrt{n2\sigma^{-2}}} = \frac{\sigma}{\sqrt{2n}}\ .$$

(5) Given that $\psi = g(\sigma) = \log(\sigma)$, we derive the estimated standard error of $\psi^* = \log(\sigma^*)$ via the Delta method as follows:

$$\widehat{\mathsf{se}}_n(\widehat{\Psi}_n) = |g'(\sigma)|\widehat{\mathsf{se}}_n(\widehat{\Sigma}_n) = \left|\frac{\partial}{\partial\sigma}\log(\sigma)\right|\frac{\sigma}{\sqrt{2n}} = \frac{1}{\sigma}\frac{\sigma}{\sqrt{2n}} = \frac{1}{\sqrt{2n}}\ .$$

(6) Finally, the 95% confidence interval for ψ^* is $\widehat{\psi}_n \pm 1.96\widehat{\mathsf{se}}_n(\widehat{\Psi}_n) = \log(\widehat{\sigma}_n) \pm 1.96\frac{1}{\sqrt{2n}}$.

3.5.12 Confidence Sets for Multiparameter Models

We will extend the Fisher Information and Delta method to models with more than one parameter:

$$X_1, X_2, \ldots, X_n \overset{IID}{\sim} f(x;\theta^*), \qquad \theta^* := (\theta_1^*, \theta_2^*, \ldots, \theta_k^*) \in \Theta \subset \mathbb{R}^k\ .$$

Let, the ML estimator of the fixed and possibly unknown vector-valued parameter θ^* be:

$$\widehat{\Theta}_n := \left(\widehat{\Theta}_{1,n}, \widehat{\Theta}_{2,n}, \ldots, \widehat{\Theta}_{k,n}\right), \qquad \widehat{\Theta}_n := \widehat{\Theta}_n(X_1, X_2, \ldots, X_n) : \mathbb{X}_n \mapsto \Theta$$

and the ML estimate based on n observations $x_1, x_2, \ldots, x_n$ be:

$$\widehat{\theta}_n := \left(\widehat{\theta}_{1,n}, \widehat{\theta}_{2,n}, \ldots, \widehat{\theta}_{k,n}\right), \qquad \widehat{\theta}_n := \widehat{\theta}_n(x_1, x_2, \ldots, x_n) \in \Theta\ .$$

Let the log-likelihood function and its Hessian matrix $H = (H_{i,j})_{i,j=1,2,\ldots,k}$ of partial derivatives be:

$$\ell_n(\theta) := \ell_n(\theta_1, \theta_2, \ldots, \theta_k) := \sum_{i=1}^{n} \log(f(x_i; (\theta_1, \theta_2, \ldots, \theta_k))), \qquad H_{i,j} := \frac{\partial}{\partial \theta_i} \frac{\partial}{\partial \theta_j} \ell_n(\theta_1, \theta_2, \ldots, \theta_k) \ ,$$

respectively, provided the log-likelihood function is sufficiently smooth.

Definition 21 (Fisher Information Matrix) *The Fisher Information matrix is:*

$$I_n(\theta) := I_n(\theta_1, \theta_2, \ldots, \theta_k) = - \begin{bmatrix} E_\theta(H_{1,1}) & E_\theta(H_{1,2}) & \cdots & E_\theta(H_{1,k}) \\ E_\theta(H_{2,1}) & E_\theta(H_{2,2}) & \cdots & E_\theta(H_{2,k}) \\ \vdots & \vdots & \ddots & \vdots \\ E_\theta(H_{k,1}) & E_\theta(H_{k,2}) & \cdots & E_\theta(H_{k,k}) \end{bmatrix} \tag{3.26}$$

and its matrix inverse is denoted by $I_n^{-1}(\theta)$.

Proposition 9 (Asymptotic Normality of MLE in Multiparameter Models) *Let*

$$X_1, X_2, \ldots, X_n \overset{IID}{\sim} f(x_1; \theta_1^*, \theta_2^*, \ldots, \theta_k^*), \qquad \theta^* = (\theta_1^*, \theta_2^*, \ldots, \theta_k^*) \in \mathbf{\Theta} \subset \mathbb{R}^k \ ,$$

for some fixed and possibly unknown $\theta^* \in \mathbf{\Theta} \subset \mathbb{R}^k$. Then, under appropriate regularity conditions:

$$\widehat{\Theta}_n := \left(\widehat{\Theta}_{1,n}, \widehat{\Theta}_{2,n}, \ldots, \widehat{\Theta}_{k,n} \right) \rightsquigarrow \text{Normal}(\theta^*, I_n^{-1})$$

In other words, the vector-valued estimator $\widehat{\Theta}_n$ converges in distribution to the multivariate Normal *distribution centred at the unknown parameter θ^* with the variance-covariance matrix given by inverse Fisher Information matrix I_n^{-1}. Furthermore, let $I_n^{-1}(j,j)$ denote the j^{th} diagonal entry of I_n^{-1}. In this case:*

$$\frac{\widehat{\Theta}_{j,n} - \theta_j^*}{\sqrt{I_n^{-1}(j,j)}} \rightsquigarrow \text{Normal}(0,1)$$

and the approximate covariance of $\widehat{\Theta}_{i,n}$ and $\widehat{\Theta}_{j,n}$ is:

$$\mathsf{Cov}(\Theta_{i,n}, \Theta_{j,n}) \approxeq I_n^{-1}(i,j) \ .$$

Now, let us look at a way of obtaining ML estimates and confidence sets for functions of θ. Suppose the real-valued function $g(\theta) = \psi : \mathbf{\Theta} \mapsto \mathbf{\Psi}$ maps points in the k-dimensional parameter space $\mathbf{\Theta} \subset \mathbb{R}^k$ to points in $\mathbf{\Psi} \subset \mathbb{R}$. Let the gradient of g be

$$\bigtriangledown g(\theta) := \bigtriangledown g(\theta_1, \theta_2, \ldots, \theta_k) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} g(\theta_1, \theta_2, \ldots, \theta_k) \\ \frac{\partial}{\partial \theta_2} g(\theta_1, \theta_2, \ldots, \theta_k) \\ \vdots \\ \frac{\partial}{\partial \theta_k} g(\theta_1, \theta_2, \ldots, \theta_k) \end{pmatrix} \ .$$

Proposition 10 (Multiparameter Delta Method) *Suppose:*

1. $X_1, X_2, \ldots, X_n \overset{IID}{\sim} f(x_1; \theta_1^*, \theta_2^*, \ldots, \theta_k^*), \qquad \theta^* = (\theta_1^*, \theta_2^*, \ldots, \theta_k^*) \in \mathbf{\Theta} \subset \mathbb{R}^k,$

2. *Let $\widehat{\Theta}_n$ be a ML estimator of $\theta^* \in \mathbf{\Theta}$ and let $\widehat{\theta}_n$ be its ML estimate, and*

3. *Let $g(\theta) = \psi : \mathbf{\Theta} \mapsto \mathbf{\Psi} \subset \mathbb{R}$ be a smooth function such that $\bigtriangledown g(\widehat{\theta}_n) \neq 0$.*

Then:

1. *$\widehat{\Psi}_n = g(\widehat{\Theta}_n)$ is the ML estimator and $\widehat{\psi}_n = g(\widehat{\theta}_n)$ is the ML estimate of of $\psi^* = g(\theta^*) \in \mathbf{\Psi}$,*

2. *The standard error of the ML estimator of ψ^* is:*

$$\widehat{\mathsf{se}}_n(\widehat{\Psi}_n) = \sqrt{\left(\bigtriangledown g(\widehat{\theta}_n)\right)^T I_n^{-1}(\widehat{\theta}_n) \left(\bigtriangledown g(\widehat{\theta}_n)\right)},$$

3. *The ML estimator of ψ^* is asymptotically normal, i.e.:*

$$\frac{\widehat{\Psi}_n - \psi^*}{\widehat{\mathsf{se}}_n(\widehat{\Psi}_n)} \rightsquigarrow \text{Normal}(0,1) \ ,$$

4. *And a $1-\alpha$ confidence interval for ψ^* is:*

$$\widehat{\psi}_n \pm z_{\alpha/2} \widehat{\mathsf{se}}_n(\widehat{\Psi}_n)$$

Let us put the theory to practice in the problem of estimating the coefficient of variation from samples of size n from an RV.

Example 3.5.13 (Estimating the Coefficient of Variation of a Normal(μ^*, σ^{*2}) RV) *Let*

$$\psi^* = g(\mu^*, \sigma^*) = \sigma^*/\mu^*, \qquad X_1, X_2, \ldots, X_n \overset{IID}{\sim} \text{Normal}(\mu^*, \sigma^{*2}) \ .$$

We do not know the fixed parameters (μ^*, σ^*) and are interested in estimating the coefficient of variation ψ^* based on n IID samples $x_1, x_2, \ldots, x_n$. We have already seen that the ML estimates of μ^* and σ^* are:

$$\widehat{\mu}_n = \overline{x}_n := \frac{1}{n}\sum_{i=1}^n x_i, \qquad \widehat{\sigma}_n = s_n := \sqrt{\frac{1}{n}\sum_{i=1}^n (x_i - \widehat{\mu}_n)^2} \ .$$

Thus, the ML estimate of $\psi^* = \sigma^*/\mu^*$ is:

$$\widehat{\psi}_n = \frac{\widehat{\sigma}_n}{\widehat{\mu}_n} = \frac{s_n}{\overline{x}_n}$$

We can now derive the standard error of the ML estimator $\widehat{\Psi}_n$ by first computing $I_n(\mu, \sigma)$, $I_n^{-1}(\mu, \sigma)$, and $\bigtriangledown g(\mu, \sigma)$. A careful computation shows that:

$$I_n(\mu, \sigma) = \begin{bmatrix} \frac{n}{\sigma^2} & 0 \\ 0 & \frac{2n}{\sigma^2} \end{bmatrix}, \qquad I_n^{-1}(\mu, \sigma) = \frac{1}{n}\begin{bmatrix} \sigma^2 & 0 \\ 0 & \frac{\sigma^2}{2} \end{bmatrix}, \qquad \bigtriangledown g(\mu, \sigma) = \begin{pmatrix} -\frac{\sigma}{\mu^2} \\ \frac{1}{\mu} \end{pmatrix} \ .$$

Therefore, the standard error of interest is:

$$\widehat{\mathsf{se}}_n(\widehat{\Psi}_n) = \sqrt{\left(\bigtriangledown g(\widehat{\theta}_n)\right)^T I_n^{-1}(\widehat{\theta}_n) \left(\bigtriangledown g(\widehat{\theta}_n)\right)} = \frac{1}{\sqrt{n}}\sqrt{\frac{1}{\widehat{\mu}_n^4} + \frac{\widehat{\sigma}_n^2}{2\widehat{\mu}_n^2}}$$

and the 95% confidence interval for the unknown coefficient of variation ψ^* is:

$$\widehat{\psi}_n \pm z_{\alpha/2} \widehat{\mathsf{se}}_n(\widehat{\Psi}_n) = \frac{s_n}{\overline{x}_n} \pm z_{\alpha/2} \left(\frac{1}{\sqrt{n}}\sqrt{\frac{1}{\widehat{\mu}_n^4} + \frac{\widehat{\sigma}_n^2}{2\widehat{\mu}_n^2}}\right)$$

Let us apply these results to $n = 100$ simulated samples from Normal$(100, 10^2)$ as follows.

CoeffOfVarNormal.m

```
n=100; % sample size
Mustar=100; % true mean
Sigmastar=10; % true standard deviation
rand('twister',67345); Us=rand(1,100); % draw some Uniform(0,1) samples
x=arrayfun(@(u)(Sample1NormalByNewRap(u,Mustar,Sigmastar^2)),Us); % get normal samples
Muhat=mean(x) % sample mean is MLE of Mustar
Sigmahat=std(x) % sample standard deviation is MLE for Sigmastar
Psihat=Sigmahat/Muhat % MLE of coefficient of variation std/mean
Sehat = sqrt((1/Muhat^4)+(Sigmahat^2/(2*Muhat^2)))/sqrt(n) % standar error estimate
ConfInt95=[Psihat-1.96*Sehat, Psihat+1.96*Sehat] % 1.96 since 1-alpha=0.95
```

```
>> CoeffOfVarNormal
Muhat =  100.3117
Sigmahat =   10.9800
Psihat =    0.1095
Sehat =    0.0077
ConfInt95 =    0.0943    0.1246
```

3.5.14 Parametric Bootstrap for Confidence Sets

The **bootstrap** is a statistical method for estimating standard errors and confidence sets of statistics, such as estimators. This is based on computationally intensive simulation and is much easier than the the variance calculation based on Fisher Information and/or the Delta method. Let $T_n := T_n((X_1, X_2, \ldots, X_n))$ be a statistic (i.e. any function of the data $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} F(x; \theta^*)$). Suppose we want to know its variance $V_{\theta^*}(T_n)$. If our statistic T_n is one with an analytically unknown variance, then we can use the bootstrap to estimate it. The bootstrap concept has the following two basic steps:

Step 1: Estimate $V_{\theta^*}(T_n)$ with $V_{\widehat{\theta}_n}(T_n)$, where $\widehat{\theta}_n$ is an estimate of θ^* based on maximum likelihood or the method of moments.

Step 2: Approximate $V_{\widehat{\theta}_n}(T_n)$ using simulated data from the "Bootstrap World."

For example, if $T_n = \overline{X}_n$, the sample mean, then in Step 1, $V_{\widehat{\theta}_n}(T_n) = n^{-1}\sum_{i=1}^{n}(x_i - \overline{x}_n)$ is the sample variance. Thus, in this case, Step 1 is enough. However, when the statistic T_n is more complicated, say $T_n = \widetilde{X}_n = F^{[-1]}(0.5)$, the sample median, then we may not be able to write down a simple expression for $V_{\widehat{\theta}_n}(T_n)$ and may need Step 2 of the bootstrap.

$$\begin{aligned} \text{Real World Data come from} \quad & F(\theta^*) \implies X_1, X_2, \ldots, X_n \implies T_n((X_1, X_2, \ldots, X_n)) = t_n \\ \text{Bootstrap World Data come from} \quad & F(\widehat{\theta}_n) \implies X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet \implies T_n((X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet)) = t_n^\bullet \end{aligned}$$

To simulate $X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet$ from $F(\widehat{\theta}_n)$, we must have a simulation algorithm that allows us to draw IID samples from $F(\theta)$, for instance the inversion sampler. In summary, the algorithm for Bootstrap Variance Estimation is:

Step 1: Draw $X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet \sim F(\widehat{\theta}_n)$

Step 2: Compute $t_n^\bullet = T_n((X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet))$

Step 3: Repeat Step 1 and Step 2 B times, for some large B, say $B \geq 1000$, to get $t_{n,1}^\bullet, t_{n,2}^\bullet, \ldots, t_{n,B}^\bullet$

Step 4: We can estimate the bootstrap confidence intervals in several ways:

(a) The $1-\alpha$ normal-based bootstrap confidence interval is:

$$C_n = [T_n - z_{\alpha/2}\widehat{se}_{boot}, T_n + z_{\alpha/2}\widehat{se}_{boot}] \ ,$$

where the bootstrap-based standard error estimate is:

$$\widehat{se}_{boot} = \sqrt{v_{boot}} = \sqrt{\frac{1}{B}\sum_{b=1}^{B}\left(t^{\bullet}_{n,b} - \frac{1}{B}\sum_{r=1}^{B} t^{\bullet}_{n,r}\right)^2}$$

(b) The $1-\alpha$ percentile-based bootstrap confidence interval:

$$C_n = [\widehat{G^{\bullet}}_n^{-1}(\alpha/2), \widehat{G^{\bullet}}_n^{-1}(1-\alpha/2)],$$

where $\widehat{G^{\bullet}}_n$ is the empirical DF of the bootstrapped $t^{\bullet}_{n,1}, t^{\bullet}_{n,2}, \ldots, t^{\bullet}_{n,B}$ and $\widehat{G^{\bullet}}_n^{-1}(q)$ is the q^{th} sample quantile (3.9) of $t^{\bullet}_{n,1}, t^{\bullet}_{n,2}, \ldots, t^{\bullet}_{n,B}$.

Let us apply the bootstrap method to the previous problem of estimating the standard error of the coefficient of variation from $n = 100$ samples from Normal$(100, 10^2)$ RV. The confidence intervals from bootstrap-based methods are similar to those from the Delta method.

CoeffOfVarNormalBoot.m

```
n=100; Mustar=100; Sigmastar=10; % sample size, true mean and standard deviation
rand('twister',67345);
x=arrayfun(@(u)(Sample1NormalByNewRap(u,Mustar,Sigmastar^2)),rand(n,1)); % normal samples
Muhat=mean(x) Sigmahat=std(x) Psihat=Sigmahat/Muhat % MLE of Mustar, Sigmastar and Psistar
Sehat = sqrt((1/Muhat^4)+(Sigmahat^2/(2*Muhat^2)))/sqrt(n) % standard error estimate
% 95% Confidence interval by Delta Method
ConfInt95DeltaMethod=[Psihat-1.96*Sehat, Psihat+1.96*Sehat] % 1.96 since 1-alpha=0.95
B = 1000; % B is number of bootstrap replications
% Step 1: draw n IID samples in Bootstrap World from Normal(Muhat,Sigmahat^2)
xBoot = arrayfun(@(u)(Sample1NormalByNewRap(u,Muhat,Sigmahat^2)),rand(n,B));
% Step 2: % Compute Bootstrapped Statistic Psihat
PsihatBoot = std(xBoot) ./ mean(xBoot);
% 95% Normal based Confidence Interval
SehatBoot = std(PsihatBoot); % std of PsihatBoot
ConfInt95BootNormal = [Psihat-1.96*SehatBoot, Psihat+1.96*SehatBoot] % 1-alpha=0.95
% 95% Percentile based Confidence Interval
ConfInt95BootPercentile = ...
    [qthSampleQuantile(0.025,sort(PsihatBoot)),qthSampleQuantile(0.975,sort(PsihatBoot))]
```

```
>> CoeffOfVarNormal
Muhat =  100.3117
Sigmahat =   10.9800
Psihat =    0.1095
Sehat =    0.0077
ConfInt95DeltaMethod =    0.0943    0.1246
ConfInt95BootNormal =    0.0943    0.1246
ConfInt95BootPercentile =    0.0946    0.1249
```

3.6 Non-parametric Estimation

So far, we have been interested in some estimation problems involved in parametric experiments. In parametric experiments, the parameter space Θ can have many dimensions, but these are finite. For example, in the n IID Bernoulli experiment, which is:

$$X_1, \ldots, X_n \overset{IID}{\sim} \text{Bernoulli}(\theta), \qquad \theta \in \Theta = [0,1] \subset \mathbb{R}^1 \ ,$$

the parameter space Θ has dimension 1. Similarly, in the n IID Normal(μ, σ^2) and the n IID Lognormal(λ, ζ), experiments:

$$X_1, \ldots, X_n \overset{IID}{\sim} \text{Normal}(\mu, \sigma^2), \qquad (\mu, \sigma^2) \in \Theta = (-\infty, +\infty) \times (0, +\infty) \subset \mathbb{R}^2$$
$$X_1, \ldots, X_n \overset{IID}{\sim} \text{Lognormal}(\lambda, \zeta), \qquad (\lambda, \zeta) \in \Theta = (0, +\infty) \times (0, +\infty) \subset \mathbb{R}^2$$

the parameter space is of dimension 2.

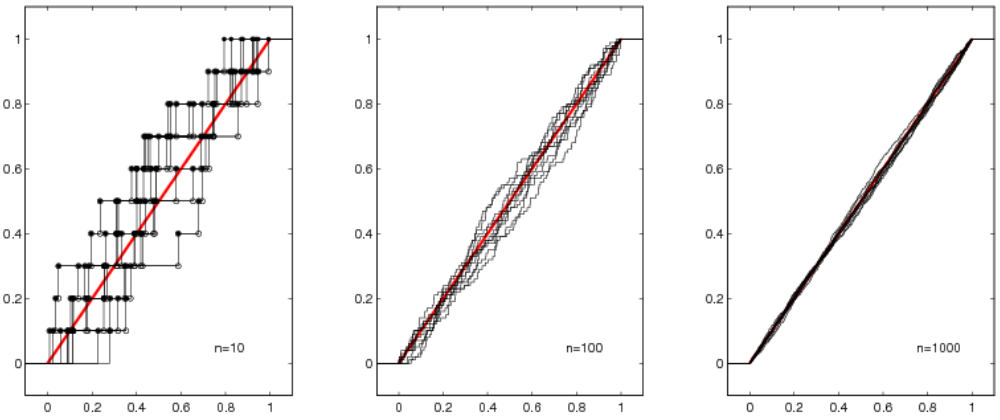
An experiment with an infinite dimensional parameter space Θ is said to be **non-parametric** . Next we consider a non-parametric experiment in which n IID samples are drawn according to some fixed and possibly unknown DF F^* from the space of **All Distribution Functions**:

$$\boxed{X_1, X_2, \ldots, X_n \overset{IID}{\sim} F^*, \qquad F^* \in \Theta = \{\text{All DFs}\} := \{\, F(x;F) : F \ is \ a \ DF \,\}}$$

where the DF $F(x;F)$ is indexed or parameterised by itself. Thus, the parameter space $\Theta = \{\text{All DFs}\}$ is the **infinite dimensional** space of **All DFs**. In this section, we look at estimation problems in non-parametric experiments with an infinite dimensional parameter space. That is, we want to estimate the DF F^* from which our IID data are drawn.

The next proposition is often referred to as the **fundamental theorem of statistics** and is at the heart of non-parametric inference, empirical processes, and computationally intensive bootstrap techniques.

Figure 3.9: Plots of ten distinct ECDFs $\widehat{F}_n$ based on 10 sets of n IID samples from Uniform$(0,1)$ RV X, as n increases from 10 to 100 to 1000. The DF $F(x) = x$ over $[0,1]$ is shown in red. The script of Labwork 5.0.17 was used to generate this plot.



Proposition 11 (Gilvenko-Cantelli Theorem) *Let* $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} F^*$. *Then:*

$$\sup_x |\widehat{F}_n(x) - F^*(x)| \xrightarrow{P} 0 \ .$$

Heuristic Interpretation of the Gilvenko-Cantelli Theorem: As the sample size n increases, the empirical distribution function $\widehat{F}_n$ converges to the true DF F^* in probability, as shown in Figure 3.9.

Proposition 12 (The Dvoretzky-Kiefer-Wolfowitz (DKW) Inequality) *Let* $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} F^*$. *Then, for any* $\epsilon > 0$*:*

$$P\left(\sup_x |\widehat{F}_n(x) - F^*(x)| > \epsilon\right) \leq 2\exp\left(-2n\epsilon^2\right) \tag{3.27}$$

Recall that $\sup(A)$ *or supremum of a set* $A \subset \mathbb{R}$ *is the least upper bound of every element in* A.

3.6.1 Estimating DF

Let $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} F^*$, where F^* is some particular DF in the space of all possible DFs, i.e. the experiment is non-parametric. Then, based on the data sequence $X_1, X_2, \ldots, X_n$ we want to estimate F^*.

For any fixed value of x, the expectation and variance of the empirical DF (3.8) are:

$$E\left(\widehat{F}_n(x)\right) = F^*(x) \implies \mathsf{bias}_n\left(\widehat{F}_n(x)\right) = 0 \tag{3.28}$$

$$V\left(\widehat{F}_n(x)\right) = \frac{F^*(x)(1-F^*(x))}{n} \implies \lim_{n\to\infty} \mathsf{se}_n\left(\widehat{F}_n(x)\right) = 0 \tag{3.29}$$

Therefore, by Proposition 4, the empirical DF evaluated at x, i.e. $\widehat{F}_n(x)$ is an asymptotically consistent estimator of the DF evaluated at x, i.e. $F^*(x)$. More formally, (3.28) and (3.29), by Proposition 4, imply that for any fixed value of x:

$$\widehat{F}_n(x) \xrightarrow{P} F^*(x) \ .$$

We are interested in a point estimate of the entire DF F^*, i.e. $F^*(x)$ over all x. A point estimator $T_n = T_n(X_1, X_2, \ldots, X_n)$ of a fixed and possibly unknown $F \in \{\text{All DFs}\}$ is the empirical DF $\widehat{F}_n$. This estimator has an asymptotically desirable property:

$$\sup_x |\widehat{F}_n(x) - F^*(x)| \xrightarrow{P} 0$$

because of the Gilvenko-Cantelli theorem in Proposition 11. Thus, we can simply use $\widehat{F}_n$, based on the realized data $(x_1, x_2, \ldots, x_n)$, as a point estimate of F^*.

On the basis of the DKW inequality (3.27), we can obtain a $1-\alpha$ confidence set or confidence interval $C_n(x) := [\underline{C}_n(x), \overline{C}_n(x)]$ about our point estimate of F:

$$\begin{aligned} \underline{C}_n(x) &= \max\{\widehat{F}_n(x) - \epsilon_n, 0\}, \\ \overline{C}_n(x) &= \min\{\widehat{F}_n(x) + \epsilon_n, 1\}, \\ \epsilon_n &= \sqrt{\frac{1}{2n}\log\left(\frac{2}{\alpha}\right)} \ . \end{aligned} \tag{3.30}$$

It follows from (3.27) that for any fixed and possibly unknown F^*:

$$P\left(\underline{C}_n(x) \leq F^*(x) \leq \overline{C}_n(x)\right) \geq 1-\alpha \ .$$

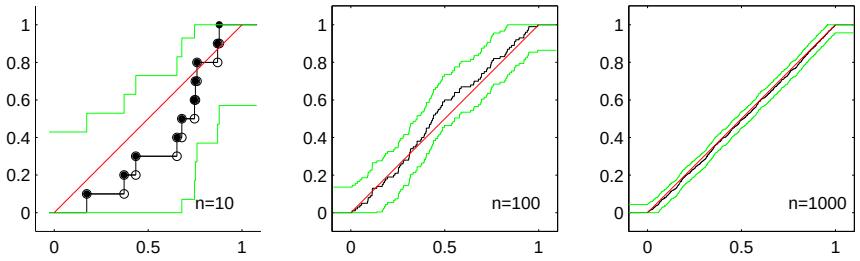
Let us look at a simple example next.

Labwork 3.6.2 (Estimating the DF of Uniform$(0,1)$ **RV)** *Consider the problem of estimating the DF of* Uniform$(0,1)$ *RV* U *on the basis of* $n_{=}10$ *samples. We use the function* `ECDF` *of Labwork 5.0.11 and* MATLAB*'s built-in function* `stairs` *to render the plots. Figure 3.10 was generated by* `PlotUniformECDFsConfBands.m` *given below.*

PlotUniformECDFsConfBands.m

```
% script PlotUniformECDFsConfBands.m to plot the ECDF from 10 and 100 samples
% from Uniform(0,1) RV
rand('twister',76534); % initialize the Uniform(0,1) Sampler
N = 3; % 10^N is the maximum number of samples from Uniform(0,1) RV
u = rand(1,10^N); % generate 1000 samples from Uniform(0,1) RV U

% plot the ECDF from the first 10 samples using the function ECDF
for i=1:N
    SampleSize=10^i;
    subplot(1,N,i)
    % Get the x and y coordinates of SampleSize-based ECDF in x1 and y1 and
    % plot the ECDF using the function ECDF
    if (i==1) [x1 y1] = ECDF(u(1:SampleSize),2,0.2,0.2);
    else
        [x1 y1] = ECDF(u(1:SampleSize),0,0.1,0.1);
        stairs(x1,y1,'k');
    end
    % Note PlotFlag is 1 and the plot range of x-axis is
    % incremented by 0.1 or 0.2 on either side due to last 2 parameters to ECDF
    % being 0.1 or 0.2
    Alpha=0.05; % set alpha to 5% for instance
    Epsn = sqrt((1/(2*SampleSize))*log(2/Alpha)); % epsilon_n for the confidence band
    hold on;
    stairs(x1,max(y1-Epsn,zeros(1,length(y1))),'g'); % lower band plot
    stairs(x1,min(y1+Epsn,ones(1,length(y1))),'g'); % upper band plot
    axis([-0.1 1.1 -0.1 1.1]);
    axis square;
    x=[0:0.001:1];
    plot(x,x,'r'); % plot the DF of Uniform(0,1) RV in red
    LabelString=['n=' num2str(SampleSize)];
    text(0.75,0.05,LabelString)
    hold off;
end
```

Figure 3.10: The empirical DFs $\widehat{F}_n^{(1)}$ from sample size $n = 10, 100, 1000$ (black), is the point estimate of the fixed and known DF $F(x) = x, x \in [0,1]$ of Uniform$(0,1)$ RV (red). The 95% confidence band for each $\widehat{F}_n$ are depicted by green lines.

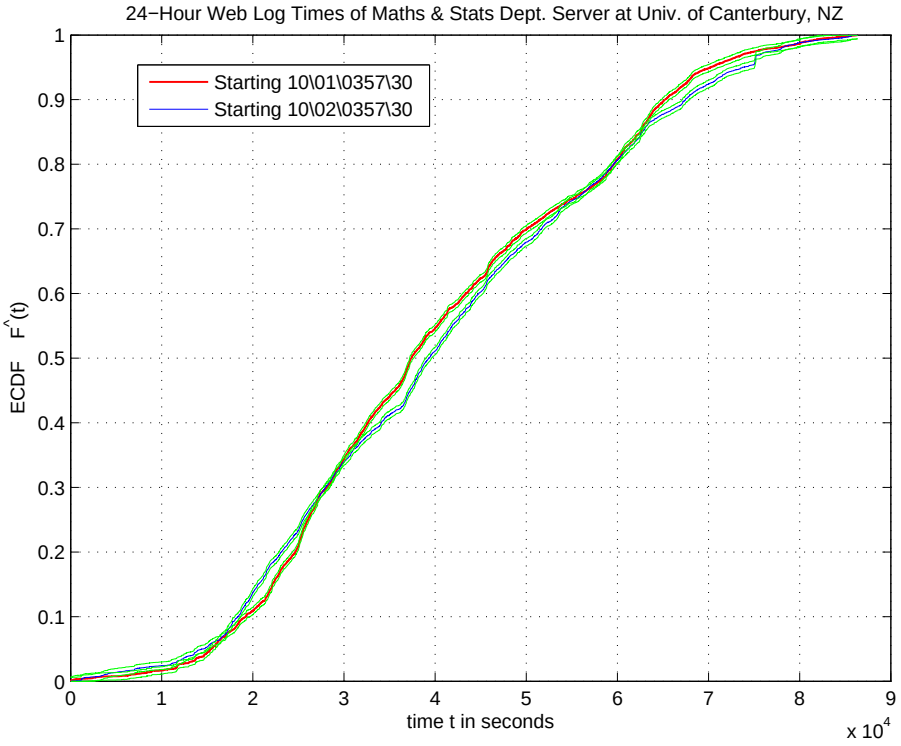


Next we look at a more interesting example involving real-world data.

Example 3.6.3 *First take a look at Data 5.0.18 to understand how the web login times to our Maths & Stats Department's web server (or requests to our WWW server) were generated. Figure 3.11 shows the login times in units of seconds over a 24 hour period starting at 0357 hours*

and 30 seconds (just before 4:00AM) on October 1st, 2007 (red line) and on October 2nd, 2007 (magenta). If we assume that some fixed and unknown DF $F^{(1)}$ specifies the distribution of login

Figure 3.11: The empirical DFs $\widehat{F}_{n_1}^{(1)}$ with $n_1 = 56485$, for the web log times starting October 1, and $\widehat{F}_{n_2}^{(2)}$ with $n_2 = 53966$, for the web log times starting October 2. Their 95% confidence bands are indicated by the green.



times for October 1st data and another DF $F^{(2)}$ for October 2nd data, then the non-parametric point estimates of $F^{(1)}$ and $F^{(2)}$ are simply the empirical DFs $\widehat{F}_{n_1}^{(1)}$ with $n_1 = 56485$ and $\widehat{F}_{n_2}^{(2)}$ with $n_2 = 53966$, respectively, as depicted in Figure 3.11. See the script of `WebLogDataProc.m` *in Data 5.0.18 to appreciate how the ECDF plots in Figure 3.11 were made.*

3.6.4 Plug-in Estimators

Recall from Section 3.2 that a statistical functional is simply any function of the DF F. For example, the median $T(F) = F^{[-1]}(1/2)$ is a statistical functional. The idea behind the plug-in estimator for a statistical functional is simple: just plug-in the point estimate $\widehat{F}_n$ instead of the unknown DF F^* to estimate the statistical functional of interest.

Definition 22 (Plug-in Estimator) *The plug-in estimator $\widehat{T}_n(X_1, \ldots, X_n) : \mathbb{X}_n \mapsto \mathbb{T}_n$ of a statistical functional of interest $T(F^*)$ is defined by:*

$$\widehat{T}_n(X_1, \ldots, X_n) = T(\widehat{F}_n) \ .$$

Labwork 3.6.5 (Plug-in Estimate for Median of Web Login Data) *Compute the plug-in estimates for the median for each of the data arrays:*

`WebLogSeconds20071001035730` *and* `WebLogSeconds20071002035730`

that can be loaded into memory by following the commands in the first 13 *lines of the script file* `WebLogDataProc.m` *of Data 5.0.18.*

Note that any statistical functional can be estimated using the plug-in estimator. However, to produce a $1-\alpha$ confidence set for the plug-in point estimate, we need bootstrap methods.

3.6.6 Non-parametric Bootstrap for Confidence Sets

The **bootstrap** is a statistical method for estimating standard errors and confidence sets of statistics, such as estimators.

Let $T_n := T_n((X_1, X_2, \ldots, X_n))$ be a statistic, i.e. any function of the data $X_1, X_2, \ldots, X_n \overset{\text{IID}}{\sim} F^*$. Suppose we want to know its variance $V_{F^*}(T_n)$, which clearly depends on the fixed and possibly unknown DF F^*. If our statistic T_n is one with an analytically unknown variance, then we can use the bootstrap to estimate it. The bootstrap idea has the following two basic steps:

Step 1: Estimate $V_{F^*}(T_n)$ with $V_{\widehat{F}_n}(T_n)$.

Step 2: Approximate $V_{\widehat{F}_n}(T_n)$ using simulated data from the "Bootstrap World."

For example, if $T_n = \overline{X}_n$, in Step 1, $V_{\widehat{F}_n}(T_n) = s_n^2/n$, where $s_n^2 = n^{-1}\sum_{i=1}^n (x_i - \overline{x}_n)$ is the sample variance and $\overline{x}_n$ is the sample mean. In this case, Step 1 is enough. However, when the statistic T_n is more complicated (e.g. $T_n = \widetilde{X}_n = F^{[-1]}(0.5)$), the sample median, then we may not be able to find a simple expression for $V_{\widehat{F}_n}(T_n)$ and may need Step 2 of the bootstrap.

$$\begin{array}{rcccc} \text{Real World Data come from} & F & \implies & X_1, X_2, \ldots, X_n & \implies & T_n((X_1, X_2, \ldots, X_n)) = t_n \\ \text{Bootstrap World Data come from} & \widehat{F}_n & \implies & X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet & \implies & T_n((X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet)) = t_n^\bullet \end{array}$$

Observe that drawing an observation from the ECDF $\widehat{F}_n$ is equivalent to drawing one point at random from the original data (think of the indices $[n] := \{1, 2, \ldots, n\}$ of the original data $X_1, X_2, \ldots, X_n$ being drawn according to the equi-probable de Moivre$(1/n, 1/n, \ldots, 1/n)$ RV on $[n]$). Thus, to simulate $X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet$ from $\widehat{F}_n$, it is enough to drawn n observations with replacement from $X_1, X_2, \ldots, X_n$.

In summary, the algorithm for Bootstrap Variance Estimation is:

Step 1: Draw $X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet \sim \widehat{F}_n$

Step 2: Compute $t_n^\bullet = T_n((X_1^\bullet, X_2^\bullet, \ldots, X_n^\bullet))$

Step 3: Repeat Step 1 and Step 2 B times, for some large B, say $B > 1000$, to get $t_{n,1}^\bullet, t_{n,2}^\bullet, \ldots, t_{n,B}^\bullet$

Step 4: Several ways of estimating the bootstrap confidence intervals are possible:

(a) The $1-\alpha$ Normal-based bootstrap confidence interval is:

$$C_n = [T_n - z_{\alpha/2}\widehat{se}_{boot}, T_n + z_{\alpha/2}\widehat{se}_{boot}] \ ,$$

where the bootstrap-based standard error estimate is:

$$\widehat{se}_{boot} = \sqrt{v_{boot}} = \sqrt{\frac{1}{B}\sum_{b=1}^B \left(t_{n,b}^\bullet - \frac{1}{B}\sum_{r=1}^B t_{n,r}^\bullet\right)^2}$$

(b) The $1-\alpha$ percentile-based bootstrap confidence interval is:

$$C_n = [\widehat{G^\bullet}_n^{-1}(\alpha/2), \widehat{G^\bullet}_n^{-1}(1-\alpha/2)],$$

where $\widehat{G^\bullet}_n$ is the empirical DF of the bootstrapped $t^\bullet_{n,1}, t^\bullet_{n,2}, \ldots, t^\bullet_{n,B}$ and $\widehat{G^\bullet}_n^{-1}(q)$ is the q^{th} sample quantile (3.9) of $t^\bullet_{n,1}, t^\bullet_{n,2}, \ldots, t^\bullet_{n,B}$.

Labwork 3.6.7 *Implement the above algorithm in* MATLAB *to estimate the bootstrap-based standard error of the plug-in estimate for the median of n IID observations from some chosen DF F^*. Choices for F^* include* Exponential($\lambda^* = 1$) *or* Lognormal$(0, 1)$.

Labwork 3.6.8 (Confidence Interval for Median Estimate of Web Login Data) *Find the* 95% *Normal-based bootstrap confidence interval as well as the* 95% *percentile-based bootstrap confidence interval for our plug-in estimate of the median for each of the data arrays:*

`WebLogSeconds20071001035730` *and* `WebLogSeconds20071002035730` .

Once again, the arrays can be loaded into memory by following the commands in the first 13 *lines of the script file* `WebLogDataProc.m` *of Section 5.0.18. Produce four intervals (two for each data-set). Do the confidence intervals for the medians for the two days intersect?*

```
>> WebLogDataProc % load in the data
>> Medianhat = median(WebLogSeconds20071001035730) % plug-in estimate of median
Medianhat =       37416
>> % store the length of data array
>> K=length(WebLogSeconds20071001035730)
K =       56485
>> B= 1000 % Number of Bootstrap replications
B =        1000
>> BootstrappedDataSet = WebLogSeconds20071001035730([ceil(K*rand(K,B))]);
>> size(BootstrappedDataSet) % dimension of the BootstrappedDataSet
ans =       56485        1000
>> BootstrappedMedians=median(BootstrappedDataSet); % get the statistic in Bootstrap world
>> % 95% Normal based Confidence Interval
>> SehatBoot = std(BootstrappedMedians); % std of BootstrappedMedians
>> % 95% C.I. for median from Normal approximation
>> ConfInt95BootNormal = [Medianhat-1.96*SehatBoot, Medianhat+1.96*SehatBoot]
ConfInt95BootNormal =    37242    37590
>> % 95% Percentile based Confidence Interval
ConfInt95BootPercentile = ...
    [qthSampleQuantile(0.025,sort(BootstrappedMedians)),...
    qthSampleQuantile(0.975,sort(BootstrappedMedians))]
ConfInt95BootPercentile =   37239    37554
```

Chapter 4

Hypothesis Testing

The subset of **all posable hypotheses** that remain **falsifiable** is the space of **scientific hypotheses**. Roughly, a falsifiable hypothesis is one for which a statistical experiment can be designed to produce data that an experimenter can use to falsify or reject it. In the statistical decision problem of hypothesis testing, we are interested in empirically falsifying a scientific hypothesis, i.e. we attempt to reject an hypothesis on the basis of empirical observations or data. Usually, the hypothesis we attempt to reject is called the **null hypothesis** or H_0 and its complement is called the **alternative hypothesis** or H_1. For example, consider the following two hypotheses:

Null Hypothesis H_0: The average waiting time at an Orbiter bus stop is 10 minutes.
Alternative Hypothesis H_1: The average waiting time at an Orbiter bus stop is not 10 minutes.

Suppose, we have observations of n waiting times $x_1, x_2, \ldots, x_n$ and are willing to assume a parametric model, say,

$$X_1, X_2, \ldots, X_n \overset{IID}{\sim} \text{Exponential}(\lambda^*)$$

with an unknown and fixed $\lambda^* \in \mathbf{\Lambda} = (0, \infty)$. Since the parameter λ of an Exponential(λ) RV is the reciprocal of the mean waiting time, we can formalise the above hypothesis testing problem of H_0 versus H_1 as follows:

$$H_0 : \lambda^* = \lambda_0 = 1/10 \qquad \text{versus} \qquad H_1 : \lambda^* \neq \lambda_0 = 1/10$$

For instance, if the sample mean $\overline{x}_n$ is much smaller or much larger than 10 minutes then we may be inclined to reject the null hypothesis that the average waiting time is 10 minutes. We will learn to formally test hypotheses in the sequel.

More generally, suppose $X_1, X_2, \ldots, X_n \overset{IID}{\sim} F(x_1; \theta^*)$, with an unknown and fixed $\theta^* \in \mathbf{\Theta}$. Let us partition the parameter space $\mathbf{\Theta}$ into $\mathbf{\Theta}_0$, the null parameter space, and $\mathbf{\Theta}_1$, the alternative parameter space, ie,

$$\mathbf{\Theta}_0 \cup \mathbf{\Theta}_1 = \mathbf{\Theta}, \qquad \text{and} \qquad \mathbf{\Theta}_0 \cap \mathbf{\Theta}_1 = \emptyset \ .$$

Then, we can formalise testing the null hypothesis versus the alternative as follows:

$$H_0 : \theta^* \in \mathbf{\Theta}_0 \qquad \text{versus} \qquad H_1 : \theta^* \subset \mathbf{\Theta}_1 \ .$$

The basic idea involves finding an appropriate rejection region $\mathbb{X}_R$ within the data space $\mathbb{X}$ and rejecting H_0 if the observed data $(x_1, x_2, \ldots, x_n)$ falls inside the rejection region $\mathbb{X}_R$,

$$\textbf{If } (x_1, x_2, \ldots, x_n) \in \mathbb{X}_R \subset \mathbb{X} \qquad \textbf{then } \text{reject } H_0 \qquad \textbf{else } \text{fail to reject } H_0.$$

Typically, the rejection region $\mathbb{X}_R$ is of the form:

$$\mathbb{X}_R := \{x : T(x) > c\}$$

where, T is the **test statistic** and c is the **critical value**. Thus, the problem of finding $\mathbb{X}_R$ boils down to that of finding T and c that are appropriate. Once the rejection region is defined, the possible outcomes of a hypothesis test are summarised in the following table.

Table 4.1: Outcomes of an hypothesis test.

	Fail to Reject H_0	Reject H_0
H_0 is True	OK	Type I Error
H_1 is True	Type II Error	OK

Definition 23 (Power, Size and Level of a Test) *The* **power function** *of a test with rejection region $\mathbb{X}_R$ is*

$$\beta(\theta) := P_\theta(x \in \mathbb{X}_R) \ . \tag{4.1}$$

That is, $\beta(\theta)$, the power of the test at the parameter value θ, is the probability that the observed data x, whose distribution is specified by θ, will fall in the rejection region $\mathbb{X}_R$.

The **size** *of a test with rejection region $\mathbb{X}_R$ is*

$$\alpha := \sup_{\theta \in \boldsymbol{\Theta}_0} \beta(\theta) := \sup_{\theta \in \boldsymbol{\Theta}_0} P_\theta(x \in \mathbb{X}_R) \ . \tag{4.2}$$

A test is said to have **level** $\overline{\overline{\alpha}}$ *if its size α is less than or equal to $\overline{\overline{\alpha}}$, i.e.:*

$$\overline{\overline{\alpha}} \geq \alpha := \sup_{\theta \in \boldsymbol{\Theta}_0} \beta(\theta) := \sup_{\theta \in \boldsymbol{\Theta}_0} P_\theta(x \in \mathbb{X}_R) \ . \tag{4.3}$$

Let us familiarize ourselves with some terminology in hypothesis testing next.

Table 4.2: Some terminology in hypothesis testing.

$\boldsymbol{\Theta}$	Test: H_0 versus H_1	Nomenclature
$\boldsymbol{\Theta} \subset \mathbb{R}^m, m \geq 1$	$H_0 : \theta^* = \theta_0$ versus $H_1 : \theta^* \neq \theta_1$	Simple Hypothesis Test
$\boldsymbol{\Theta} \subset \mathbb{R}^m, m \geq 1$	$H_0 : \theta^* \in \boldsymbol{\Theta}_0$ versus $H_1 : \theta^* \in \boldsymbol{\Theta}_1$	Composite Hypothesis Test
$\boldsymbol{\Theta} \subset \mathbb{R}^1$	$H_0 : \theta^* = \theta_0$ versus $H_1 : \theta^* \neq \theta_0$	Two-sided Hypothesis Test
$\boldsymbol{\Theta} \subset \mathbb{R}^1$	$H_0 : \theta^* \geq \theta_0$ versus $H_1 : \theta^* < \theta_0$	One-sided Hypothesis Test
$\boldsymbol{\Theta} \subset \mathbb{R}^1$	$H_0 : \theta^* \leq \theta_0$ versus $H_1 : \theta^* > \theta_0$	One-sided Hypothesis Test

Chapter 5

Appendix

Labwork 5.0.9 *Here are the functions to evaluate the pdf and DF of a* Normal(μ, σ^2) *RV* X *at a given* x.

NormalPdf.m

```
function fx = NormalPdf(x,Mu,SigmaSq)
% Returns the Pdf of Normal(Mu, SigmaSq), at x,
% where Mu=mean and SigmaSq = Variance
%
% Usage: fx = NormalPdf(x,Mu,SigmaSq)
if SigmaSq <= 0
   error('Variance must be > 0')
   return
end

Den = ((x-Mu).^2)/(2*SigmaSq);
Fac = sqrt(2*pi)*sqrt(SigmaSq);

fx = (1/Fac)*exp(-Den);
```

NormalCdf.m

```
function Fx = NormalCdf(x,Mu,SigmaSq)
% Returns the Cdf of Normal(Mu, SigmaSq), at x,
% where Mu=mean and SigmaSq = Variance using
% MATLAB's error function erf
%
% Usage: Fx = NormalCdf(x,Mu,SigmaSq)
if SigmaSq <= 0
   error('Variance must be > 0')
   return
end

Arg2Erf = (x-Mu)/sqrt(SigmaSq*2);
Fx = 0.5*erf(Arg2Erf)+0.5;
```

Plots of the pdf and DF of several Normally distributed RVs depicted in Figure 2.5 were generated using the following script file:

PlotPdfCdfNormal.m

```
% PlotPdfCdfNormal.m script file
% Plot of some pdf's and cdf's of the Normal(mu,SigmaSq) RV X
%
x=[-6:0.0001:6]; % points from the subset [-5,5] of the support of X
subplot(1,2,1) % first plot of a 1 by 2 array of plots
plot(x,NormalPdf(x,0,1),'r') % pdf of RV Z ~ Normal(0,1)
hold % to superimpose plots
```

```
plot(x,NormalPdf(x,0,1/10),'b') % pdf of RV X ~ Normal(0,1/10)
plot(x,NormalPdf(x,0,1/100),'m') % pdf of RV X ~ Normal(0,1/100)
plot(x,NormalPdf(x,-3,1),'r--') % pdf of RV Z ~ Normal(-3,1)
plot(x,NormalPdf(x,-3,1/10),'b--') % pdf of RV X ~ Normal(-3,1/10)
plot(x,NormalPdf(x,-3,1/100),'m--') % pdf of RV X ~ Normal(-3,1/100)
xlabel('x')
ylabel('f(x; \mu, \sigma^2)')
legend('f(x;0,1)','f(x;0,10^{-1})','f(x;0,10^{-2})','f(x;-3,1)','f(x;-3,10^{-1})','f(x;-3,10^{-2})')
subplot(1,2,2) % second plot of a 1 by 2 array of plots
plot(x,NormalCdf(x,0,1),'r') % DF of RV Z ~ Normal(0,1)
hold % to superimpose plots
plot(x,NormalCdf(x,0,1/10),'b') % DF of RV X ~ Normal(0,1/10)
plot(x,NormalCdf(x,0,1/100),'m') % DF of RV X ~ Normal(0,1/100)
plot(x,NormalCdf(x,-3,1),'r--') % DF of RV Z ~ Normal(-3,1)
plot(x,NormalCdf(x,-3,1/10),'b--') % DF of RV X ~ Normal(-3,1/10)
plot(x,NormalCdf(x,-3,1/100),'m--') % DF of RV X ~ Normal(-3,1/100)
xlabel('x')
ylabel('F(x; \mu, \sigma^2)')
legend('F(x;0,1)','F(x;0,10^{-1})','F(x;0,10^{-2})','F(x;-3,1)','F(x;-3,10^{-1})','F(x;-3,10^{-2})')
```

Labwork 5.0.10 *Here are the functions to evaluate the pdf and DF of an* Exponential(λ) *RV* X *ata given* x *(point or a vector).*

ExponentialPdf.m

```
function fx = ExponentialPdf(x,Lambda)
% Returns the Pdf of Exponential(Lambda) RV at x,
% where Lambda = rate parameter
%
% Usage: fx = ExponentialPdf(x,Lambda)
if Lambda <= 0
   error('Rate parameter Lambda must be > 0')
   return
end

fx = Lambda * exp(-Lambda * x);
```

ExponentialCdf.m

```
function Fx = ExponentialCdf(x,Lambda)
% Returns the Cdf of Exponential(Lambda) RV at x,
% where Lambda = rate parameter
%
% Usage: Fx = ExponentialCdf(x,Lambda)
if Lambda <= 0
   error('Rate parameter Lambda must be > 0')
   return
end
Fx = 1.0 - exp(-Lambda * x);
```

Plots of the pdf and DF of several Exponentially distributed RVs at four axes scales that are depicted in Figure 2.3 were generated using the following script file:

PlotPdfCdfExponential.m

```
% PlotPdfCdfExponential.m script file
% Plot of some pdf's and cdf's of the Exponential(Lambda) RV X
%
x=[0:0.0001:100]; % points from the subset [0,100] of the support of X
subplot(2,4,1) % first plot of a 1 by 2 array of plots
plot(x,ExponentialPdf(x,1),'r:','LineWidth',2) % pdf of RV X ~ Exponential(1)
hold on % to superimpose plots
plot(x,ExponentialPdf(x,10),'b--','LineWidth',2) % pdf of RV X ~ Exponential(10)
plot(x,ExponentialPdf(x,1/10),'m','LineWidth',2) % pdf of RV X ~ Exponential(1/10)
xlabel('x')
```

```
ylabel('f(x; \lambda)')
legend('f(x;1)','f(x;10)','f(x;10^{-1})')
axis square
axis([0,2,0,10])
title('Standard Cartesian Scale')
hold off

subplot(2,4,2)
semilogx(x,ExponentialPdf(x,1),'r:','LineWidth',2) % pdf of RV X ~ Exponential(1)
hold on % to superimpose plots
semilogx(x,ExponentialPdf(x,10),'b--','LineWidth',2) % pdf of RV X ~ Exponential(10)
semilogx(x,ExponentialPdf(x,1/10),'m','LineWidth',2) % pdf of RV X ~ Exponential(1/10)
%xlabel('x')
%ylabel('f(x; \lambda)')
%legend('f(x;1)','f(x;10)','f(x;10^{-1})')
axis square
axis([0,100,0,10])
title('semilog(x) Scale')
hold off

subplot(2,4,3)
semilogy(x,ExponentialPdf(x,1),'r:','LineWidth',2) % pdf of RV X ~ Exponential(1)
hold on % to superimpose plots
semilogy(x,ExponentialPdf(x,10),'b--','LineWidth',2) % pdf of RV X ~ Exponential(10)
semilogy(x,ExponentialPdf(x,1/10),'m','LineWidth',2) % pdf of RV X ~ Exponential(1/10)
%xlabel('x');
%ylabel('f(x; \lambda)');
%legend('f(x;1)','f(x;10)','f(x;10^{-1})')
axis square
axis([0,100,0,1000000])
title('semilog(y) Scale')
hold off

x=[ [0:0.001:1] [1.001:1:100000]]; % points from the subset [0,100] of the support of X
subplot(2,4,4)
loglog(x,ExponentialPdf(x,1),'r:','LineWidth',2) % pdf of RV X ~ Exponential(1)
hold on % to superimpose plots
loglog(x,ExponentialPdf(x,10),'b--','LineWidth',2) % pdf of RV X ~ Exponential(10)
loglog(x,ExponentialPdf(x,1/10),'m','LineWidth',2) % pdf of RV X ~ Exponential(1/10)
%xlabel('x')
%ylabel('f(x; \lambda)')
%legend('f(x;1)','f(x;10)','f(x;10^{-1})')
axis square
axis([0,100000,0,1000000])
title('loglog Scale')
hold off

x=[0:0.0001:100]; % points from the subset [0,100] of the support of X
subplot(2,4,5) % second plot of a 1 by 2 array of plots
plot(x,ExponentialCdf(x,1),'r:','LineWidth',2) % cdf of RV X ~ Exponential(1)
hold on % to superimpose plots
plot(x,ExponentialCdf(x,10),'b--','LineWidth',2) % cdf of RV X ~ Exponential(10)
plot(x,ExponentialCdf(x,1/10),'m','LineWidth',2) % cdf of RV X ~ Exponential(1/10)
xlabel('x')
ylabel('F(x; \lambda)')
legend('F(x;1)','f(x;10)','f(x;10^{-1})')
axis square
axis([0,10,0,1])
hold off

subplot(2,4,6) % second plot of a 1 by 2 array of plots
semilogx(x,ExponentialCdf(x,1),'r:','LineWidth',2) % cdf of RV X ~ Exponential(1)
hold on % to superimpose plots
semilogx(x,ExponentialCdf(x,10),'b--','LineWidth',2) % cdf of RV X ~ Exponential(10)
semilogx(x,ExponentialCdf(x,1/10),'m','LineWidth',2) % cdf of RV X ~ Exponential(1/10)
```

```
%xlabel('x')
%ylabel('F(x; \lambda)')
%legend('F(x;1)','F(x;10)','F(x;10^{-1})')
axis square
axis([0,100,0,1])
%title('semilog(x) Scale')
hold off

subplot(2,4,7)
semilogy(x,ExponentialCdf(x,1),'r:','LineWidth',2) % cdf of RV X ~ Exponential(1)
hold on % to superimpose plots
semilogy(x,ExponentialCdf(x,10),'b--','LineWidth',2) % cdf of RV X ~ Exponential(10)
semilogy(x,ExponentialCdf(x,1/10),'m','LineWidth',2) % cdf of RV X ~ Exponential(1/10)
%xlabel('x');
%ylabel('F(x; \lambda)');
%legend('F(x;1)','F(x;10)','F(x;10^{-1})')
axis square
axis([0,10,0,1])
%title('semilog(y) Scale')
hold off

x=[ [0:0.001:1] [1.001:1:100000]]; % points from the subset of the support of X
subplot(2,4,8)
loglog(x,ExponentialCdf(x,1),'r:','LineWidth',2) % cdf of RV X ~ Exponential(1)
hold on % to superimpose plots
loglog(x,ExponentialCdf(x,10),'b--','LineWidth',2) % cdf of RV X ~ Exponential(10)
loglog(x,ExponentialCdf(x,1/10),'m','LineWidth',2) % cdf of RV X ~ Exponential(1/10)
%xlabel('x')
%ylabel('F(x; \lambda)')
%legend('F(x;1)','F(x;10)','F(x;10^{-1})')
axis square
axis([0,100000,0,1])
%title('loglog Scale')
hold off
```

Labwork 5.0.11 *A* MATLAB *function to plot the empirical DF* (3.8) *of n user-specified samples. Read the following M-file for the algorithm:*

ECDF.m

```
function [x1 y1] = ECDF(x, PlotFlag, LoxD, HixD)
% return the x1 and y1 values of empirical CDF
% based on samples in array x of RV X
% plot empirical CDF if PlotFlag is >= 1
%
% Call Syntax: [x1 y1] = ECDF(x, PlotFlag, LoxD,HixD);
% Input      : x = samples from a RV X (a vector),
%              PlotFlag is a number controlling plot (Y/N, marker-size)
%              LoxD is a number by which the x-axis plot range is extended to the left
%              HixD is a number by which the x-aixs plot range is extended to the right
% Output     : [x1 y1] & empirical CDF Plot IF PlotFlag >= 1
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

R=length(x);       % assume x is a vector and R = Number of samples in x
x1=zeros(1,R+2);
y1=zeros(1,R+2);        % initialize y to null vectors
for i=1:1:R        % loop to append to x and y axis values of plot
y1(i+1)=i/R;                   % append equi-increments of 1/R to y
end                % end of for loop
x1(2:R+1)=sort(x);        % sorting the sample values
x1(1)=x1(2)-LoxD; x1(R+2)=x1(R+1)+HixD;   % padding x for emp CDF to start at min(x) and end at max(x)
y1(1)=0; y1(R+2)=1;         % padding y so emp CDF start at y=0 and end at y=1

% to make a ECDF plot for large number of points set the PlotFlag<1 and use
% MATLAB's plot function on the x and y values returned by ECDF -- stairs(x,y)
```

```
if PlotFlag >= 1       % Plot customized empirical CDF if PlotFlag >= 1
    %newplot;
    MSz=10/PlotFlag;   % set Markersize MSz for dots and circles in ECDF plot
                       % When PlotFlag is large MSz is small and the
                       % Markers effectively disappear in the ecdf plot
    R=length(x1);       % update R = Number of samples in x
        hold on            % hold plot for superimposing plots

    for i=1:1:R-1
        if(i>1 && i ~= R-1)
            plot([x1(i),x1(i+1)],[y1(i),y1(i)],'k o -','MarkerSize',MSz)
        end
        if (i< R-1)
          plot(x1(i+1),y1(i+1),'k .','MarkerSize',2.5*MSz)
        end
        plot([x1(i),x1(i+1)],[y1(i),y1(i)],'k -')
        plot([x1(i+1),x1(i+1)],  [y1(i),y1(i+1)],'k -')

    end

    hold off;
end
```

Ideally, this function needs to be rewritten using primitives such as MATLAB*'s* `line` *commands.*

Labwork 5.0.12 *Let us implement Algorithm 6 as the following* MATLAB *function:*

qthSampleQuantile.m

```
function qthSQ = qthSampleQuantile(q, SortedXs)
%
% return the q-th Sample Quantile from Sorted array of Xs
%
% Call Syntax: qthSQ = qthSampleQuantile(q, SortedXs);
%
% Input      : q = quantile of interest, NOTE: 0 <= q <= 1
%              SortedXs = sorted real data points in ascending order
% Output     : q-th Sample Quantile, ie, inverse ECDF evaluated at q

% store the length of the the sorted data array SortedXs in n
N = length(SortedXs);
Nminus1TimesQ = (N-1)*q; % store (N-1)*q in a variable
Index = floor(Nminus1TimesQ); % store its floor in a C-style Index variable
Delta = Nminus1TimesQ - Index;
if Index == N-1
    qthSQ = SortedXs(Index+1);
else
    qthSQ = (1.0-Delta)*SortedXs(Index+1) + Delta*SortedXs(Index+2);
end
```

Labwork 5.0.13 *Figure 3.5 was made with the following script file.*

LevyDensityPlot.m

```
x=linspace(-9,9,1500);y=x;
[X, Y]=meshgrid(x,y);
Z1 = (cos((0*X)+1) + 2*cos((1*X)+2) + 3*cos((2*X)+3) + 4*cos((3*X)+4) + 5*cos((4*X)+5));
Z2 = (cos((2*Y)+1) + 2*cos((3*Y)+2) + 3*cos((4*Y)+3) + 4*cos((5*Y)+4) + 5*cos((6*Y)+5));
Temp=50;
Z = exp(-(Z1 .* Z2 + (X + 1.42513) .^2 + (Y + 0.80032) .^ 2)/Temp);
mesh(X,Y,Z)
caxis([0, 10]);
rotate3d on
```

Labwork 5.0.14 *The negative of the Levy density* (**??**) *is encoded in the following M-file as a function to be passed to* MATLAB*'s* `fminsearch`*.*

NegLevyDensity.m

```
function NegLevyFunVal = NegLevyDensity(parameters);
X=parameters(1); Y=parameters(2); Temp=50.0;
Z1 = (cos((0*X)+1) + 2*cos((1*X)+2) + 3*cos((2*X)+3) + 4*cos((3*X)+4) + 5*cos((4*X)+5));
Z2 = (cos((2*Y)+1) + 2*cos((3*Y)+2) + 3*cos((4*Y)+3) + 4*cos((5*Y)+4) + 5*cos((6*Y)+5));
NegLevyFunVal = -exp(-(Z1 .* Z2 + (X + 1.42513) .^2 + (Y + 0.80032) .^ 2)/Temp);
```

Labwork 5.0.15 *Figure 3.6 was made with the following script file.*

LogNormalLogLklPlot.m

```
% Plots the log likelihood of LogNormal(lambd, zeta),
% for observed data vector x IIDLogNormal(lambd,zeta),
rand('twister',001);
x=exp(arrayfun(@(u)(Sample1NormalByNewRap(u,10.36,0.26^2)),rand(1,100)));
% log likelihood function
lambda=linspace(5,15.0,200);
zeta=linspace(0.1, 2,200);
[LAMBDA, ZETA]=meshgrid(lambda,zeta);
LAMBDA3=repmat(LAMBDA,[1 1 length(x)]);
ZETA3=repmat(ZETA,[1 1 length(x)]);

xx=zeros([1 1 length(x)]);xx(:)=x;
x3=repmat(xx,[length(lambda) length(zeta) 1]);
%l = -sum(log((1 ./ (sqrt(2*pi)*zeta) .* x) .* exp((-1/(2*zeta^2))*(log(x)-lambda).^2)));
LOGLKL = sum(log((1 ./ (sqrt(2*pi)*ZETA3) .* x3) .* exp((-1/(2*ZETA3.^2)).*(log(x3)-LAMBDA3).^2)),3);
LOGLKL(LOGLKL<0)=NaN;

caxis([0 0.1]*10^3);colorbar
axis([0 15 0 2 0 0.1*10^3])
clf; meshc(LAMBDA, ZETA, LOGLKL);
rotate3d on;
```

Labwork 5.0.16 *Figure 3.8 was made with the following script file.*

BernoulliMLEConsistency.m

```
clf;%clear any figures
rand('twister',736343); % initialize the Uniform(0,1) Sampler
N = 3; % 10^N is the maximum number of samples from RV
J = 100; % number of Replications for each n
u = rand(J,10^N); % generate 10X10^N samples from Uniform(0,1) RV U
p=0.5; % set p for the Bernoulli(p) trials
PS=[0:0.001:1]; % sample some values for p on [0,1] to plot likelihood
for i=1:N
    if(i==1) Pmin=0.; Pmax=1.0; Ymin=-70; Ymax=-10; Y=linspace(Ymin,Ymax,J); end
    if(i==2) Pmin=0.; Pmax=1.0; Ymin=-550; Ymax=-75; Y=linspace(Ymin,Ymax,J); end
    if(i==3) Pmin=0.3; Pmax=0.8; Ymin=-900; Ymax=-700; Y=linspace(Ymin,Ymax,J); end
    n=10^i;% n= sample size, ie, number of Bernoulli trials
    subplot(1,N,i)
    if(i==1) axis([Pmin Pmax Ymin -2]); end
    if(i==2) axis([Pmin Pmax Ymin -60]); end
    if(i==3) axis([Pmin Pmax Ymin -685]); end
    EmpCovSEhat=0; % track empirical coverage for SEhat
    EmpCovSE=0; % track empirical coverage for exact SE
    for j=1:J
     % transform the Uniform(0,1) samples to n Bernoulli(p) samples
     x=floor(u(j,1:n)+p);
     s = sum(x); % statistic s is the sum of x_i's
     % display the outcomes and their sum
     %display(x)
     %display(s)
     MLE=s/n; % Analyticaly MLE is s/n
     se = sqrt((1-p)*p/n); % standard error from known p
     sehat = sqrt((1-MLE)*MLE/n); % estimated standard error from MLE p
```

```
     Zalphaby2 = 1.96; % for 95% CI
     if(abs(MLE-p)<=2*sehat) EmpCovSEhat=EmpCovSEhat+1; end
     line([MLE-2*sehat MLE+2*sehat],[Y(j) Y(j)],'Marker','+','LineStyle',':','LineWidth',1,'Color',[1 .0 .0])
     if(abs(MLE-p)<=2*se) EmpCovSE=EmpCovSE+1; end
     line([MLE-2*se MLE+2*se],[Y(j) Y(j)],'Marker','+','LineStyle','-')
     % l is the Log Likelihood of data x as a function of parameter p
     l=@(p)sum(log(p ^ s * (1-p)^(n-s)));
     hold on;
     % plot the Log Likelihood function and MLE
     semilogx(PS,arrayfun(l,PS),'m','LineWidth',1);
     hold on; plot([MLE],[Y(j)],'.','Color','c'); % plot MLE
    end
    hold on;
    line([p p], [Ymin, l(p)],'LineStyle',':','Marker','none','Color','k','LineWidth',2)
    %axis([-0.1 1.1]);
    %axis square;
    LabelString=['n=' num2str(n) '  ' 'Cvrg.=' num2str(EmpCovSE) '/' num2str(J) ...
        ' ~=' num2str(EmpCovSEhat) '/' num2str(J)];
    %text(0.75,0.05,LabelString)
    title(LabelString)
    hold off;
end
```

Labwork 5.0.17 *The following script was used to generate the Figure 3.9.*

GilvenkoCantelliUnif01.m

```
% from Uniform(0,1) RV
rand('twister',76534); % initialize the Uniform(0,1) Sampler
N = 3; % 10^N is the maximum number of samples from Uniform(0,1) RV
u = rand(10,10^N); % generate 10 X 10^N samples from Uniform(0,1) RV U
x=[0:0.001:1];
% plot the ECDF from the first 10 samples using the function ECDF
for i=1:N
    SampleSize=10^i;
    subplot(1,N,i)
    plot(x,x,'r','LineWidth',2); % plot the DF of Uniform(0,1) RV in red
    % Get the x and y coordinates of SampleSize-based ECDF in x1 and y1 and
    % plot the ECDF using the function ECDF
    for j=1:10

     hold on;
     if (i==1) [x1 y1] = ECDF(u(j,1:SampleSize),2.5,0.2,0.2);
     else
        [x1 y1] = ECDF(u(j,1:SampleSize),0,0.1,0.1);
        stairs(x1,y1,'k');
     end
    end
%     % Note PlotFlag is 1 and the plot range of x-axis is
%     % incremented by 0.1 or 0.2 on either side due to last 2 parameters to ECDF
%     % being 0.1 or 0.2
%     Alpha=0.05; % set alpha to 5% for instance
%     Epsn = sqrt((1/(2*SampleSize))*log(2/Alpha)); % epsilon_n for the confidence band
     hold on;
%     stairs(x1,max(y1-Epsn,zeros(1,length(y1))),'g'); % lower band plot
%     stairs(x1,min(y1+Epsn,ones(1,length(y1))),'g'); % upper band plot
    axis([-0.1 1.1 -0.1 1.1]);
    %axis square;
    LabelString=['n=' num2str(SampleSize)];
    text(0.75,0.05,LabelString)
    hold off;
end
```

Data 5.0.18 (Our Maths & Stats Dept. Web Logs) *We assume access to a* `Unix` *terminal (*`Linux`, `Mas OS X`, `Sun Solaris`, *etc). We show how to get your hands dirty with web logs that track among others, every* `IP` *address and its time of login to*

our department web server over the world-wide-web. The raw text files of web logs may be manipulated but they are typically huge files and need some `Unix` *command-line utilities.*

```
rsa64@mathopt03:~> cd October010203WebLogs/
rsa64@mathopt03:~/October010203WebLogs> ls -al
-rw-r--r--+  1 rsa64 math 7527169 2007-10-04 09:38 access-07_log.2
-rw-r--r--+  1 rsa64 math 7727745 2007-10-04 09:38 access-07_log.3
```

The files are quite large over 7.5 MB each. So we need to compress it. We use the `gzip` *and* `gunzip` *utility in any* `Unix` *environment to compress and decompress these large text files of web logs. After compression the file sizes are more reasonable.*

```
rsa64@mathopt03:~/October010203WebLogs> gzip access-07_log.3
rsa64@mathopt03:~/October010203WebLogs> gzip access-07_log.2
rsa64@mathopt03:~/October010203WebLogs> ls -al
-rw-r--r--+  1 rsa64 math 657913 2007-10-04 09:38 access-07_log.2.gz
-rw-r--r--+  1 rsa64 math 700320 2007-10-04 09:38 access-07_log.3.gz
```

Next we show you some entries in the very tail end of one of these files:

```
rsa64@mathopt03:~/October010203WebLogs> zcat access-07_log.3.gz | tail
.
.
66.228.166.160 - - [02/Oct/2007:03:55:27 +1300] "GET / HTTP/1.0" 200 10133
192.197.69.29 - - [02/Oct/2007:03:55:27 +1300] "GET /images/uc/list.gif HTTP/1.1" 200 44
.
.
.
65.55.208.20 - - [02/Oct/2007:03:57:00 +1300] "GET /MATH371/07/S1/C/ HTTP/1.0" 200 9689
```

For confidentiality reasons we may not be able to make the raw data publicly available. Also, we are only interested in the inter-login time at our department's web-server. So we can "pipe" the output of `zcat` *which cancatenates the contents of a zipped file to the terminal using the pipe command* `|` *into another command-line utility called* `awk` *which is a pattern scanning and processing language to pull out the date information alone. We further pipe the output of* `awk` *into a stream editor called* `sed` *to get the time output in numbers formatted as* `Year month day hour minute seconds` *or* `YYYY MM DD HH MM SS`*. Finally, we can obtain the login times to our web-server for 2 full 24 hour cycles. The first log file begins at* `03 57 30` *hours on* `01/Oct/2007` *while the second log file begins at* `03 57 30` *hours on* `02/Oct/2007`*, as shown in the output below (only the beginning and ending of each file is shown).*

```
rsa64@mathopt03:~/October010203WebLogs> zcat access-07_log.3.gz | grep ' 200 '
| awk '{ print \$4}' | sed -e  's/\[\([0-9]\{2\}\)\/\([a-Z]\{3\}\)\/\([0-9]\{4\}\)
:\([0-9]\{2\}\): \([0-9]\{2\}\):\([0-9]\{2\}\)/\3 10 \1 \4 \5 \6/'
2007 10 01 03 57 40
2007 10 01 03 57 41
.
.
.
2007 10 02 03 56 46
2007 10 02 03 57 00
rsa64@mathopt03:~/October010203WebLogs> zcat access-07_log.2.gz | grep ' 200 '
| awk '{ print \$4}'| sed -e 's/\[\([0-9]\{2\}\)\/\([a-Z]\{3\}\)\/\([0-9]\{4\}\)
:\([0-9]\{2\}\):\([0-9]\{2\}\):\([0-9]\{2\}\)/\3 10 \1 \4 \5 \6/'
2007 10 02 03 57 48
2007 10 02 03 58 31
.
.
.
2007 10 03 03 56 21
2007 10 03 03 56 52
```

Finally, there are 56485 *and* 53966 *logins for the two 24-hour cycles, starting* `01/Oct` *and* `01/Oct`*, respectively. We can easily get these counts by further piping the previous output into the line counting utility* `wc` *with the -l option. All the* `Unix` *command-line tools mentioned earlier can be learned by typing* `man` *followed by the tool-name, for eg. type* `man sed` *to learn about the usage of* `sed` *ata* `Unix` *command shell. We further pipe the output of login times for the two 24-hour cycles starting* `01/Oct` *and* `02/Oct` *in format* `YYYY MM DD HH MM SS` *to* `| sed -e 's/2007 10 //' > WebLogTimes20071001035730.dat` *and* `... > WebLogTimes20071002035730.dat`*, respectively to strip away the redundant information on* `YYYY MM` *, namely* `2007 10` *, and only save the relevant information of* `DD HH MM SS` *in files named* `WebLogTimes20071001035730.dat` *and* `WebLogTimes20071002035730.dat`*, respectively. These two files have the data of interest to us. Note that the size of these two uncompressed final data files in plain text are smaller than the compressed raw web log files we started out from.*

```
rsa64@mathopt03:~/October010203WebLogs> ls -al
-rw-r--r--+  1 rsa64 math 677820 2007-10-05 15:36 WebLogTimes20071001035730.dat
-rw-r--r--+  1 rsa64 math 647592 2007-10-05 15:36 WebLogTimes20071002035730.dat
-rw-r--r--+  1 rsa64 math 657913 2007-10-04 09:38 access-07_log.2.gz
-rw-r--r--+  1 rsa64 math 700320 2007-10-04 09:38 access-07_log.3.gz
```

Now that we have been familiarized with the data of login times to our web-server over 2 24-hour cycles, let us do some statistics. The log files and basic scripts are courtesy of the Department's computer systems administrators Paul Brouwers and Steve Gourdie. This data processing activity was shared in such detail to show you that statistics is only meaningful when the data and the process that generated it are clear to the experimenter. Let us process the data and visualize the empirical distribution functions using the following script:

WebLogDataProc.m

```
load WebLogTimes20071001035730.dat % read data from first file
% multiply day (October 1) by 24*60*60 seconds, hour by 60*60 seconds,
% minute by 60 seconds and seconds by 1, to rescale time in units of seconds
SecondsScale1 = [24*60*60; 60*60; 60; 1;];
StartTime1 = [1 3 57 30] * SecondsScale1; % find start time in seconds scale
%now convert time in Day/Hours/Minutes/Seconds format to seconds scale from
%the start time
WebLogSeconds20071001035730 = WebLogTimes20071001035730 * SecondsScale1 - StartTime1;

% repeat the data entry process above on the second file
load WebLogTimes20071002035730.dat %
SecondsScale1 = [24*60*60; 60*60; 60; 1;];
StartTime2 = [2 3 57 30] * SecondsScale1;
WebLogSeconds20071002035730 = WebLogTimes20071002035730 * SecondsScale1 - StartTime2;

% calling a more efficient ECDF function for empirical DF's
[x1 y1]=ECDF(WebLogSeconds20071001035730,0,0,0);
[x2 y2]=ECDF(WebLogSeconds20071002035730,0,0,0);
stairs(x1,y1,'r','linewidth',1) % draw the empirical DF for first dataset
hold on;
stairs(x2,y2,'b') % draw empirical cdf for second dataset

% set plot labels and legends and title
xlabel('time t in seconds')
ylabel('ECDF    F^\^(t)')
grid on
legend('Starting 10\01\0357\30', 'Starting 10\02\0357\30')
title('24-Hour Web Log Times of Maths & Stats Dept. Server at Univ. of Canterbury, NZ')

%To draw the confidence bands
Alpha=0.05; % set alpha
% compute epsilon_n for first dataset of size 56485
Epsn1 = sqrt((1/(2*56485))*log(2/Alpha));
stairs(x1,max(y1-Epsn1,zeros(1,length(y1))),'g') % lower 1-alpha confidence band
stairs(x1,min(y1+Epsn1,ones(1,length(y1))),'g') % upper 1-alpha confidence band

% compute epsilon_n for second dataset of size 53966
Epsn2 = sqrt((1/(2*53966))*log(2/Alpha));
stairs(x2,max(y2-Epsn2,zeros(1,length(y2))),'g') % lower 1-alpha confidence band
stairs(x2,min(y2+Epsn2,ones(1,length(y2))),'g') % upper 1-alpha confidence band
```