Week 2

Exploratory Data Analysis for Univariate Data: Graphical Summaries

So why we need Descriptive Statistics? You can ask - if we have the data, if we have all observations we want, why we need to describe that data? Say, if we have, for a particular student, all his/her grades for all courses, why we need to describe them? And why people calculate the GPA? Hopefully, you will not give the answer to me - you will answer this question by yourself.

Well, of course, having all the datapoints is very nice. But...

Example, Dataset: Look at the following data:
29.70, 29.72, 30.37, 30.19, 30.12, 30.28, 29.88, 29.72, 29.80, 29.99, 29.56, 29.55, 28.99, 29.22, 28.27, 28.05, 27.45, 27.88, 27.94, 29, 27.57, 27.01, 26.15, 26.78, 25.91, 25.53, 25.78, 25.20, 24.55, 24.49, 25.14, 23.82, 23.93, 24.87, 24.18, 24.39, 22.98, 23.83, 20.79, 20.14, 20.49, 18.21, 18.19, 17.88, 17.71, 17.82, 17.5, 17.45, 18.54, 18.76, 16.26, 16.13, 15.64, 15.64

Well, I can guess (and even bet on that for a Bounty chocolate stick) that you have not read all the data values ◡̈ (don't even try to go back to read and win the bet ◡̈). In fact, this is a real example of a data - the weekly close prices (closing prices) for the time period Feb 03, 2017 - Feb 03, 2018 for the General Electric Company stock. Of course, having all the values is very valuable, but if you will show this to your clients, nobody will be happy. So describing - visualizing the data or summarizing a data is important for communicating, and maybe not a trivial task.

For example, we can display the above GE Stock price dataset in a graphical form, and (hopefully) everybody will get the picture: see the Figure 2.1 below.

R code, GE Stock Price: To obtain the GE Stock price Figure 2.1, use:

```
x <- c(29.70, 29.72, 30.37, 30.19, 30.12, 30.28, 29.88, 29.72, 29.80, 29.99, 29.56,
       29.55, 28.99, 29.22, 28.27, 28.05, 27.45, 27.88, 27.94, 29, 27.57, 27.01,
       26.15, 26.78, 25.91, 25.53, 25.78, 25.20, 24.55, 24.49, 25.14, 23.82, 23.93,
       24.87, 24.18, 24.39, 22.98, 23.83, 20.79, 20.14, 20.49, 18.21, 18.19, 17.88,
       17.71, 17.82, 17.5, 17.45, 18.54, 18.76, 16.26, 16.13, 15.64, 15.64)

plot(x, type = "l", lwd = 2, col = "blue", main = "GE Weekly closing prices,
     Feb 03, 2017 - Feb 03, 2018", xlab = "No. of the week in the inverse order",
     ylab = "GE Stock Price")
```

Of course, it will be much better to draw like in the[1] Fig. 2.2.

[1]You can find more info about the "quantmod" package at https://www.quantmod.com/examples/intro/.

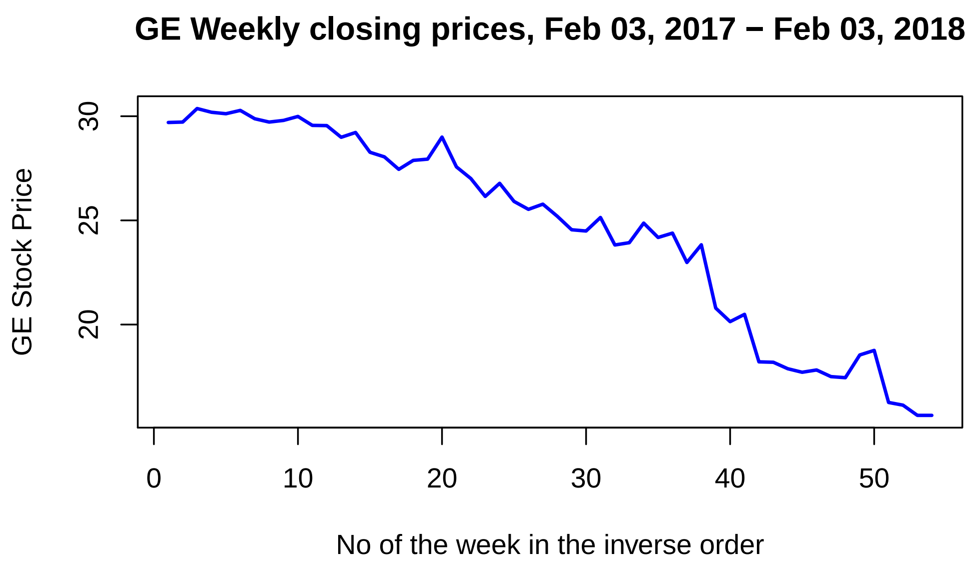


Fig. 2.1: GE Stock Price Data

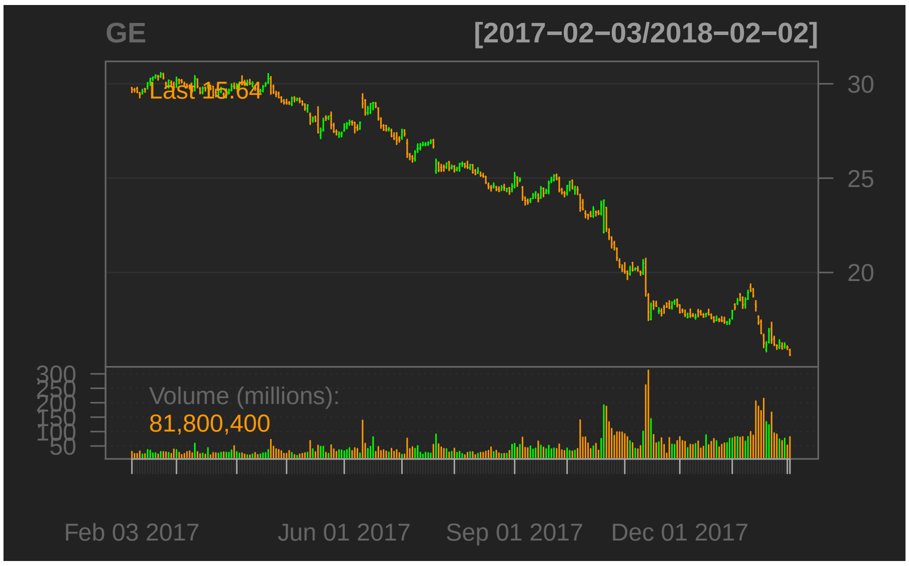


Fig. 2.2: GE Stock Price Data, using the **R**'s quantmod package

R CODE, GE STOCK PRICE, QUANTMODE PACKAGE: To obtain the GE Stock price Figure 2.2, you need to first install the "quantmod" package. To that end, uncomment the "install.packages" line below and run it.

```
#GE Stock Prices
#You need to install quantmod package, if it is not already installed
#install.packages("quantmod")
library(quantmod) # Some kind of include command from C++
start <- as.Date("2017-02-03")  #Time period beginning
end <- as.Date("2018-02-03")    #Time period end
```

```
getSymbols("GE", src="yahoo", from = start, to = end)
barChart(GE)
```

By the way, in the above example we have a so-called *time series data*. Right this moment we have a course on Times Series at AUA.

The moral of the story: data description and representation is important!

So let's start. Here, for the beginning, we will assume that we have a univariate numerical data (observations, dataset), $x_1, x_2, ..., x_n$, where $x_i \in \mathbb{R}$. In this case we will say that we are given a (univariate, 1D) dataset x. Here x_i-s can be the values of some variable for different observations:

EXAMPLE, 1D DATASET: Say, we are given a dataset x, where x_k is the AUA Math Test grade for the student k, $k = 1, ..., 100$.

EXAMPLE, 1D DATASET: Say, we are doing a salary size study for Math professors in Armenia. We choose a sample of size 10 (some portion of Math Professors $\smile$), ask them about their salary, and record them as x_1, x_2, ..., x_{10}, where x_1 is the salary of the first Prof, x_2 for the second Prof etc.

Btw, will it be a nice study, if we will choose at random 10 CS students, give them an instruction to ask one of their Math instructors about the salary?

As we have seen above, without any organization of the data it is difficult to get a sense of what is going on in the data: what is the the most typical value, or how concentrated or spread are values about that typical value, or if there are gaps in values etc. So we want to have some simple characteristics that will help to make summaries. There are various graphical representation methods and numerical summaries that help us to draw conclusion about the density/frequency distribution of our data.

So we will give below:

- Graphical Summaries, Graphical Representation Methods;
- Numerical Summaries, Numerical Description Methods.

2.1 Frequency Tables

Before doing a graphical representation, we can get an information about the data by calculating how frequently different numbers appear in our dataset. Of course, this will be meaningful in the case of discrete variable, since for a continuous variable, mostly you will have every datapoint appearing only once[2].

We assume that we have a univariate discrete numerical data $x_1, x_2, ..., x_n$, where $x_i \in \mathbb{R}$, for $i = 1, ..., n$.

[2]Say, if x is a variable showing the lifetime for a light bulb, then, if you are not making rough rounding, say, if you fix the exact (as precise as you can) lifetime of a bulb - something like 2 years, 1 months, 12 days, 23 hours, 22 minutes, 45 seconds and 210 milliseconds, for some bulb, then it is highly unprobable that you will have 2 bulbs with the same lifetime.

Definition 2.1. *The **frequency** of a value t in observations $x_1, x_2, ..., x_n$ is the number of times t occurs in observations:*

$$\textit{Frequency of } t = \textit{number of occurrences of } t \textit{ in data.}$$

Definition 2.2. *The **relative frequency** (or percentage) of a value t in observations $x_1, x_2, ..., x_n$ is the ratio of frequency of t divided by the total number of observations, n:*

$$\textit{Relative Frequency of } t = \frac{\textit{Frequency of } t}{\textit{Total Number of Observations}} = \frac{\textit{Frequency of } t}{n}.$$

EXAMPLE, FREQUENCY AND RELATIVE FREQUENCY TABLE: Consider the following dataset: the number of A students in my different courses:

$$9, 18, 20, 11, 15, 13, 14, 12, 10, 8, 26, 6, 14, 6$$

To construct the Frequency or Relative Frequency tables, it is convenient to sort our dataset[3]. The sorted dataset is

$$6, 6, 8, 9, 10, 11, 12, 13, 14, 14, 15, 18, 20, 26.$$

Now, the frequency of 6 is 2, since we have 2 sixes in or dataset. The frequency of 8 is 1 etc. The relative frequency of 6 is $\frac{2}{14} = \frac{1}{7}$, since we have 14 observations. Similarly, the relative frequency of 8 is $\frac{1}{14}$ and so on. The result will be:

Table 2.1: The Frequency/Relative Frequency Table

Value	Frequency	Relative Frequency
6	2	2/14
8	1	1/14
9	1	1/14
10	1	1/14
11	1	1/14
12	1	1/14
13	1	1/14
14	2	2/14
15	1	1/14
18	1	1/14
20	1	1/14
26	1	1/14

Remark, Frequency/Relative Frequency Properties: Please note that the sum of all frequencies will give the number of all observations:

$$\sum_{x_k \in x} \text{Frequency of } x_k = n,$$

and the sum of all relative frequencies will sum up to 1:

$$\sum_{x_k \in x} \text{Relative Frequency of } x_k = 1.$$

R code, Frequency Table: To get the frequency table in **R**, you can use the *table* command:

```
#Frequency Table
x <- c(1,2,3,1,2,4,5,3,6,2,3,4,1,5,2,2,3)
table(x)
```

Here the unique data values and frequencies are not accessible (say, if I want to add all frequencies, or to calculate the relative frequencies). Another way of doing this is the following:

```
x <- c(1,2,3,1,2,4,5,3,6,2,3,4,1,5,2,2,3)
y <- as.data.frame(table(x))
y  #Displaying the Frequency Table
a <- as.numeric(y$x) #Choosing the values of x
b <- y$Freq #Choosing the frequencies
c <- b/length(x) #Calculating Relative frequencies
y$RelFreq <- c   #Adding a new column to the data frame (table)
y  #Displaying the result
```

2.2 Graphical Representation of Data: Histogram for a Discrete Variable

Now, to give a graphical representation for a Discrete Variable frequency table, one draws the frequency histogram or the relative frequency histogram.

Assume we have some observations of a particular discrete variable: $x_1, x_2, ..., x_n$, with $x_k \in \mathbb{R}$.

Definition 2.3. *The **frequency histogram** for this data is the function* $h : \mathbb{R} \to \mathbb{N} \cup \{0\}$ *defined by*

$$h_{freq}(x) = \textit{the number of occurences of } x \textit{ in our dataset} = \textit{the frequency of } x, \qquad x \in \mathbb{R}.$$

Mathematically, one can define

$$h_{freq}(x) = \sum_{i=1}^{n} \mathbb{1}(x_i = x), \qquad x \in \mathbb{R},$$

where $\mathbb{1}(\cdot)$ is the truth indicator function defined by

$$\mathbb{1}(\text{TRUE}) = 1 \qquad \text{and} \qquad \mathbb{1}(\text{FALSE}) = 0.$$

In other words, this is just the same as the frequency, but in the form of a function defined on $\mathbb{R}$[4].

EXAMPLE, INDICATOR FUNCTION: As an example how the Truth Indicator Function works, assume $t = 3$. Then

$$\mathbb{1}(t < 5) = \mathbb{1}(\text{TRUE}) = 1, \qquad \text{but} \qquad \mathbb{1}(t > 4) = 0.$$

Now, we can define the relative frequency histogram in a similar way:

Definition 2.4. *The* ***relative frequency histogram*** *for the above dataset is the function* $h : \mathbb{R} \to [0, +\infty)$ *defined by*

$$h_{relfreq}(x) = \textit{the relative frequency of } x = \frac{1}{n} \cdot \sum_{i=1}^{n} \mathbb{1}(x_i = x), \qquad x \in \mathbb{R}.$$

The following properties are very easy to prove:

Proposition 2.1. *For any dataset* $x_1, ..., x_n$,

a. $0 \leqslant h_{freq}(x) \leqslant n$ *for any* $x \in \mathbb{R}$;

b. $\sum_{x \in \mathbb{R}} h_{freq}(x) = n$[5].

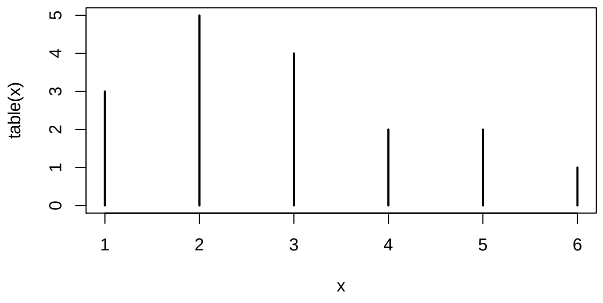
c. $0 \leqslant h_{relfreq}(x) \leqslant 1$ *for any* $x \in \mathbb{R}$;

d. $\sum_{x \in \mathbb{R}} h_{relfreq}(x) = 1$.

Now, having the histogram (or the frequency table), we can represent it on the Cartesian plane by using the Line Graph or a Bar Graph. We put the values of our variable on the OX axis, and for each value, we draw a line of height equal to the frequency (or the relative frequency, if we want to have these values) of that value in the dataset. Below are some examples for the Frequency Histogram, and you can easily update for the Relative ones.

R CODE, LINE GRAPH: The following simple code draws a Line Graph:

```
#Line Graph, Simple Version
x <- c(1,2,3,1,2,4,5,3,6,2,3,4,1,5,2,2,3)
plot(table(x))
```

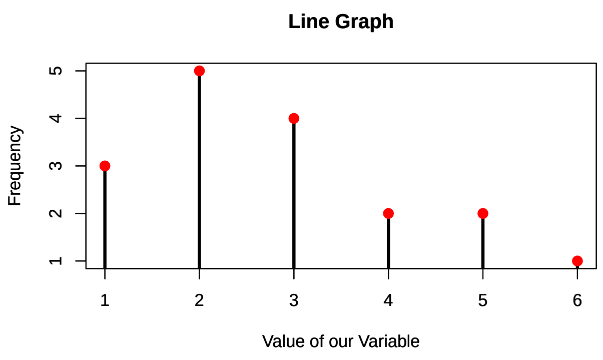


[4]Well, you can say that the above definition of the frequency was also a definition of a function. I will have nothing to argue in this case. Btw, people sometimes define the frequency of x to be 0, if x is not in our dataset.

[5]In fact, the sum involves only finitely many number of non-zero values, so the sum is correctly defined.

R code, Line Graph: Another way to draw a Line Graph:

```
#Line Graph, Another Version
x <- c(1,2,3,1,2,4,5,3,6,2,3,4,1,5,2,2,3)
y <- as.data.frame(table(x))
a <- as.numeric(y$x)
b <- y$Freq
plot(a,b,type="h", lwd=3,  xlab = "Value of
our Variable", ylab = "Frequency", main
= "Line Graph")
points(a,b, pch=16, cex=1.4, col="red")
```



R code, BarPlot: Barplot is almost the same as the Line Graph above with one difference: it plots rectangles (bars) of small width instead of lines. Here is an example how to do a barplot for the data above. The plot is given in Fig. 2.3.

```
#BarPlot
x <- c(1,2,3,1,2,4,5,3,6,2,3,4,1,5,2,2,3)
z <- table(x)
barplot(z, main = "BarPlot Example")
```

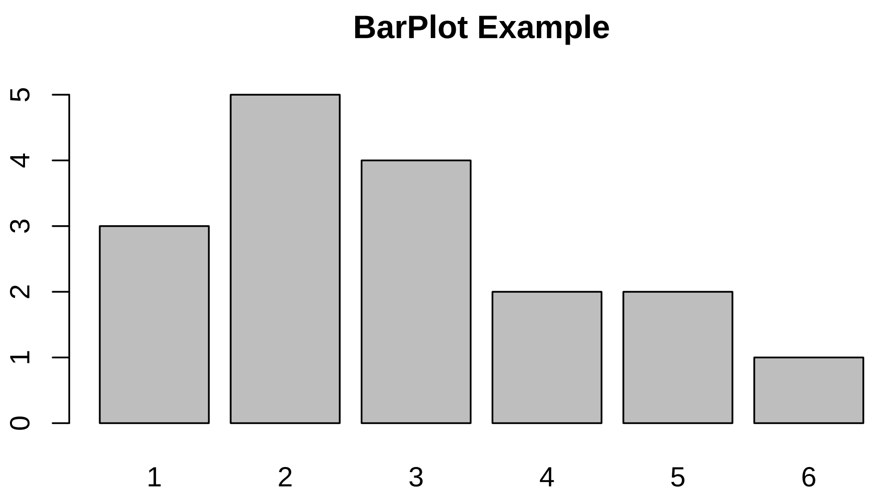


Fig. 2.3: Barplot Example

R code, Frequency Polygon: Another way to graph the frequency table is to give the Frequency Polygon. We plot the points with the value of our variable as the x-coordinate of a point, and the frequency of x as the y-coordinate, we join that points by line segments, and this is the Frequency Polygon.

```
#Frequency Polygon
x <- c(1,2,3,1,2,4,5,3,6,2,3,4,1,5,2,2,3)
y<-as.data.frame(table(x))
a<-as.numeric(y$x)
b<-y$Freq
plot(a,b,type = "l", lwd = 3, main = "Frequency Polygon", xlab = "Value of our
Variable", ylab = "Frequency")
points(a,b, pch=16, cex=1.4, col="red")
```

The result is in Fig. 2.4.

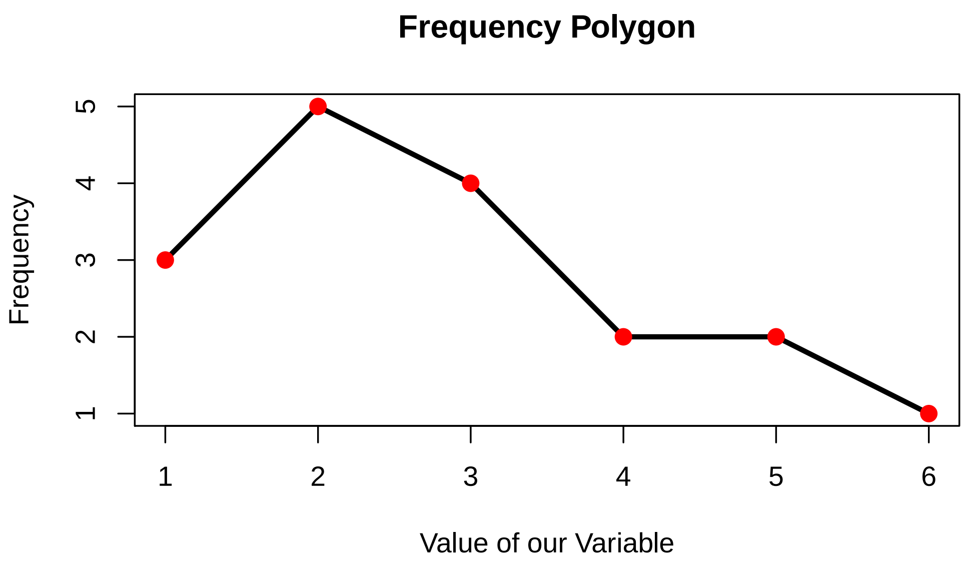


Fig. 2.4: Frequency Polygon Example

2.3 Graphical Representation of Data: Histogram for a Grouped Data / Continuous Variable

Now assume we have some finite number of observations on some continuous variable, $x_1, ..., x_n$. The general idea is the following: we want to group our data into some subgroups, and then visualize.

Example, Grouping: Say, our variable is an age, and we want to group to: ages between 0 and 6 years, between 6 and 15 years, 15 and 120 years; or, our variable is the blood (systolic) pressure, we group into categories: pressure lower than 80, between 80 and 100, between 100 and 130, higher than 130; or, our variable is the wage of a person, and we group them into: wage between 30,000AMD and 100,000AMD, between 100,000AMD and 700,000AMD, higher than 700,000AMD etc.

To that end, generally, we first fix an interval (usually, a closed interval) containing all observations: $x_i \in I$, for any $i = 1, ..., n$. For example, we can choose $I = [\min x_k, \max x_k]$ or a larger interval. Then we choose a finite partition of the interval I: we choose a finite number of disjoint intervals $I_0, I_1, ..., I_k$, which are called **class intervals** or **bins**, in such a way that

$$I_p \cap I_q = \varnothing, \text{ for } p \neq q, \qquad \text{and} \qquad \bigcup_{j=0}^{k} I_j = I.$$

We will assume that I_j includes its left endpoint but not the right endpoint[6] (except the case when I_j is the rightmost interval - in that case I_j includes also the right endpoint).

Example, Class Intervals or Bins: Assume our observations lie in the interval $I = [-3, 22]$. Then we can choose the following partition:

$$I_1 = [-3, 1), \ I_2 = [1, 4), \ I_3 = [4, 12), \ I_4 = [12, 22].$$

Or, we can choose for the same example,

$$I_1 = [-3, 10), \quad I_2 = [10, 20), \quad I_3 = [20, 22].$$

Example, Class Intervals or Bins: For the ages example above, we can take $I = [0, 120]$ and the following partition as bins:

$$I_1 = [0, 6), \ I_2 = [6, 15), \ I_3 = [15, 120].$$

Now, having the class intervals (bins), we calculate the number n_j of datapoints x_i lying in I_j:

$$n_j = \text{the number of data points in } I_j = \sum_{i=1}^{n} \mathbb{1}(x_i \in I_j), \qquad j = 0, 1, 2, ..., k.$$

Definition 2.5. *The* ***frequency histogram*** *of our continuous (or a grouped) data* $x_1, ..., x_n$ *is the piecewise constant function*

$$h_{freq}(x) = n_j, \qquad \forall x \in I_j, \quad j = 1, 2, ..., k.$$

In other words, frequency histogram shows the number of observations in our dataset in each bin, in each class interval. One also defines $h_{freq}(x) = 0$ for all $x \notin I$.

[6]This is just a convention, the left-end inclusion convention. Sometimes people choose the right-end inclusion convention. Important is to know and to state in your example which convention are you using.

Example, Frequency Histogram for a Continuous or a Grouped Data: Consider the following example: or dataset is

$$1,3,4,2,5,6,7,5,3,4,3,5,6,4,5,6,5,7,6,7,8.$$

We choose $I = [0,10]$, which covers the range of our datapoints. Next, we choose some partition of I, say,

$$I_1 = [0,4), \quad I_2 = [4,7), \quad I_3 = [7,10].$$

Then we calculate n_j-s. We have 3 bins, so we need to calculate n_1, n_2 and n_3. For our ease, it will be convenient to sort our dataset:

$$1,2,3,3,3,4,4,4,5,5,5,5,5,6,6,6,6,7,7,7,8$$

- n_1 = the number of observations in $I_1 = [0,4) = 5$;
- n_2 = the number of observations in $I_2 = [4,7) = 12$;
- n_3 = the number of observations in $I_3 = [7,10] = 4$.

Now, the frequency histogram for our data is:

$$h_{freq}(x) = \begin{cases} 5, & x \in I_1 = [0,4) \\ 12, & x \in I_2 = [4,7) \\ 4, & x \in I_3 = [7,10] \\ 0, & \text{otherwise} \end{cases}$$

See below for the code and the graph.

R code, Histogram of a Continuous or a Grouped Data:

```
#Frequency Histogram
x <- c(1,3,4,2,5,6,7,5,3,4,3,5,6,4,5,6,5,7,6,7,8)
bins <- c(0,4,7,10)
hist(x, breaks = bins, freq = T, right = F)
# right = F is for excluding the right-endpoints,
# freq = T is to draw the frequencies
```

The result is in Fig. 2.5.

Similarly, we define the relative frequency histogram for a continuous data:

Definition 2.6. *The **relative frequency histogram** of our continuous data $x_1, ..., x_n$ is the piecewise constant function*

$$h_{relfreq}(x) = \frac{n_j}{n}, \qquad \forall x \in I_j, \quad j = 1,2,...,k.$$

In other words, the relative frequency histogram shows the relative frequency (the percentage) of observations in our dataset in each bin, in each class interval.

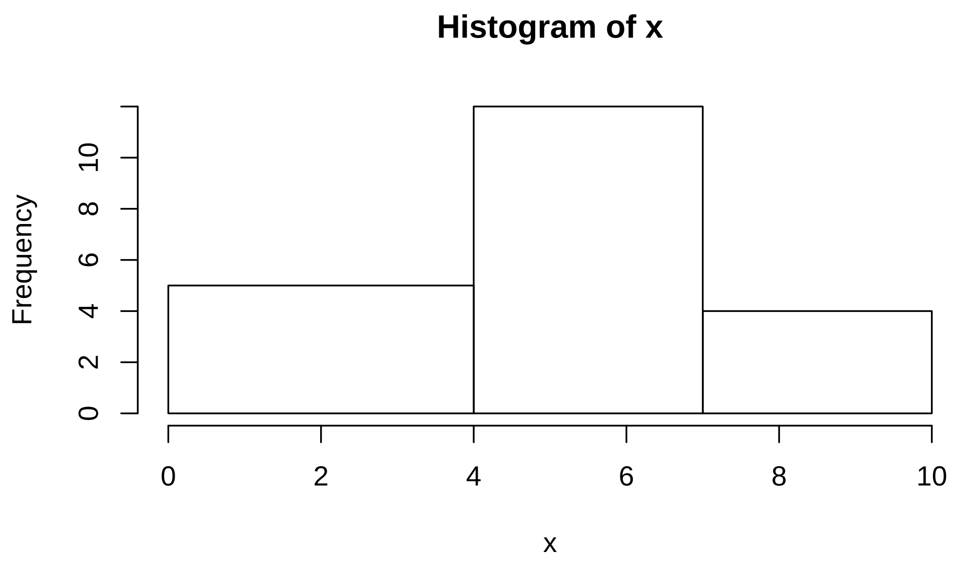


Fig. 2.5: Frequency Histogram

Example, Relative Frequency Histogram for a Continuous or a Grouped Data: For the above example, we will have (note that the number of observations is $n = 21$):

$$h_{relfreq}(x) = \begin{cases} 5/21, & x \in I_1 = [0,4) \\ 12/21, & x \in I_2 = [4,7) \\ 4/21, & x \in I_3 = [7,10] \\ 0, & \text{otherwise} \end{cases}$$

The graph of this function will be the scaled version of the frequency histogram.

For the grouped data case, one defines also the density histogram, which is an important characteristic for our data, as we will see soon:

Definition 2.7. *The* ***density histogram*** *or the* ***normalized relative frequency histogram*** *of our data* $x_1, ..., x_n$ *is the piecewise constant function*

$$h_{dens}(x) = \frac{n_j}{n} \cdot \frac{1}{\mathrm{length}(I_j)}, \qquad \forall x \in I_j.$$

Here $\mathrm{length}(I_j)$ *is the length of the interval* I_j*. Also we define* $h(x) = 0$*, if* $x \notin I$*.*

In other words,

$$h_{dens}(x) = \frac{\text{Number of datapoints in } I_j}{\text{Total number of elements}} \cdot \frac{1}{\mathrm{length}(I_j)}, \qquad \forall x \in I_j.$$

Example, Density Histogram for a Continuous or a Grouped Data: Again, for the above example, we will have

$$\mathrm{length}(I_1) = 4 - 0 = 4, \quad \mathrm{length}(I_2) = 7 - 4 = 3, \quad \mathrm{length}(I_3) = 10 - 7 = 3,$$

so

$$h_{dens}(x) = \begin{cases} \frac{5}{21} \cdot \frac{1}{4}, & x \in I_1 = [0,4) \\ \frac{12}{21} \cdot \frac{1}{3}, & x \in I_2 = [4,7) \\ \frac{4}{21} \cdot \frac{1}{3}, & x \in I_3 = [7,10] \\ 0, & \text{otherwise} \end{cases}$$

The graph of the density histogram function will NOT be, in general, the scaled version of the frequency histogram (unless the lengths of all class intervals are the same). See below for the code and graph.

R code, Density Histogram:

```
#Density Histogram
x <- c(1,3,4,2,5,6,7,5,3,4,3,5,6,4,5,6,5,7,6,7,8)
bins <- c(0,4,7,10)
hist(x, breaks = bins, right = F)
# right = F is for excluding the right-endpoints
```

The result is in Fig. 2.6.

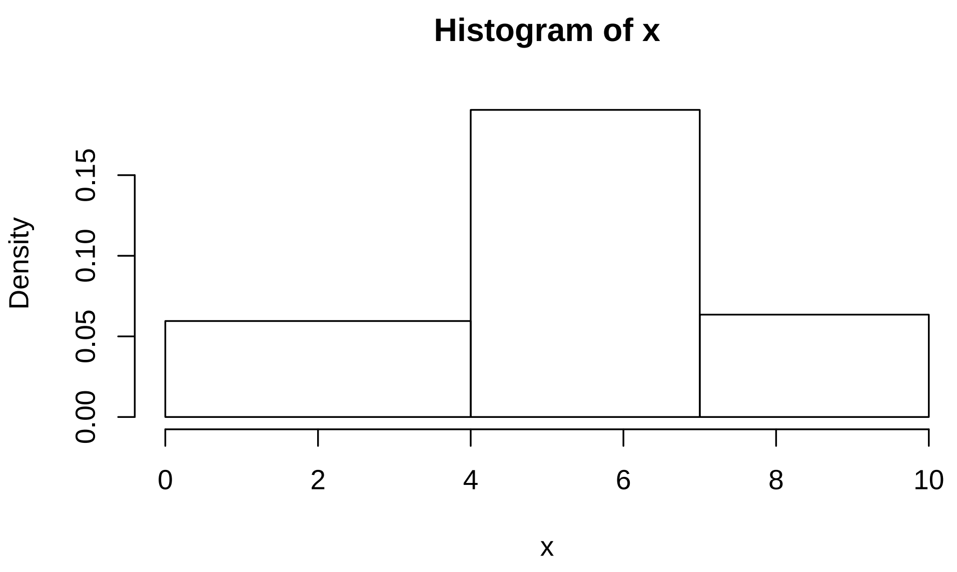


Fig. 2.6: Density Histogram

The idea of the normalization is that in this case we will have that the area under the graph of the density histogram is 1, i.e., the sum of areas of rectangles represented in the histogram is equal to 1. This makes the density histogram to be some kind of PDF for our data. In fact, when the data comes from some continuous distribution, then the density histogram is an approximation for the PDF of that distribution. We will talk about this again later.

R CODE, HISTOGRAMS: Here are some methods to draw histograms in **R**:

```
#Histogram, basic version
#R itself chooses the bins
x <- rexp(200, rate = 2) #Generates 200 (pseudo)random numbers from the Exp(2) distrib
hist(x) #Basic histogram

#Histogram, v2
bins=c(0,1,3,5) #creates a bin-endpoints vector
hist(x, breaks = bins) #Histogram with custom bins: our bins are [0,1], (1,3], (3,5]
hist(x, breaks = bins, right = F) #Histogram with custom bins:
                                  #our bins are [0,1), [1,3), [3,5]

#Histogram, v3
bins=seq(from=0,to=4, by=0.1) #creates a vector c(0, 0.1, 0.2, ..., 3.9, 4)
                              #equivalent to Matlab's bins = 0:0.1:4
hist(x, breaks = bins)        #Histogram with custom bins:
                              #our bins are [0,0.1], (0.1,0.2],... [3.9, 4]

#Caution: bins need to cover the range:
bins = c(0,1,2)
hist(x, breaks = bins) #This will sometimes give an error
#The point is that we will sometimes obtain numbers x outside the range [0,2]
#So take care to include all datapoints in bins.

#Histogram, v4
x <- rnorm(100) #generating a sample of 100 random numbers from the Standard Normal
hist(x) #Basic Histogram
bins=seq(from=min(x)-1,to=max(x)+1,by=0.8) #Bins with length 0.8,
   #covering the whole range of x's
hist(x, breaks = bins) #Histogram with our custom bins

#Histogram, v5
x <- c(1,2,3,1,2,4,5,3,6,2,3,4,1,5,2,2,3)
hist(x) #Please note that the y-axis shows the frequencies
#another example
x <- rnorm(1400)
hist(x)  #Here again, the y-axis shows the frequencies
hist(x, freq = TRUE)  #The freq parameter controls whether the frequencies
                      #will be displayed or the densities
                      #Here you will have the same result as without the freq parameter
hist(x, freq = FALSE) #This will give the Density histogram
hist(x, freq = FALSE, col = "green") #Guess what is doing this :)
```

Now, let us be back to our density histograms, and talk about the effect of normalization. Recall that any PDF $f(x)$ of a random variable X satisfies the following conditions:

- $f(x) \geqslant 0$, for any $x \in \mathbb{R}$ (in fact, a.e. in $\mathbb{R}$);
- $\displaystyle\int_{-\infty}^{+\infty} f(x)dx = 1$.

It is a notable (and important) fact that is that the sum of the areas of all rectangles (boxes) of the density histogram, i.e., the area under the density histogram, is 1. In fact:

Proposition 2.2. *If $h_{dens}(x)$ is the density histogram for the dataset $x_1, x_2, ..., x_n$, constructed using the bins $I_1, I_2, ..., I_m$, then*

- $h_{dens}(x) \geqslant 0$, *for all* $x \in \mathbb{R}$;
- $\displaystyle\int\limits_{-\infty}^{+\infty} h_{dens}(x)dx = 1$.

Proof. The first part is obvious, and the second part can be obtained in the following way:

$$\int\limits_{-\infty}^{+\infty} h_{dens}(x)dx = \int\limits_{I} h_{dens}(x)dx = \sum_j \int_{I_j} h_{dens}(x)dx = \sum_j \int_{I_j} \frac{n_j}{n}\frac{1}{|I_j|}dx = \sum_j \frac{n_j}{n}\frac{1}{|I_j|} \cdot |I_j| = \sum_j \frac{n_j}{n} = 1.$$

Also, you can get this result by calculating the sum of the areas of all rectangles representing the histogram. □

Now, if you have the density histogram for some dataset, then it is not too hard to find the relative frequencies of datapoint in each bin:

The relative frequency of datapoints in each class interval (bin) is the **area** of the rectangle over that interval

This is very easy to see, since if we are considering the class interval I_j, then the relative frequency of datapoints lying in I_j is $\frac{n_j}{n}$. On the other hand,

$$\text{Area of the rectangle over the interval } I_j = \text{height}(I_j) \cdot \text{width}(I_j) =$$

$$= \frac{n_j}{h}\frac{1}{\text{length}(I_j)} \cdot \text{length}(I_j) = \frac{n_j}{n}.$$

Example, Reading relative frequencies from the Histogram: Fig. 2.7 shows the density histogram for some dataset (in fact, this is the **R** dataset *state.x77*, the *Murder* column, see the dataset and its documentation). The relative frequency of datapoints in $[10, 12)$ is the area of the rectangle over that interval, approximately, $0.12 \cdot (12 - 10) = 0.24$. In other words, approximately 24% of our data is in $[10, 12)$. Below please find the code drawing the histogram and checking that exactly 24% of all datapoints are in $[10, 12)$.

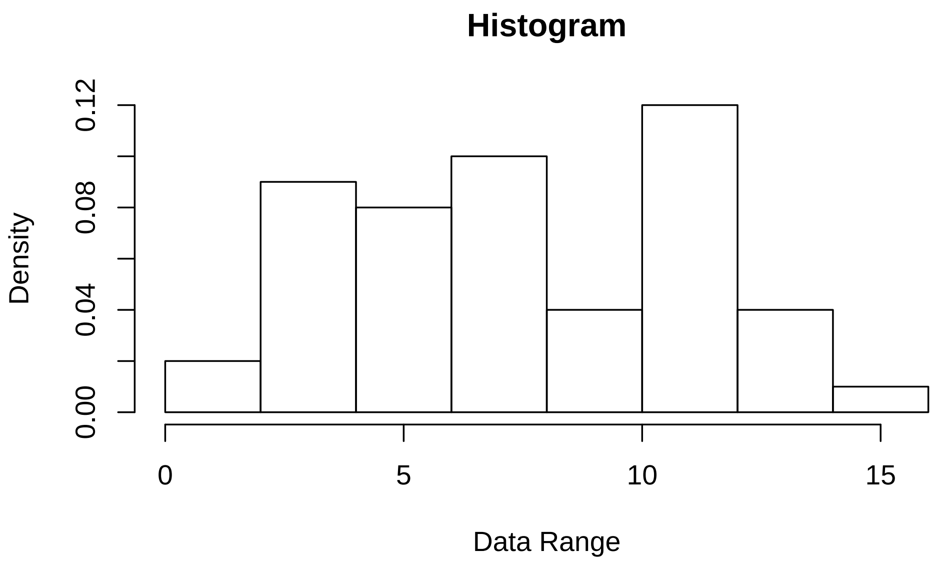


Fig. 2.7: Density Histogram for some Data

R code, Reading relative frequencies from the Histogram:

```
#Reading relative frequencies from the Histogram
x <- data.frame(state.x77) #Transforming the dataset into a data frame
y <- x$Murder # y will be the column (feature) Murder
hist(y, freq = FALSE, main = "Histogram", xlab = "Data Range")
n <- length(y) # total number of datapoints
n.j <- length(y[(y>=10)&(y<12)]) #the number of datapoints in [10,12)
rel.freq <- n.j/n # the ratio of n.j and n, the relative frequency
rel.freq
```

The result is: 0.24

Remark, Histogram and PDF, similarities: Assume we have a dataset $x_1, ..., x_n$, and $h_{dens}(x)$ is its density histogram, constructed using the class intervals I_j, $j = 1, ..., m$. As we have stressed above, h_{dens} is some sort of PDF for our data. And, in some sense, it has many properties of the PDF. Besides the above two properties, we can get the following: recall that if $f(x)$ is the PDF of the r.v. X, then

$$\mathbb{P}(X \in [a, b]) = \int_{[a,b]} f(x)dx.$$

For the density histogram, the above property of calculation of the relative frequencies can be written as:

$$\mathbb{P}(x_k \in I_j) = \int_{I_j} h_{dens}(x)dx = \frac{n_j}{n}.$$

The meaning of this is that if we will make a new r.v. taking the values $x_1, x_2, ..., x_n$, with uniform probabilities $\frac{1}{n}$, then the probability that this new r.v. will be inside I_j is the integral of h_{dens} over that interval. Or, put in other way, if we are picking at random (uniformly!) one of the points x_k, then the probability that we will get a number inside I_j is $\int_{I_j} h_{dens}(x)dx =$ the relative frequency of our datapoints in I_j.

In some sense, the density histogram gives us an approximation and estimation of the PDF behind the data. The idea is that if we will assume that our data comes from some probability distribution, then, under some mild conditions, the density histogram will approximate the PDF of that distribution.

R code, Density Histogram: Here is the example of PDF estimation by a histogram: we generate 1000 random samples from $\text{Exp}(1)$ distribution, and compare the normalized histogram with its PDF.

```
#Histogram approximation of the PDF
#First we plot the graph of Exp(1) distribution PDF
plot(dexp, lwd = 3, col = 'red', xlim = c(0,3), ylim = c(0,1))
#Now we generate a sample of size 1000 from that distribution
x <- rexp(1000)
par(new = TRUE) #This is to keep the previous plot, and to draw over that plot
hist(x, breaks = seq(-3,10,0.2), freq = FALSE, xlim = c(0,3), ylim = c(0,1))
```

The obtained picture is given in Fig. 2.8. By the way, every time you will run this code, it will generate new sample of size 1000, so the picture will not be the same. You can try to run this part many times to see that, in general, Density histogram gives an approximation for the PDF. Also, you can try to increase the number of random numbers, and to decrease the bin lengths.

Please note also that in the Fig. 2.8 the y-labels overlap. you can correct this by giving, say, the same *ylab = 'Density'* parameter in the *plot* and *hist* commands. Try that by yourself. You can try also to run without *xlim* or *ylim* parameters to see why I am using them.

It turns out that to have a good representation for a histogram, one needs to choose the class intervals wisely[7]. The next example is an example of small bins.

R code, Small Bin Sizes:

```
#Histogram with small bins
x <- 0.3*rnorm(300)+0.7*runif(300, -1,1)
hist(x, breaks = seq(min(x)-1,max(x)+1, by = 0.01))
```

See the result in Fig. 2.9.

In fact, if you have a lot of data, then using small bins is something like to consider all your datapoints, one by one. Something like to say: "I have 1 datapoint with the value 0.002, 1 datapoint with the value 0.0019, another one with 0.0022, etc...". The idea of histogram is to group wisely your

[7]Experiment with http://www.shodor.org/interactivate/activities/Histogram/

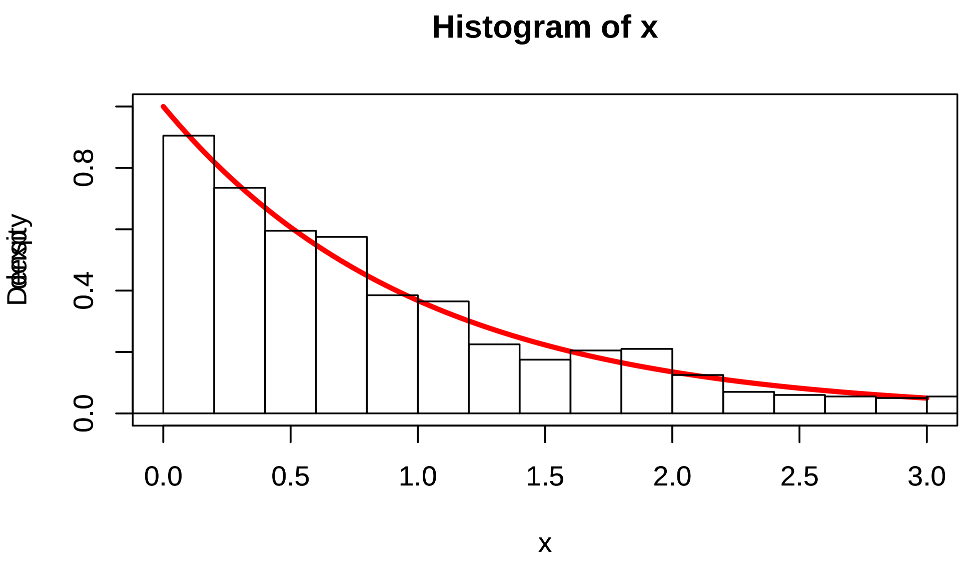


Fig. 2.8: Density Histogram approximation of PDF

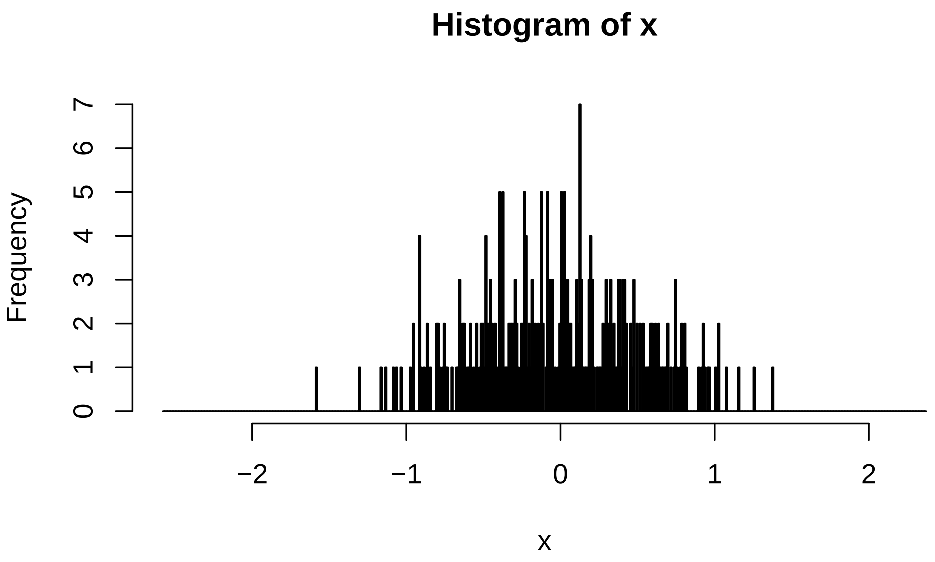


Fig. 2.9: Histogram with small bins

datapoints to keep all important features and information. So taking bin widths small enough will keep the structure of your dataset, and give a lot of information about your dataset, but choosing very small bins will be an overdose[8] ⌣̈. Usually, one takes the lengths of all class intervals to be the same number h, and in that case h is called the bin width. You can consult the literature or Wiki page `https://en.wikipedia.org/wiki/Histogram` for the choice of bin width h.

[8]An anecdote comes to my mind: students ask their professor: what is the best length for a final project paper? Professor replies - your papers need to be like bikinis: long enough to cover the subject, but short enough to be interesting ⌣̈

So what can be seen using histograms? A lot of things. Among them, histograms can be used to determine if the data:

- is symmetric about some point or is skewed to left or right
- is spread out or concentrated at some point
- has some gaps
- has values far apart from others, has outliers (anomalies)
- is unimodal, bimodal or multimodal[9]

Also, if using the frequency histogram, one can also see the most frequent values range.

R code, Histogram for Precipitation Data:

```
#precip is a standard dataset in R, showing the average amount of precipitation
#(rainfall) in inches for each of 70 United States (and Puerto Rico) cities.
hist(precip, breaks = seq(0,70,5), col = "blue")
```

The result is represented in Fig. 2.10. You can see two gaps - between 0 and 5 and 60 and 65 (so there was no city with average precipitation between 60 and 65, say). The most frequent average precipitation is between 35 and 40 (we have 13 cities having this much average precipitations). The data is 3-modal (?), most of the values are between 30 and 50, and also that one of the cities was unlucky with a flood ⌣̈

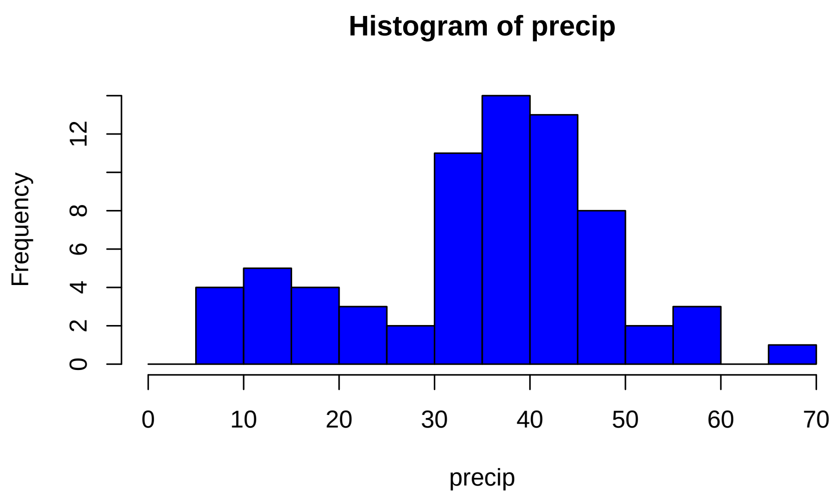


Fig. 2.10: Precipitation data histogram

[9]Well, you can ask - why we need to know if our data is unimodal or bimodal etc. The idea is that later we will try to fit a distribution to a data: say, we have a data, and we want to find a model, theoretical distribution best describing the data. If the data is, say, bimodal, then it is not a clever idea to try to use the Normal distribution (because Normal Distribution is unimodal). What can be done for, say, bimodal dataset? - We can try, for example, the mixture of 2 Gaussian (i.e., Normal) distributions - some weighted sum of two Normal distributions.

Remark, Histogram vs Barplot: Please note that Barplot and Density Histogram are different things. Barplot is not grouping the data, it just makes a rectangle over the datapoint with the height, which is the frequency (or the relative frequency) of the value. Also, the widths of that rectangles are not important. But this is not the case for Histograms.

2.4 Supplementary: Histograms in Image Analysis

One of the usage of Histograms is in Digital Photography and Image Analysis - the Image Histogram. To make the Histogram for a, say, grayscale image, we take as a variable the level of gray (tone) in the image, ranging from 0 to 255 (0-black, 255-white), and the frequency of the value $x \in \{0, 1, 2, ..., 255\}$ shows how many pixels in our image have the gray level (tone) x. You can find histograms in Digital Cameras, and in software like Adobe Photoshop, IrfanView etc.

Fig. 2.14 gives an example of image histogram. This image is taken from the book **Gonzalez, R.C. and Woods, R.E.**, *Digital Image Processing*, 2008, Pearson/Prentice Hall. See Section 3.3 of that book for Histogram Processing methods. You can also read about the Image Histogram and about the Exposure and Contrast at http://www.cambridgeincolour.com/tutorials/histograms1.htm.

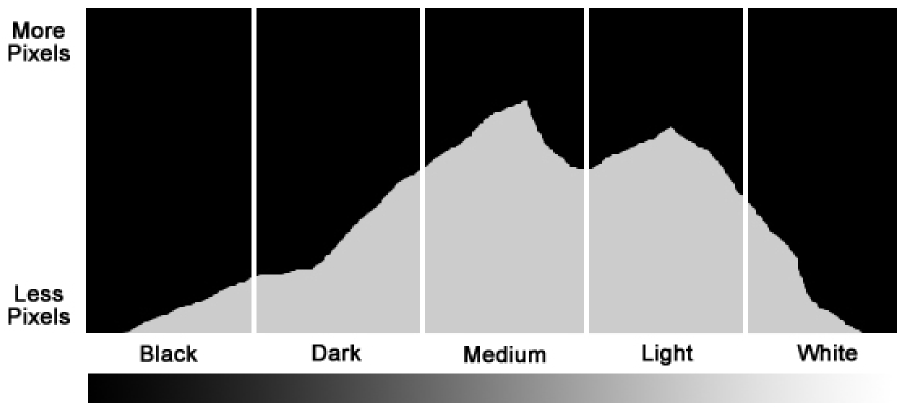


Fig. 2.11: Image Histograms, Explanation: Very few Black areas; An increase in Dark areas; Much more Medium areas; Around the same amount of Light areas; Very few White areas. Source: http://seeinginmacro.com/exposure-histogram-in-photography/

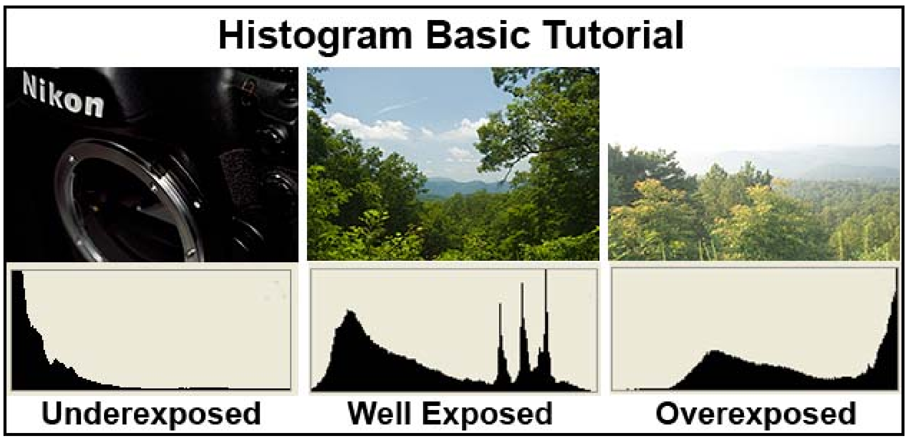


Fig. 2.12: Image Histograms, Explanation

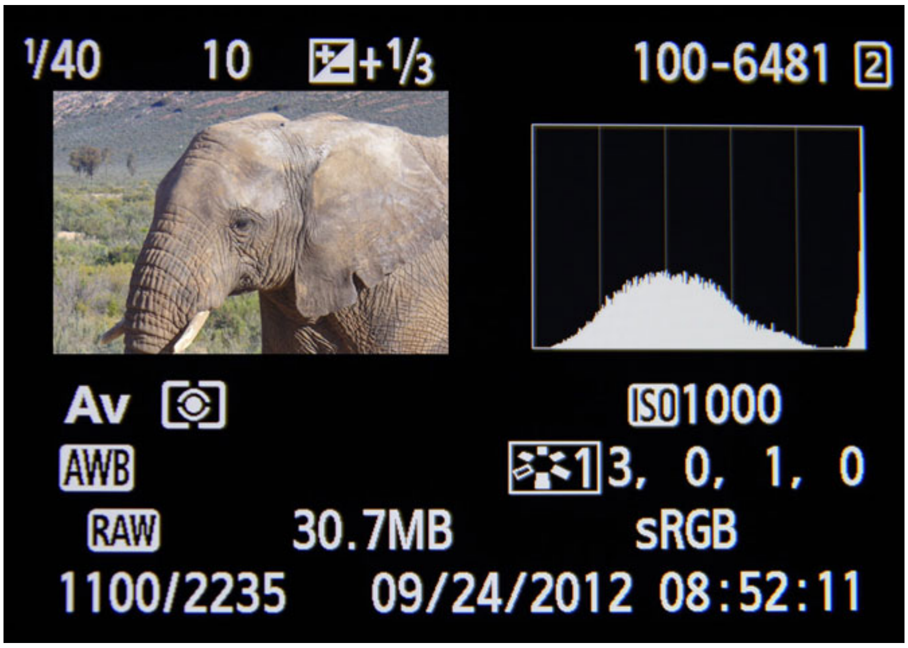


Fig. 2.13: Camera Image Histogram

2.5 Stem-and-Leaf Plots

Stem-and-Leaf plot is another visual representation for a univariate numerical dataset. In fact, it can be viewed as some kind of inverted (rotated) histogram for a small or moderate size dataset.

Assume we have a numerical dataset $x_1, x_2, ..., x_n$, and n is not too large. Drawing the Stem-and-Leaf plot starts by dividing our numbers x_k into Stems and Leafs, and then we will represent each number x_k in the form

$$\text{Stem} \,|\, \text{Leaf}$$

The *Leaf* needs to be a single digit, and *Stem* needs to be an integer (with not too many digits). And each Stem-and-Leaf plots need to contain a key explaining how the datapoints are divided into Stems and Leafs, or, which is the same, how to "recover" the dataset using the Stem-and-Leaf plot. I am using "recover" in quotes, since sometimes we do some rounding before plotting the Stem-and-Leaf plot, so exact recovery will not be possible in that case.

Now, let me explain how to construct Stem-and-Leaf plots, and why the Leaf needs to be a single digit.

Example, Stem-and-Leaf plot: Assume we have the following dataset:

$$20, 21, 34, 21, 45, 20, 23, 21, 32, 33, 33, 31, 20, 21, 23$$

Our dataset consists of 2-digit integers, so it is natural to take last digits as Leafs, and first digits as Stems. Say, the number 20 will be represented as

$$2\,|\,0$$

The key here will be that $2\,|\,0$ will be the number 20, or the Leaf shows the units digit, and the Stem shows the decimal digit.

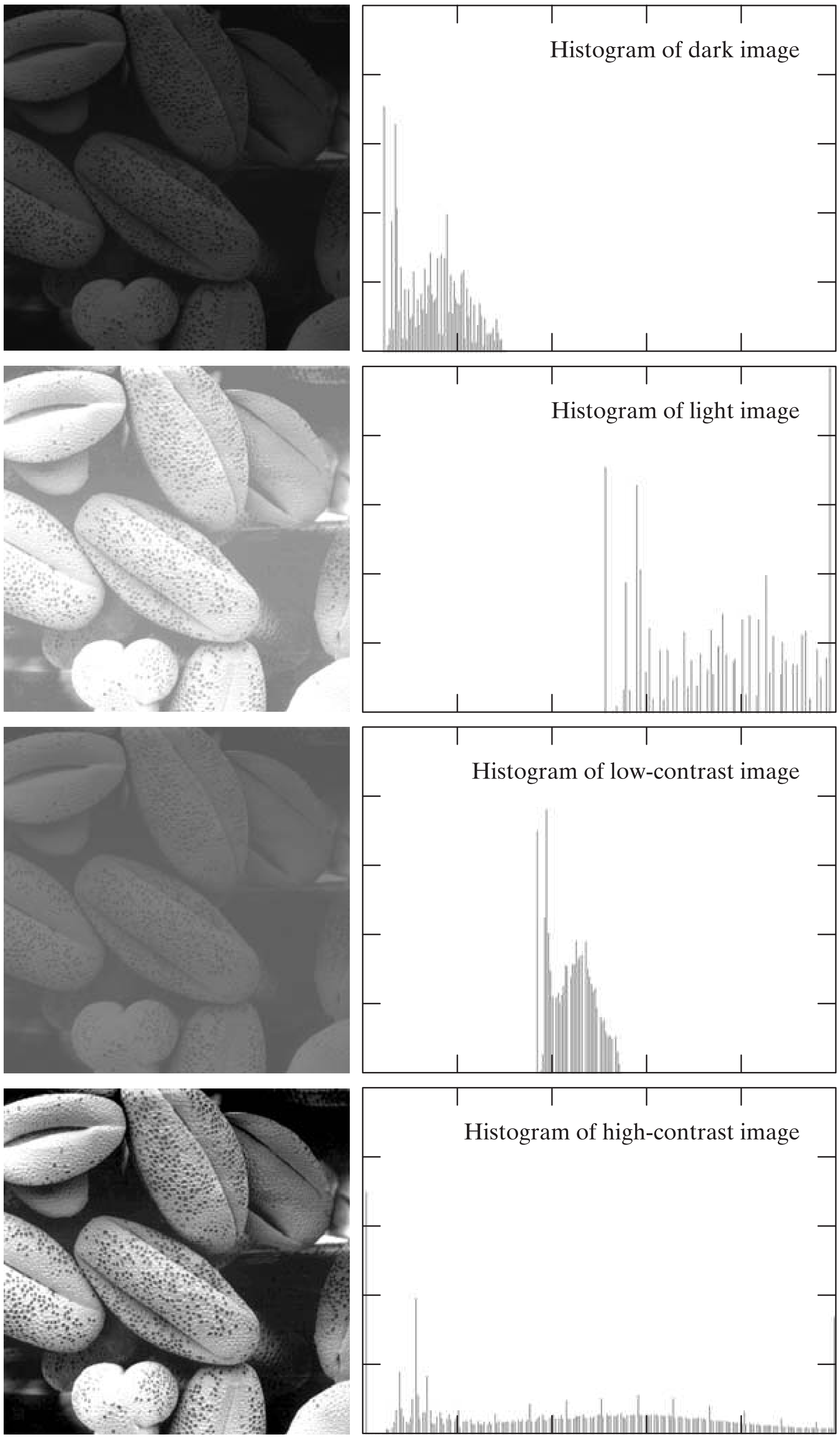


FIGURE 3.16 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms.

Fig. 2.14: Image Histogram, From Gonzalez, Woods, Digital Image Processing

Now, we need some grouping of data. For the Stems we choose, in some sense, bins: say, all datapoints from the interval $[20, 30)$ will be displayed in one row, all datapoints from the interval $[30, 40)$ will be on the next row etc. Say, the next datapoint 21 will be represented as

$$2|1$$

and we represent the pair $20, 21$ in the following way:

$$2|01$$

We will read this "back" as 20 and 21 - this is why we need to choose Leafs to have only one digit. So we have 1 stem here - 2, and we have 2 Leafs here - 0 and 1, so we recover 20 and 21.

Now, say, the next number is 34. So the Stem is 3 and the Leaf for this number is 4. Because our number is not in the first "bin", is not in $[20, 30)$, we will write it on the next row. So the Stem-and-Leaf plot for 20, 21 and 34 will be

```
2 | 01
3 | 4
```

Now, hope things are clear. To obtain the Stem-and-Leaf plot for our dataset, we first sort it:

$$20, 20, 20, 21, 21, 21, 21, 23, 23, 31, 32, 33, 33, 34, 45$$

Now, the Stem-and-Leaf plot will be:

```
2 | 000111133
3 | 12334
4 | 5
```

and the Key is that $2|0$ is 20. **R** is giving the Key in other way, see the code below.

R code, Stem-and-Leaf plot: Let us construct the Stem-and-Leaf plot for the above dataset using **R**:

```
#Stem-and-Leaf Plot
x <- c(20, 21, 34, 21, 45, 20, 23, 21, 32, 33, 33, 31, 20, 21, 23)
stem(x)
```

The result will be:

```
  The decimal point is 1 digit(s) to the right of the |

  2 | 000111133
  3 | 12334
  4 | 5
```

So **R** is giving the place of the decimal point - it is 1 digit to the right of |, i.e., for 2|0 we will have "20." .

Now, another example:

Example, Stem-and-Leaf plot: Say, we want to give the Stem-and-Leaf plot for the dataset

$$3.172, 3.145, 3.442, 3.124, 3.456, 2.312, 2.443, 2.111, 2$$

First we sort our dataset:

$$2, 2.111, 2.312, 2.443, 3.124, 3.145, 3.172, 3.442, 3.456$$

The we need to choose Stems and Leafs. Say, we can choose the Leaf to be the last digit, and all other digits will be in the Stem. For example, the Stem-and-Leaf plot representation for 3.172 will be

$$317|2$$

with the key that this is our number 3.172, or, in the **R** format, we need to say that the decimal point is 2 units left to |. But this is not a good choice for Leafs and Stems. Usually, one first rounds the numbers (well, if it is important not to do a rounding, if it is important to keep the original numbers, choose Stems and Leafs as above): say, we can round to

$$2, 2.1, 2.3, 2.4, 3.1, 3.1, 3.2, 3.4, 3.5$$

Now, it is very natural to choose Leafs the digits after the decimal point, and Stems will be the numbers before the decimal point. Say, for the datapoint 2 we will have $2|0$ (since $2 = 2.0$), and for the datapoint 2.1 we will have $2|1$. Next, we need to choose groupings or bins: say, naturally, we can choose to represent in the same row the number in $[2, 3)$ and in $[3, 4)$: in this case our dataset Stem-and-Leaf plot will be:

```
2 | 0134
3 | 11245
```

and the Key is that $2|0$ is 2.0.

If we will choose another bins, say, we will group in a row datapoints in $[2, 2.5)$, $[2.5, 3)$, $[3, 3.5)$ and $[3.5, 4)$, then we will have the following Stem-and-Leaf plot for our dataset:

```
2 | 0134
2 |
3 | 1124
3 | 5
```

and again the Key is that $2|0$ is 2.0. Below please find the **R** code for this example - **R** will do this last grouping.

R code, Stem-and-Leaf plot: Let us construct the Stem-and-Leaf plot for the above dataset using **R**:

```
#Stem-and-Leaf Plot
x <- c(3.172, 3.145, 3.442, 3.124, 3.456, 2.312, 2.443, 2.111, 2)
stem(x)
```

The result will be:

```
  The decimal point is at the |
  2 | 0134
  2 |
  3 | 1124
  3 | 5
```

So **R** is giving the place of the decimal point - it is at the |, i.e., for 2 | 0 we will have "2.0".

R CODE, STEM-AND-LEAF PLOT: You can control the Stem-and-Leaf Plot (choice of grouping "bins") in **R**. Please run and compare the following code results, for the above example:

```
#Stem-and-Leaf Plot, different scale parameters
x <- c(3.172, 3.145, 3.442, 3.124, 3.456, 2.312, 2.443, 2.111, 2)
stem(x)
stem(x, scale = 1)
stem(x, scale = 2)
stem(x, scale = 0.5)
```

Now, lets do the inverse thing: read the data from the Stem-and-Leaf plot:

EXAMPLE, READING STEM-AND-LEAF PLOT: Consider the following Stem-and-Leaf plot obtained in **R**:

```
  The decimal point is 1 digit(s) to the right of the |

   0 | 24004678
   2 | 002466668822244466
   4 | 002668024466
   6 | 046806
   8 | 04523
  10 |
  12 | 0
```

The Key says how to read the data: say, $0|2$ means $02 = 2$, and $6|4$ means 64. So we can recover our dataset (if no roundings were made): the first elements are 02=2 and 04=4, but the next 0 is confusing - it is not in the increasing order. But the point is that we need to take into account our bins - here **R** is grouping with class intervals (look at the stems!) $[0, 20)$, $[20, 40)$, $[40, 60)$,..., $[120, 140)$. So in the first row, after 2,4, the third element 0 means 10 (!!). Then again 10, then 14, 16, etc. So the first row of our Stem-and-Leaf plot can be decoded as

$$2, 4, 10, 10, 14, 16, 17, 18.$$

The second row ten will be decoded as

$$20, 20, 22, 24, 26, 26, 26, 26, 28, 28, 32, 32, 32, 34, 34, 34, 36, 36$$

The row with Stem 8 is obtained from the datapoints

$$80, 84, 85, 92, 93$$

and the last datapoint in our dataset is 120 (think: why not 130? ◡̈). The above Stem-and-Leaf plot was obtained using the code give below.

Uff!

R code, The above example code: To obtain the above Stem-and-Leaf plot, I have used the *cars* dataset form **R**:

```
#Stem-and-Leaf plot example
cars  #This will show the standard loaded dataset cars
x <- cars$dist # we take the dist column of cars dataset
x #this will show the vector x
sort(x) #sort the dataset, so it will be easy to compare with the Stem-and-Leaf plot
stem(x) #Stem-and-Leaf plot with default scale 1
```

Voila!

Next, you can try, as a continuation of the same example, to run

```
stem(x, scale = 2)
```

R code, Stem-and-Leaf plot vs Histogram: Let us get the Stem-and-Leaf plot and Histogram for the same dataset *faithful*, which is one of the standard embedded datasets in **R** (it shows the waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA.)

```
#Stem-and-Leaf plot vs. histogram
faithful #faithful dataset of R
duration = faithful$eruptions #the eruptions column data of the dataset
stem(duration)
hist(duration, breaks = seq(from = 1.4,to = 5.8, by =0.2))
#stem(duration, scale = 2)  #you can try this also
```

The results are given in Fig. 2.16-2.15. Note the shape similarity (well, if you will rotate one of the graphs ◡̈)!!

Remark, Difference between SaL and Hist: One advantage of Stem-and-Leaf plot against the histogram is that it shows the original dataset (after, maybe, some rounding), shows all the datapoints (so in case of Stem-and-Leaf plot you can "recover" all your datapoints, but not for the histogram). The disadvantage is that Stem-and-Leaf plot will not work for a large dataset, and, also, you need to calculate by hand, if necessary, the frequencies, relative frequencies etc, in this case.

2.6 Supplementary: Kernel Density Estimation

As we have seen above, we can get an information about the density of our dataset using the density histogram or Stem-and-Leaf plots. Stem-and-Leaf plot works only for small size datasets, and one

```
The decimal point is 1 digit(s) to the left of the |

16 | 070355555588
18 | 000022233333335577777777888822335777888
20 | 00002223378800035778
22 | 0002335578023578
24 | 00228
26 | 23
28 | 080
30 | 7
32 | 2337
34 | 250077
36 | 0000823577
38 | 2333335582225577
40 | 0000003357788888002233555577778
42 | 03335555778800233333555577778
44 | 02222335557780000000023333357778888
46 | 0000233357700000023578
48 | 00000022335800333
50 | 0370
```

Fig. 2.15: Stem-and-Leaf plot

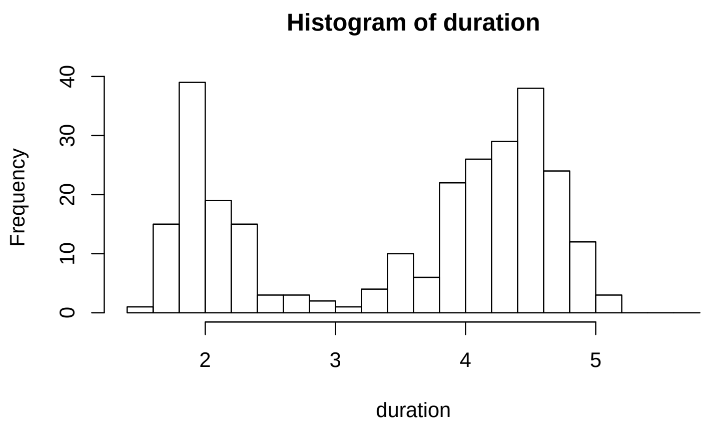


Fig. 2.16: Histogram

of the drawbacks of the histogram is that it is not smooth - it is a piecewise constant (stepwise) function. Another issue with histograms is that it depends on the choice of the class intervals. Usually, one fixes a step-size h and takes class intervals to be adjacent intervals of width h, starting from some point a, i.e., intervals of the form $[a, a+h]$, $(a+h, a+2h]$, $(a+2h, a+3h]$, ... But in this case the form of the histogram will depend on a and h. If, say, we will fix h, then by changing a, we will obtain different histograms. This is why (and not only ☺) people try to construct a smoothed version of the histogram. Kernel Density Estimate[10] (KDE) is some kind of smoothed histogram.

To explain the idea of KDE, and where it comes from, first let me give another type of histogram. Let $x_1, x_2, ..., x_n$ be our datapoints. First, I want to give our ordinary Density histogram, but for equal-length bins: choose an origin a (this can be, for example, $\min\{x_k\}$), choose a bin size $h > 0$, and take

$$I_1 = [a, a+h], \quad I_2 = (a+h, a+2h], \quad I_3 = (a+2h, a+3h], ...$$

as our class intervals (bins). Then

$$h_{dens}(x) = \frac{\text{Number of datapoints in the same bin as } x}{\text{Total number of points}} \cdot \frac{1}{length(I_j)} =$$

$$= \frac{\text{Number of datapoints in the same bin as } x}{nh}.$$

Now, this definition depends on the choice of a, the starting point. Another way to define a histogram is the following:

[10]It is called an Estimate, because it is an estimate for the PDF behind our data.

Definition 2.8. *The **Naive Estimator** of the PDF is the following function*[11]

$$h_{naive}(x) = \frac{\textit{Number of datapoints in } \left(x - \frac{h}{2}, x + \frac{h}{2}\right]}{nh}.$$

This will again produce a piecewise constant function.

EXAMPLE, NAIVE PDF ESTIMATOR: Here we need to have an example. Write that by yourself!

Next, let us rewrite the definition of the Naive Estimator using the following function (called a rectangular kernel with bandwidth 1):

$$k(t) = \begin{cases} 1, & \text{if } t \in \left(-\frac{1}{2}, \frac{1}{2}\right] \\ 0, & \text{otherwise.} \end{cases}$$

Then[12]

$$h_{naive}(x) = \frac{1}{nh} \cdot \sum_{i=1}^{n} k\left(\frac{x - x_i}{h}\right).$$

Now, to obtain a smooth version of this, one chooses a smooth Kernel function $K(t)$, here, a function with

$$K(t) \geqslant 0, t \in \mathbb{R}, \qquad \text{and} \qquad \int_{-\infty}^{+\infty} K(t)dt = 1.$$

For example, we can take the Gaussian Kernel

$$K(t) = \frac{1}{\sqrt{2\pi}} \cdot e^{-t^2/2}, \qquad t \in \mathbb{R},$$

or any other PDF.

Next, one defines the Kernel Density Estimator with Kernel K as

$$KDE_K(x) = KDE(x) = \frac{1}{nh} \cdot \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right).$$

It is easy to see that $KDE(x)$ will give a PDF, i.e., will be nonnegative and will integrate to 1:

$$\int_{-\infty}^{+\infty} KDE(x)dx = \frac{1}{nh} \cdot \sum_{i=1}^{n} \int_{-\infty}^{+\infty} K\left(\frac{x - x_i}{h}\right) dx = \frac{1}{n} \cdot \sum_{i=1}^{n} \int_{-\infty}^{+\infty} K\left(\frac{x - x_i}{h}\right) d\frac{x - x_i}{h} \overset{u = \frac{x - x_i}{h}}{=\!=\!=\!=}$$

[11] The idea of this comes from the following consideration: Assume that our data comes from a realizations of a random variable X, with PDF $f(x)$. We want to estimate f, having that realizations $x_1, ..., x_n$. Now, we have

$$\mathbb{P}\left(x - \frac{h}{2} < X < x + \frac{h}{2}\right) = \int_{x-\frac{h}{2}}^{x+\frac{h}{2}} f(t)dt \approx f(x) \cdot h.$$

Hence,

$$f(x) \approx \frac{\mathbb{P}\left(x - \frac{h}{2} < X < x + \frac{h}{2}\right)}{h},$$

and also

$$\mathbb{P}\left(x - \frac{h}{2} < X < x + \frac{h}{2}\right) \approx \text{The relative frequency of our data in } \left(x - \frac{h}{2}, x + \frac{h}{2}\right] = \frac{\text{Number of datapoints in } \left(x - \frac{h}{2}, x + \frac{h}{2}\right]}{\text{Total Number of datapoints}}.$$

[12] Check this!

$$= \frac{1}{n} \cdot \sum_{i=1}^{n} \int_{-\infty}^{+\infty} K(u)du = \frac{1}{n} \cdot \sum_{i=1}^{n} 1 = 1.$$

Like in the case of the Density histogram, where that histogram was depending on the bins choice, the KDE depends on the choice of $h > 0$. h is called the **bandwidth**, and its estimation is another story.

R CODE, KDE: To construct the KDE for a dataset, one can use **R** command *density*:

```
#Kernel Density Estimator
x <- c(1,1,1,2,2,2,2,3,3,3,3,3,3,3,3,3,4,4,4,5,6,7,8,9,10)
d <- density(x)
plot(d, lwd = 2)
```

The result is given in Fig. 2.17 (and the default kernel **R** is using is Gaussian, you can change that by giving the parameter *kernel*). Note that we have a lot of 3-s in our dataset, so KDE is giving high value close to 3.

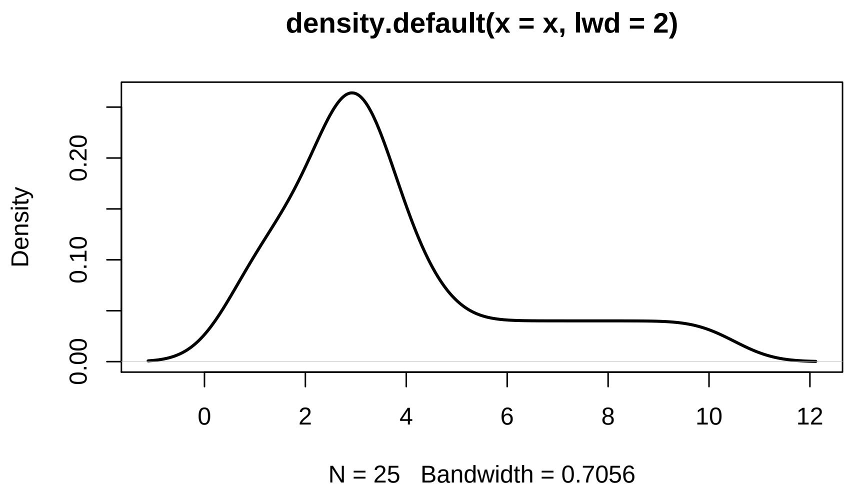


Fig. 2.17: Kernel Density Estimator

R CODE, DENSITY HISTOGRAM VS KDE: Now, lets us draw a Density Histogram vs KDE:

```
#Kernel Density Estimator vs Density Histogram
x <- c(1,1,1,2,2,2,2,3,3,3,3,3,3,3,3,3,4,4,4,5,6,7,8,9,10)
d <- density(x)
hist(x, freq = FALSE, xlim = c(-1,11), ylim = c(0,0.3), col = "blue", main = "")
par(new = TRUE)
plot(d, lwd = 3, col = "red", xlim = c(-1,11), ylim = c(0,0.3), main = "")
```

The result is given in Fig. 2.18.

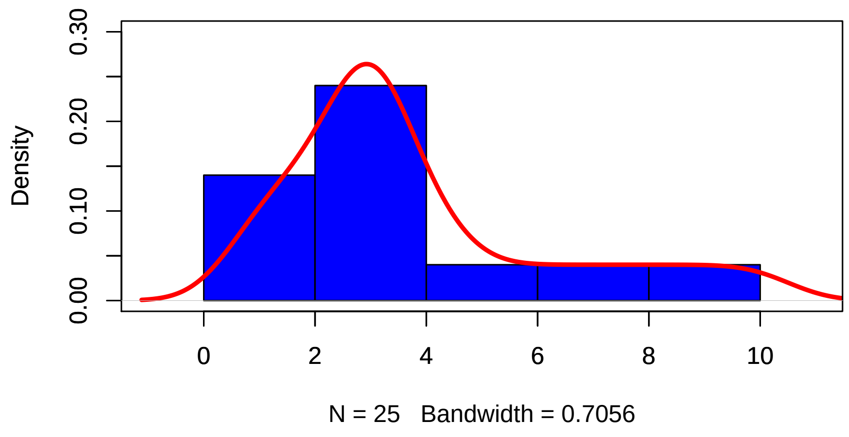


Fig. 2.18: Density Histogram vs KDE

2.7 Empirical Distribution Function

The density histogram and KDE were to estimate the probability density, the PDF of the distribution behind the data. Now we want to estimate the CDF driving the data. And even if we do not have some distribution behind our data, we can construct so kind of CDF function, which will give us another graphical representation for our dataset.

Assume our dastaset consists of points $x_1, x_2, ..., x_n$.

Definition 2.9. *The* ***Empirical Distribution Function, ECDF*** *or the* ***Cumulative Histogram*** $\mathtt{ecdf}(x)$ *of our data* $x_1, ..., x_n$ *is defined by*

$$\mathtt{ecdf}(x) = \frac{\textit{number of elements in our dataset} \leqslant x}{\textit{the total number of elements in our dataset}} =$$
$$= \frac{\textit{number of elements in our dataset} \leqslant x}{n} = \frac{\sum_{i=1}^{n} \mathbb{1}(x_i \leqslant x)}{n}, \qquad \forall x \in \mathbb{R}.$$

Example, ECDF: Assume our dataset is:

$$-1, 3, 2, 1, 3, 1, 0.$$

In order to construct the ECDF, first we rearrange these numbers in the increasing order (this will simplify our calculations):

$$-1, 0, 1, 1, 2, 3, 3.$$

Here, n, the total number of elements in our dataset is $n = 7$. Let us calculate first, say, $\mathtt{ecdf}(-1.3)$:

$$\mathtt{ecdf}(-1.3) = \frac{\text{number of elements} \leqslant -1.3}{n} = \frac{0}{7} = 0.$$

In fact, if $x < -1$, then $\mathtt{ecdf}(x) = 0$, since no element in our dataset is less than x. Now, if $x = -1$, then

$$\mathtt{ecdf}(-1) = \frac{\text{number of elements} \leqslant -1}{n} = \frac{1}{7},$$

since only one element in our dataset is $\leqslant -1$. Similarly, for any $x \in [-1, 0)$, we will have $\mathtt{ecdf}(x) = \frac{1}{7}$. In the same vain we can obtain the value of our ECDF at any point:

$$\mathtt{ecdf}(x) = \begin{cases} 0, & \text{if } x < -1 \\ \frac{1}{7}, & \text{if } -1 \leqslant x < 0 \\ \frac{2}{7}, & \text{if } 0 \leqslant x < 1 \\ \frac{4}{7}, & \text{if } 1 \leqslant x < 2 \\ \frac{5}{7}, & \text{if } 2 \leqslant x < 3 \\ 1, & \text{if } x \geqslant 3 \end{cases}$$

Remark, ECDF and Discrete r.v. CDF: In some sense, the ECDF is the CDF of the r.v. X which takes the value x_i with the probability $\frac{1}{n}$ (if two or more values x_i coincide, then one needs to count every occurrence of x_i). In other way, we make a r.v. X with values x_i (taking only distinct values x_i), and with probabilities

$$\mathbb{P}(X = x_i) = \text{the relative frequency of } x_i.$$

Say, for the example above, for the dataset

$$-1, 3, 2, 1, 3, 1, 0$$

we define the following r.v. X:

Values of X	-1	0	1	2	3
$\mathbb{P}(X = x)$	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{2}{7}$	$\frac{1}{7}$	$\frac{2}{7}$

since we have $n = 7$ datapoints, and -1, 0 and 2 appear only once, and 1 and 3 appear twice in our dataset. Now, the Empirical CDF of our dataset $\mathtt{ecdf}(x)$ will coincide with the CDF $F_X(x)$ of X:

$$\mathtt{ecdf}(x) = F_X(x), \qquad x \in \mathbb{R}.$$

Example, ECDF and Discrete r.v. CDF: The Empirical CDF for the dataset $1, 5, 5, 9$ is the same as the CDF for the r.v. X with values $1, 5, 9$ and with probabilities

$$\mathbb{P}(X = 1) = \frac{1}{4}, \qquad \mathbb{P}(X = 5) = \frac{2}{4}, \qquad \mathbb{P}(X = 9) = \frac{1}{4}.$$

i.e., for the r.v. X with the PMF

Values of X	1	5	9
$\mathbb{P}(X = x)$	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{1}{4}$

R code, ECDF: The R Code to graph the ECDF for the data above is

```
#Empirical CDF
x <- c(-1, 3 , 2, 1, 3, 1, 0)
f<-ecdf(x)
plot(f)
```

The result is given in Fig. 2.19.

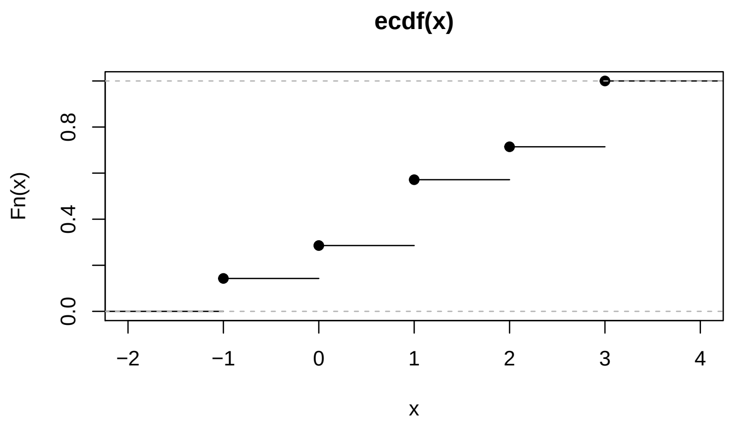


Fig. 2.19: Empirical CDF

The remarkable property of ECDF $\mathrm{ecdf}(x)$ is that if our dataset $x_1, ..., x_n$ is a sample obtained from a r.v. with the CDF $F(x)$, then $\mathrm{ecdf}(x)$ is a good estimate (approximation) for $F(x)$, and, in fact, $\mathrm{ecdf}(x)$ approaches to $F(x)$[13]. Below I am giving one example, and you can try to run that for different distributions, and for different (increasing) number of sample points:

R code, ECDF approximation of the Theoretical CDF:

```
#Empirical CDF approximation of Theoretical CDF
#First we plot the CDF of the Standard Normal Distribution
plot(pnorm, lwd = 3, col = 'red', xlim = c(-3,3), ylim = c(0,1), ylab = "ecdf and CDF")
x <- rnorm(30) #Now we are taking a sample of size 30 from N(0,1)
f<-ecdf(x) #f will be the ECDF of our data x
par(new = TRUE) #this is to keep the previous graph and to draw over it
plot(f, xlim = c(-3,3), ylim = c(0,1), ylab = "ecdf and CDF")
```

The result is presented in Fig. 2.20. Try to rerun several times (since every time **R** will generate a new sample), increase the number of sample points, and try for different distributions!

Later we will talk about BoxPlots, another graphical Representation (Description) of our data.

[13]This is the famous **Glivenko-Cantelli Theorem**.

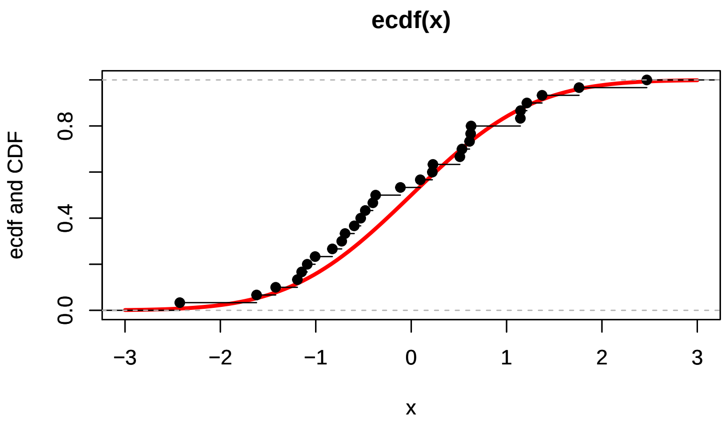


Fig. 2.20: Empirical CDF approximation of the $\mathcal{N}(0,1)$ CDF