Descriptive Statistics using R

Pramudita Satria Palar, Ph.D.

15/8/2022

1 Introduction

Introductory statistics usually begins with descriptive statistics. Although this is not always the case, using descriptive statistics as a starting point is actually nice because we are more familiar with the concept of, e.g., "average" and "spread" within samples than a population. The essence of descriptive statistics is to explain the data with summary statistics/important number or through the use of visualization tools such as hitogram. The data size might reach hundreds, thousands, or even more, which makes simple data inspection difficult (you surely do not want to look at the data row by row). The analyzed data itself can represent a population or is a population itself (this concept will be clearer later when discuss the concept of population and sample).

In general, there are several types descriptive statistics to describe our data, including measures of central tendency and measures of variability. When we have data, intuitively we want to know the average and the spread of the data, which is why we start with the concept of *centerness* and *spread*. In some occassions, we also want to know the skewness of the data and other important measures, such as *kurtosis*.

Some notes before you read this tutorial:

- The notation $\mathcal{N}(\text{mean}, \text{sd})$ denotes a normal distribution with a specified mean and standard deviation.
- We will use some plotting tools in this document. Please see my other Markdown file to learn about R's plotting feature.
- You need to install some additional packages such as `moments` that we use to calculate skewness.
- When random data is generated (e.g., via `rnorm()` function), expect to see different results than what are shown in this document. This is normal, the important thing is that you understand the essence and the concept.
- I highly suggest you to learn the basics of R first. Prior experience with other high-level languages such as MATLAB or Python will be extremely useful.

2 Measures of central tendency

We will begin with descriptive statistics, in which our aim is to obtain several numbers / coefficients that describe our data. By obtaining these coefficients, you will get a first impression on the nature of your data (e.g., what is the central tendency of the data?).

Let us begin with the measures of central tendency. The central tendency, as the name suggests, is a measure that shows the "centerness" of your data. Let us denote the collected data into a vector $X = \{x_1, x_2, \ldots, x_n\}$ (n is the number of observation), and now we want to calculate the central tendency of X. Several popular measures of central tendency that you can use include the following:

- Mean ($\bar{X}$): where $\bar{X} = (\sum_{i=1}^{n} x_i)/n$.

- Median ($\tilde{X}$): By median, the data is sorted with ascending order. This sorted data is then splitted into the higher half and the lower half. The point where the data is splitted is called the median.
- Mode: The most frequent value in your data set.

For example, if your data is $X = \{4, 3, 2, 1, 5, 7, 6, 7, 7\} (n = 9)$, then

- $\bar{X} = (4 + 3 + 2 + 1 + 5 + 7 + 6 + 7 + 7)/9 = 4.6667$
- Your median is 5 ($\tilde{X} = 5$). You got this by sorting your data first, i.e, $\{1, 2, 3, 4, \mathbf{5}, 6, 7, 7, 7\}$. See that 5 is in the center.
- Your mode is 7, as you can see that it appears 3 times.

As a side note, it is possible that you have data with no mode, especially when your data is generated from continuous processes (e.g., from temperature measurement data).

Let us begin by defining X as a vector using `c()`:

```
X <- c(4,3,2,1,5,7,6,7,7)
```

You can check if you input `X` correctly by typing `X` in your R console. Alternatively, you can use `print(X)`:

```
print(X)
```

```
## [1] 4 3 2 1 5 7 6 7 7
```

Another side note, most of the R functions that we will use take numerical data (i.e., `numeric`) as their inputs. For a data frame, the column or the variable need to be extracted first in a form of numerical data (using `'$'`). Notice that some advanced features of R can directly take data frame as the input. However, this tutorial mostly focuses on numerical data so as to make it easier to grasp the concept. I need to mention here that it is also possible to import the data from spreadsheet or CSV files into your R session for further processing in R, which is actually a more typical way of inputting the data into the R session.

R has several built-in functions for calculating the mean and the median from data, namely

- `mean()` to calculate (arithmetic) mean.
- `median()` to calculate median.

See an example below:

```
mean(X)
```

```
## [1] 4.666667
```

```
median(X)
```

```
## [1] 5
```

The `mean()` function can have an additional argument `trim`, in which this argument is used to calculate the trimmed mean according to the given fraction of trimmed data. For example, if you want to calculate the trimmed mean with trimmed data 20% on both sides, we need to type `mean(X,trim=0.2)` or simply `mean(X,0.2)`.

Another important feature of `mean()` and `median()` is that these functions can calculate the statistics even when some entries are invalid (`NA`). An extra argument `na.rm = TRUE` is needed to accomplish such a task. For example, to calculate mean:

```r
X_NA <- c(4,3,2,1,5,7,NA,6,7,7) # data with NA
mean(X_NA) # will result in NA
```

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

```r
mean(X_NA,na.rm=TRUE) # throwing away NA before calculating the mean
```

```
## [1] 4.666667
```

R does not provide a built-in function to calculate mode. It is relatively easy to code a function for calculating mode. However, for the sake of simplicity, this tutorial only focuses on R's built functions.

3 Minimum and maximum

Frequently, we want to know the maximum and the minimum of the data. R provides intuitive functions to do just that, namely, `min()` and `max()`. Similar to other functions, invalid entries can be skipped by including `na.rm=TRUE` as an extra argument. Using our previous `X` data:

```r
min(X)
```

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

```r
max(X)
```

```
## [1] 7
```

4 Quantiles, percentiles, dan quantiles

To grasp the concept of interquartile range (IQR), we need to know the concept of quartile first. By quartile, we mean that the (sorted) data is divided into four equal parts (quarters). Quartile itself is a type of quantile, which is the percent of points below the given value. For example, 0.4 (or 40%) quantile is the value at which 40% percent of the data fall below that particular value. There are various types of quantile, some of them are:

- **Percentiles**, the 100-quantiles.
- **Deciles**, the 10-quantiles.
- **Quartiles**, the 4-quantiles.

The z-quantiles, where z is an integer, is defined such that the data is equally divided into z parts. As an example, *quartile* divides the data into four parts with three important numbers, that is, the *lower quartile* (Q1), *median* (Q2), and *upper quartile* (Q3). In other words, Q1, Q2, and Q3, are, respectively, the 0.25, 0.5, and, 0.75 quantile. As the name suggests, percentiles divide the data 100 parts and each value of the percentile indicates the value below which a given percentage of observations in a group of observations fall. To take an example the 20th percentile is the value 20% of the observations fall below that value.

The following code calculates quantiles from data geenerated from a standard normal distribution $\mathcal{N}(0, 1)$. For that purpose, the random data will be generated using `rnorm()`. The quantile is then subsequently calculated using `quantile()`, which takes minimum two arguments, namely, `x` (your data), and `probs` (a

value where the quantile is calculated). It is possible to use several values for `probs` in a form of vector. As an example, we can use `probs=c(0.25,0.5,0.75)` to calculate quartiles. Notice that any value between 0 and 1 is valid for `probs`.

```
X <- rnorm(10000,mean=0,sd=1) # Data from standard normal distribution

quantile(X,c(0.25,0.5,0.75)) # Calculate quartiles
```

```
##          25%          50%          75%
## -0.681101757  0.007245521  0.688435112
```

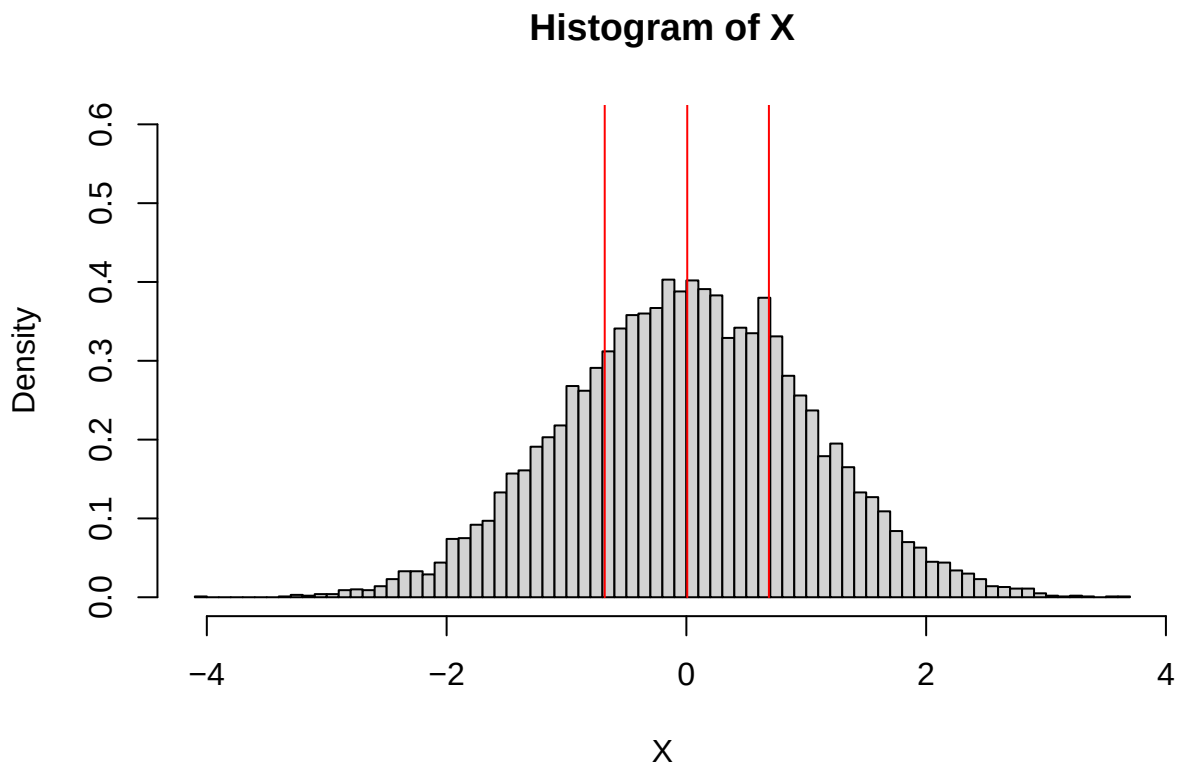
```
quantile(X,c(0.1,0.2,0.3,0.7,0.85)) # Calculate quantiles for arbitrary values
```

```
##        10%        20%        30%        70%        85%
## -1.3057896 -0.8546008 -0.5257077  0.5584532  1.0456874
```

The following plot draws the quartiles to illustrate the concept of quantiles (the red lines are the quartiles):

```
QUART <- quantile(X,c(0.25,0.5,0.75)) # Calculate the quartiles
hist(X,100,freq=FALSE,ylim=c(0,0.6)) # Draw a histogram

lines(c(QUART[1],QUART[1]),c(0,1),col="red")
lines(c(QUART[2],QUART[2]),c(0,1),col="red")
lines(c(QUART[3],QUART[3]),c(0,1),col="red")
```



5 Measures of variability

Besides measures of central tendency, it is also important to understand the dispersion of the data around the "centre". Measures of variability that are commonly used in practice are as follows:

- **Standar deviasi** (σ), probably the most popular measure, where $\sigma(X) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{X})}$ (sample standard deviation).
- **Range**, defined as $\text{Range}(X) = \max(X) - \min(X)$.
- **IQR (interquartile range)**, $(IQR = Q3 - Q1)$.

The standard deviation is calculated using R's built-in function `sd()`. Range is easily calculated by `max()-min()`. Also, there is a built-in function `range()` but it is used to output both the minimum and the maximum, not the actual range. Lastly, `IQR()` outputs the interquartile range. See the following example, where the data is generated from $\mathcal{N}(8, 2)$:

```
X <- rnorm(10000,mean=5,sd=2) # Data from a normal distribution

sd(X) # Calculate standard deviation
```

```
## [1] 1.988286
```

```
max(X)-min(X) # Calculate range
```

```
## [1] 14.41967
```

```
IQR(X) # Calculate interquartile range
```

```
## [1] 2.728333
```

As an alternative to standard deviation, the variance of the data can also be computed ($\text{Var}(X) = \sigma^2(X)$) using a built-in function `var()`:

```
var(X) # Calculate variance
```

```
## [1] 3.953283
```

6 Measures of asymmmetry

For a given data set, there is a possibility that the data is *skewed*. We might have some of the following questions in mind: "is the data leaning more toward the left side, or the right side?". Imagine that you are talking about your salary per week and then you collect the data for about two or three years. You obviously want your salary data to lean more toward the right side, that is, you frequently obtained high salary per week. To measure that, you need to measure the asymmetry in your data by calculating the skewness (s) of your data set, reads as

$$s = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{X})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{X})^2\right)^{3/2}}$$

The value of the skewness (s) might indicate three different trends:

- If $s > 0$ (positive skew), the data leans more toward the left side and the tail is heavier on the right side.
- If $s < 0$ (negative skew), the data leans more toward the right side and the tail is heavier on the left side.
- If $s = 0$, or close to zero, your data is symmetric (for example, a normal distribution is a symmetric distribution).

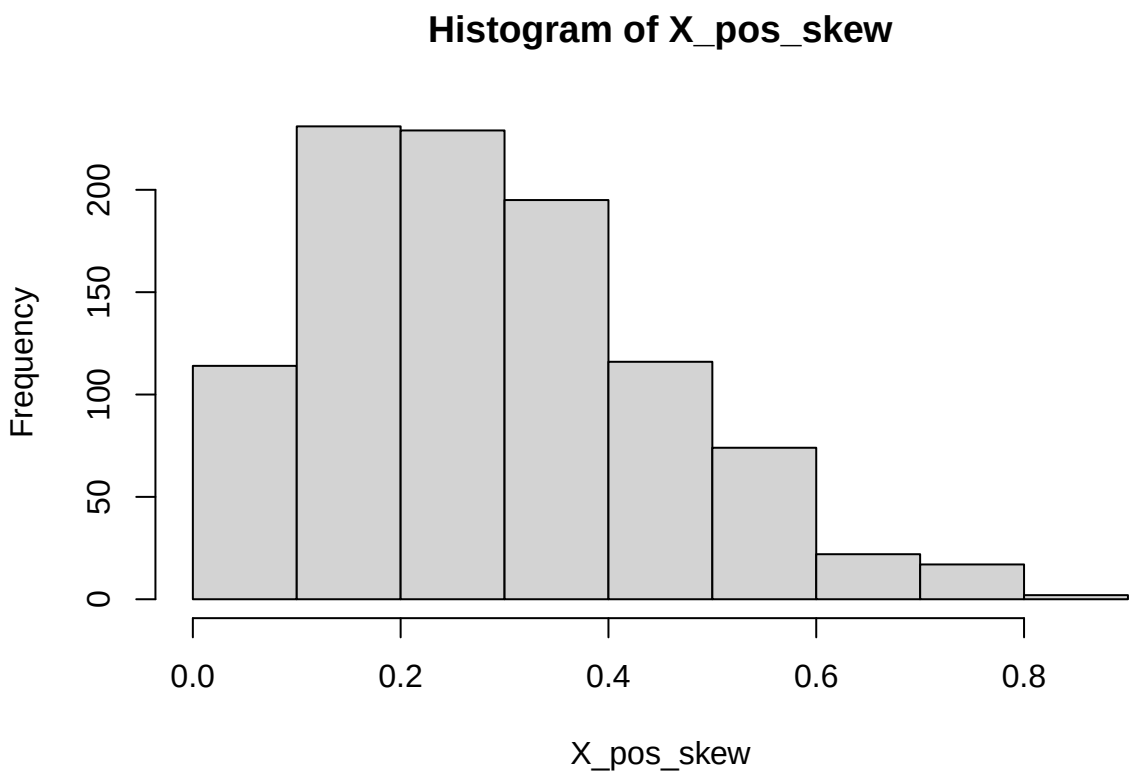
R does not provide a built-in function to calculate skewness, so we will import an extra package `moments` that needs to be installed in your system first. To install `moments`, type `install.packages("moments")` followed by `library("moments")` in your R console first. For demonstration, we will generate data from a beta distribution with various parameters to adjust the skewness. There are two non-negative parameters for a beta distribution, namely `shape1` and `shape2` which appear as the exponent of the random variable and manage the distribution's shape. For ease of notation, let us denote `shape1` and `shape2` as α and β, respectively. A beta distribution is then defined as Beta(α, β). Try the following example:

```
library(moments)

n <- 1000 # Sample size
X_pos_skew <- rbeta(n,2,5) # Positive skew, Beta(2,5)
X_neg_skew <- rbeta(n,5,2) # negative skew, Beta(5,2)
X_no_skew <- rbeta(n,5,5) # zero skew (or close to zero), Beta(5,5)
```

You can visualize the distribution of the data by using `hist()`. The next code draws a histogram for `X_pos_skew` (please also try doing the same procedure for `X_neg_skew` and `X_no_skew`):

```
hist(X_pos_skew)
```



Next, the skewness is calculated by using `skewness()` from the package `moments`:

```
skewness(X_pos_skew)
```

```
## [1] 0.6375409
```

```
skewness(X_neg_skew)
```

```
## [1] -0.5603034
```

```
skewness(X_no_skew)
```

```
## [1] -0.004951315
```

7 Korelasi Pearson

When analyzing two different data sets (say, X and Y), you might have interest in investigating the relationship between the two data sets. You might have questions such as: (1) If I increase X , will Y decrease?, (2) Does changing X really affect Y ?, (3) if there is indeed a relationship, how strong is the relationship?, et cetra.

One way to answer such questions is to use **Pearson correlation coefficient** which measures the linear relationship between two datasets. The Pearson correlation coefficient of two variables, say X and Y, is calculated as

$$\rho_{X,Y} = \text{corr}(X,Y) = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

where cov is covariance, $\mathbb{E}$ denotes expectation, μ_X is the mean of population X, μ_Y is the mean of population Y. Notice that the equation above is used to calculate the correlation between populations.

In practice, the size of the data is finite. Hence, $\rho_{X,Y}$ is estimated from sample as follows:

$$\rho_{X,Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{Y})^2}}$$

The value of $\rho_{X,Y}$ always falls between -1 and +1. In this regard, $\rho_{X,Y} = 1$ indicates an exact linear positive relationship, while $\rho_{X,Y} = -1$ indicates an exact linear negative relationship. If $\rho_{X,Y} \approx 0$, or very close to zero, then no correlation exists. The (absolute) magnitude of $\rho_{X,Y}$ indicates the strength of the positive/negative relationship. As a rule of thumb, we call $|\rho_{X,Y}| > 0.5$ for a strong correlation while $|\rho_{X,Y}| \leq 0.5$ indicates a weak correlation.

The R function to use is `cor` that takes minimum two arguments, that is, the two data sets (which is X and Y for our case). For demonstration, we define X as $X = (1, 3, 5, 8, 9, 11, 12)$. The other data set Y is generated according to one of the following scenarios:

- $Y = X + 2 + \mathcal{N}(0, 1)$, for a strong and positive correlation,
- $Y = \mathcal{N}(0, 1)$, where Y is a random noise (i.e., no correlation),
- $Y = -(X + 2) + \mathcal{N}(0, 1)$, for a strong and negative correlation,
- $Y = X + 2 + \mathcal{N}(8, 1)$, for a weak and positive correlation, and
- $Y = -(X + 2) + \mathcal{N}(8, 1)$, for a weak and negative correlation,

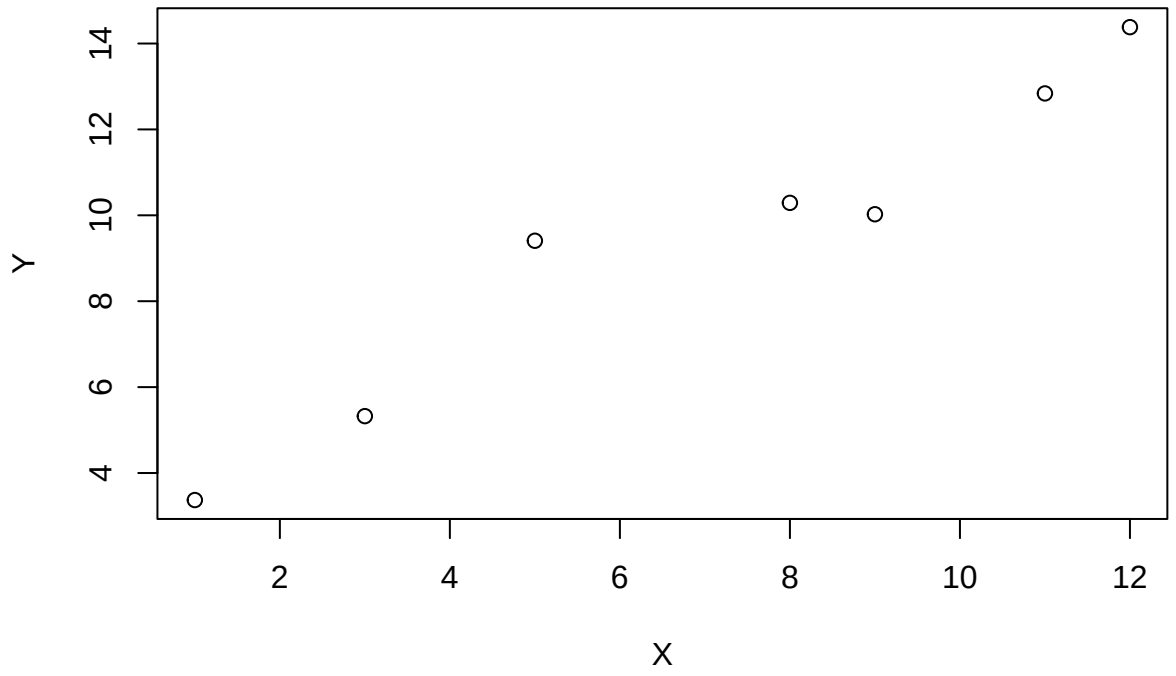
In addition, a scatter plot between X and Y will also be drawn so that you can see the meaning of Pearson correlation coefficient. I suggest you to try changing `Y` (just delete `#` from the following snippet) for a different correlation scenario:

```
X <- c(1,3,5,8,9,11,12)
Y <- X + 2 + rnorm(length(X)) # + correlation (strong)
# Y <- rnorm(length(X)) # No correlation
# Y <- -(X + 2) + rnorm(length(X)) # - correlation (strong)
# Y <- X + 2 + 8*rnorm(length(X)) # + correlation (weak)
# Y <- -(X + 2) + 8*rnorm(length(X)) # - correlation (weak)

cor(X,Y) # Calculate the correlation
```

```
## [1] 0.9692164
```

```
plot(X,Y) # Plot X vs Y
```



8 using `summary()` function

One of the most important R functions for descriptive statistics is `summary()`. This function simultaneously calculates the minimum, the maximum, the three quartiles, and the mean:

```r
X <- rnorm(10000,mean=8,sd=2) # Data from a normal distribution N(8,2)
summary(X) # Calculate the summary of X
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.311   6.637   7.973   7.977   9.278  16.645
```

9 Note: Saving the statistics into a variable

Notice that throughout this tutorial we directly output the statistics. However, it is actually much more convenient if we save these values into variables for further processing. As an example, the following code saves the calculation into several variables:

```r
result.mean <- mean(X)
result.mean <- median(X)
result.summary <- summary(X)
```

10 Final thoughts

That is the end of this short tutorial on how to do descriptive statistics using R. The main purpose of descriptive statistics, as the name implies, is to describe data in terms of only a few significant numbers. R is a very convenient software (call it programming language if you wish) to use for this purpose. Furthermore, R is also convenient to use when the size of your data is relatively large, which is not very convenient to process if you use spreadsheets.

You might have several questions such as: "Cannot all of this be done in a spreadsheet software alone?". The answer is, of course you can. However, R is very convenient because all calculations can be done by typing scripts. Apart from that, R has a lot of statistical features that standard Spreadsheet software does not provide. The plotting and data visualization features of R are also much better than most commercially available spreadsheet softwares (feel free to disagree). Not to mention that R is free.

That's all folks!