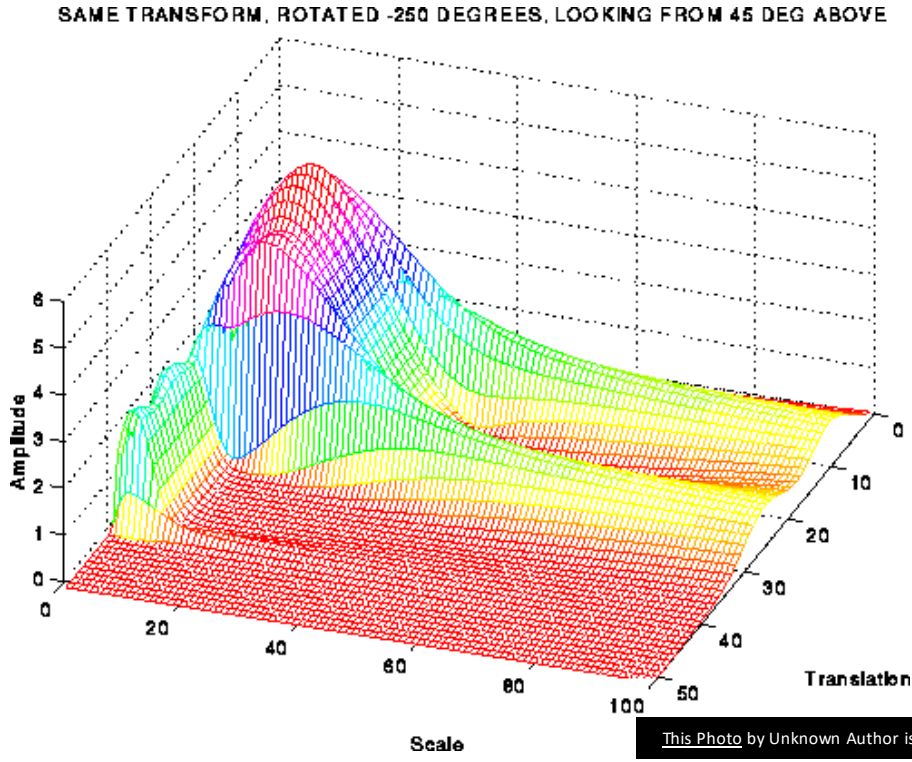


SIT718 - Data Transformations - Week 1

Delaram Pahlevani

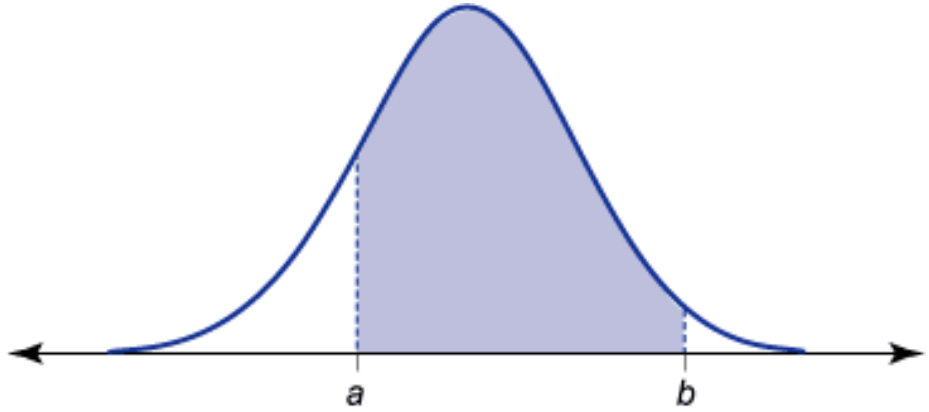
Probability distributions

A probability distribution is a mathematical function that provides the probabilities of occurrence of different possible outcomes in an experiment. In more technical terms, the probability distribution is a description of a random phenomenon in terms of the probabilities of events.

Example: if the random variable X is used to denote the outcome of a coin toss ("the experiment"), then the probability distribution of X would take the value 0.5 for X = heads, and 0.5 for X = tails (assuming the coin is fair).

Image Source: wikipedia

Continuous probability distribution



Main properties:

- continuous types of data or random variables.
- different kinds of distributions
- cannot be expressed in tabular form similar to discrete probability distribution
- an equation or formula is used to describe a continuous probability distribution (probability density function).

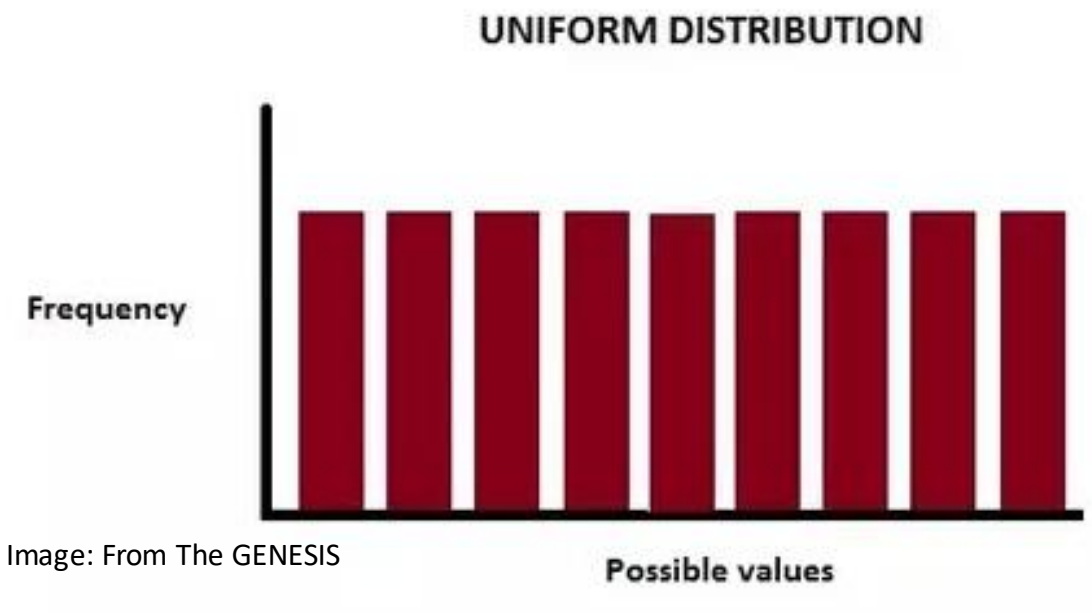


Image: From The GENESIS

Uniform distribution

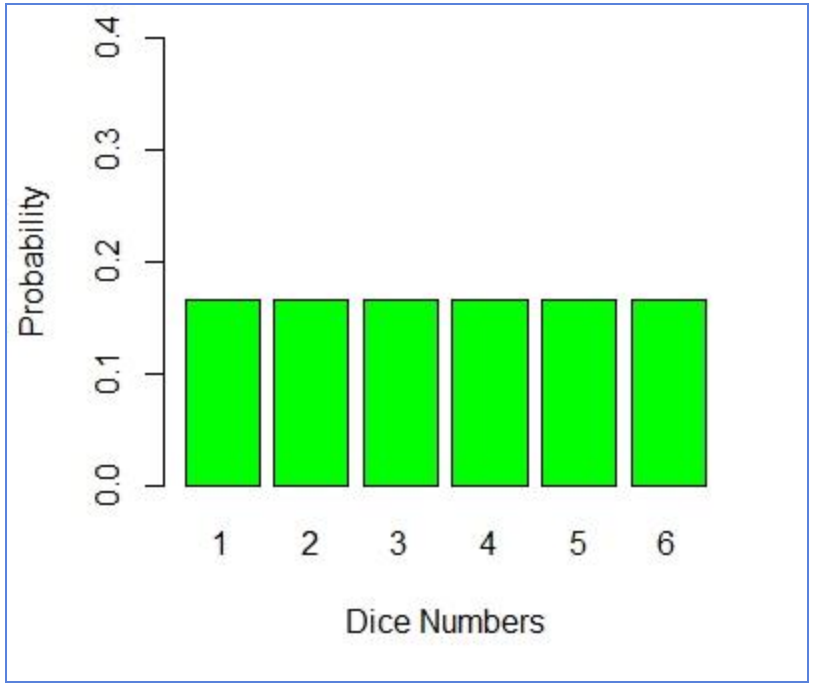
This is a distribution that describes every event having an equal probability of happening.

Example 1: Probabilities of a Dice Roll

```
barplot(c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6),
      names.arg=c("1", "2", "3", "4", "5", "6"),
      main="Probability of Each Number of a Dice Occurring on a
Single Dice Roll", xlab="Dice Numbers",
      ylab="Probability",
      ylim=c(0,1), col = "green"
)

barplot
```

```
barplot(height, width = 1, space = NULL, names.arg = NULL, legend.text = NULL, beside = FALSE, horiz = FALSE, density = NULL, angle = 45, col = NULL, border = par("fg"), main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, xpd = TRUE, log = "", axes = TRUE, axisnames = TRUE, cex.axis = par("cex.axis"), cex.names = par("cex.axis"), inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0, add = FALSE, args.legend = NULL, ...)
```



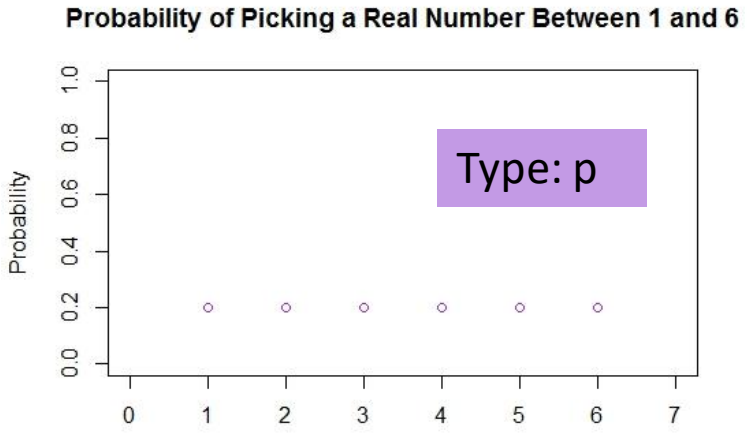
Example 2: Probability of Picking a Real Number Between 1 and 6

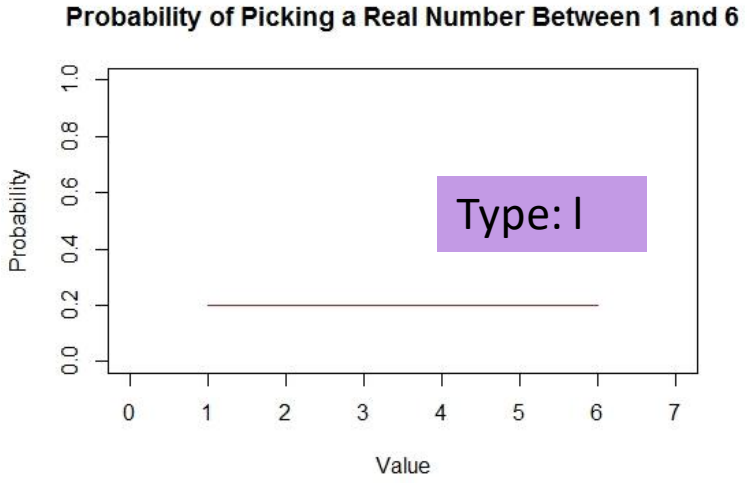
```
plot(c(1:6), array(1/5, 6), type="l", xlim=c(0,7), ylim=c(0,1),
xlab="Value", ylab="Probability", main="Probability of Picking a
Real Number Between 1 and 6" )
```

Plot(x coordinates, y coordinates, type = “p”, “l”, “b” and etc, main title, sub = sub title)

```
polygon(c(1,1,6,6), c(0,0.2, 0.2, 0), angle=30, density=10 , col =
"pink")
```

```
polygon(x, y = NULL, density = NULL, angle = 45, border =
NULL, col = NA, lty = par("lty"), ..., fillOddEven = FALSE)
```

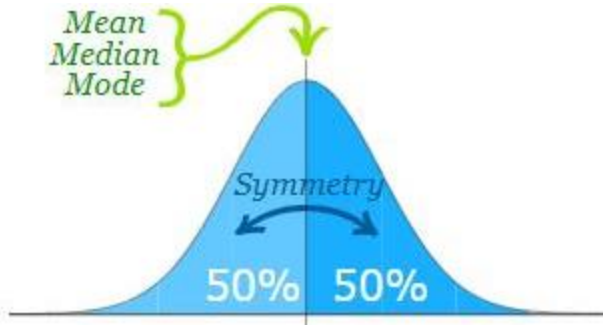


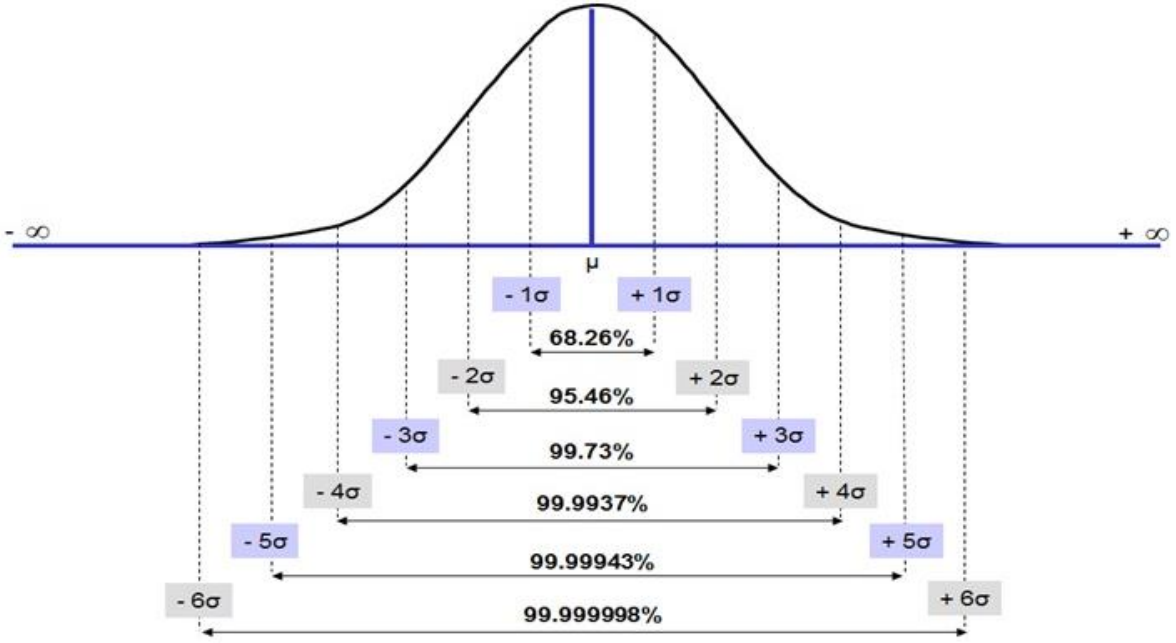


Normal Distribution

Main properties:

- ✓ Normal distributions are symmetric around their mean.
- ✓ The mean, median, and mode of a normal distribution are equal.
- ✓ The area under the normal curve is equal to 1.0.
- ✓ Normal distributions are denser in the centre and less dense in the tails
- ✓ Normal distributions are defined by two parameters, the mean (μ) and the standard deviation (σ)
- ✓ 68% of the area of a normal distribution is within one standard deviation of the mean
- ✓ Approximately 95% of the area of a normal distribution is within two standard deviations of the mean.





Standard deviation

The standard deviation is a single number that allows us to discern how spread out our data is. The variance and standard deviation are technically not aggregation functions as they do not satisfy the boundary properties.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Standardisation

Standardisation is the process of converting a set of data to a N(0,1) distribution i.e. a normal distribution with mean **0** and standard deviation **1**. It can help us make decisions about our data.

So to convert a value to a Standard Score ("z-score"):

- first subtract the mean,
- then divide by the Standard Deviation

And doing that is called "Standardizing":

$$z = \frac{x - \mu}{\sigma}$$

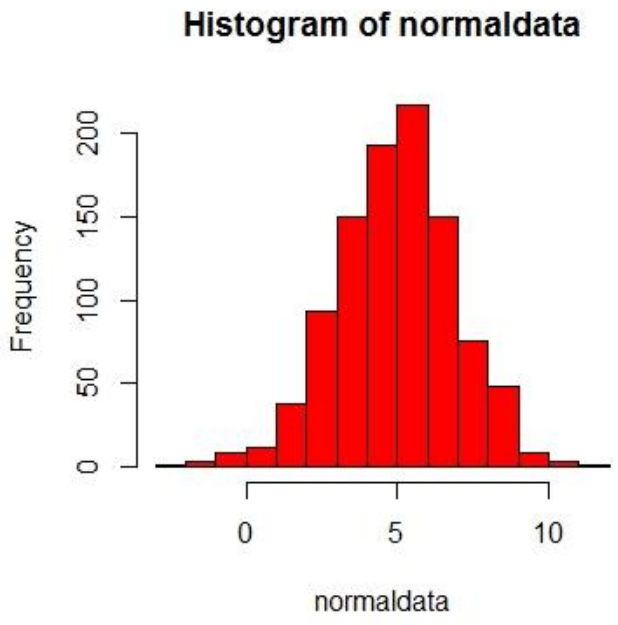
- **z** is the "z-score" (Standard Score)
- **x** is the value to be standardized
- **μ** is the mean
- **σ** is the standard deviation

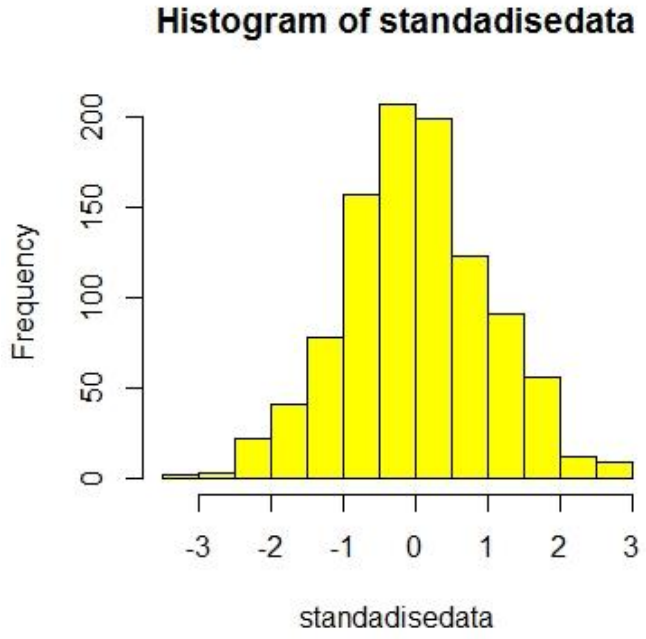
Rnorm function in RStudio

Task: Create a sample of 1000 numbers with mean 5 and standard deviation 2 which are normally distributed.

Generating numbers with normal distribution
rnorm(n,m,s)

```
normaldata = rnorm(1000, 5, 2)
m = 5
s =2
standadisedata = (normaldata - m)/s
hist(normaldata , col = "red")
hist(standadisedata, col = "yellow")
```





Change entries in a vector

R Exercise 1 Create the vector and change the entries.

```
a <- array(0,20)
a[5] <- 1
a[c(3,7,11)] <- c(2,6,1)
a[17:20] <- c(1,2,1,4)
```

Your expected output when you type a and press enter should now be

0 0 2 0 1 0 6 0 0 0 1 0 0 0 0 0 1 2 1 4

- Column bind using cbind function
- Row bind using rbind function

R Exercise 2 Assign the vectors and perform the cbind() and rbind operations.

```
a <- c(1,2,3,7,9)
a <- cbind(a, c(21,2,1,5,6))
a <- cbind(a, c(2,-1,5,0,-1))
a <- cbind(a,c(1,9,7,2,1),array(6,5))
b <-c(3,6,1,9, 2)
b <- rbind(b, c(3,2,1,8,9))
b <- rbind(b, c(4,1,12,1,2))
```

Multidimensional array

R Exercise 3 Create a 3*4 array and then assign values to different entries using the following.

```
A[3,1] <- 4
A[1,] <- c(1,2,3,4)
A[,2] <- c(6,5,4)
A[2:3,3:4] <- array(-1,c(2,2))
```

Your final matrix should appear as

```
     [,1] [,2] [,3] [,4]
[1,]    1    6    3    4
[2,]    0    5   -1   -1
[3,]    4    4   -1   -1
```

We can also consider entire columns, e.g. the second using A[,2] and entire rows using A[1,].

Rank-based scores – functions in Rstudio

- Sort(vector): reordering a vector into increasing
- Order(vector, decreasing = TRUE): indices from highest to lowest, or lowest to highest
- Rank(vector): relative ranking of the variables

Power mean in RStudio

R Exercise 7- Define the power mean as a function in R and check the following:

PM(c(3,2,7),2)
PM(c(1,0,7),0)
PM(c(0.28,0.4,0.47),-557)
PM(c(0.28,0.4,0.47),-558)

```
PM = function(x,p){
if(p == 0) {
prod(x)^(1/length(x))
}
else {
(mean(x^p))^(1/p)}
}
```

Plot for function in RStudio

To plot the function f(t) = t^2 for these values, we can either use

```
x = (1:100)/100
plot(x^2)


y=array(0,length(x))
for(i in 1:length(x))
{
y[i]= (x[i]^2)
}
plot(x,y)
```