Probability Distributions in R

Probability Distributions in R

Working with probability distributions

When we have a probability distribution there are several operations that we can do conditioning on certain parameters values:

- generate random x values
- calculate the density of a certain x value
- calculate the cumulative probability of a certain x value
- calculate the x value associated to a certain cumulative probability

Probability Distributions in R

In R there are several probability distributions (PD) implemented as functions. Basically the corresponding equation of the PD is converted into R code. For example, the Gaussian distribution Probability Density Function (PDF) is represented in Equation 1.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

Gaussian distribution example

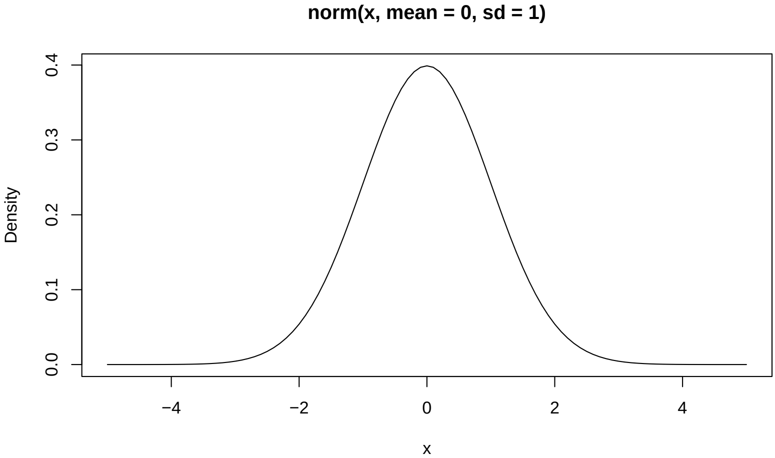
Let's convert the Equation 1 into R code. Our variable is x then we have μ and σ that are the mean and standard deviation of the Gaussian distribution.

```
norm <- function(x, mean = 0, sd = 1){
    1 / sqrt(2 * pi * sd^2) * exp(-((x - mean)^2)/(2 * sd^2))
}

norm(0)
#> [1] 0.3989423
norm(2)
#> [1] 0.05399097
norm(-1)
#> [1] 0.2419707
```

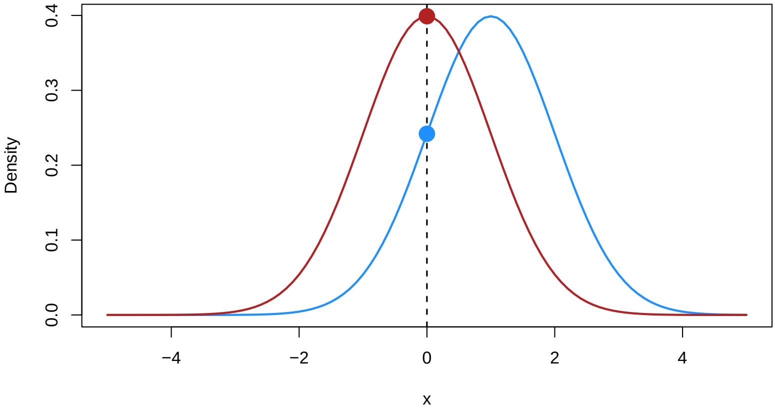
Gaussian distribution example

With the previous code we are calculating the probability density of a certain value given the parameters. Let's use `norm()` for a sequence of values and plot the results.



Gaussian distribution example

Clearly, if we change the parameters, the calculated densities will be different. For example:



Gaussian distribution in R

Fortunately we do not need to write the probabilities distribution manually but a lot of them are already included in R. For example, the `norm()` function can be replaced by `dnorm()`.

```
norm(0, 1, 2)
```

```
#> [1] 0.1760327
```

```
dnorm(0, 1, 2)
```

```
#> [1] 0.1760327
```

`d`, `q`, `r` and `p` functions

`d`, `q`, `r` and `p` functions

Actually in R there are already implemented a lot of probability distributions. This document https://cran.r-project.org/web/views/Distributions.html provides a very comprehensive overview.

The general idea is always the same, regardless the distribution:

- generate random x values **there is the `r` function**
- calculate the density of a certain x value **there is the `d` function**
- calculate the cumulative probability of a certain x value **there is the `p` function**
- calculate the x value associated to a certain cumulative probability **there is the `q` function**

`d`, `q`, `r` and `p` functions

The combination is `d`, `p`, `q` or `r` + the function contaning the equations of that specific distribution. Thus we can use `dnorm()`, `pnorm()`, `qnorm()` and `rnorm()`.

Maximum Likelihood

The `d` function provides the probability density (or likelihood) of a certain value(s) fixing the parameters. What about fixing the value(s) and changing the parameters?

Let's assume we have $n = 10$ values from a Normal distribution with unknown parameters:

```
#>  [1] 15.06 13.66  7.77 12.26  9.72 13.68 11.68 13.33  0.62
#> [10]  2.23
```

We can calculate the mean and standard deviation:

```
mean(x)
#> [1] 10
sd(x)
#> [1] 5
```

Maximum Likelihood

Now, we can calculate the likelihood of the 10 values. Which values should we used for the parameters? We can try different values for μ and σ:

```
dnorm(x, 0, 1)
#>  [1] 2.214419e-50 1.152300e-41 3.086863e-14 9.204670e-34
#>  [5] 1.226524e-21 9.441835e-42 1.004709e-30 1.086009e-39
#>  [9] 3.291893e-01 3.344474e-02
dnorm(x, 10, 5)
#>  [1] 0.04780475 0.06100511 0.07223775 0.07204280 0.07966271
#>  [6] 0.06087474 0.07543414 0.06394678 0.01373134 0.02382822
dnorm(x, -5, 2)
#>  [1] 2.836064e-23 2.457018e-20 2.795900e-10 1.343146e-17
#>  [5] 3.454794e-13 2.295467e-20 1.603083e-16 1.166393e-19
#>  [9] 3.848397e-03 2.916100e-04
```

Maximum Likelihood

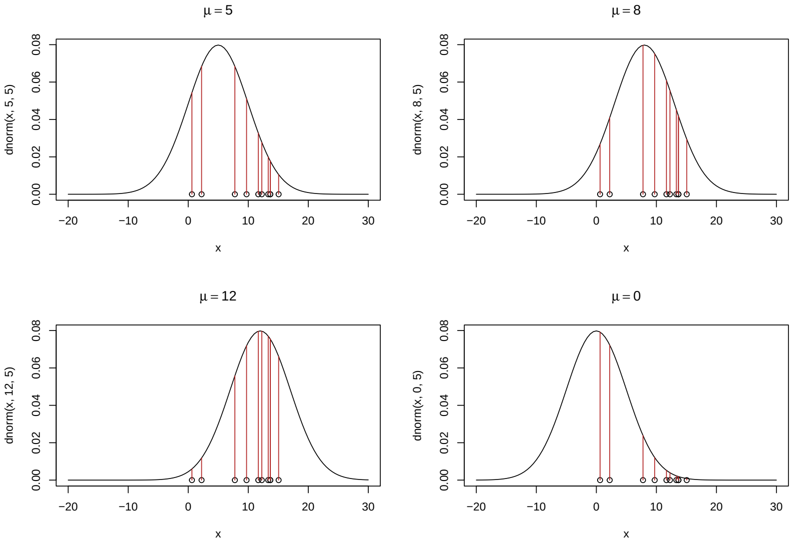
We can take the product (or the sum of the log-transformed values):

```r
prod(dnorm(x, 0, 1))
#> [1] 1.008627e-270
prod(dnorm(x, 10, 5))
#> [1] 1.16165e-13
prod(dnorm(x, -5, 2))
#> [1] 4.354571e-142
```

Maximum Likelihood

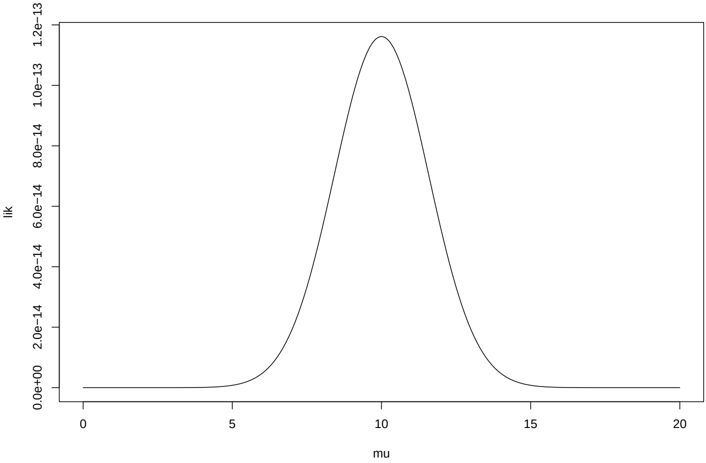
What about varying a parameter, e.g., μ? We can fix the σ to a certain value, for example 5.

Maximum Likelihood



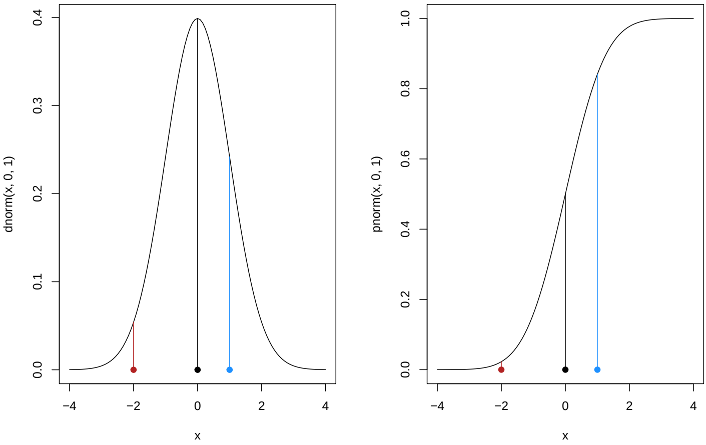
Maximum Likelihood

There is a point where the likelihood is maximised. The point is when the sum of the heights of the red segments is maximised.



Cumulative distribution

With the `p` function we calculate the cumulative probability associated with a given value.



Inverse Cumulative distribution

The `q` function is basically the inverse of the `p` function. We want to know which is the `x` value associated with a given cumulative probability. One is just the inverse of the other.

```
p <- pnorm(-0.5, 0, 1)
p # % of area on the left of -0.5 (given mu and sigma)
```

```
#> [1] 0.3085375
```

```
q <- qnorm(p, 0, 1)
q # value associated with p%  of cumulative probability
```

```
#> [1] -0.5
```

If you remember from Psychometrics courses, these are respectively the percentile and the rank percentile.

Generating numbers

Finally the `r` function can generate random numbers, fixing the parameters values. This is the core of Monte Carlo simulations.

```r
x <- rnorm(100, 10, 5)
head(x)
```

```
#> [1]  6.4320970 15.0558407 10.6856186 -0.9744109 10.6790606
#> [6] 15.6645378
```

```r
summary(x)
```

```
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -0.9744  6.2342 10.4991  9.9260 13.1631 22.1059
```

```r
# new values everytime you run the command
summary(rnorm(100, 10, 5))
```

```
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  -1.324   6.700   9.963   9.952  12.518  23.421
```

Discrete distributions

The same functions can also be used with discrete probability distributions. For example the Binomial or the Poisson distributions.

```r
rpois(n = 10, lambda = 20)
```

```
#>  [1] 12 17 26 26 26 35 24 21 18 19
```

```r
dpois(x = 10, lambda = 20)
```

```
#> [1] 0.005816307
```

```r
ppois(q = 10, lambda = 20)
```

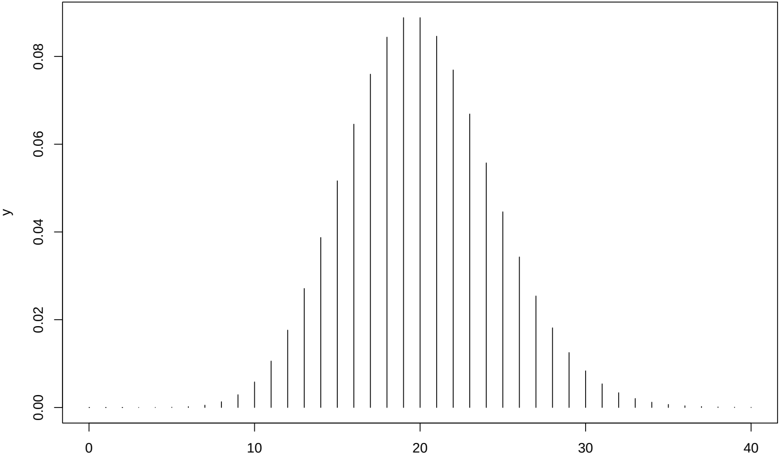
```
#> [1] 0.01081172
```

```r
qpois(p = 0.5, lambda = 20)
```

```
#> [1] 20
```

Discrete distributions, Poisson

In the Poisson distributions we are counting the number of events.
We can have 10 or 11 events, not 10.5.



Discrete distributions, Binomial

In the Binomial distribution we are counting the number of successes for a total number of trials. Also here we can have 10 successes or 11, not 10.5.