Statistical Methods

Lab 1

V. Gioia (and N. Torelli and G. Di Credico)

vincenzo.gioia@units.it

08/10/2024

Office hours: Friday, 5.00 - 6.30 pm Get in touch for availability outside the scheduled office hours

Materials: All the material can be found in the Moodle page

1. Pre-LAB R code: To complete (together) some lines of code
2. Post-LAB R code: It will replace the R code pre-LAB after the lecture
3. Report pre-LAB: Missing some parts
4. Report post-LAB: It will replace the Report pre-LAB after the lecture

First Lab's goals. Learn about:

1. **Central Limit Theorem** (CLT)
2. **Bivariate Normal distribution**

Contents

Central Limit Theorem

Recall the CLT

Let $X_1, \ldots, X_n$ be a sequence of independent and identically distributed (iid) random variables (rv) from a distribution having

$$\mathbb{E}(X_i) = \mu, \qquad \mathbb{V}(X_i) = \sigma^2 < \infty, \quad i = 1, \ldots, n$$

For large n ($\dot{\sim}$ indicates convergence in distribution)

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i \dot{\sim} \mathcal{N}(\mu, \sigma^2/n) \quad S_n = \sum_{i=1}^{n} X_i \dot{\sim} \mathcal{N}(n\mu, n\sigma^2)$$

- The CLT supports the normal approximation to the distribution of a rv that can be seen as the sum of other rvs
- The CLT is useful for computing some quantities

Example

Let X and Y be two independent rv, such that $X \sim \text{Bin}(n, p)$ and $Y \sim \text{Bin}(m, q)$

We are interested in computing the probability $P(X > Y)$. **The Normal approximation is the simplest way to compute** $P(X > Y)$

- $X \approx \mathcal{N}(np, np(1-p)), \qquad Y \approx \mathcal{N}(mq, mq(1-q))$
- $W = X - Y$ is still normal (well known probability result), having

$$\mu_W = \mu_X - \mu_Y = np - mq \qquad \sigma^2_W = \sigma^2_X + \sigma^2_Y = np(1-p) + mq(1-q)$$

- $P(X > Y) = P(W > 0)$ can be computed easily

Waterpolo match example

- **Problem**: Tomorrow two professional Italian waterpolo teams, Posillipo and Pro Recco, compete against each other.
- **Goal**: We are interested in computing the probability that Posillipo win the next match against Pro Recco

Assumptions and quantity of interest

- Let X (Y) be the random number of **goals scored** by Posillipo (Pro Recco), and assume that X and Y follow two independent Binomial distributions
- X (Y) represents the number of shots converted in goal on the total number of shots n (m) made by Posillipo (Pro Recco), having probability p (q)
- Probability that Posillipo win: $P(X > Y) = P(X - Y > 0) =?$

- We adopt a simplification, and we treat the quantities p, q, m, n as *known*. For instance, based on historical experience we fix

$$p = 0.5, \quad q = 0.7, \quad n = 20, \quad m = 20$$

- Let W be the r.v., such that $W = X - Y$, we can easily compute the probability of interest ($P(X - Y > 0) = P(W > 0) = ?$) leveraging the CLT

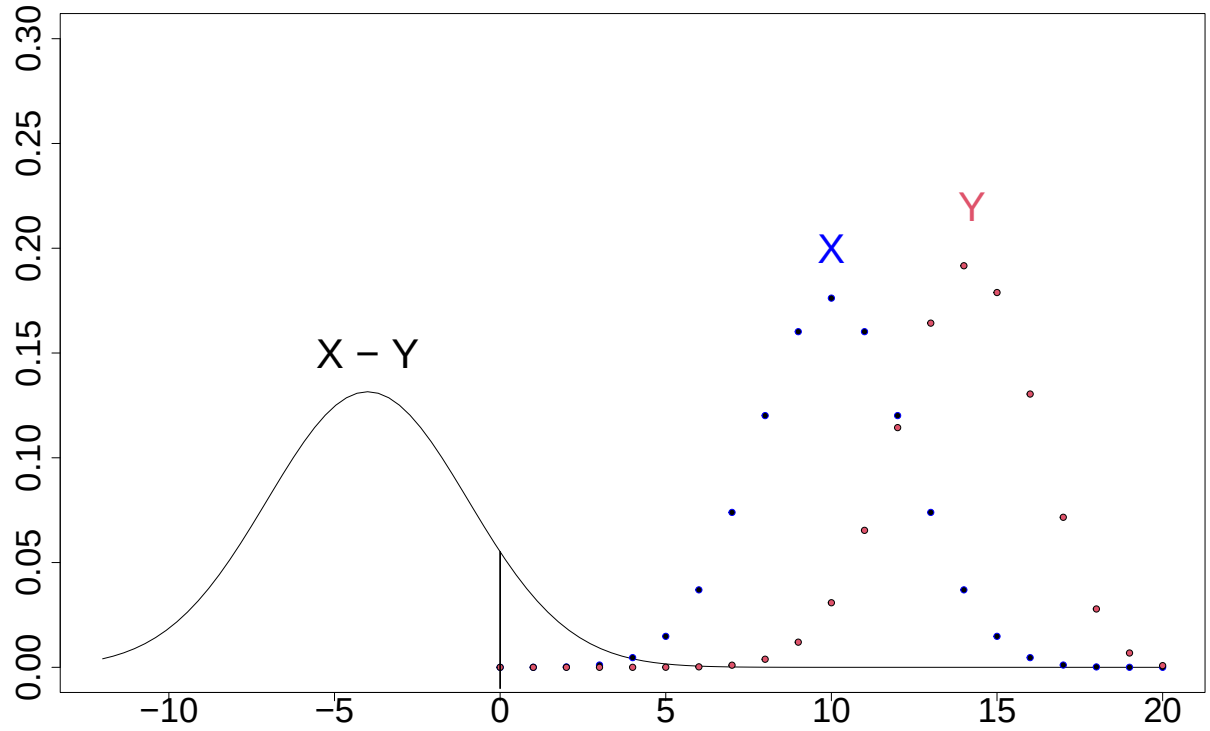
$$W \approx \mathcal{N}(\mu_W = \mu_X - \mu_Y, \sigma_W^2 = \sigma_X^2 + \sigma_Y^2)$$

```
p <- 0.5
q <- 0.7
n <- m <- 20
mW <-  p * n - q * m
varW <- n * p * (1 - p) + m * q * (1 - q)
sdW <- sqrt(varW)
# P(X > Y) = P(W > 0) = ?
PWin_P <- pnorm(0, mean = mW,
                sd = sdW,
                lower.tail = FALSE)
PWin_P
```

This value of the probability is based on the assumptions we did. However, according to our guessing there is low probability that Posillipo win the next match against Pro Recco

Visualise the pmf of X (red) and Y (blue) and the pdf of $W = X - Y$

```
curve(dnorm(x, mW, sdW), xlim = c(-12, 20), ylim = c(0, 0.3),
      xlab = "", ylab = "", cex.axis = 2.5)
points(0 : n, dbinom(0 : n, n, p), pch = 21, bg = 1, col = "blue")
points(0 : m, dbinom(0 : m, m, q), pch = 21, bg = 2)
segments(x0 = 0, y0 = -0.01, x1 = 0, y1 = dnorm(0, mW, sdW), lwd = 2)
text(14.25, 0.22, "Y", cex = 3, col = 2)
text(10, 0.2, "X", cex = 3, col = "blue")
text(-4, 0.15, "X - Y", cex = 3, col = 1)
```



Approximation with CLT: A Note on Continuity correction

When discrete distributions are approximated by continuous distributions, it is a good practice to apply a *correction* (c.c.). In this case:

$$P(X > Y) \approx P(W > 0) \overset{c.c.}{\approx} P(W > 0.5)$$

```
PWin_P_cc <- pnorm(0.5, mean = mW,
                   sd = sdW,
                   lower.tail = FALSE)
PWin_P_cc
```

```
## [1] 0.06895673
```

```
PWin_P # Without c.c.
```

```
## [1] 0.09362452
```

In general

Let X a discrete r.v. that you approximate with a continuos one ($\tilde{X}$), the continuity corrections are

$$P(X > x) \approx P(\tilde{X} > x) \overset{c.c}{\approx} P(\tilde{X} > x + 0.5)$$
$$P(X \geq x) \approx P(\tilde{X} \geq x) \overset{c.c}{\approx} P(\tilde{X} > x - 0.5)$$
$$P(X \leq x) \approx P(\tilde{X} \leq x) \overset{c.c}{\approx} P(\tilde{X} < x + 0.5)$$
$$P(X < x) \approx P(\tilde{X} < x) \overset{c.c}{\approx} P(\tilde{X} < x - 0.5)$$

Potential limits of our probabilistic model

There are several potential limits of our probabilistic model, but one can deserve your attention: **Normal approximation when n is not large**

We are leveraging the CLT for approximating the Binomial distribution with the Gaussian one, but when n is not large the approximation can be poor

The question on "how large should be n to consider good the approximation?" deserve a comment: **It depends on specific conditions** and only sometimes you can follow the rule of thumb that n greater than 30/50 guarantees a good approximation

Let's explore the quality of the approximation using our example

- The Normal approximation is the simplest way to compute $P(W > 0)$, but you can obtain the result differently
- Is the difference $X - Y$ a well-known probability distribution? **No, you must compute the probability law**
- You can obtain the result (with a bit of effort) by using probability calculus

$$P(W = w) = \begin{cases} \sum_{x=0}^{m} \binom{n}{x} p^x (1-p)^{n-x} \binom{m}{x+w} q^{x+w} (1-q)^{m-(x+w)} & \text{if} \quad w \leq 0 \\ \sum_{x=0}^{n} \binom{n}{x+w} p^{x+w} (1-p)^{n-(x+w)} \binom{m}{x} q^x (1-q)^{m-x} & \text{if} \quad w > 0 \end{cases}$$

There is much space to improve this naive R implementation below

```
Px <- dbinom(0 : n, n, p = p)
Py <- dbinom(0: m, m, p = q)
Pw <- rep(NA, 2 * n + 1)

count <- 1; count2 <- 2 * n + 1
for(i in 0 : n){
 idx1 <- 1 : (i + 1)
 idx2 <- (n + 1 - i) : (n + 1)
 if(i == n){
   Pw[count] <- sum(Px[idx1] * Py[idx2])
 } else {
 Pw[count] <- sum(Px[idx1] * Py[idx2])
 Pw[count2] <- sum(Px[idx2] * Py[idx1])
 }
 count <- count + 1
 count2 <- count2 - 1
}
sum(Pw) # check
```

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

Compare approximate solutions (with and without c.c.) with the exact one

Just a slight difference between the exact solution and the approximated one

```
sum(Pw[(n + 2) : length(Pw)]) # P(W>0 )
```

```
## [1] 0.06995932
```

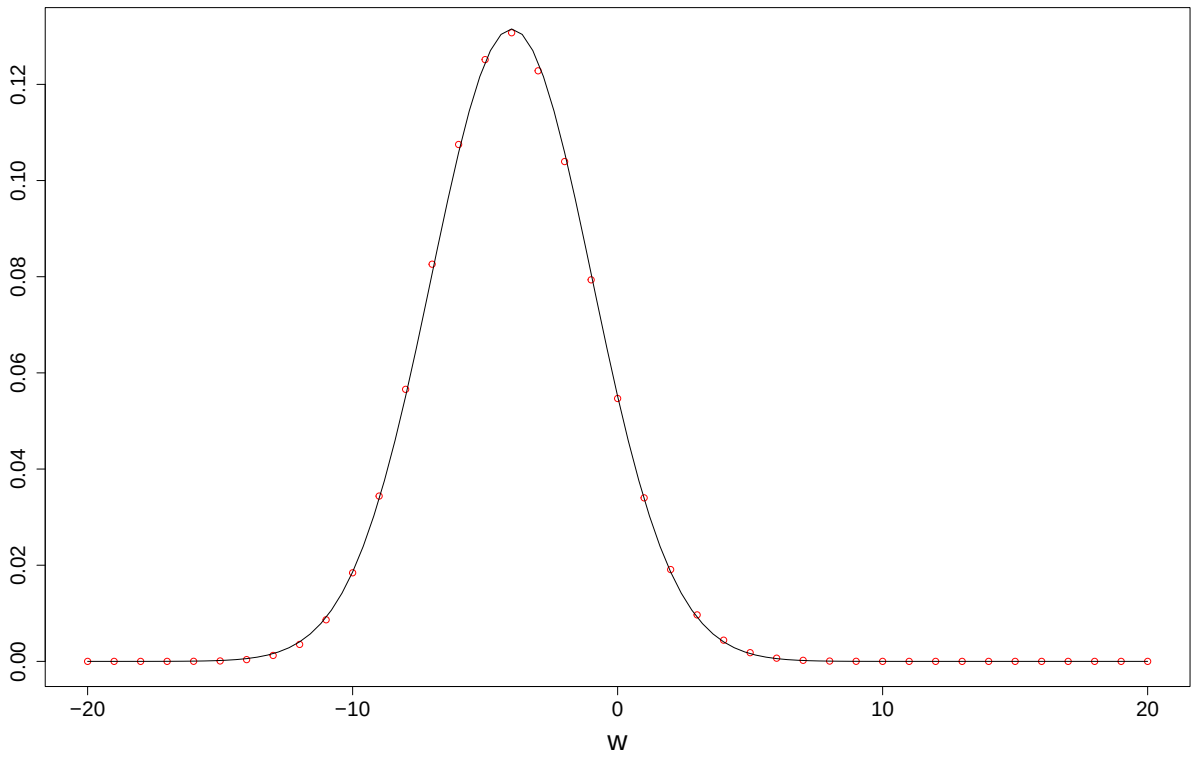
```
PWin_P                        # appr. P(W>0 )
```

```
## [1] 0.09362452
```

```
PWin_P_cc                     # appr. with c.c. P(W>0 )
```

```
## [1] 0.06895673
```

```
plot(-n : m, Pw, xlab = "w", ylab = "", col = "red", cex.axis = 1.5, cex.lab = 2)
curve(dnorm(x, mW, sdW), add = TRUE)
```



Let's explore a case with lower n

- Leaving unchanged the rest, consider $n = m = 5$
- The quality of the approximation (without c.c.) deteriorates
- Note I omitted the computation of the exact solution, you will find only the final result; in the R code you will find a smart way to do it

```
n <- m <- 5
mW <- n * p - m * q
sdW <- sqrt(n * p * (1-p) + m * q * (1-q))
PWin_P <- pnorm(0, mW, sdW, lower.tail = FALSE)
PWin_P_cc <- pnorm(0 + 0.5, mW, sdW, lower.tail = FALSE)
```

```
sum(Pw[(n + 2) : length(Pw)]) # P(W>0)
```

```
## [1] 0.1606744
```

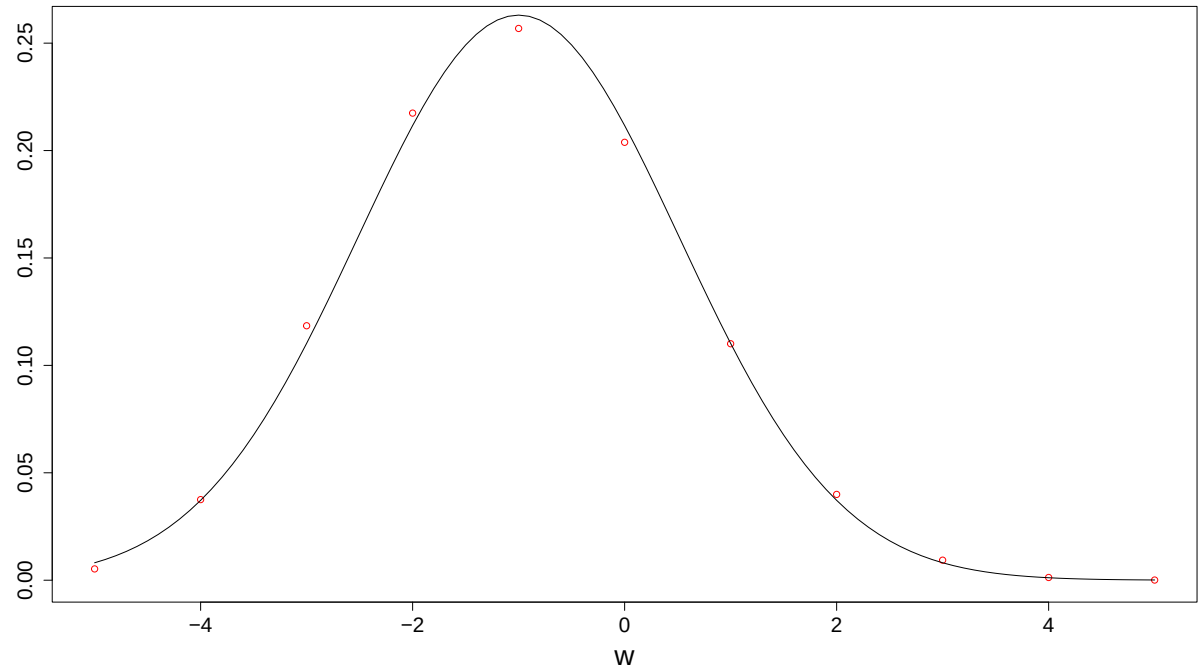
```
PWin_P # P(W>0) approx.
```

```
## [1] 0.2548257
```

```
PWin_P_cc # P(W>0) approx. c.c.
```

```
## [1] 0.1613143
```

```
plot(-n : m, Pw, xlab = "w", ylab = "", col = "red", cex.axis = 1.5, cex.lab = 2)
curve(dnorm(x, mW, sdW), add = TRUE)
```



Let's explore a case with larger n

- Leaving unchanged the rest, consider $n = m = 100$
- The quality of the approximation (without c.c.) improves remarkably
- Note I omitted the computation of the exact solution, you will find only the final result; in the R code you will find a smart way to do it

```
n <- m <- 100
mW <- n * p - m * q
sdW <- sqrt(n * p * (1 - p) + m * q * (1 - q))
PWin_P <- pnorm(0, mW, sdW, lower.tail = FALSE)
PWin_P_cc <- pnorm(0 + 0.5, mW, sdW, lower.tail = FALSE)
```

```
sum(Pw[(n + 2) : length(Pw)]) # P(W>0 )
```

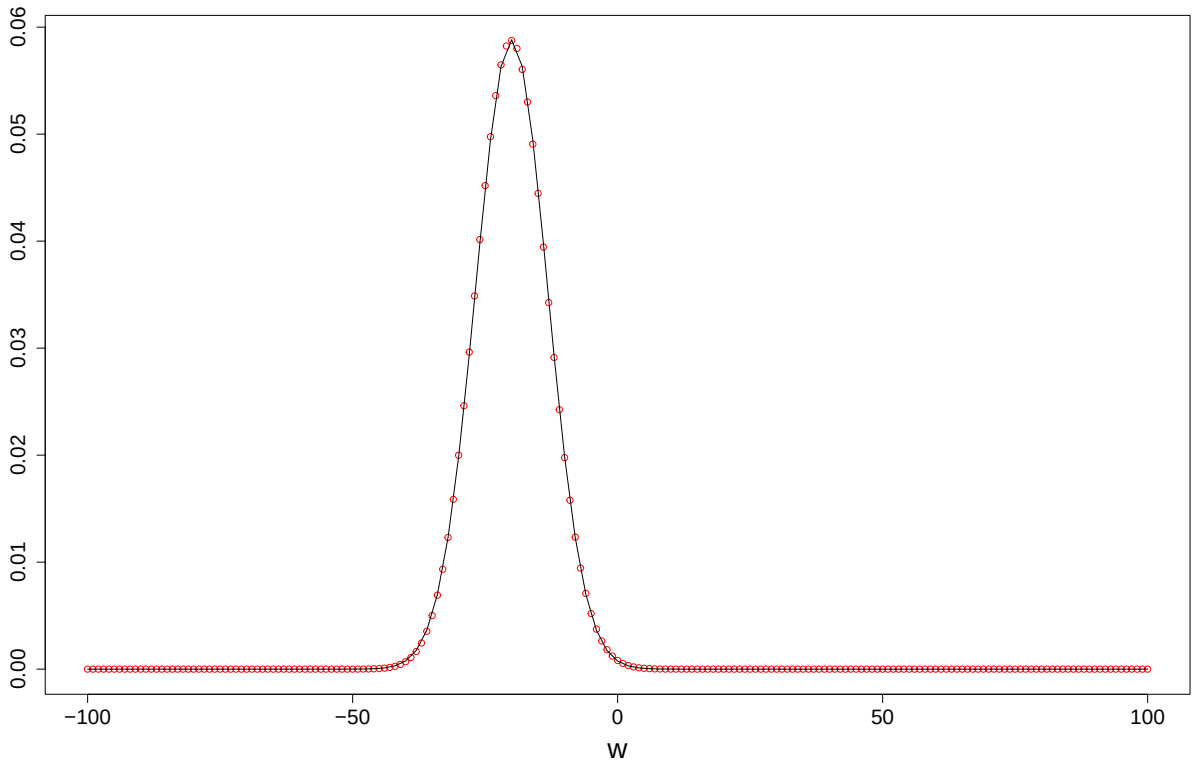
```
## [1] 0.001365914
```

```
PWin_P
```

```
## [1] 0.00159485
```

```
PWin_P_cc
```

```
## [1] 0.001253232
```



Extra: An alternative approach to the waterpolo match example?

Rather than specifying in advance the total number of unknown shots and the converting shots probabilities, i.e. 4 parameters, one could be tempted to use a more flexible distribution accounting just for the *scoring intensity*, regardless of the number of shots

For this purpose, the Poisson distribution seems suitable. We may assume two independent Poisson distributions for the number of goals of the upcoming match: $X \sim \mathcal{P}(\lambda),\ Y \sim \mathcal{P}(\mu)$

At this stage, we need to specify the rates, for instance upon our own knowledge about waterpolo goal abilities we fix $\lambda = 5, \mu = 7$

- Leveraging the Poisson assumption on the number of goals scored, how can we estimate now the winning probability for Posillipo,

$$P(X > Y) = P(X - Y > 0) = ?$$

- Obtain $P(X - Y > 0)$ using the exact and approximate solution and compare them
- Hint: We may use the following probability result: $Z = X - Y \sim \mathcal{PD}(\lambda - \mu, \lambda + \mu)$, where $\mathcal{PD}$ stands for the **Poisson difference** distribution, also known as **Skellam** distribution, with mean $\lambda - \mu$ and variance $\lambda + \mu$ (use the **skellam** R package)

Learn more: Skellam distribution

Multivariate Normal Distribution

Recall

- $\mathbf{X} = (X_1, \ldots, X_d)^\top$ (d-dimensional random vectors), $\mathbf{X} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- Mean vector: $\mu = (\mu_1, \ldots, \mu_d)^\top$, where $\mu_j = \mathbb{E}(X_j)$, $j = 1, \ldots, d$
- Covariance matrix: $\boldsymbol{\Sigma}$ is a $d \times d$ matrix whose entries Σ_{jk} represents

$$\Sigma_{jj} = \sigma_j^2 = \mathbb{V}(X_j), \quad \Sigma_{jk} = \Sigma_{kj} = \sigma_{jk} = \operatorname{cov}(X_j, X_k)$$

- Density function

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Bivariate Normal Distribution

- Multivariate normal with $d = 2$
- $\mathbf{X} = \begin{pmatrix} \mathbf{X_1} \\ \mathbf{X_2} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix} \right)$
- Determinant of $\boldsymbol{\Sigma}$ (ρ is the correlation coefficient):

$$|\boldsymbol{\Sigma}| = \sigma_1^2\sigma_2^2 - (\sigma_{12})^2 = \sigma_1^2\sigma_2^2\left(1 - \frac{(\sigma_{12})^2}{\sigma_1^2\sigma_2^2}\right) = \sigma_1^2\sigma_2^2(1-\rho^2)$$

- Precision matrix: $\boldsymbol{\Sigma}^{-1} = \frac{1}{\sigma_1^2\sigma_2^2(1-\rho^2)} \begin{pmatrix} \sigma_2^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_1^2 \end{pmatrix}$
- Density function

$$f_{\mathbf{X}}(\mathbf{x}) = \left((2\pi)\sigma_1\sigma_2\sqrt{1-\rho^2}\right)^{-1} e^{-\frac{1}{2(1-\rho^2)}\left(\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2 - 2\rho\frac{(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2}\right)}$$

Bivariate Normal Distribution - Special case

Function for computing the value of the density function for a bivariate Gaussian in (x_1, x_2) having mean vector components equal to zero and unity variances, that is

$$f_{(X_1,X_2)}(x_1, x_2) = \frac{1}{(2\pi)\sqrt{1-\rho^2}} e^{-\frac{1}{2(1-\rho^2)}(x_1^2 + x_2^2 - 2\rho x_1 x_2)}$$

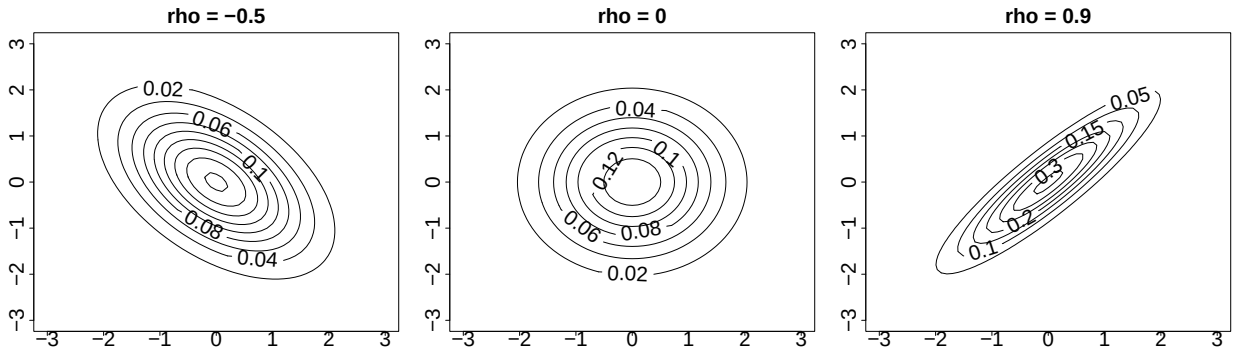
```r
NBiv <- function(x, y, rho){
  den <- (2 * pi * sqrt(1 - rho^2))
  num <- exp(-0.5 * (1 - rho^2)^(-1) * (x^2 + y^2 - 2 * rho * x * y))
  return(num/den)
}
```

Visualise the bivariate normal distribution in a 3D plots

```r
xx <- yy <- seq(-3, 3, 0.1)
z <- outer(xx, yy, NBiv, rho = 0.5)
par(mfrow = c(1, 2))
persp(xx, yy, z, xlab = "x", ylab = "y", zlab = "f(x,y)", cex.lab = 3)
persp(xx, yy, z, theta = 30, phi = 30, xlab = "x", ylab = "y",
      zlab = "f(x,y)", cex.lab = 3)
```

Visualise the bivariate normal distribution in a 2D plots

```r
rho <- c(-0.5, 0, 0.9)
z <- list()
for(j in 1 : 3) z[[j]] <- outer(xx, yy, NBiv, rho = rho[j])
par(mfrow = c(1, 3))
for(j in 1 : 3) contour(xx, yy, z[[j]], main = paste0("rho = ", rho[j]),
                        cex.main = 3, cex.axis = 3, labcex = 2)
```



Some properties of $\mathbf{X} = (X_1, X_2)^\top \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Marginal distributions

$$f_{X_j}(x_j) = \int_{-\infty}^{+\infty} f(x_j, x_k) dx_k$$

So $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$

Conditional distributions

$$f_{X_j|X_k}(x_j|x_k) = \frac{f_{(X_j,X_k)}(x_j, x_k)}{f_{X_k}(x_k)}$$

So (here there was a typo in the pre-LAB version)

- $X_1|X_2 = x_2 \sim \mathcal{N}(\mu_1 + \rho\frac{\sigma_1}{\sigma_2}(x_2 - \mu_2), \sigma_1^2(1-\rho^2))$
- $X_2|X_1 = x_1 \sim \mathcal{N}(\mu_2 + \rho\frac{\sigma_2}{\sigma_1}(x_1 - \mu_1), \sigma_2^2(1-\rho^2))$

Random generation

Leverage $(\tilde{X}_1, \tilde{X}_2)^\top = \boldsymbol{\Sigma}^{-1/2}(\mathbf{X} - \boldsymbol{\mu}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

mvtnorm R package

- The function we built above is only able to handle a special case
- Here we use the capabilities of **mvtnorm** (install/load it)

```
library(mvtnorm)
```

```
## Warning: il pacchetto 'mvtnorm' è stato creato con R versione 4.3.3
```

Example - Anthropometric data

- Anthropometric data analysis represents an important stumbling block of statistics analysis (around 19th and early 20th Centuries, Quetelet, Galton, Pearson, etc. investigates biometric measurements)
- Here, we consider two measurements, the weight (in kg) and the height (in cm) of an adult. We are simplifying a bit the phenomenon (both are dependent on the gender, geographical characteristics, and so on)
- Let X_1 and X_2 the r.v. representing the height and the weight measurements. We assume the random vector (X_1, X_2) follows a bivariate normal distribution
- Consider: $\mu_1 = 176, \mu_2 = 85.5, \sigma_1 = 7, \sigma_2 = 14.2, \rho = 0.47$

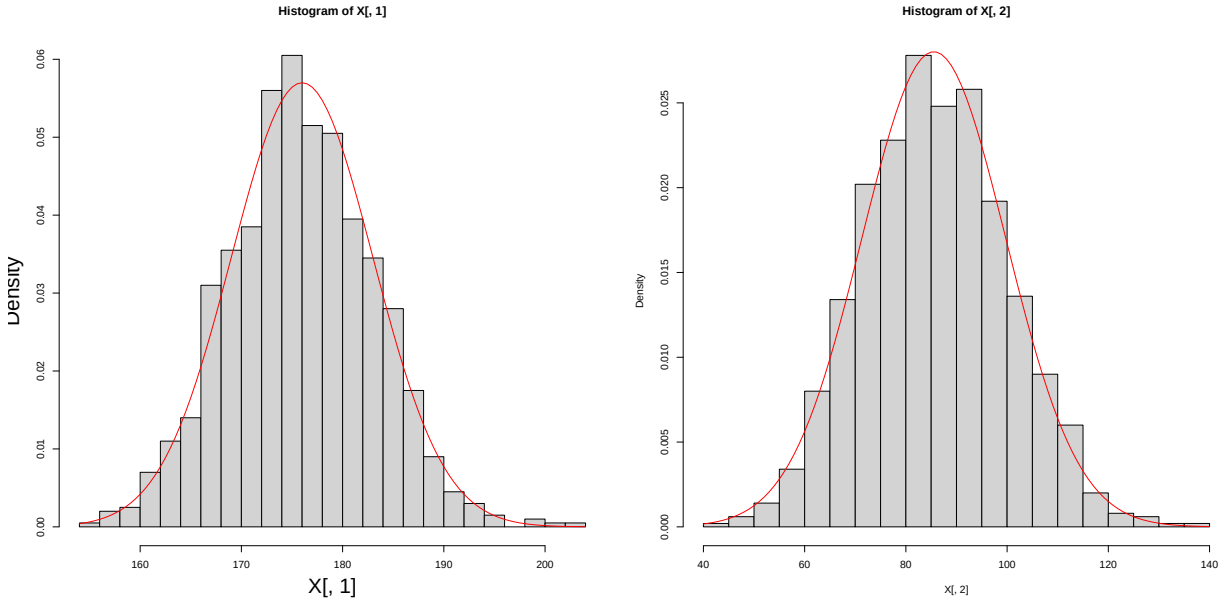
```
mu_h <- 176
mu_w <- 85.52
sd_h <- 7
sd_w <- 14.24
rho <- 0.47
```

rmvnorm requires the mean vector and the covariance matrix

```
cov <- rho * sd_h * sd_w
Sigma <- matrix(c(sd_h^2, cov, cov, sd_w^2), 2,2, byrow=T)
set.seed(13)
X <- rmvnorm(1000, c(mu_h, mu_w), Sigma)
```

Graphical check for marginals

```
par(mfrow = c(1,2))
hist(X[,1], prob = TRUE, breaks = 30, cex.lab = 2)
curve(dnorm(x, mu_h, sd_h), col = "red", add=TRUE)
hist(X[,2], prob = TRUE, breaks = 30)
curve(dnorm(x, mu_w, sd_w), col = "red", add=TRUE)
```



Note: the variance of the conditional distribution is lower of the variance of the marginal distribution

```
x1.fix <- 180
muw_c <- mu_w + rho * (sd_w/sd_h) * (x1.fix - mu_h)
sdw_c <- sqrt(sd_w^2*(1-rho^2))
c(sd_w, sdw_c)
```

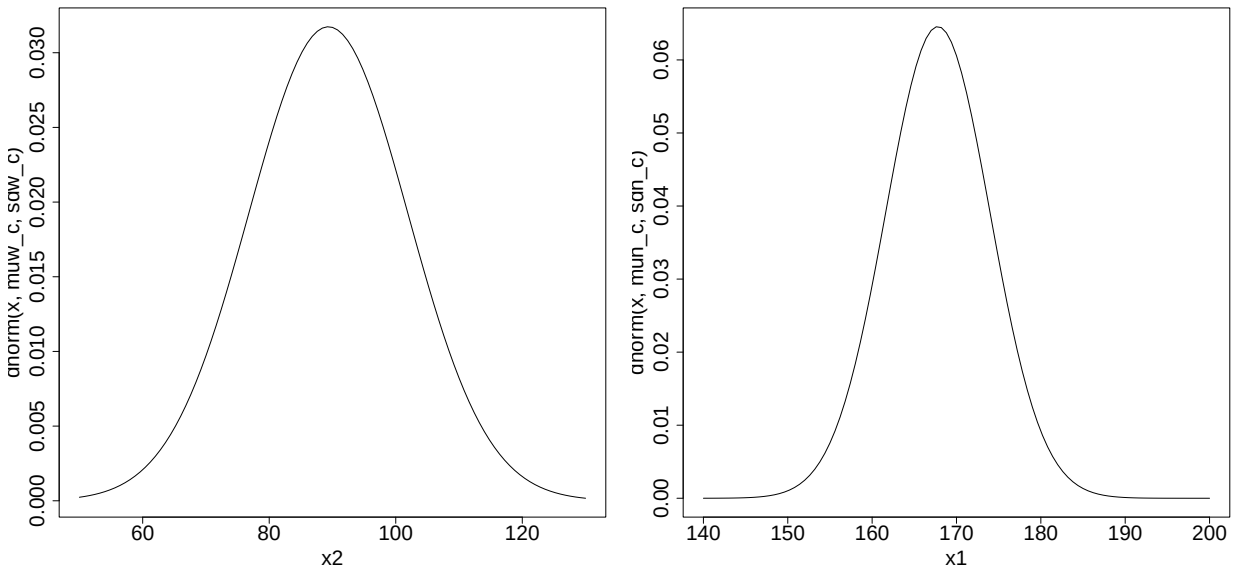
```
## [1] 14.24000 12.56917
```

```
x2.fix <- 50
muh_c <- mu_h + rho * (sd_h/sd_w) * (x2.fix - mu_w)
sdh_c <- sqrt(sd_h^2*(1-rho^2))
c(sd_h, sdh_c)
```

```
## [1] 7.000000 6.178665
```

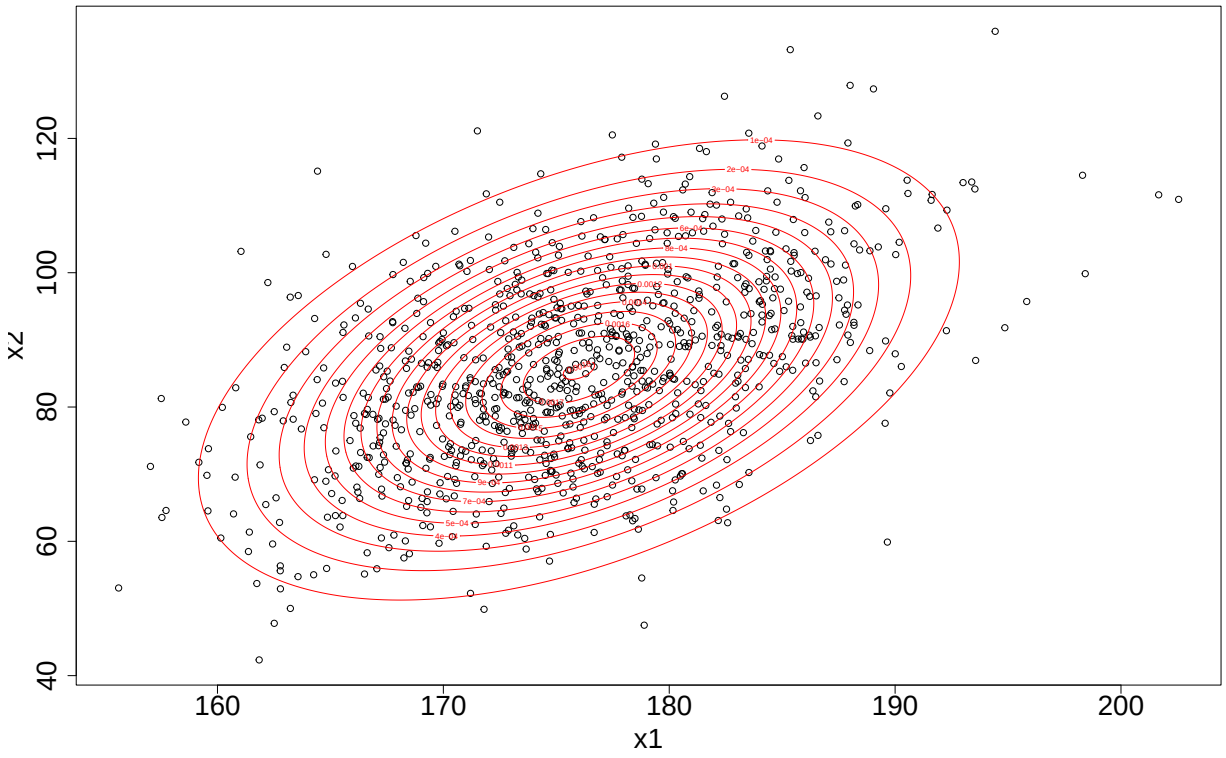
Graphical representation of the conditional distributions

```
par(mfrow = c(1,2))
curve(dnorm(x, muw_c, sdw_c), from = 50, to = 130,
      cex.lab = 2, cex.axis = 2, xlab = "x2")
curve(dnorm(x, muh_c, sdh_c), from = 140, to = 200,
      cex.lab = 2, cex.axis = 2, xlab = "x1")
```



Scatterplot and contour plot

```
xx <- seq(min(X[, 1]), max(X[, 1]), length.out = 500)
yy <- seq(min(X[, 2]), max(X[, 1]), length.out = 500)
zz <- outer(X = xx, Y = yy, FUN = function(x,y)
  dmvnorm(cbind(x,y), mean = c(mu_h, mu_w), sigma = Sigma))
plot(X[,1], X[,2], xlab = "x1", ylab = "x2", cex.axis = 2, cex.lab = 2)
contour(xx, yy, zz, add = TRUE, col = "red", nlevels = 20)
```



Extra: Inverse sampling metdod

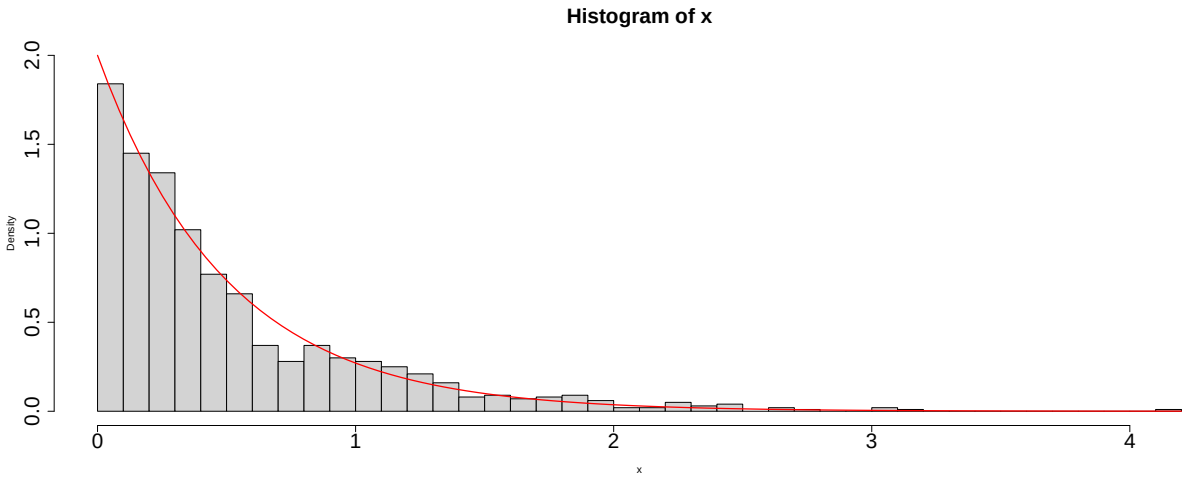
Inverse sampling method: Generate a value from a distribution, with known and easily to derive quantile function, starting from the uniform generator.

Generation from exponential distribution

- Let $U \sim U(0,1)$ (continuous uniform in the interval (0,1))
- The random variable X: $X = -\frac{\log(U)}{\theta} \sim \mathcal{E}(\theta)$

Fix $\theta = 2$; generate 1000 samples; compare (graphically) the simulated distribution and the target distribution

```
set.seed(13)
theta <- 2
R <- 1000
x <- -log(runif(R))/theta
hist(x, prob = T, breaks = 30, cex.main = 2, cex.axis = 2, ylim = c(0,2))
curve(dexp(x, theta), from = 1e-9, add = T, lwd = 2, col = "red")
```



Extra exercises

Let $U_i \sim U(0,1)$, $i = 1, \ldots, n$, be iid rv. Fix $\theta = 2$

Exercise 1: Minimum of exponential r.v. The r.v. $Z = \min\{X_1, X_2, \ldots, X_n\}$, where $X_i = -\log(U_i)/\theta$, follows an exponential distribution with parameter $n\theta$ ($\mathcal{E}(n\theta)$)

Exercise 2: Sum of exponential r.v. The distribution of $Y = \sum_{i=1}^{n} X_i$ is a Gamma distribution with shape parameter equal to n and rate parameter θ ($\mathcal{G}a(n, \theta)$)

Learn more: Relationships among univariate probability distributions