Statistical methods

Laboratory 1

N. Torelli, G. Di Credico, V. Gioia

vincenzo.gioia@units.it

Office hour: Friday, 17.00 - 18.30

09/10/2022

Contents

Central Limit Theorem (CLT)

Let $X_1, X_2, \ldots, X_n$, be a sequence of independent and identically distributed (iid) random variables (rv) from a distribution with mean μ and finite variance σ^2.

For large n, the sample average

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

where $\dot{\sim}$ indicates convergence in distribution. Equivalently

$$S_n = \sum_{i=1}^{n} X_i \dot{\sim} \mathcal{N}(n\mu, n\sigma^2)$$

The CLT supports the normal approximation to the distribution of a rv that can be viewed as the sum of other rv.

Approximation with CLT: application

The approximation above is useful in statistics for computing some quantities. For instance, let X and Y be two independent Binomial rv, such that $X \sim \text{Bin}(n, p)$ and $Y \sim \text{Bin}(m, q)$.

If we are interested in computing the probability $P(X > Y)$ the Normal approximation is the simplest way to do it. We can approximate:

$$X \approx \mathcal{N}(np, np(1-p)), \qquad Y \approx \mathcal{N}(mq, mq(1-q)).$$

Then, by using a well known probability result, the difference $W = X - Y$ of two independent normal distributions with means μ_X, μ_Y and variances σ_X^2, σ_Y^2, respectively, is **still a normal distribution** with mean $\mu_W = \mu_X - \mu_Y$ and variance $\sigma_W^2 = \sigma_X^2 + \sigma_Y^2$.

In such a case,

$$\mu_W = \mu_X - \mu_Y = np - mq, \qquad \sigma_W^2 = \sigma_X^2 + \sigma_Y^2 = np(1-p) + mq(1-q).$$

Approximation with CLT: real application with waterpolo goals

Tomorrow two professional Italian waterpolo teams, Posillipo and Pro Recco, compete against each other. Let X and Y be the random *goals scored* by Posillipo and Pro Recco, respectively.

We assume that X, Y follow two independent Binomial distributions. Thus, X and Y represent the number of shots converted in goal on the total number of shots n, m made by Posillipo and Pro Recco, with probabilities p and q, respectively.

Before the match, the number of shots is *unknown*. In what follows, we adopt a simplification, and we treat the quantities p, q, m, n as *known*, for instance fixing them upon historical experience: $p = 0.5, q = 0.7, n = 20, m = 20$.

We want to investigate the Posillipo probability of winning the next match against Pro Recco, that is

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

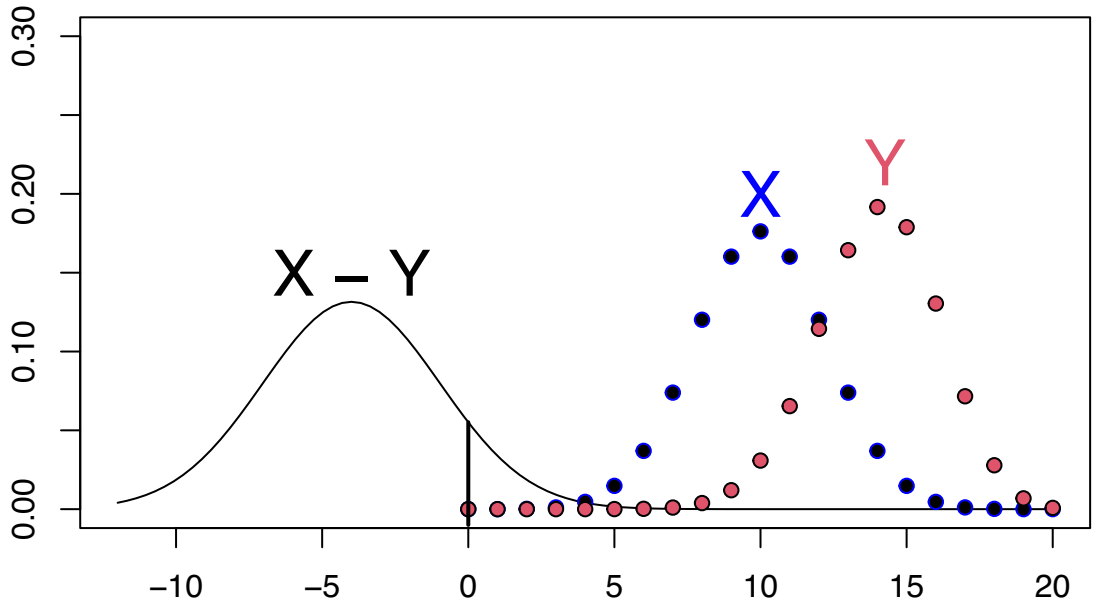
So, let W be the r.v., such that $W = X - Y$. We could compute the law of this rv but using the Normal approximation, $W \approx \mathcal{N}(\mu_W = \mu_X - \mu_Y, \sigma^2_W = \sigma^2_X + \sigma^2_Y)$, we can esily compute such a probability of interest.

```
p <- 0.5
q <- 0.7
n <- m <- 20
mW <-  p * n - q * m
sdW <- sqrt(n * p * (1 - p) + m * q * (1 - q))
# Probability that  W = X-Y > 0 (Posillipo win the match)
PWin_P <- pnorm(0, mean = mW, sd = sdW, lower.tail = FALSE)
PWin_P
```

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

Here, we show the probability mass functions of X and Y and the probability density function of $W = X - Y$.

```
# pdf of W
curve(dnorm(x, mW, sdW), xlim = c(-12, 20), ylim = c(0, 0.3),
      xlab = "", ylab = "", cex.lab = 1.25)

# pmf of X and Y
points(0 : 20, dbinom(0 : n, n, p), pch = 21, bg = 1, col = "blue")
points(0 : 20, dbinom(0 : m, m, q), pch = 21, bg = 2)

segments(0, -0.01,               #(x_0, y_0)
         0, dnorm(0, mW, sdW),   #(x_1, y_1)
         lwd = 2)

text(14.25, 0.22, "Y", cex = 2, col = 2)
text(10, 0.2, "X", cex = 2, col = "blue")
text(-4, 0.15, "X - Y", cex = 2, col = 1)
```



Note: when discrete distributions are approximated by continuous distributions, it is a good practice to apply a *continuity correction* (c.c.). In this case: $P(X > Y) \approx P(W > 0) \overset{c.c.}{\approx} P(W > 0.5)$.

```
# Probability that Posillipo win the match,
# taking into account the continuity correction
PWin_P_cc <-  pnorm(0.5, mean = mW, sd = sdW, lower.tail = FALSE)
PWin_P_cc
```

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

Waterpolo goals: what if we use a Poisson distribution?

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, in order to make the same prediction as before, we need to specify the rates, for instance upon our own knowledge about waterpolo abilities: $\lambda = 5, \mu = 7$.

How can we estimate now the winning probability for Posillipo, $P(X > Y) = P(X - Y > 0)$?

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 $\lambda + \mu$. (see e.g. https://www.jstor.org/stable/2981372)

```
# Probability that Posillipo win the match via Skellam distribution
library(skellam)
lambda <- 5
mu <- 7

PWin_P_sk <- pskellam(0, lambda, mu, lower.tail = FALSE)
PWin_P_sk
```

```
## [1] 0.2336875
```

It's your turn

With the same settings above ($\lambda = 5, \mu = 7$), try to use the normal approximation of the Poisson distribution, that is if $X \sim Poisson(\lambda)$, for large λ, it holds $X \approx \mathcal{N}(\mu = \lambda, \sigma^{\in} = \lambda)$, to compute the winning probability for Posillipo and compare the results with those obtained by using the Skellam distribution. Then, plot the probability mass functions of X, Y and Z. Try to overlap also the normal approximation that you obtained.

```
# Probability that Posillipo win the match, by using
# the normal approximation to the Poisson distribution
mW <- lambda - mu
sdW <- sqrt(lambda + mu)

# Without continuity correction
PWin_P_napp <- pnorm(0, mW , sdW, lower.tail = FALSE)
PWin_P_napp
```

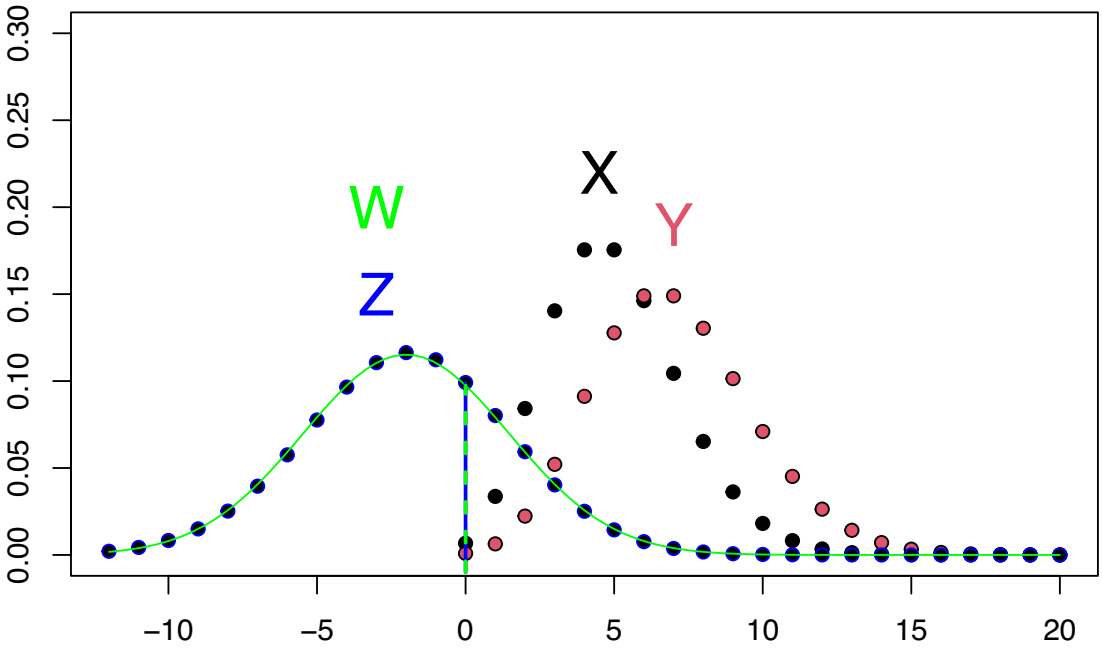
```
## [1] 0.2818514
```

```
# With continuity correction
PWin_P_napp_cc <- pnorm(0.5, mW, sdW, lower.tail = FALSE)
PWin_P_napp_cc
```

```
## [1] 0.2352432
```

```
# pmf of X
plot(0 : 20, dpois(0 : 20, lambda), xlim = c(-12, 20),
     ylim=c(0, 0.3), ylab = "", xlab = "",
     pch = 21, bg = 1, cex.lab = 1.25)
# pmf of Y
points(0 : 20, dpois(0 : 20, mu), pch = 21, bg = 2)
# pmf of Z
points(-12 : 20, dskellam(-12 : 20, lambda, mu), pch = 21,
       bg = 1, col = "blue" )
```

```r
# pdf of W
curve(dnorm(x, mW, sdW), add = TRUE, col = "green")

segments(0, -0.01,                      #(x_0, y_0)
         0, dskellam(0, lambda, mu),    #(x_1, y_1)
         lwd = 2, col = "blue" )
segments(0, -0.01,
         0, dnorm(0, mW, sdW),
         lwd = 2, col = "green", lty = "dashed")

text(4.5, 0.22, "X", cex = 2, col = 1)
text(7, 0.19, "Y", cex = 2, col = 2)
text(-3.0, 0.15, "Z", cex = 2, col = "blue")
text(-3.0, 0.20, "W", cex = 2, col = "green")
```



Law of large numbers

Let $X_1, \ldots, X_n, \ldots$, be a sequence of independent and identically distributed r.v., such that $E[X_i] = \mu$ and $V[X_i] = \sigma^2 < +\infty$. Then, given the sample mean

$$\bar{X}_n = \frac{1}{n} \sum_{i=1} X_i,$$

we have that, for $\epsilon > 0$,

$$\lim_{n \to \infty} P(|\bar{X}_n - \mu| \geq \epsilon) = 0.$$

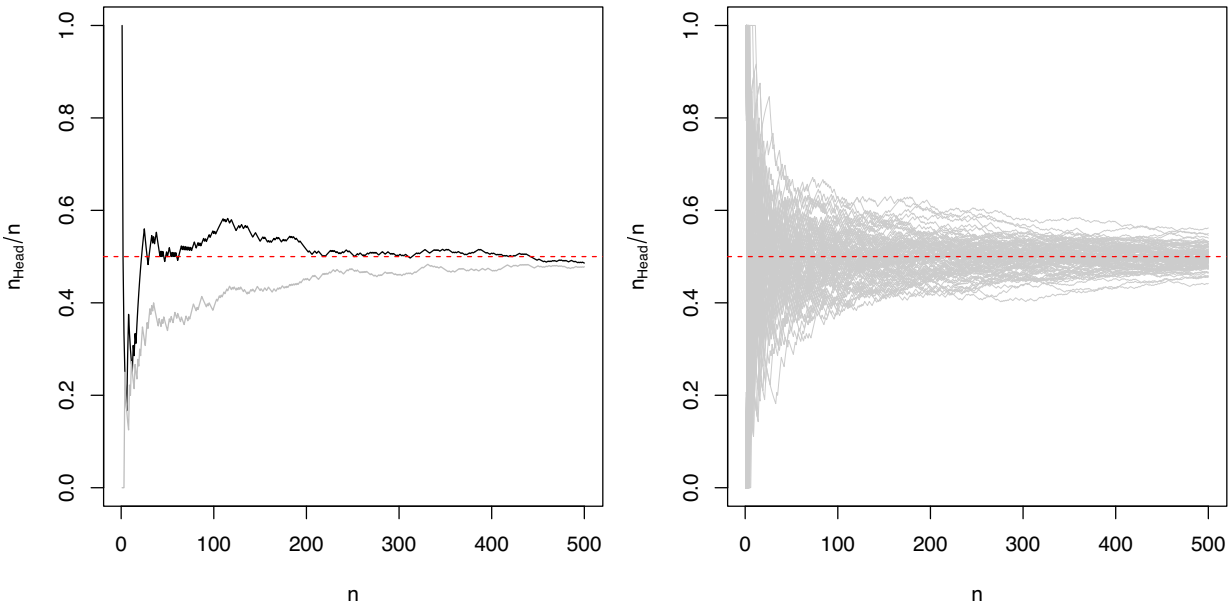
In practice, suppose tossing a coin n times and let k the number of heads obtained. Then k/n is the proportion of heads obtained in n tosses. If the coin is fair then we can guess that the proportion is not too far from 0.5 (unlikely it should be exactly 0.5). Although it could happen than, due to the chance, we can observe anomalies represented by large or small number of heads, by increasing the number of tosses such a strange behavior disappear.

Then, let $X_1, X_2, \ldots, X_n$, be a sequence of iid rv such that $X_i \sim \text{Be}(1/2)$. The number of heads obtained in n tosses is $\sum_{i=1}^n X_i$, and the proportion of heads $\bar{X}_n = (\sum_{i=1}^n X_i)/n$.

```
LLN_ex <- function(n, p = 0.5, seed = 1234){
  par(mfrow = c(1, 2), mar = c(4, 4, 1, 1))
  set.seed(seed)
  #Each trajectory is a realization of a sequence of rv
  # First trajectory
  x <- sample(c(0, 1), n, prob = c(1 - p, p), replace = T)
  plot(1 : n, cumsum(x)/(1 : n), type = "l", ylim = c(0, 1),
       ylab = expression(n[Head]/n), xlab = "n")
  abline(h = .5, lty = 2, col = "red")

  # Second trajectory
  x <- sample(c(0, 1), n, prob = c(1 - p, p), replace = T)
  points(1 : n, cumsum(x)/(1 : n), type = "l", col = "grey")

  plot(1 : n, cumsum(x)/(1 : n), type = "l", ylim = c(0, 1),
     ylab = expression(n[Head]/n), xlab="n", col = "gray80")

  # Instead, considering 100 trajectories
  for(i in 1 : 100){
    x <- sample(c(0, 1), n, prob = c(1 - p, p), replace = T)
    points(1 : n, cumsum(x)/(1 : n), type = "l", col = "gray80", lwd = .5)
  }
  abline(h = .5, lty = 2, col = "red")
}
```

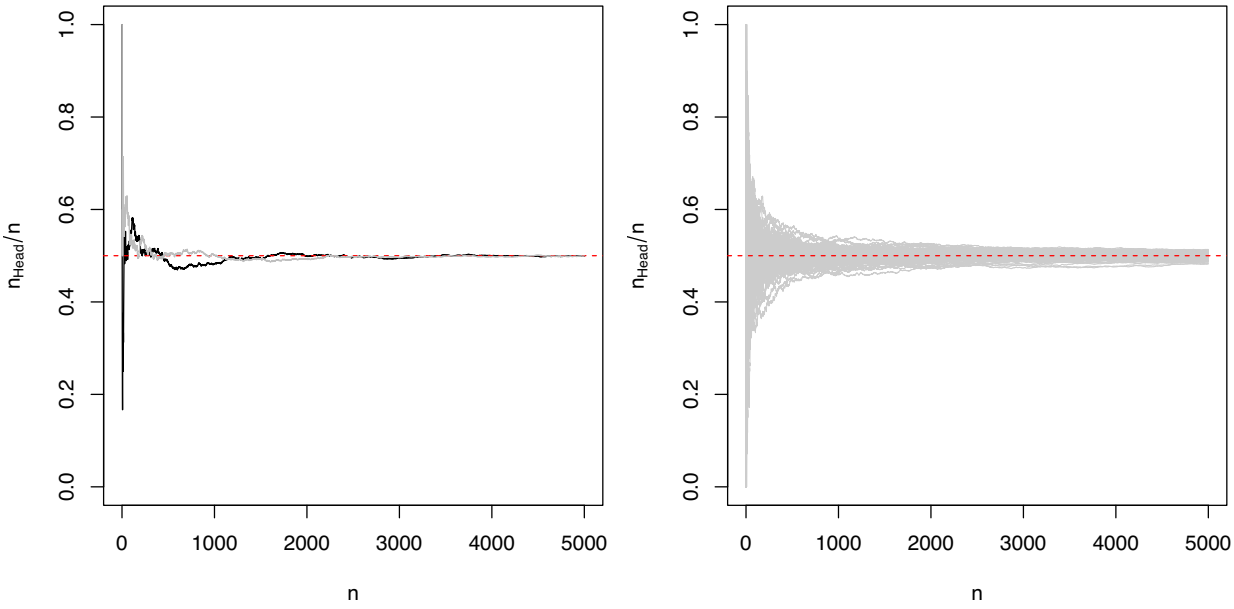
Consider $n = 500$

```
n <- 500
LLN_ex(n)
```



Now increase $n = 5000$

```
n <- 5000
LLN_ex(n)
```



Monte Carlo simulation

Simulation is a well-known technique designed to approximate a process and to retrieve general results by assuming to observe the process several times. We rely on the so called **Monte Carlo** simulation, a wide class of simulation procedures based on sampling independent and identically distributed values from a process —precisely, from the underlying presumibly true probability distribution of the process— and computing numerical outputs. The steps of a Monte Carlo simulation are:

- Generate n independent and identically distributed values from a process;
- Compute a summary for this sample, a statistic;
- Repeat the steps above R times and obtain a sample distribution for the statistic.

Distribution of the sample mean

Suppose that $Y_1, \ldots, Y_n$, is sequence of iid rv from

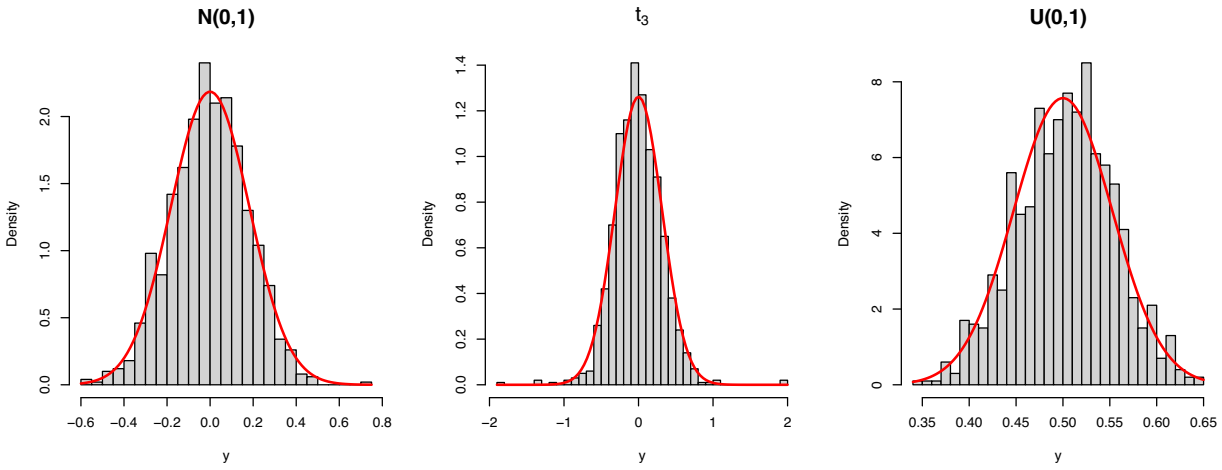
- Case 1: $\mathcal{N}(0, 1)$ (i.e. standard normal);
- Case 2: t_3 (i.e. t-student with 3 degrees of freedom);
- Case 3: $\mathrm{U}(0, 1)$ (i.e. standard continuous uniform).

Which is the distribution of the sample mean $\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$?

Before answering, let's investigate this issue via simulation, with number of simulations $R = 1000$ and sample size $n = 30$.

```
set.seed(1234)
R <- 1000
n <- 30

#generate the sample of size n (for the 3 distributions) R times
samples <- array(0, c(3, n, R))
for(i in 1 : R){
  samples[1, ,i] <- rnorm(n, 0, 1)
  samples[2, ,i] <- rt(n, df = 3)
  samples[3, ,i] <- runif(n, 0, 1)
}

#compute the sample mean and the sample variance
samples_stat <- array(0, c(3, 2, R))
for(i in 1 : R) {
  samples_stat[, 1, i] <- apply(samples[, , i], 1, mean)
  samples_stat[, 2, i] <- apply(samples[, , i], 1, var)
}
```

```r
#visualize the results
par (mfrow=c(1,3))

hist(samples_stat[1, 1, ], nclass = 30, probability = TRUE,
     xlab="y", main= "N(0,1)", cex.main = 1.5)
curve(dnorm(x, 0, sqrt(1/n)), add = TRUE, col = "red", lwd = 2)

hist(samples_stat[2, 1, ], nclass = 30, probability = TRUE,
     xlab = "y", main = expression(t[3]), cex.main = 1.5)
curve(dnorm(x, 0, sqrt((3/((3 - 2) * n)))), add = TRUE, col = "red", lwd = 2)

hist(samples_stat[3, 1, ], nclass = 30, probability = TRUE,
     xlab = "y", main = "U(0,1)", cex.main = 1.5)
curve(dnorm(x, 1/2, sqrt(1/(12 * n))), add = TRUE, col = "red", lwd = 2)
```



In the Gaussian case the **sample mean** $\overline{Y}$ for iid values still follows a normal distribution, precisely $\mathcal{N}(\mu, \sigma^2/n)$.

For other distributions, this result holds only asymptotically, when $n \to \infty$, due to the CLT theorem. When the sampling does not come from a Normal distribution $\overline{Y} \dot{\sim} \mathcal{N}(\mu, \sigma^2/n)$.

Obviously, overlapping the Gaussian density over the histograms, require obtaining the parameters of the appropriate Gaussian distribution. Thus, for each distribution we must obtain $E[\bar{Y}_n]$ and $Var(\bar{Y}_n)$.

Distribution of the sample variance under the Gaussian case

We know that $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2$ is an unbiased estimator for the variance (an estimator is said to be **unbiased** iff $E(\widehat{\theta}) = \theta$), and this is exactly the output provided by R through the function var() applied to a sample vector.

In the example above, we have already stored the variance. If the assumed true generating model above is normal (case 1), we know that the distribution of the sample variance is proportional to a χ^2 distribution with $n-1$ degrees of freedom:

$$\frac{(n-1)S^2}{\sigma^2} \sim \chi^2_{n-1}$$

Note
Let X a r.v. with pdf $f_X(x)$, then $Y = g(X)$, with $g(\cdot)$ an invertible function, has pdf

$$f_Y(y) = f_X(g^{-1}(y))\left|\frac{d}{dy}g^{-1}(y)\right|$$

So, in our case let $X = \frac{(n-1)S^2}{\sigma^2} \sim \chi^2_{n-1}$, then $S^2 = Y = \frac{\sigma^2}{n-1}X$ is such that

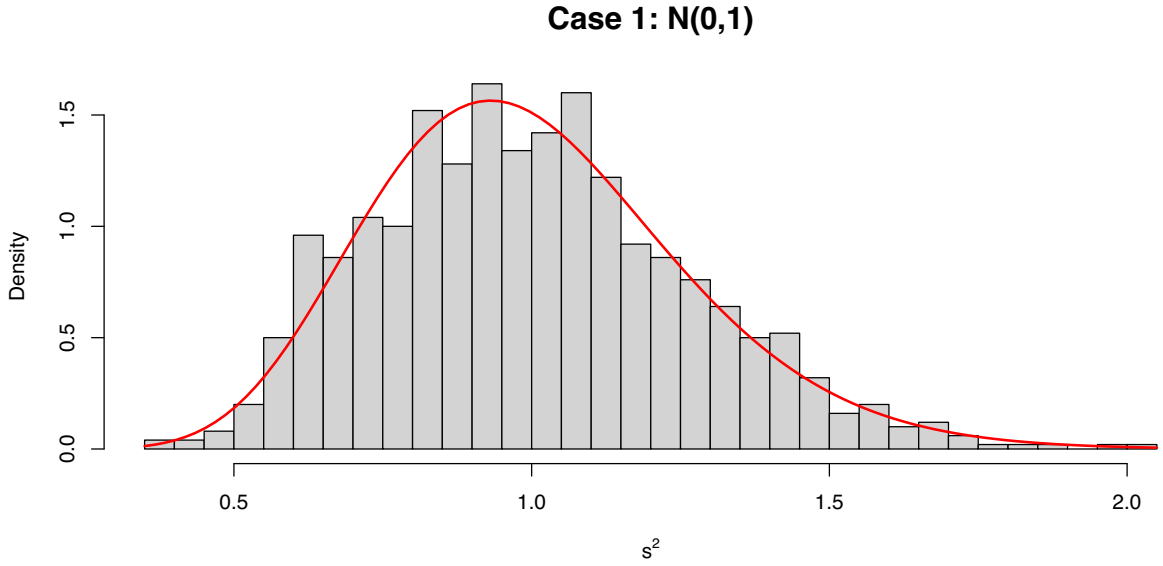
$$f_Y(y) = f_X\left(\frac{n-1}{\sigma^2}y\right)\frac{n-1}{\sigma^2},$$

since $g^{-1}(y) = \frac{n-1}{\sigma^2}y$ and $\frac{d}{dy}g^{-1}(y) = \frac{n-1}{\sigma^2}$

It's your turn

Visualise the sample distribution of the sample variance and overlap the corresponding distribution.

```
par (mfrow = c(1, 1))
sigma <- 1
hist(samples_stat[1, 2, ], nclass = 30, probability = TRUE,
     xlab = expression(s^2), main = "Case 1: N(0,1)", cex.main = 1.5)
curve(((n - 1)/sigma^2) * dchisq(x * ((n - 1)/sigma^2), df = n - 1),
      add = TRUE, col = "red", lwd = 2)
```



Extra Exercise: sample variance distribution under non normal cases

Consider the following paper https://www.tandfonline.com/doi/abs/10.1080/00031305.2014.966589 (in the Lab1 folder you can find the pdf version). It is useful to understand the distribution of the sample variance $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2$ when the Y_i's are not normally distributed.

Then, visualize the sample distribution of the sample variances for the case 2 and 3 via histograms; overlap the probability density function of the sample variance estimators according to the results of the aforementioned paper and compare them with those obtained by considering the distribution of the sample variance for the normal case.

Some relationship among probability distributions

During the theory lectures, you have seen how to generate a value from a distribution, with known and easily to derive quantile function, starting from the uniform generator. Such an approach is called **inverse sampling method**.

Thus, let $U_i \sim U(0,1)$, $i = 1, \ldots, n$, be iid rv. We can verify that the random variable X defined as:

$$X = -\frac{\log(U_1)}{\theta}$$

follows an exponential distribution with parameter θ.

Extra exercise

Now, fix $\theta = 2$ and verify that

- The random variable $Z = \min\{X_1, X_2, \ldots, X_n\}$, where $X_i = -\log(U_i)/\theta$, follows an exponential distribution with parameter $n\theta$.
- The distribution of the random variable $Y = \sum_{i=1}^{n} X_i$ is a Gamma distribution with parameters shape=n and rate=θ.

You can see such relationships among probability distributions from the chart https://www.math.wm.edu/~leemis/chart/UDR/UDR.html

Basic concepts of estimation

Point estimation

Let $\hat{\theta}$ be the point estimator for the parameter θ. An estimator is said to be

- **unbiased** if and only if $E(\hat{\theta}) = \theta$
- **consistent** if $\hat{\theta} \xrightarrow{P} \theta$, as $n \to \infty$ or, equivalently, if $\text{var}(\hat{\theta}) \to 0$, as $n \to \infty$.

There are some properties that we would ensure for an estimator: **low variance** and **low bias**. But, there is a tradeoff between unbiasedness and low variance, so we would usually seek to get both (to some extent); ideally, we would target a small **Mean Squared Error** (**MSE**)

$$\text{MSE}(\hat{\theta}) = \text{E}\{(\hat{\theta} - \theta)^2\} = \{\text{E}(\hat{\theta}) - \theta\}^2 + \text{var}(\hat{\theta}) = (\text{Bias})^2 + \text{Variance}$$

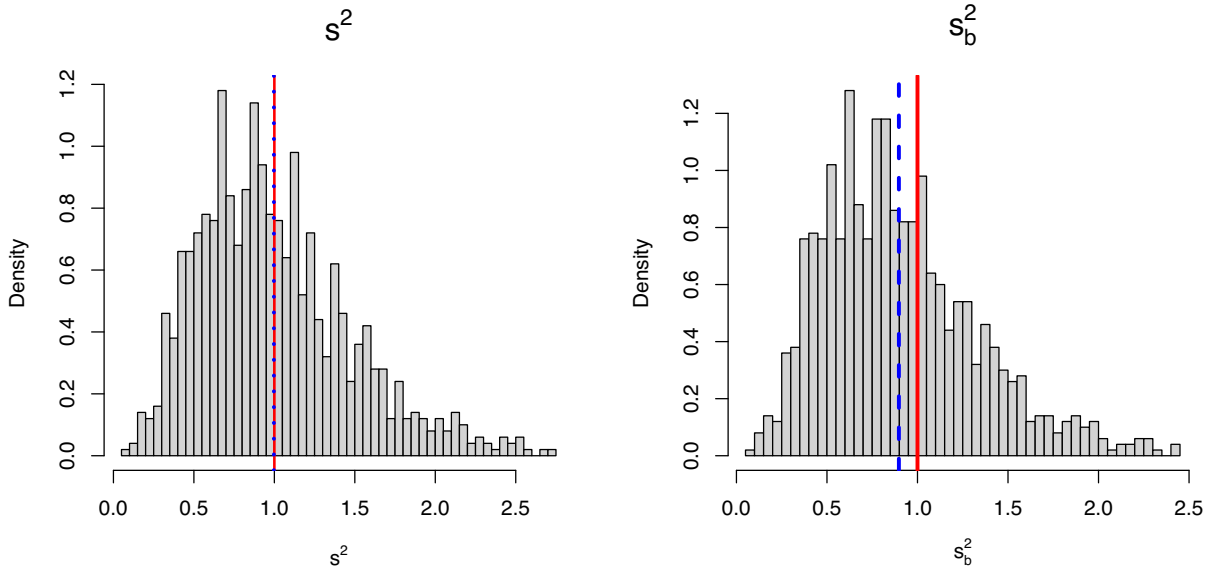
Application: Comparison of unbiased and biased sample variance estimators

Related to the distribution of the sample variance in the normal case, we could also rely on the biased estimator $S_b^2 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2$.

Check the biased nature of S_b^2 via MC simulation, generating $n = 10$ iid values from a normal distribution. Compare the results with the unbiased estimator S^2

```
#Initial settings
set.seed(2)
R <- 1000
n <- 10
mu <- 0
sigma <- 1

# Save the results in two vectors:
# s2: unbiased sample variance estimates
# s2_b: biased sample variance estimates
s2 <- rep(0, R)
s2_b <- rep(0, R)

# For each replication we generate 10 samples from
# a normal r.v. with mean mu and variance sigma^2
# and we compute the four sample variance estimates
for(i in 1 : R) {
  y <- rnorm(n, mu, sigma)
  s2[i] <- var(y)
  s2_b[i] <- var(y) * (n - 1)/n
}
```

```r
s2_mean <- mean(s2)
s2_b_mean <- mean(s2_b)

#plot s2
par(mfrow = c(1, 2), oma = c(0, 0, 0, 0))
hist(s2, breaks = 50, xlab = expression(s^2), probability = TRUE,
     main = expression(s^2), cex.main = 1.5)
#in red the true mean, in blue the estimated mean
abline(v = sigma^2, col = "red", lwd = 2)
abline(v = s2_mean, col = "blue", lwd = 3, lty = 3)

#plot s2 biased
hist(s2_b, breaks = 50, xlab = expression(s[b]^2), probability = TRUE,
     main = expression(s[b]^2), cex.main = 1.5)
#in red the true mean, in blue the estimated mean
abline(v = sigma^2, col = "red", lwd = 3)
abline(v = s2_b_mean, col = "blue", lwd = 3, lty = 2)
```



Application: Comparison of four estimators of the population mean

As we already know from the theory, the sample mean is an excellent estimator for the population mean. But we will consider now other estimators. Consider the case of a normal distribution $\mathcal{N}(\mu, \sigma^2)$ and the following estimators:

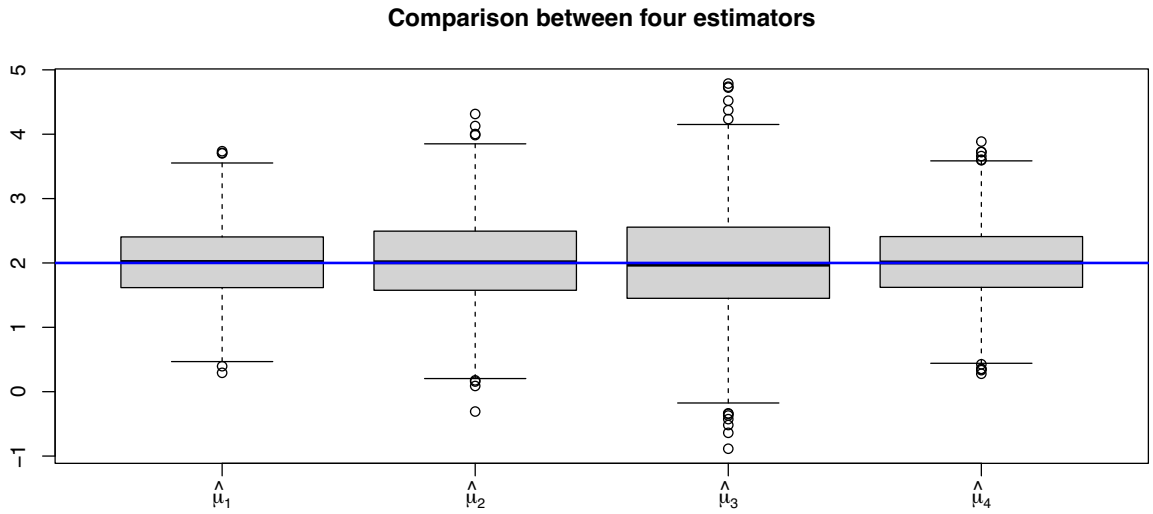
- sample mean:
$$\hat{\mu}_1 = \frac{1}{n}\sum_{i=1}^{n} x_i$$
- sample median:
$$\hat{\mu}_2 = \text{Median}(x)$$
- the semi-sum of the sample minimum and the maximum
$$\hat{\mu}_3 = \frac{\min(x) + \max(x)}{2} = \frac{x_{(1)} + x_{(n)}}{2}$$
- the 10% trimmed sample mean: the idea of the trimmed sample mean is to compute the mean discarding the 10% of data from the lower tail and the 10% from the upper tail (the following definition covers only the case when n is even and $0.1n$ is an integer)
$$\hat{\mu}_4 = \frac{1}{0.8n}\sum_{i=0.1n}^{0.9n} x_{(i)}$$

with $x_{(i)}$ the i-th order statistic.

We aim to compare the four estimators in terms of unbiasedness and efficiency.

```
R <- 1000   # Number of replications
n <- 10     # Sample size
mu <- 2     # Population mean
sigma <- 2  # Population standard deviation

est <- matrix(0, R, 4)

set.seed(1234)

for (i in 1 : R) {
 x <- rnorm(n, mu, sigma)
 est[i, 1] <- mean(x)
 est[i, 2] <- median(x)
 est[i, 3] <- (min(x) + max(x))/2
 est[i, 4] <- mean(x, trim = 0.1)
}

par(mfrow = c(1, 1), xaxt = "n")
boxplot(est, main="Comparison between four estimators")
par(xaxt = "s")
```

```
axis(1, 1 : 4, c(expression(hat(mu)[1]), expression(hat(mu)[2]),
                 expression(hat(mu)[3]), expression(hat(mu)[4])) )
abline(h = mu, lwd = 2, col = "blue")
```




```
# Bias
bias <- apply(est, 2, mean) - mu
bias
```

```
## [1]  0.012231786  0.032274611 -0.005109588  0.016567130
```

```
# Variances
variance <- apply(est, 2, var)
variance
```

```
## [1] 0.3674287 0.5111770 0.7202595 0.3870022
```

All the estimators appear unbiased and the sample mean register the lowest estimated sample variance, which is a good approximation of $\sigma^2/n = 0.4$.

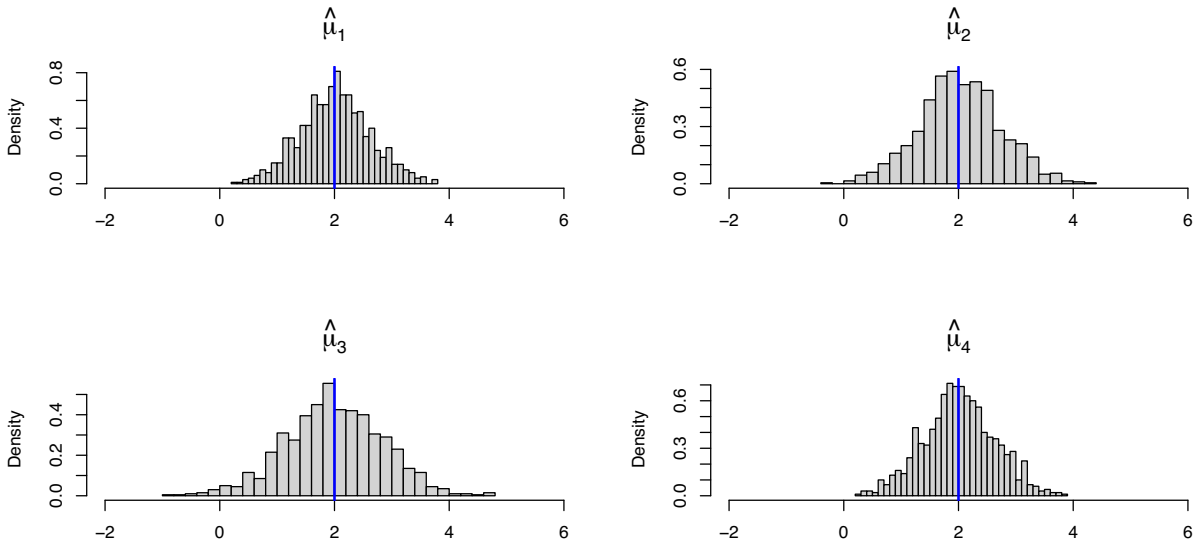
It's your turn

Let's check now whether all the estimators are consistent. For checking this statement, $n = 10$ is extremely low and need to be increased, let's say $n = 200$.

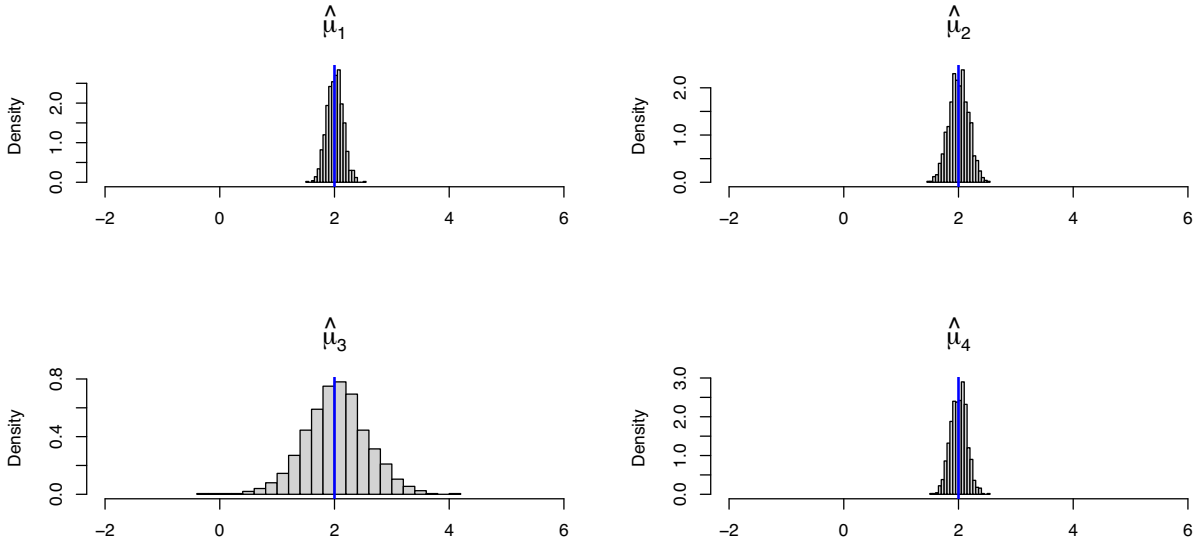
Compare the simulated values for $n = 10$ with those obtained for $n = 200$ by using histograms. What do you expect?

```
n <- 200
mu <- 2
sigma <- 2
```

```r
# We repeat the previous steps considering n=200
est_200 <- matrix(0, R, 4)
for (i in 1 : R) {
 x <- rnorm(n, mu, sigma)
 est_200[i, 1] <- mean(x)
 est_200[i, 2] <- median(x)
 est_200[i, 3] <- (max(x) + min(x))/2
 est_200[i, 4] <- mean(x, trim = 0.1)
}

# n =10
par(mfrow = c(2, 2))
for (j in 1 : 4) {
 hist(est[,j], nclass = 30,  probability = TRUE,
      main = substitute(hat(mu)[j] , list(j = j)),
      xlab = "", xlim = c(-2, 6), cex.main = 1.5)
 abline(v = mu, col = "blue", lwd = 2)
}
```




```r
# n= 200
for(j in 1 : 4){
 hist(est_200[,j], nclass = 30,  probability = TRUE,
      main=substitute(hat(mu)[j] , list(j = j)),
      xlab = "", xlim = c(-2, 6), cex.main = 1.5)
 abline(v = mu, col = "blue", lwd = 2)
}
```




```
# Variances
variance <- apply(est_200, 2, var)
variance
```

```
## [1] 0.01903574 0.02883970 0.30210689 0.02013795
```

Some notes on R Markdown

R Markdown is a powerful tool to create reports, by integrating text, formulas, R code, plots, links and so on.

Getting start:

- Create an R Markdown file (File, New File)
- Choose the title, the author/authors and the selected output
- The (default) header includes what you choose on the previous step
- We can add subtitle, fontsize and several output related options (not considered here)

The compilation is done by clicking on the knit button (there you can choose between html or pdf, despite your initial choice).

In moodle (Lab1 folder) you will find some materials on the basic use of R Markdown, but the web is a great mine of information.

The R code, in R Markdown term a chunk: each chunk is delimited by the symbols "‘ and must start with the syntax `{r chunk_name }`

You can mask the code by adding the argument `echo = FALSE` and you can avoid the evaluation by using `eval = FALSE`.

For a better rendering, it is better to introduce the option `fig.align = 'center'` in the options of knitr function otherwise you can use it as argument of `{r chunk_name }`.

In the Lab1 folder, you can also find an example on how generating slides using beamer in R Markdown (see the folder ExampleRMD_beamer).