PSTAT160A Stochastic Processes
Section 3

Authors: *John Inston, Denisse Alejandra Escobar Parra*
Date: *October 14, 2025*

Probability Distributions in R

R provides a unified system for working with probability distributions. Each distribution (Normal, Binomial, Poisson, etc.) has a *root name*, and four associated commands obtained by adding one of the prefixes `d`, `p`, `q`, or `r`.

Table 1: Table of prefixes with descriptions.

Prefix	Meaning	Description
`d`	Density / Mass	Gives $f(x)$ or $P(X = x)$.
`p`	Cumulative probability	Computes $P(X \leq q)$.
`q`	Quantile (inverse CDF)	Finds q such that $P(X \leq q) = p$.
`r`	Random generation	Simulates n samples from the distribution.

Table 2: Common probability distributions in `R` and their root names.

Distribution	Root	Distribution	Root
Beta	`beta`	Log-normal	`lnorm`
Binomial	`binom`	Multinomial	`multinom`
Cauchy	`cauchy`	Negative Binomial	`nbinom`
Chi-squared	`chisq`	Normal	`norm`
Exponential	`exp`	Poisson	`pois`
F	`f`	Student's t	`t`
Gamma	`gamma`	Uniform	`unif`
Geometric	`geom`	Weibull	`weibull`
Hypergeometric	`hyper`	—	—

Example 1. Generate 5 random numbers distributed as Poisson with $\lambda = 3$.

```r
rpois(5,3)
```

```
[1] 1 5 6 4 3
```

Example 2. Compute $\mathbb{P}(X \leq 2)$ with $X \sim Bin(10, 0.3)$.

```r
pbinom(2,10,0.3)
```

```
[1] 0.3827828
```

Example 3. Find the 90th percentile of a $Gamma(2, 1)$ distribution.

```r
qgamma(0.9, 2, 1)
```

```
[1] 3.88972
```

Example 4. Find $f_X(2.5)$ where $X \sim exp(1)$.

```r
dexp(2.5,1)
```

```
[1] 0.082085
```

Exercises

1. Use `rbinom(10, size = 10, prob = 0.5)` to generate random values. Change the probability parameter to 0.2 and 0.8. What do you notice about the results?
2. Simulate `x <- rnorm(1000, 0, 1)`. Plot a histogram of `x` and overlay the theoretical density using:

```r
hist(x, freq = FALSE)
curve(dnorm(x, 0, 1), add = TRUE, col = "blue")
```

3. Compute the proportion of simulated values falling between -1 and 1, and compare to `pnorm(1) - pnorm(-1)`.

Plots and grahs

Plots and Graphs in R

A central part of data analysis is *visualizing* numerical results. In R, the command `plot()` is extremely versatile: it can display functions, data points, time series, or relationships between variables.

Other common plotting functions include `curve()` for smooth functions and `hist()` for histograms.

Basic usage of `plot()`

The general form is:

```
plot(x, y, type = "p", main = "Title", xlab = "X-axis", ylab = "Y-axis")
```

Some common values for the argument `type` are:

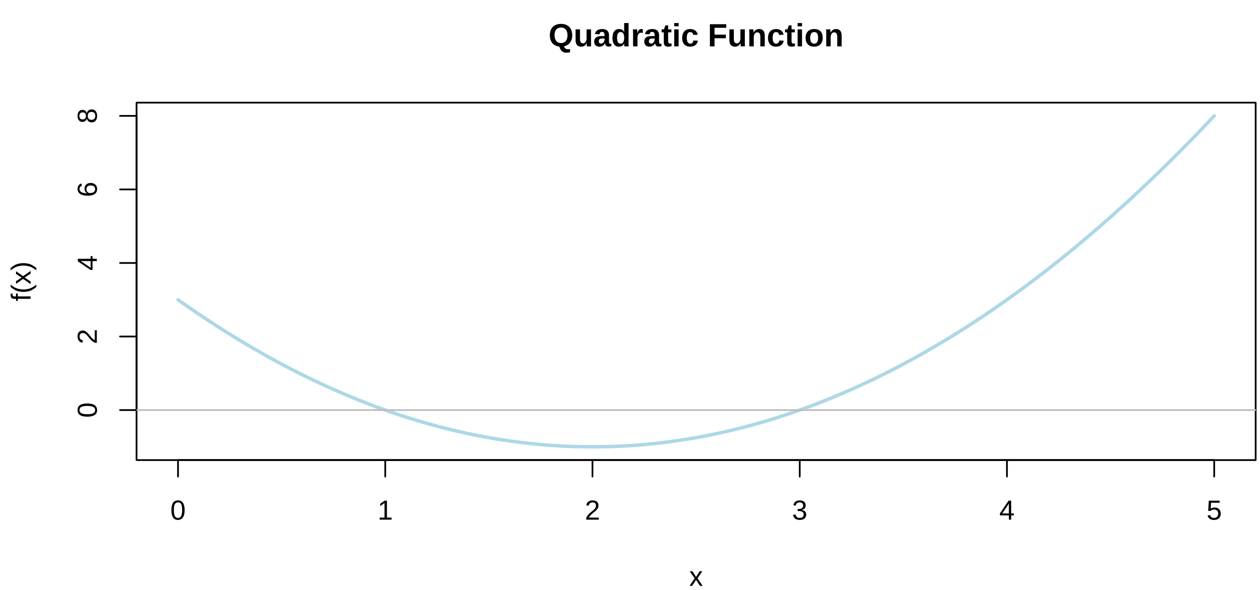
- `p`: points (the default)
- `l`: lines
- `b`: both points and lines

You can also customize colors, labels, or add reference lines with `col`, `xlab`, `ylab`, and `abline()`.

Example 1. Plotting a quadratic function

Use `curve()` to graph $f(x) = x^2 - 4x + 3$ for $x \in [0, 5]$:

```
curve(x^2 - 4*x + 3, from = 0, to = 5,
      main = "Quadratic Function",
      xlab = "x", ylab = "f(x)", col = "lightblue", lwd = 2)
abline(h = 0, col = "gray")
```

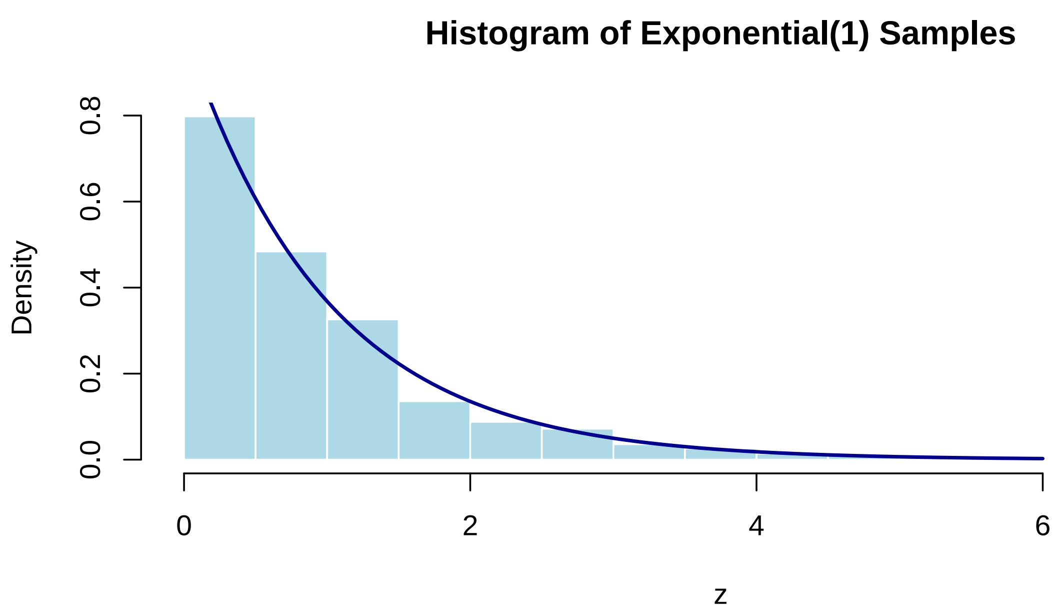


This produces a smooth curve of the function and adds a horizontal line at $y = 0$.

Example 2. Histogram and density curve

The command `hist()` displays the distribution of simulated or observed data. To overlay a continuous density curve, set `freq = FALSE` so that the histogram represents relative frequencies.

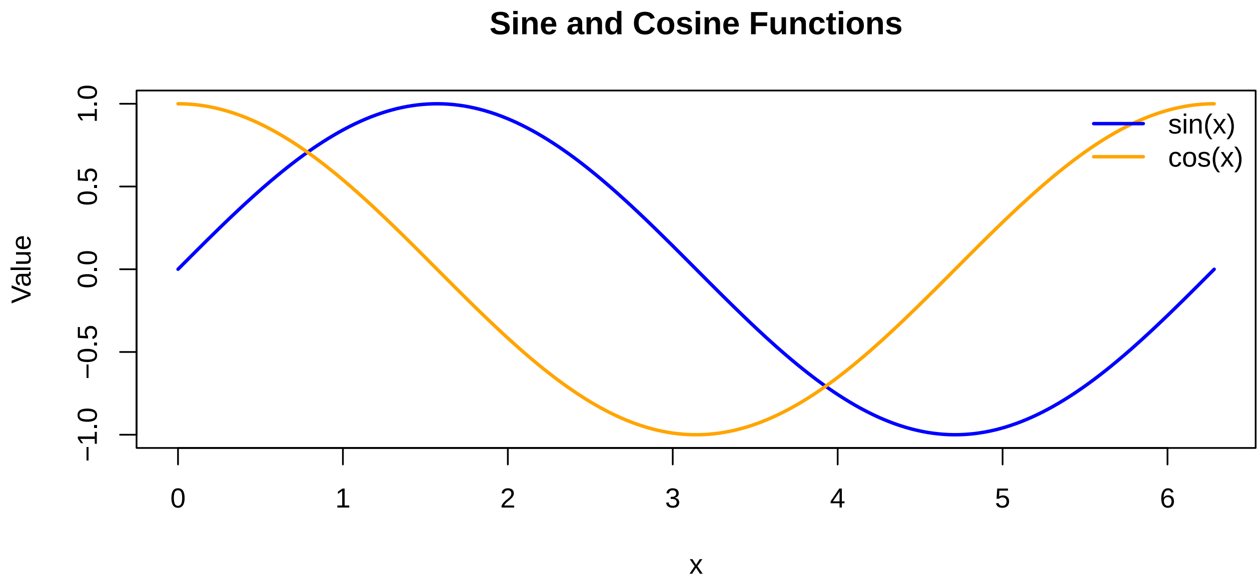
```
z <- rexp(1000, rate = 1)   # Exponential(1) samples
hist(z, freq = FALSE, main = "Histogram of Exponential(1) Samples",
     xlab = "z", col = "lightblue", border = "white")
curve(dexp(x, 1), from = 0, to = 6, add = TRUE, col = "darkblue", lwd = 2)
```



Example 3. Multiple lines in the same plot

Use `lines()` to add more series to an existing plot. Below we compare $\sin(x)$ and $\cos(x)$ on the same axes.

```
x <- seq(0, 2*pi, length.out = 200)
plot(x, sin(x), type = "l", col = "blue", lwd = 2,
     main = "Sine and Cosine Functions", ylab = "Value", xlab = "x")
lines(x, cos(x), col = "orange", lwd = 2)
legend("topright", legend = c("sin(x)", "cos(x)"),
       col = c("blue", "orange"), lwd = 2, bty = "n")
```



Exercises

1. Plot the cubic function $f(x) = x^3 - 3x$ for $x \in [-3, 3]$. Add horizontal and vertical reference lines at 0 using `abline()`.

2. Simulate 1,000 values from a Normal$(5, 2^2)$ distribution. Create a histogram with density scaling (`freq = FALSE`) and overlay the theoretical normal density using `curve()`.

3. Plot the functions $\sin(x)$, $\cos(x)$, and $\sin(x) + \cos(x)$ on the same graph with different colors and a legend.

Script Files in R

When working with many R commands, it is convenient to save them in a *script file*. A script is simply a plain text file (usually with extension `.R`) that stores R code. This allows you to edit, organize, and re-run code easily, without typing each command into the console every time.

Creating and running a script in RStudio

1. Go to **File** → **New File** → **R Script**.
2. Type your commands in the editor pane.
3. Highlight the lines you want to execute, and press `Cmd + Enter` (Mac) or `Ctrl + Enter` (Windows).
4. Save the file with a meaningful name, such as `my\_script.R`.

Alternatively, once a script is saved, you can execute the entire file directly from the console:

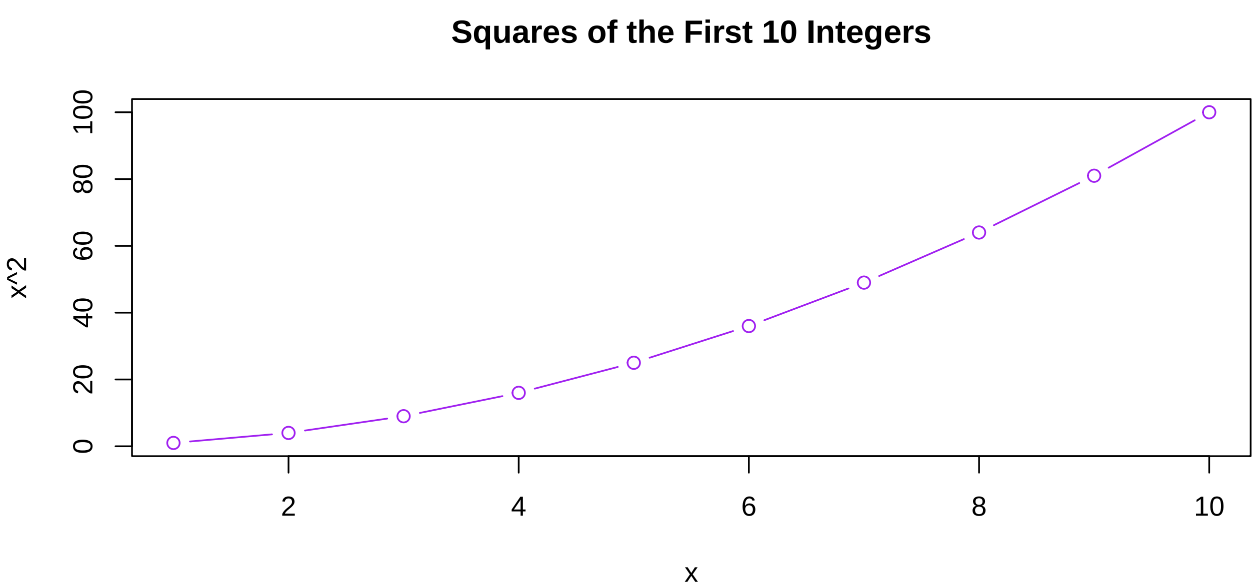
```
source("my_script.R")
```

Using scripts helps to keep your workflow organized, reproducible, and easier to share.

Example 1. Simple sequence operations

Create a new file named `sequences.R` and write the following commands:

```
x <- 1:10
y <- x^2
plot(x, y, type = "b", col = "purple",
     main = "Squares of the First 10 Integers",
     xlab = "x", ylab = "x^2")
```



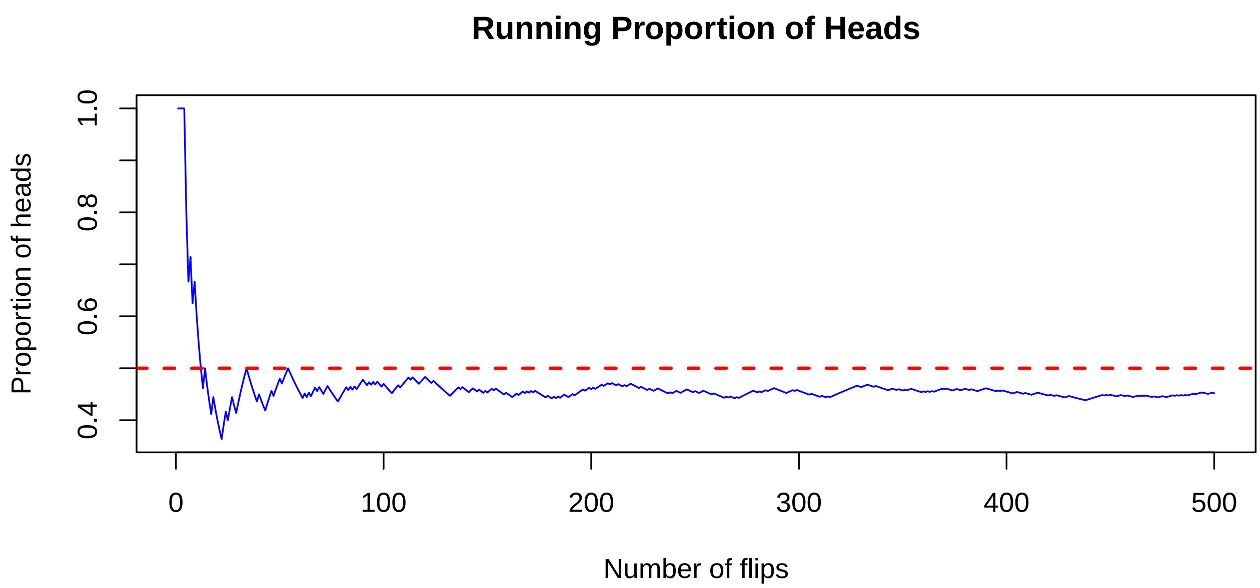
Save the file and run it using `source("sequences.R")`. The resulting plot should appear in your RStudio plotting window.

Example 2. Simulating and plotting coin flips

Consider 500 independent coin flips, with 1 representing heads and 0 tails.
We can compute and plot the running proportion of heads.

```
n <- 500
flips <- sample(0:1, n, replace = TRUE)
prop <- cumsum(flips) / (1:n)

plot(1:n, prop, type = "l", col = "blue",
     xlab = "Number of flips",
     ylab = "Proportion of heads",
     main = "Running Proportion of Heads")
abline(h = 0.5, col = "red", lwd = 2, lty = 2)
```



This visualizes the **Law of Large Numbers**: as n increases, the proportion of heads tends to 0.5.

Example 3. Writing a short script that computes summary statistics

Save the following script as `summaries.R`:

```
x <- rnorm(1000, mean = 10, sd = 2)
mean_x <- mean(x)
sd_x <- sd(x)

cat("Sample mean:", mean_x, "\n")
cat("Sample standard deviation:", sd_x, "\n")
```

When you run `source("summaries.R")`, R prints the computed sample mean and standard deviation in the console.

Exercises

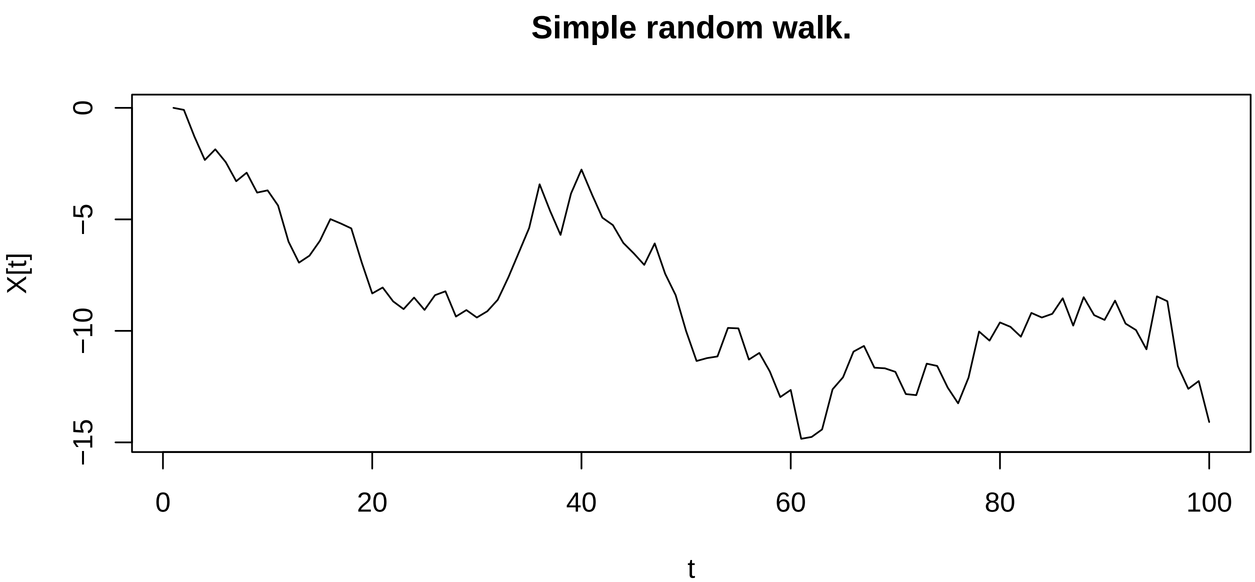
1. Modify the coin flip simulation so that the coin is *biased*, with probability of heads $p = 0.6$. Plot the running proportion of heads and compare it to the unbiased case.

Random Walks

A *random walk* is a mathematical model that describes a path consisting of a sequence of random steps. Specifically, a random walk is a sequence of random variables $\{X_t\}_{t\in\mathbb{N}}$ which are defined recursively by

$$X_t = X_{t-1} + \xi_t, \tag{1}$$

where $\{\xi_t\}_{t\in\mathbb{N}}$ is the sequence of independent and identically distributed random variables representing the random steps.



Random walks can also be represented as a cumulative sum, for example from Equation 1 we can compute

$$\begin{aligned} X_t &= X_{t-1} + \xi_t \\ &= (X_{t-1} + \xi_{t-1}) + \xi_t \\ &\vdots \\ &= X_0 + \xi_1 + \xi_2 + \ldots + \xi_t \\ &= X_0 + \sum_{i=1}^{t} \xi_i. \end{aligned} \tag{2}$$

A *realization* of a random walk is a single selection from the set of possible paths. To generate a realization of a random walk we simulate the random variables used to construct the process.

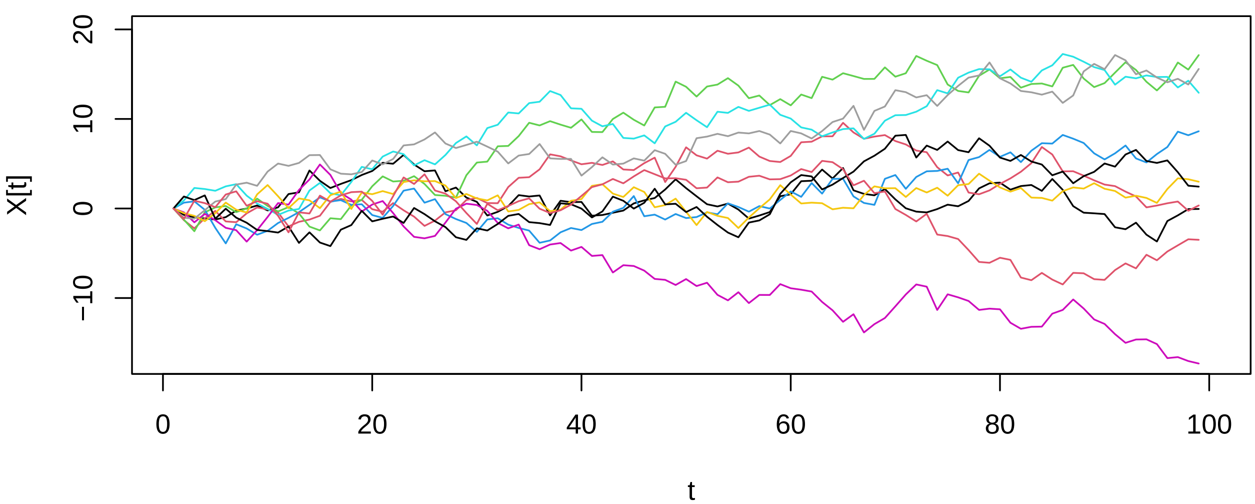
Example 1. Realizations of a random walk

We define a random walk using Equation 2 where we specify $\xi_t \sim \mathcal{N}(\mu = 1, \sigma^2 = 1)$. We can first write a function `gaussian_random_walk` generating a single realization of the process of length `T=100` with starting position `x0=0`:

```r
gaussian_random_walk <- function(mu=0,sigma=1,T=100,x0=0){
  xi <- rnorm(T-1,mu,sigma)
  x <- x0
  for(t in 2:T){
    x[t] = x[t-1] + xi[t]
  }
  return(x)
}
```

We can use this function to produce a 10 realizations of the process which we plot using `lines()`:

```r
set.seed(321)
x <- gaussian_random_walk()
plot(1:100,x,type = "l", xlab="t", ylab="X[t]", ylim=c(-17,20))
for(i in 2:10){
  lines(1:100,gaussian_random_walk(),col=i)
}
```



Exercises

1. Adjust the code above to change the step random variables to following independent and identically distribution Bernoulli random variables with $p = 0.5$.