Sums of random variables

Isadora Antoniano Villalobos

2025-11-26

1 Sums of Standard Normal Random Variables

We consider a sequence of i.i.d random variables:

$$X_i \sim N(0,1); \quad i = 1, 2, \dots$$

We use these to define a new sequence:

$$S_n = \sum_{i=1}^{n} X_i; \quad n = 1, 2, \dots$$

We know that, for each n,

$$\mathbb{E}[S_n] = 0, \quad Var[S_n] = n, \quad S_n \sim N(0, n).$$

Additionally, these variables are not independent, in fact, $S_{n+1} = S_n + X_{n+1}$, so

$$\mathbb{E}[S_n + 1|S_n] = S_n, \quad Var[S_{n+1}|S_n] = 1, \quad S_{n+1}|S_n \sim N(S_n, 1).$$

The dependence between the variables creates a very different structure o the observed sequences (or paths) compared to what we would observe if the variables were independent. We can visualize this by simulating some realizations from the sequence of sums S_n, $n = 1, 2, \dots$ and comparing them with realizations of a sequence $W_n \sim N(0, n)$ of independent random variables with the same marginal distributions.

1.1 A single realization of the X_n sequence

First, we simulate a single realization of the first $N = 1000$ variables in the initial i.i.d $X_n \sim N(0,1)$ sequence. In order to achieve reproducibility, we will fix the seed of the random number generator to ensure we always obtain the same numbers.

```
# Set random generator seed
set.seed(42)
N = 1000 # sample size
x = rnorm(N) # N i.i.d. standard Normal
```

Notice that we're using a lower case x to store the generated values, since they correspond to a single realization of the vector $X = (X_1, \dots, X_N)$

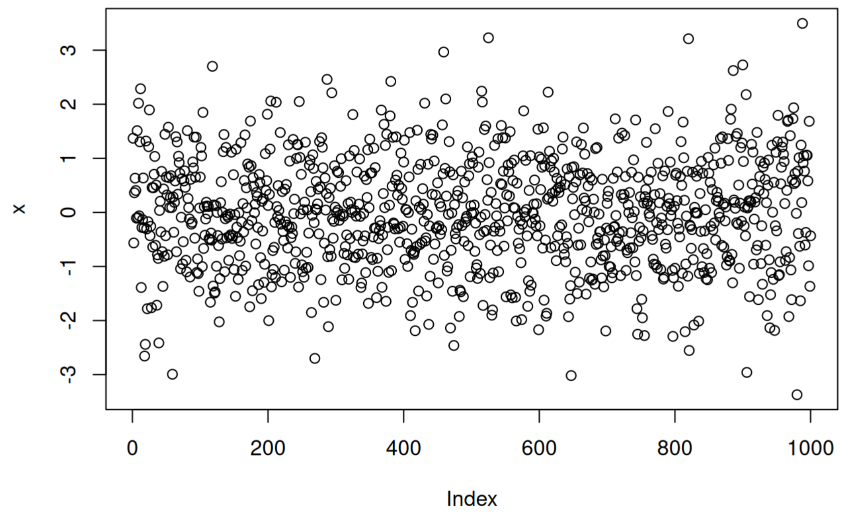
We can use the function `head` to see the initial values of the sequence we just generated:

```
head(x)
```

```
## [1]  1.3709584 -0.5646982  0.3631284  0.6328626  0.4042683 -0.1061245
```

And we can use the function `plot` to visualize all the observations:

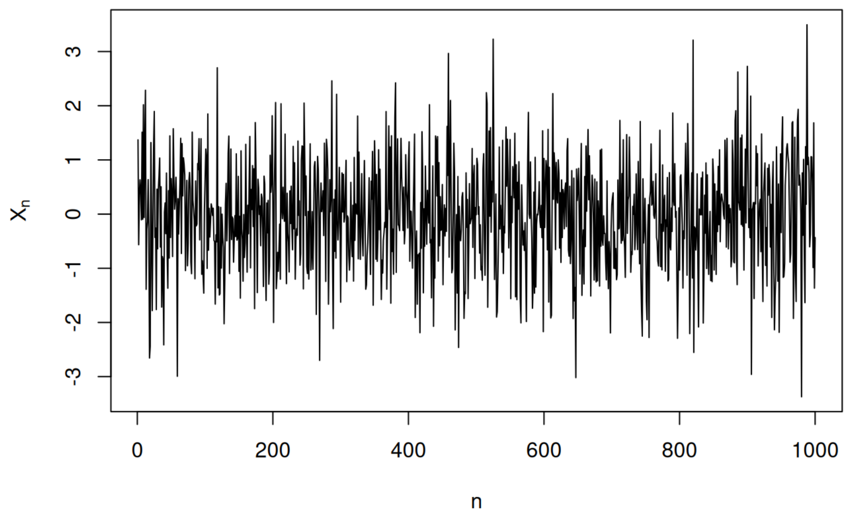
```
plot(x)
```



The dots in the plot remind us that we have individual realizations of the $N = 1000$ random variables: $X_1 = x_1, \dots, X_N = x_N$.

We could, instead, use a line to better represent the idea of a single random function $X(n) = X_n$ changing values through time. The time, in this case, is observed as fixed intervals $n = 1, 2, \dots, N$.

```
# A line plot with labels for the X and Y axis
plot(x, type= 'l', xlab = "n", ylab= expression(X[n]))
```

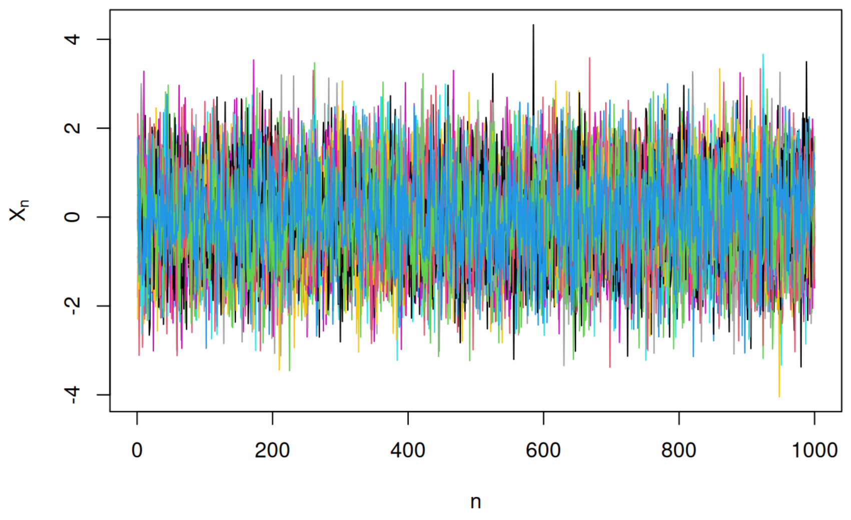


We can clearly see how the observations are centered around their common mean 0 and most of them fall within 3 standard deviations from the mean (a known characteristic of the Normal distribution).

1.2 Multiple realizations of the X_n sequence

Remember that random variables are functions of the results of an experiment. The sequence we just produced can be seen as the initial 1000 values of an infinite sequence corresponding to a single observed experiment result $\omega \in \Omega$. We can think of repeating the experiment $m = 20$ times, obtaining, each time, a different observed sequence, or path, of this process. The function `replicate` allows us to do this efficiently in R:

```
# Set random generator seed
set.seed(42)
# Sample size of the individual RVs (length of each path)
N = 1000
# Sample size of the process (number of paths)
m = 20
# Matrix with m columns, each containing the N realizations of a path
x_paths = replicate(m,rnorm(N))
# Plot of the first path with axis limits big enough to fit them all
plot(x_paths[,1], type= 'l', xlim = c(0, N), ylim=range(x_paths), xlab = "n",
ylab= expression(X[n]))
# Adding the other paths in the same graphic space, each with its own color
for (i in 2:m) {
  lines(x_paths[,i], col=i)
}
```



Notice how all the paths exhibit a similar behavior. This is due to the independence of the individual X_n variables. These sequences of i.i.d. standard normal random variables are a mathematical representation of what is commonly known as **white noise**.

1.3 A single realization of the S_n sequence

Lets now move on to the sequence of sums. The first and simplest idea to simulate a path is to start with the first $s_1 = x_1$ and successively augment the s array to include $s_{n+1} = s_n + x_{n+1}$:

```
set.seed(42)
N = 1000 # sample size
S1 = rnorm(1) # Initializing the vector of sums
for (n in 2:N) {
  S_inefficient = c(S1, rnorm(1)+ S1[n-1])
}
head(S_inefficient)
```

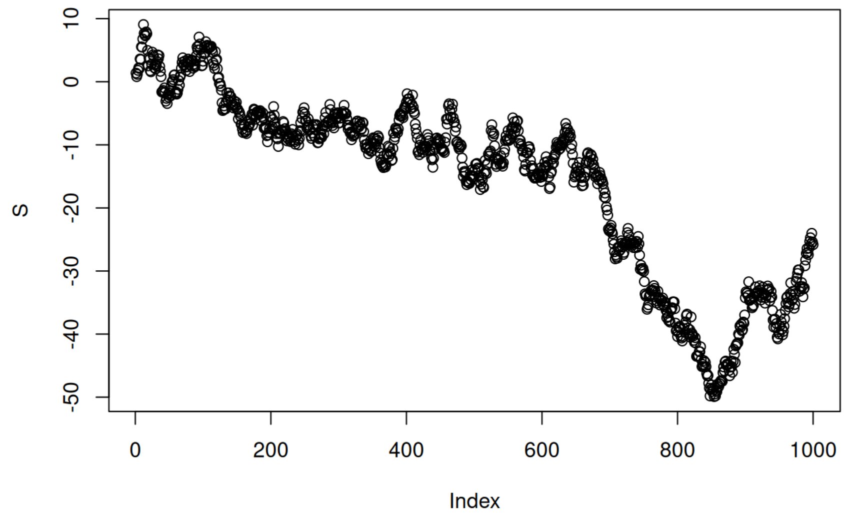
```
## [1] 1.370958       NA
```

This works but is not computationally efficient. Instead, we can use the `cumsum` function to produce exactly the accumulated sums we are interested on. Notice that by fixing the seed of the random number generator to the same value, we obtain the same sequence:

```
set.seed(42)
N=1000
S = cumsum(rnorm(N))
head(S)
```

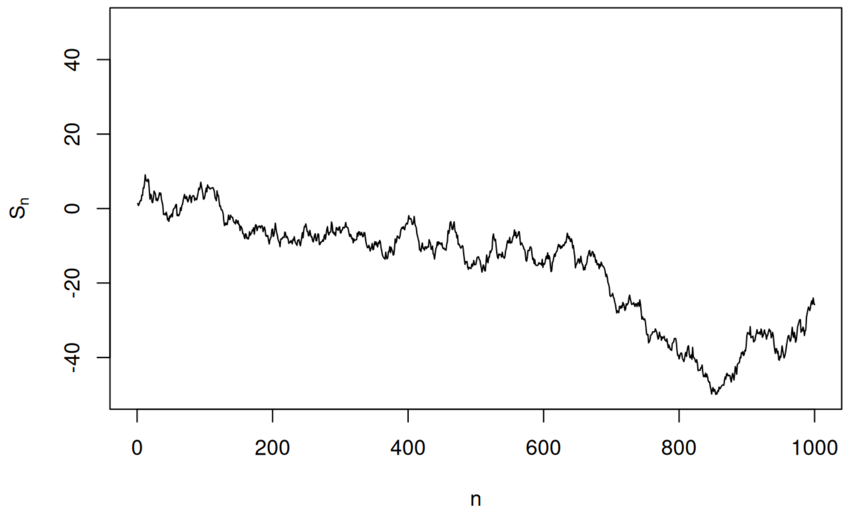
```
## [1] 1.3709584 0.8062603 1.1693887 1.8022513 2.2065196 2.1003951
```

```
# Plot our sequence of sums
plot(S)
```



Once again, we use a line plot to better visualize the path as a function of n. We can center the graph around 0 to see how far it goes above and below the mean.

```
plot(S, type= 'l', xlim = c(0, N), ylim=max(abs(S))*c(-1,1), xlab = "n", ylab
= expression(S[n]))
```



Notice how the path starts close to 0 and then drifts away much further than the X_n paths ever did. This is due to the aggregation effect and the increasing variance of S_n as n grows. However, it is hard to say much more with only one observed path. Is this a typical behavior of the paths of sums?

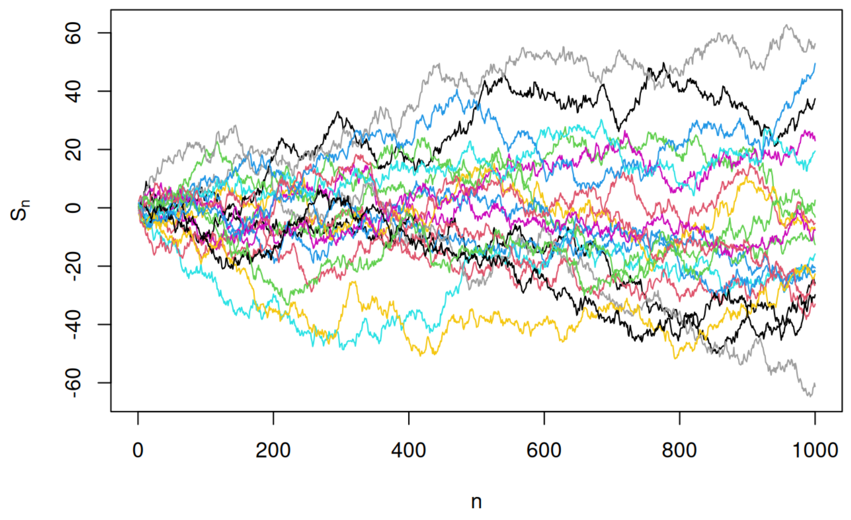
1.4 Multiple realizations of the S_n sequence

Once again, we will use the function `replicate` together with `cumsum` to make the simulation efficient:

```
set.seed(42)
m = 20 # Number of realizations of the S sequence
N=1000 # Sample size for each S realization

s_paths= replicate(m, cumsum(rnorm(N)))

# Plot our first path
plot(s_paths[,1], type= 'l', xlim = c(0, N), ylim=range(s_paths), xlab = "n",
ylab= expression(S[n]))
# Add the other paths in the same graphic space each with its own color
for (i in 2:m) {
  lines(s_paths[,i], col=i)
}
```



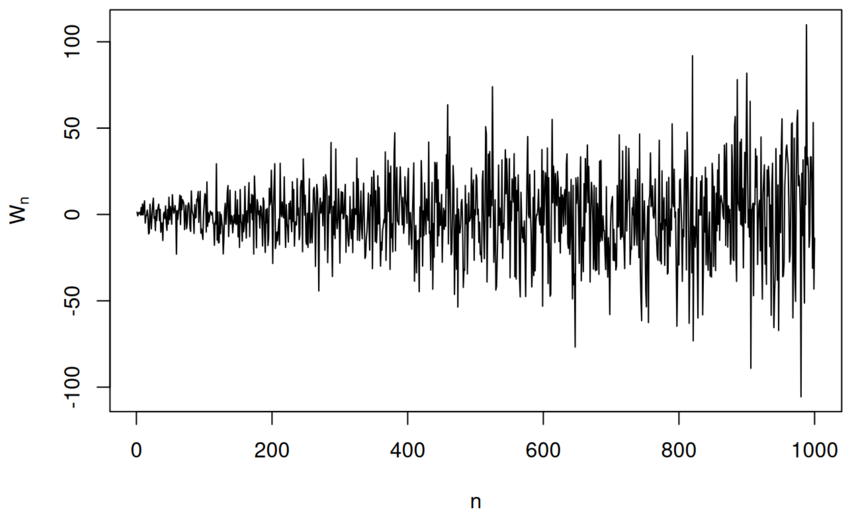
We can now clearly see the typycal behavior of the paths that we observed earlier. Indeed, they are all centered around the common mean, zero, and their variability increases with n. The effects of the dependence are harder to identify to the untrained eye. Perhaps it's best to compare with a sequence of independent random variables with the same marginal distributions:

$$W_n \sim N(0,n), \quad n=1,2,\dots$$

.

We will first produce a single realization:

```
set.seed(42)
N=1000
m=20
w = rnorm(N, 0, sqrt(c(1:N)))
plot(w, type= 'l', xlab = "n", ylab= expression(W[n]))
```



We can see how a single realization of this process produces a behavior much more erratic than any of the individual paths of dependent sums s_n. Because the marginal distributions coincide, we can still see how the observed values w_n tend to get further away from the mean, zero, as n grows. But they are able to span the whole range of values in a single realization, while we needed $m=20$ realizations of the dependent process to produce that much variability.

This shows the importance of the dependence between the S_n variables. Singularly, for each n, S_n and W_n have the same variability. But within a single path, the W_n are free to vary independently, while the S_n are conditioned by the previous observed values. The marginal distributions alone don't give us enough information about the (joint) behaviour of the process.

We don't need to produce m realizations of the W_n sequence because one is enough to see the difference between having and not having independence.

2 Controlling the variance: Law of Large Numbers and Central Limit Theorem

Notice that

$$Var[S_n] = n \xrightarrow[n\to\infty]{} \infty$$

so the sequence of random variables does not converge. Intuitively, in the limit S_n will have infinite variability.

2.1 Visualizing the Law of Large Numbers

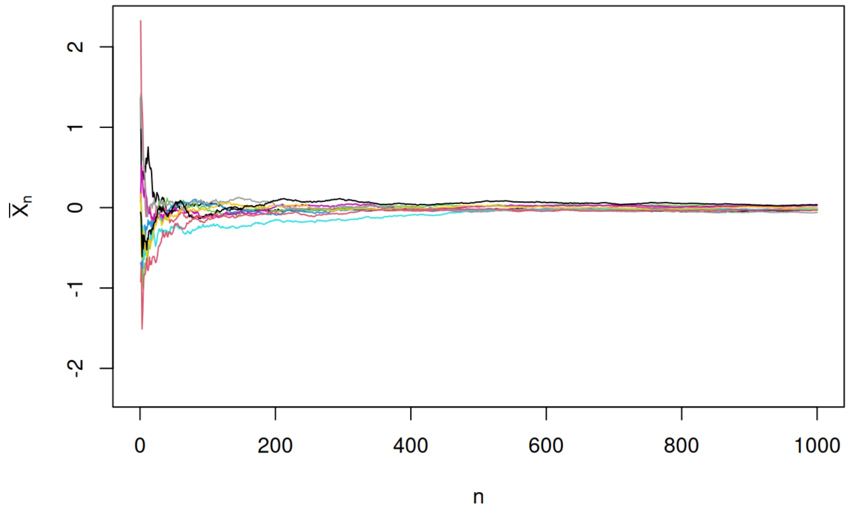
If we want to produce a convergence sequence, a first idea would be to divide S_n by n, to control the variance. This produces a new random sequence:

$$\bar{X}_n = \frac{1}{n} S_n$$

For each n, the new variable $\bar{X}_n$ is known as the *Sample mean* of the sample $X_1, \ldots, X_n$. Let's look at the observed sample means corresponding to our previously simulated samples. We can do this by fixing the same seed:

```r
set.seed(42)
N = 1000
m = 20
# Divide each sum by the corresponding sample size
sm_paths <- replicate(m, cumsum(rnorm(N)) / (1:N))

# Plot the paths in the usual way
plot(sm_paths[,1], type = 'l', xlim = c(0, N), ylim = range(sm_paths), xlab =
"n", ylab = expression(bar(X)[n]))
for (i in 2:10) {
  lines(sm_paths[,i], col = i)
}
```



The variability of the sequence has been controlled. Perhaps too much. Indeed:

$$Var[\bar{X}_n] = \frac{1}{n} \xrightarrow[n\to\infty]{} 0$$

The variance of the sequence of sample means goes to zero, while the mean is fixed. So, in the limit, the random variables collapse to the common mean, zero. The randomness disappears. This is a visual example of the *Law of Large Numbers*.

2.2 Visualizing the Central Limit Theorem

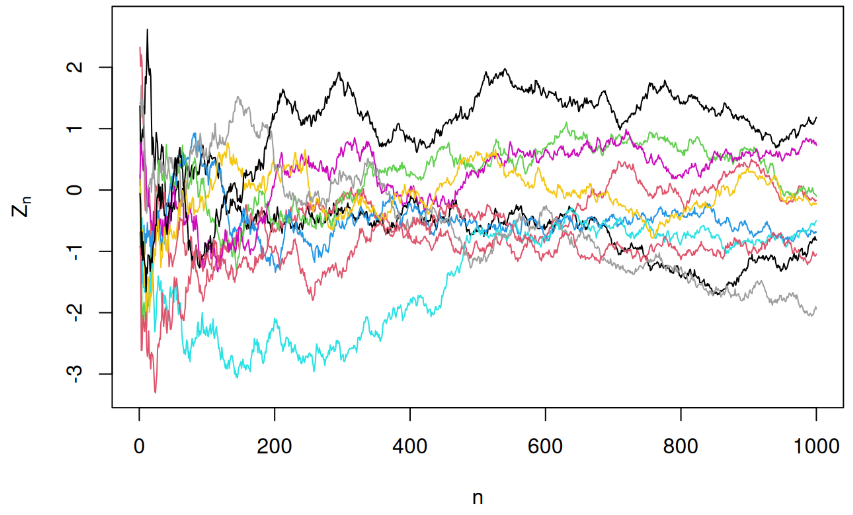
In order to control the variance of the sequence without without collapsing to a constant, we can divide S_n by something smaller, like $\sqrt{n}$. Notice that this is the same as standardizing the sequence:

$$Z_n = \frac{S_n - \mathbb{E}[S_n]}{\sqrt{Var[S_n]}} = \frac{S_n - 0}{\sqrt{n}} = \frac{1}{\sqrt{n}} S_n$$

Let's visualize the behaviour of the typical paths of the Z_n sequence:

```r
set.seed(42)
N <- 1000
m = 20
# Standardize the sums
z_paths <- replicate(m, cumsum(rnorm(N)) / sqrt(1:N))

# Plot
plot(z_paths[,1], type = 'l', xlim = c(0, N), ylim = range(z_paths), xlab =
"n", ylab = expression(Z[n]))
for (i in 2:10) {
  lines(z_paths[,i], col = i)
}
```



This time, we have generated a sequence of identically distributed standard normal random variables:

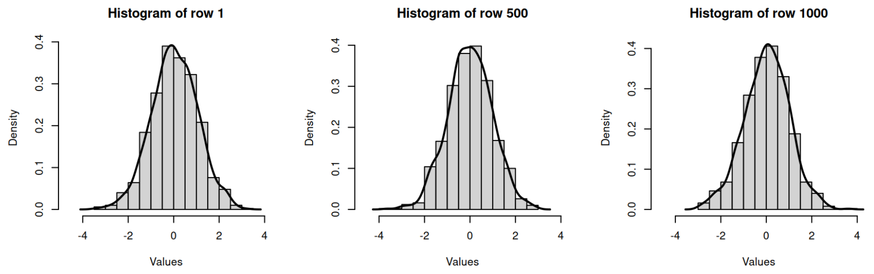
$$Z_n \sim N(0,1), \quad n = 1, 2, \ldots$$

But these are not independent, as we can see from the regularity of the individual paths.

We can visualize the marginal distribution of some of the Z_n (for example $n = 1, n = N/2, n = N$) using a *histogram*, but we will need a larger sample size:

```
set.seed(42)
N <- 1000
m = 1000
# Standardize the sums
z_paths <- replicate(m, cumsum(rnorm(N)) / sqrt(1:N))

# Rows to plot
rows_to_plot <- c(1, N/2, N)
# Three histograms side by side
par(mfrow = c(1, 3))
# Histogram for each selected row
for (r in rows_to_plot) {
  hist(z_paths[r, ],
       freq = FALSE,# density scale
       main = paste("Histogram of row", r),
       xlab = "Values",
       xlim = range(z_paths))
  lines(density(z_paths[r, ]), lwd = 2)
}
```



However, in order to understand how remarkable the **Central Limit Theorem** really is, we need to consider what happens when the original distribution of the sample X_n is not Normal, but anything else. The distribution of Z_n is no longer normal distribution, but converges to it!

$$F_{Z_n}(z) \xrightarrow[n\to\infty]{} \Phi(z)$$

The same is true for the **Law of Large Numbers**. Regardless of the distribution of the X_n, if the common mean is $\mathbb{E}[X_n] = \mu$, then

$$\bar{X}_n \xrightarrow[n\to\infty]{} \mu$$

2.3 Visualizing Central Limit Theorem and Law of large numbers for other distributions

2.3.1 Exponential

Remember

$$X \sim Exp(\lambda) \quad \Rightarrow \quad \mathbb{E}[X] = 1/\lambda, \quad Var[X] = 1/\lambda^2$$

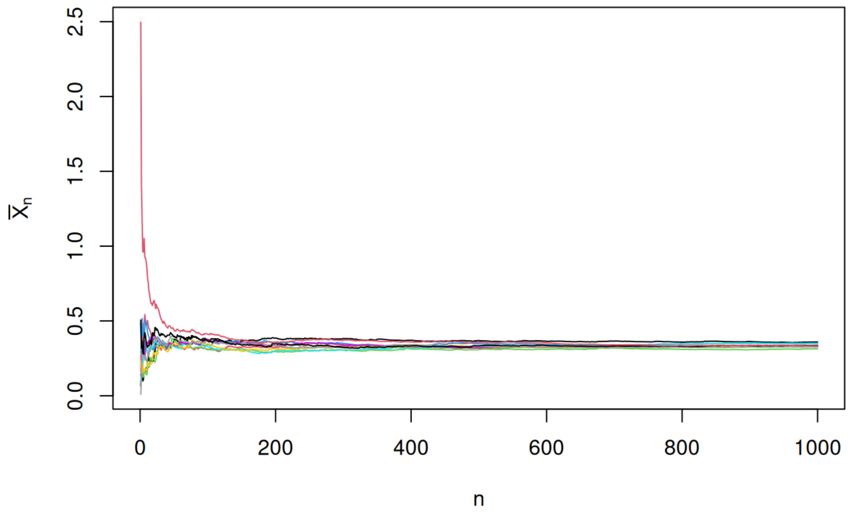
Additionally,

$$\mathbb{E}[S_n] = nE[X] = n/\lambda, \quad Var[S_n] = nVar[X] = n/\lambda^2$$

- **Law of large numbers:** We simulate $n = 20$ paths of the sequence of sample means $\bar{X}_n$

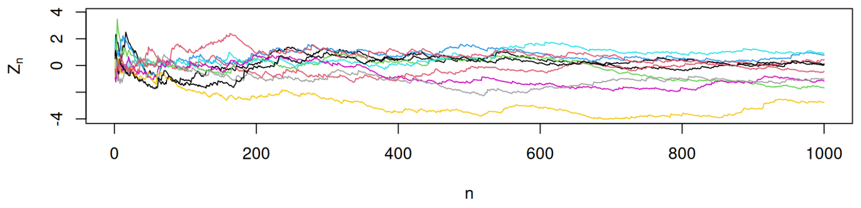
```
set.seed(42)
N = 1000
m = 20
lambda = 3
mean_s = (1:N)/lambda
sd_s = sqrt(1:N)/lambda
# Divide each sum by the corresponding sample size
sm_paths <- replicate(m, cumsum(rexp(N, lambda)) / (1:N))

# Plot the paths in the usual way
plot(sm_paths[,1], type = 'l', xlim = c(0, N), ylim = range(sm_paths), xlab =
"n", ylab = expression(bar(X)[n]))
for (i in 2:10) {
  lines(sm_paths[,i], col = i)
}
```



- **Central Limit Theorem:** We simulate $m = 20$ paths of the standardized sums, Z_n

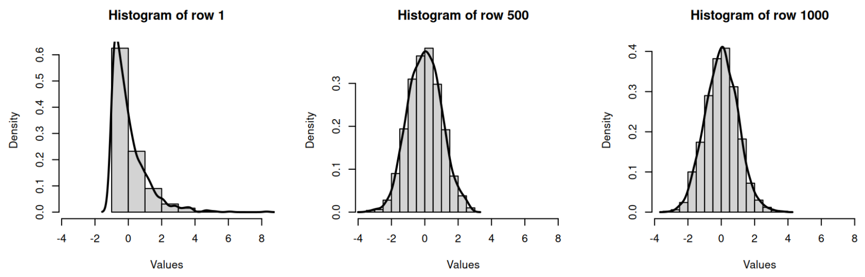
```
N = 1000
m = 20
lambda = 3
mean_s = (1:N)/lambda
sd_s = sqrt(1:N)/lambda
# Divide each sum by the corresponding sample size
z_paths <- replicate(m, (cumsum(rexp(N, lambda))-mean_s) / sd_s)
# Plot
plot(z_paths[,1], type = 'l', xlim = c(0, N), ylim = range(z_paths), xlab =
"n", ylab = expression(Z[n]))
for (i in 2:10) {
  lines(z_paths[,i], col = i)
}
```



In order to visualize the marginal distribution of Z_n converging to the normal, we simulate $m = 1000$ paths and plot histograms for selected $n = 1, N/2, N$

```
N = 1000
m = 1000
lambda = 3
mean_s = (1:N)/lambda
sd_s = sqrt(1:N)/lambda
# Divide each sum by the corresponding sample size
z_paths <- replicate(m, (cumsum(rexp(N, lambda))-mean_s) / sd_s)

# Rows to plot
rows_to_plot <- c(1, N/2, N)
# Three histograms side by side
par(mfrow = c(1, 3))
# Histogram for each selected row
for (r in rows_to_plot) {
  hist(z_paths[r, ],
       freq = FALSE,# density scale
       main = paste("Histogram of row", r),
       xlab = "Values",
       xlim = range(z_paths))
  lines(density(z_paths[r, ]), lwd = 2)
}
```



Remember that a sum of exponentials follows a gamma distribution: $S_n \sim Gamma(n, \lambda)$. So what we are seeing is a standardized Gamma distribution can be approximated by a Standard Normal if the first parameter is large enough.

2.3.2 Bernoulli

We will now replicate the experiment for Bernoulli random variables. Remember

$$X \sim Bern(p) \quad \Rightarrow \quad \mathbb{E}[X] = p, \quad Var[X] = p(1-p)$$

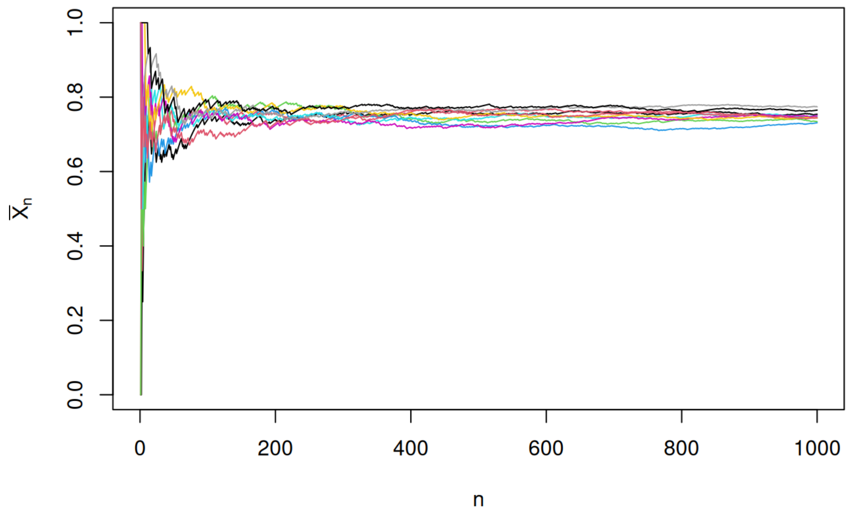
Also

$$\mathbb{E}[S_n] = nE[X] = np, \quad Var[S_n] = nVar[X] = np(1-p)$$

- **Law of Large numbers:**

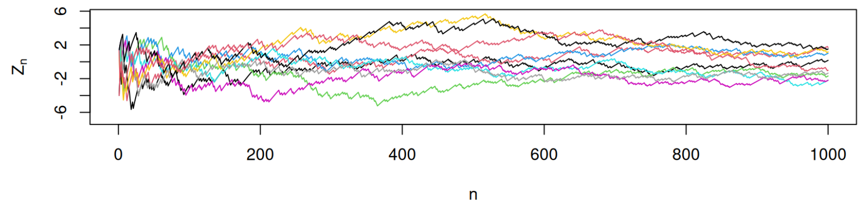
```
set.seed(42)
N = 1000
m = 20
p = 3/4
# Divide each sum by the corresponding sample size
sm_paths <- replicate(m, cumsum(rbinom(N, size =1, prob=p)) / (1:N))

# Plot the paths in the usual way
plot(sm_paths[,1], type = 'l', xlim = c(0, N), ylim = range(sm_paths), xlab =
"n", ylab = expression(bar(X)[n]))
for (i in 2:10) {
  lines(sm_paths[,i], col = i)
}
```



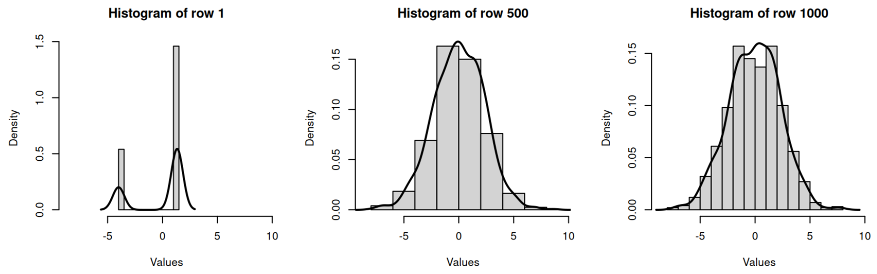
- **Central Limit Theorem:**

```
N = 1000
m = 20
lambda = 3
mean_s = (1:N)*p
sd_s = sqrt(1:N)*p*(1-p)
# Divide each sum by the corresponding sample size
z_paths <- replicate(m, (cumsum(rbinom(N, size =1, prob=p))-mean_s) / sd_s)
# Plot
plot(z_paths[,1], type = 'l', xlim = c(0, N), ylim = range(z_paths), xlab =
"n", ylab = expression(Z[n]))
for (i in 2:10) {
  lines(z_paths[,i], col = i)
}
```



```
N = 1000
m = 1000
lambda = 3
mean_s = (1:N)*p
sd_s = sqrt(1:N)*p*(1-p)
# Divide each sum by the corresponding sample size
z_paths <- replicate(m, (cumsum(rbinom(N, size =1, prob=p))-mean_s) / sd_s)

# Rows to plot
rows_to_plot <- c(1, N/2, N)
# Three histograms side by side
par(mfrow = c(1, 3))
# Histogram for each selected row
for (r in rows_to_plot) {
  hist(z_paths[r, ],
       freq = FALSE,# density scale
       main = paste("Histogram of row", r),
       xlab = "Values",
       xlim = range(z_paths))
  lines(density(z_paths[r, ]), lwd = 2)
}
```



Remember that a sum of Bernoullies follows a binomial distribution: $S_n \sim Binom(n, p)$. So what we are seeing is a standardized Binomial distribution can be approximated by a Standard Normal if the first parameter is large enough.