```
## Registering fonts with R
## Package "fontcm" already installed.
## Registering font package "fontcm" with fonts.
## Font package "fontcm" already registered in fonts database.
##
## Attaching package:  'dplyr'
## The following objects are masked from 'package:plyr':
##
##    arrange, count, desc, failwith, id, mutate, rename, summarise,
##    summarize
## The following objects are masked from 'package:stats':
##
##    filter, lag
## The following objects are masked from 'package:base':
##
##    intersect, setdiff, setequal, union
```

1 Description

The symetric random walk will be described in this document (Mt). it covers the theory of "Stochastic Calculus for finance" Tome 2 chapter 3 section 1.

The construction of the random walk depend on the evolution of a random variable X_i. The previous RV can take two value at each time, like tossing a coin. X_i can take the value 1 or -1.

$$X_i = \begin{cases} 1 \\ -1 \end{cases} \tag{1}$$

The Symetric Random Walk is constructed by summing up the different outcome of the random variable X_i from k experiments:

$$M_k = \sum_{j=1}^{k} X_j \tag{2}$$

In the following lines of code, X_i is randomly difined. The variable k ensure to have a sufficent number of periods to further generate the scaled random walk. It refers to the k of equation 2. p and q are the probability measure, respectively p chance to get value 1 and q chance to get -1 from random variable X_i.

After creating the random variable X_i it suffices to add up all the differente output we get from time 1 up to k to get a specific Symetric Random Walk.

The following outcome present a randomly generated 300 steps symmetric random walk.

Table 1: 300 steps Symmetric Random Walk

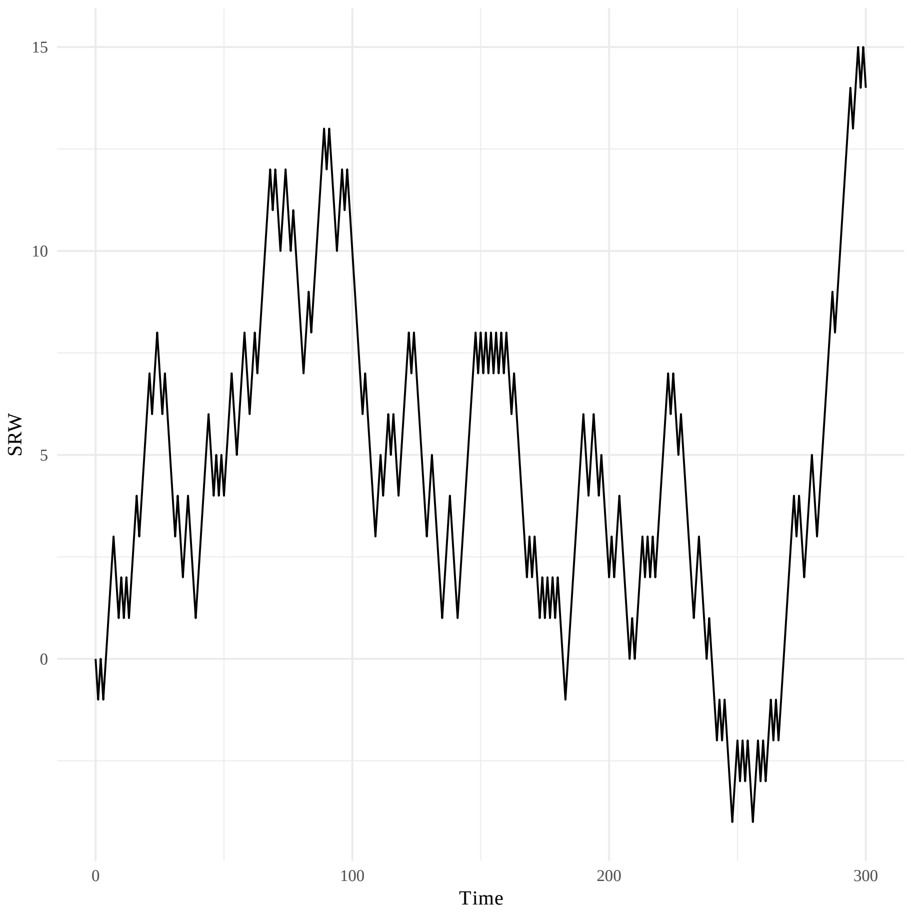


Figure 1: Symmetric Random Walk

```
# Because squared matrix dim(y) = dim(x):
dim_x <- dim_y <- 1:(k + 1) # from 1 to k+1 because we start to time zero nonrandom which e
# Create the symmetric random walk distribution:
# ifelse is vectorized and therefore is consistent with the using of outer which accept onl
Mk <- outer(dim_x,
            dim_y,
            FUN=function(r,c){ifelse(c>=r, (c-r) - (r-1), NA_integer_)})
colnames(Mk) <- paste0("F(", 1:ncol(Mk) - 1, ")")

# Create the Tex Table > tabular
Mk.tab <- xtable(Mk[1:10, 1:10], digits = 0, format = "latex")
Mk.tab
```

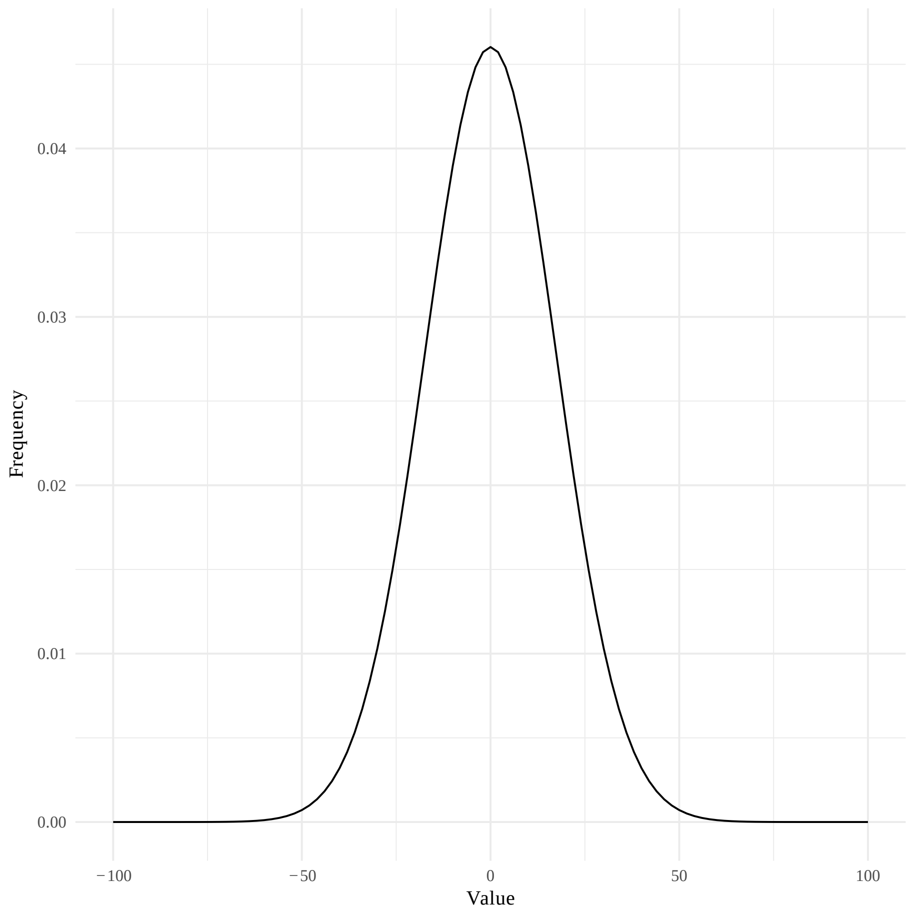
	F(0)	F(1)	F(2)	F(3)	F(4)	F(5)	F(6)	F(7)	F(8)	F(9)
1	0	1	2	3	4	5	6	7	8	9
2		-1	0	1	2	3	4	5	6	7
3			-2	-1	0	1	2	3	4	5
4				-3	-2	-1	0	1	2	3
5					-4	-3	-2	-1	0	1
6						-5	-4	-3	-2	-1
7							-6	-5	-4	-3
8								-7	-6	-5
9									-8	-7
10										-9

```r
# compute the probability measure to apply on the Random Variable Mk
fi <- outer(dim_x,
            dim_y,
            FUN = function(i, j){choose((j-1), (j-i)) * p^(j-1)})
```

```r
 range <- 1:ncol(Mk)
lastToss <- ncol(Mk)
 # Using ggplot
# data.frame which map distribution and random variable:
distributionSymRanWal <- data.frame(
  Value = Mk[range, lastToss],
  Frequency = fi[range, lastToss]
)

# For the sake of visibility the limit of X axis has been set to [-100, 100]
ggplot(data = distributionSymRanWal, aes(Value, Frequency)) +
  geom_line() +
  scale_x_continuous(limits = c(-100, 100)) +
   theme_minimal() +
  theme(text = element_text(family="CM Roman"),
        axis.title = element_text(face = "plain"))

## Warning: Removed 200 rows containing missing values (geom_path).
```



2 Martingale property

$$\mathbb{E}[X] = \sum_{i=1}^{N} p_i \times x_i \tag{3}$$

To show the martingale property we have to show that the expectation of the symmetric random walk $\mathbb{E}[X|F(0)] = M_0 = 0$

```
EM200 = sum(Mk[1:200, 200] * fi[1:200, 200]) # equal zero.
```

```
 ##
# Expectation of Mt_l at k
# denoted by: E[Mt_l|f(k)], with k < l
##
```

```r
##
# Variables
##
from <- 2 # Departure of the Expectation
k <- 4 # To get the filtration point
l <- 19 # Give the period to be expected
if(k>l)
interval <- l-k
##
# Partionated Symmetric Random Walk
##
# first the value of M at time £k£ has to be set. It means that the value at this time £k£ 
# Therefore at time £k£ the variable £M_k£ is not random.
# We can take any value we want to start with from 1 to £k + 1£.
# I choose to name this variable £from£:
from <- 2 # Departure of the Expectation
# The lenght of the path is already know: £(l - k)£:
len <- l - k
i <- from:(from + len)
j <- k:l
df <- Mk[i, j]
# names(df) <- sapply(1:ncol(df), function(x){paste0("X",x)})
##
# Con
##

# The probability from k to l start from k to l. The probability table has therefore to be 
fi_min <- fi[1:(len+1), 1:(len+1)]
# Old calculation of fi:
#
# fi <- data.frame(matrix(rep(0, (l-k + 1)^2), nrow = l-k + 1))
# for(j in 1:(l-k+1))
#   for(i in 1:j)
#     fi[i, j] <- choose((j-1), (i-1)) * p^(j-1)#((j-1)*p^(j-1))/(factorial(j-1)*factorial(
# Finally compute the expectation E[Mt_l|f(k)]:
Mk[from, k] == sum(df[, l-k+1] * fi_min[, l-k+1]) #Yeah it is a matringale

## F(3)
## TRUE
```