GETTING STARTED WITH S

Open R, then...

```
R : Copyright 2002, The R Development Core Team
Version 1.5.0  (2002-04-29)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type ‘license()’ or ‘licence()’ for distribution details.

R is a collaborative project with many contributors.
Type ‘contributors()’ for more information.

Type ‘demo()’ for some demos, ‘help()’ for on-line help, or
‘help.start()’ for a HTML browser interface to help.
Type ‘q()’ to quit R.

>
```

TO FIND OUT ABOUT THE BINOMIAL DISTRIBUTION

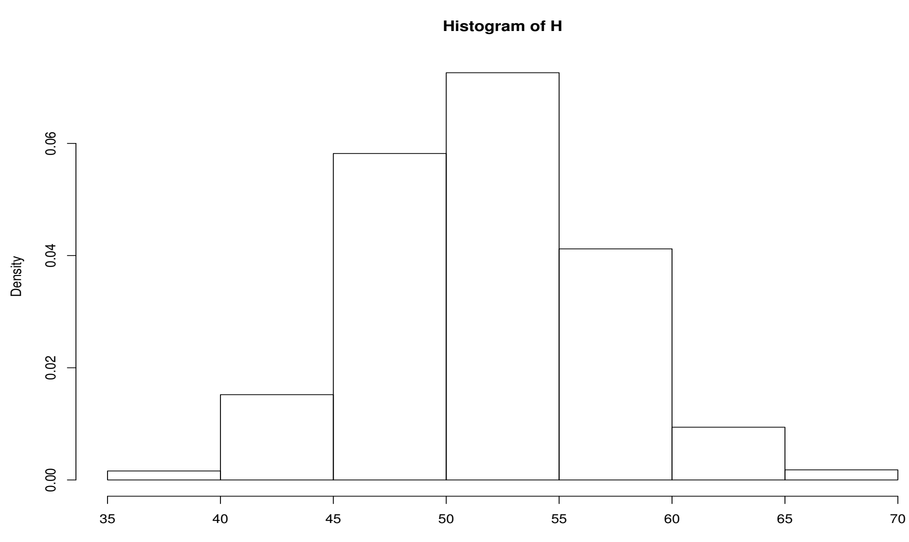
```
> help(rbinom)

Binomial                package:base                R Documentation

The Binomial Distribution

Description:

     Density, distribution function, quantile function and random
     generation for the binomial distribution with parameters `size'
     and `prob'.

Usage:

     dbinom(x, size, prob, log = FALSE)
     pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
     qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
     rbinom(n, size, prob)

Arguments:

    x, q: vector of quantiles.

       p: vector of probabilities.
```

and so on... scroll down.. give “q” to quit help

```
> n<-100
# 100 time periods
# comments are, incidentally, preceeded by a "#"

> r<-log(1.05)/n
# interest rate for the whole period is therefore given by
> exp(r*n)
[1] 1.05

> u<-1.01
> d<-0.99
# This defines the up and down movements
# the most extreme outcomes are
> u^100
[1] 2.704814
# and, of course,
> d^100
[1] 0.3660323

# since we do a lot of discounting: u-tilde and d-tilde
> ut <- exp(-r)*u
> dt <- exp(-r)*d

# the risk neutral probabilities are
> piH <- (1-dt)/(ut -dt)
> piT <- (ut-1)/(ut -dt)
# and so
> piH
[1] 0.524401

# we take out initial stock price to be
> S0 <- 100
# this sets up our basic system
```

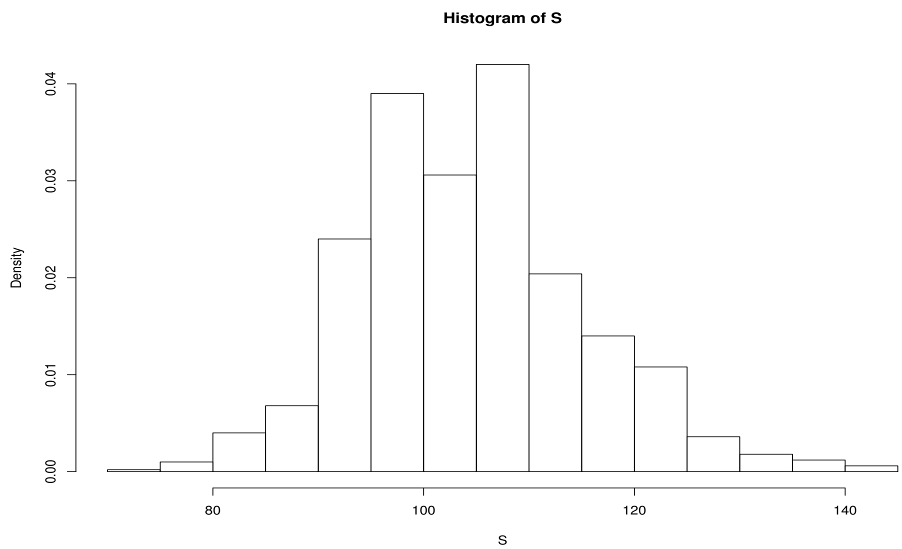
LET'S HAVE A LOOK AT WHAT OUR DISTRIBUTIONS LOOK LIKE

```
# let's generate random numbers representing the number of heads

# we generate a sample of 1000 realizations of H
> M<-1000
> H<- rbinom(M,n,piH)

> hist(H,freq=F)

# this produces the graph below
# to find out about the hist function, give the command
> help(hist)

# to save a copy of your graph, you can for example do this

> postscript("histogram.ps")
> hist(H,freq=F)
> graphics.off()
```



WHAT ABOUT THE DISTRIBUTION OF S_n?

```
# S-tilde and S
> St<- S0 * exp(H*log(ut/dt) + n*log(dt))
> S <- exp(r*n)*St
> hist(S,freq=F)
```



PRICING THE CALL OPTION
EXACT CALCULATION

```
# strike price
K <- 105

# the probability distribution of H:
# the outcome space:
omega <- c(0:100)
> omega
  [1]   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
 [19]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
 [37]  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52
 [55]  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70
 [73]  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88
 [91]  90  91  92  93  94  95  96  97  98  99 100
# probabilities of the outcomes:
> p <- dbinom(omega,n,piH)
# consists of 101 elements
# this is indeed a probability
> sum(p)
[1] 1

# the outcomes for the stock price (101 elements)
> st<- S0 * exp(omega*log(ut/dt) + n*log(dt))
> s <- exp(r*n)*st

# the payoff of the call option
> v <- pmax(s-K,0)
# max(s-K,0) gives a different answer

# the price (the discounted risk neutral expectation)
> exp(-r*n) * sum(v*p)
[1] 3.989224
```

PRICING THE CALL OPTION
MONTE CARLO SIMULATION

IF $S^{(1)}, ..., S^{(M)}$ ARE IID COPIES WITH THE DISTRIBUTION OF S, THEN

$$\hat{E}h(S) = \frac{1}{M}\sum_{i=1}^{M} h(S^{(i)})$$

APPROXIMATES THE EXPECTATION

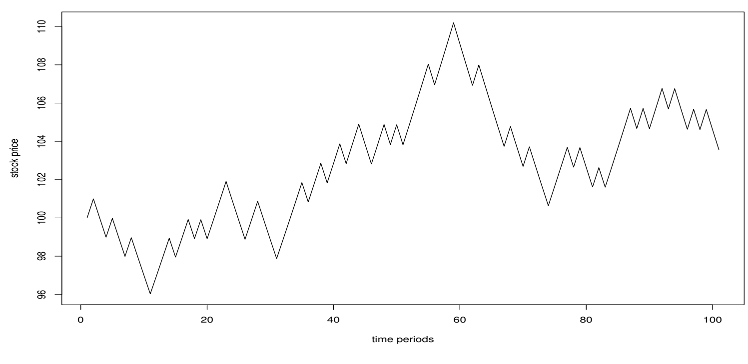
$$\hat{E}h(S) \simeq Eh(S)$$

```
# our pi is a risk neutral measure: E(S) is estimated to be
> exp(-r*n)*mean(S)
[1] 99.72981

# the options payoff
> V<-pmax(S-K,0)
# the estimated options price
> exp(-r*n)*mean(V)
[1] 3.991318

# by comparison to the true price
> 3.991318/3.989224
[1] 1.000525
```

THE APPROXIMATION GETS BETTER AS $M \to \infty$

HOW BIG DOESW M NEED TO BE?
HOW DOES ONE ASSESS THE ERROR?
A LATER LECTURE

```
# generating one realization of a path
# the process of heads and tails
> II <- rbinom(n,1,piH)
# H is now a process
> HH <- cumsum(II)
> HH
 [1]  1  1  1  2  2  2  3  3  3  3  4  5  6  6  7  8  8  9  9 10 11 12 12
[26] 13 14 14 14 14 15 16 17 18 18 19 20 20 21 22 22 23 24 24 24 25 26 26
[51] 28 29 30 31 31 32 33 34 34 34 34 35 35 35 35 35 36 36 36 37 37 37 37
[76] 40 40 41 41 41 42 42 43 44 45 46 46 47 47 48 49 49 50 50 50 51 51 52
# to start thye cumulative process in 0:
> HH <- c(0,HH)
# S-tilde and S
> SSt<- S0 * exp(HH*log(ut/dt) + c(0:100)*log(dt))
> SS <- exp(r*c(0:100))*SSt
# the maximum
> MM <- max(SS)

# have a look at the process
> plot(SS,type="l",xlab="time periods",ylab="stock price")
```



GENERATING MANY REALIZATIONS

```
# cumulate final values of the stock price in SSS,
# final values of the maximum in MMM

# first initialize
SSS <- c(1:M)*0
MMM <- c(1:M)*0

#the loop

for(i in 1:M){
II <- rbinom(n,1,piH)
HH <- cumsum(II)
HH <- c(0,HH)
SSt<- S0 * exp(HH*log(ut/dt) + c(0:100)*log(dt))
SS <- exp(r*c(0:100))*SSt
SSS[i]<-SS[n]
MMM[i] <- max(SS)
}

# the lookback option with strike K=1.1:
K <- 1.1
# payoff
V <- pmax(MMM/SSS - K ,0)
# price
exp(-r*n)*mean(V)
[1] 0.0097791

# or you can use the apply function in S

# or you can build a tree
```

RADON-NIKODYM DERIVATIVES: IMPORTANCE SAMPLING

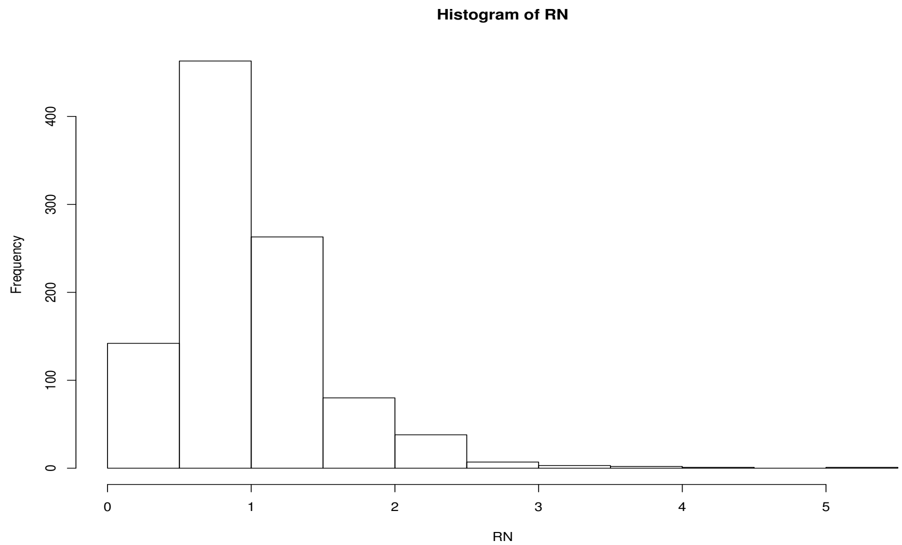
Often convenient to simulate under a different measure

$$E_\pi(S_n - K)^+ = E_Q(S_n - K)^+ \frac{d\pi}{dQ}$$

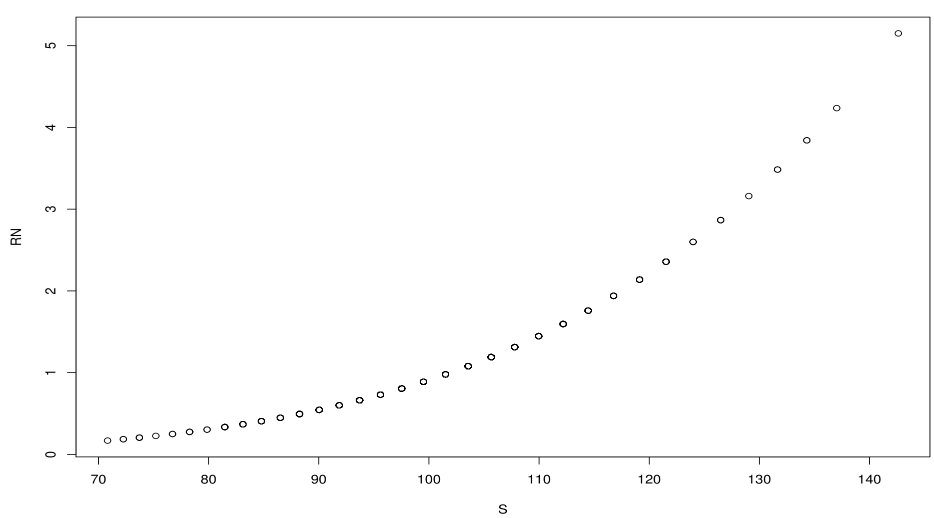
For example, in the previous sampling scheme, suppose we sample the heads and tail under a fair coin, so $Q(H) = Q(T) = 1/2$:

$$\frac{d\pi}{dQ} = \left(\frac{\pi(H)}{Q(H)}\right)^{\#H} \left(\frac{\pi(T)}{Q(T)}\right)^{\#T}$$

```
# the fair coin probabilities
QH <- 1/2
QT <- 1/2
# sample under these probabilities
H <- rbinom(M,n,QH)

# S-tilde and S
> St<- S0 * exp(H*log(ut/dt) + n*log(dt))
> S <- exp(r*n)*St

# and... the Radon-Nikodym derivative
> RN <- ((piH/QH)^H)*((piT/QT)^(n-H))

# the Radon-Nikodym derivative is a random variable
hist(RN)
```




```
# it has mean 1
> mean(RN)
[1] 1.011647

# and the R-N derivative is a function  of S
> plot(S,RN)
```



USING THE RELATION

$$E_\pi(S_n - K)^+ = E_Q(S_n - K)^+ \frac{d\pi}{dQ}$$

```
# strike price
> K <- 105
# the options payoff
> V<-pmax(S-K,0)
# the estimated options price
> exp(-r*n)*mean(V*RN)
[1] 4.075278
# by comparison to the true price
> 4.075278/3.989224
[1] 1.021573
```

This is more off, but will get better as $M \to \infty$

```
# first initialize
SSS <- c(1:M)*0
MMM <- c(1:M)*0
HHH <- c(1:M)*0

#the loop
for(i in 1:M){
II <- rbinom(n,1,QH)
HH <- cumsum(II)
HH <- c(0,HH)
SSt<- S0 * exp(HH*log(ut/dt) + c(0:100)*log(dt))
SS <- exp(r*c(0:100))*SSt
SSS[i]<-SS[n]
MMM[i] <- max(SS)
HHH[i] <- HH[n]
}
# the Radon-Nikodym derivative
> RN <- ((piH/QH)^HHH)*((piT/QT)^(n-HHH))
# a diagnostic, should be close to 1:
> mean(RN)
[1] 0.9582251

# the lookback option with strike K=1.1:
> K <- 1.1
# payoff
> V <- pmax(MMM/SSS - K ,0)
# price
> exp(-r*n)*mean(V*RN)
[1] 0.008485798
# Ratio to simulation without the RN-derivative
> 0.008485798/0.0097791
[1] 0.8677484 # not great, but you get the idea
```