Fixed Income:Black_Derman_Toy Model

YiTao Hu, Charles Rambo, Junyu(Kevin) Wu, Jin (Jane) Huangfu

20/02/2020

```
#import data
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 3.5.2
```

```
vol_ts=read_excel("Homework 6 voldat.xlsx",col_names = FALSE)
```

```
## New names:
## * `` -> ...1
```

```
vol_ts=vol_ts$...1
True_DTs = read_excel("Homework 6 pfilea.xlsx", col_names = FALSE)
```

```
## New names:
## * `` -> ...1
```

```
True_DTs=True_DTs$...1
```

Question 1

initizliation of interest rate tree

We first need to randomly initialize the short rate r_t^* at each point of time t.

```
#initizlize R*
steps=29
r_stars=abs(rnorm(n = (steps+1))/100)
```

From r_t^*, we can map all the interest rate tree with the folloing formula:

$$r_t^{id} = r_t^* e^{-2i\sigma_t\sqrt{0.5}}$$

where i is the number of doward movements from the top state. We can define a function to do the computation.

```
#define a function to compute R^id from R^*
get_Rid=function(r_star,id,vol){
  rid=r_star*exp(-2*id*vol*0.5^0.5)
  return (rid)
}
get_all_Rids=function(r_stars,vol_ts){
  #initialize the rate tree dataframe
  rateTree=data.frame(matrix(0,nrow = length(r_stars),ncol = length(r_stars)))
  colnames(rateTree)=seq(0,(length(r_stars)-1)/2,0.5)
  rateTree[1,]=r_stars

  #compute and store the value of rateTree
for (t in 2:ncol(rateTree)){
  for(i in 2:t){
    rateTree[i,t]=get_Rid(r_star = r_stars[t],id = i,vol = vol_ts[(t-1)])
  }
```

```
}
  return (rateTree)
}

Rates_tree=get_all_Rids(r_stars = r_stars,vol_ts = vol_ts)
```

Mapping interest rate tree to discount factor tree and compute recursively discount factor curve

Given the short interest rate tree, we can map the interest rate tree to a discount factor tree of 0.5 year maturity at each state s and each time step t because short term discount factor is an one-to-one mapping relationship. In particular, we will use the following formula to compute discount factor

$$D^{id}(t, 0.5) = \frac{1}{1 + r^{id}/2}$$

where id refers to the number of downward movements or state, t refers to the time step.

```
get_short_DT_tree=function(Rates_tree){
  DT_short_tree=1/(1+Rates_tree/2)
  return (DT_short_tree)
}
DT_short_tree=get_short_DT_tree(Rates_tree = Rates_tree)
```

Given the short-term discount factor tree, we can compute the cash flow tree or discount factor tree with various maturity using the following backward recursive scheme for each $D(0, T)$ of T=0.5,1,. . . ,15

$$D^s(t, T) = D^s(t, 0.5)\frac{D^{su}(t + 0.5, T - 0.5) + D^{sd}(t + 0.5, T - 0.5)}{2}$$

where s denotes the state, su denotes upward state and sd denotes doward state one step later. This formula, in essence, is equivalent to the recusive scheme in the TA notes with interest rate.

```
#define a function to compute each individual D(0,T)
compt_DOT=function(DT_short_tree,Mar){
  #initialize the matrix of D^s(t,T) matrix
  DsT_mat=DT_short_tree[1:Mar,1:Mar]
  #perform the backward inductive scheme
  for (t in Mar:2){
    for (i in 1:(t-1)){
      DsT_mat[i,(t-1)]=DsT_mat[i,(t-1)]*mean(c(DsT_mat[i,t],DsT_mat[(i+1),t]))
    }
  }
  return (DsT_mat[1,1])}
#define a function to compute all the D(0,T)
comp_all_DOT=function(DT_short_tree){
  #initialize a vector of 30 discount factor at time 0
  fitted_DTs=rep(0,ncol(DT_short_tree))
  fitted_DTs[1]=DT_short_tree[1,1]
  for (Mar in 2:length(fitted_DTs)){
    fitted_DTs[Mar]=compt_DOT(DT_short_tree = DT_short_tree,Mar = Mar)
  }
return (fitted_DTs)
}

fitted_DTs=comp_all_DOT(DT_short_tree = DT_short_tree)
```

Define Cost function and Perform Optimization to get the numerical solution

Because it is complicated to solve all the r_t^* analytically, we turn to find its numerical solution by minimizing the Mean Average Error between the fitted DTs and the true DTs.

We, first, need to defined a weighted sum error function (because the algorithm overweighs the first parameter and underweighs the last one, we want to do a sigmoid-like transformation to reverse this weighting):

```
compt_weightedError=function(True_values,fitted_values){
  WE=sum((1/(1+exp(-seq(-1.5,1.5,3.0/29))))*abs(fitted_values-True_values))
  #WE=sum(seq(2,3,1/29)*abs(fitted_values-True_values))
  return (WE)
}
WE=compt_weightedError(True_values = True_DTs,fitted_values = fitted_DTs)
```

Then, we can assimble all the function into a pipeline as a cost function or optimization objective.

```
Cost_Fun=function(r_stars,vol_ts,True_DTs){
  Rates_tree=get_all_Rids(r_stars = r_stars,vol_ts = vol_ts)
  DT_short_tree=get_short_DT_tree(Rates_tree = Rates_tree)
  fitted_DTs=comp_all_DOT(DT_short_tree = DT_short_tree)
  WE=compt_weightedError(True_values = True_DTs,fitted_values = fitted_DTs)
  return (WE)
}
Cost_Fun(r_stars = r_stars,vol_ts = vol_ts ,True_DTs = True_DTs)
```

```
## [1] 6.36783
```

Finally, we can feed the Cost Function into optimizer and try to find the optimal numerical solution.

```
#Num_sol=optim(par = r_stars,fn = Cost_Fun,method = "BFGS",vol_ts=vol_ts,True_DTs=True_DTs,control = li
Num_sol=nlminb(start = r_stars,
               objective =Cost_Fun,vol_ts=vol_ts,True_DTs=True_DTs,lower = rep(0,30),control = list(tra
```

```
##   0:     6.3678303: 0.000931861 0.00480785 0.000494980 0.00274852 0.0127616 0.0177531 0.00280396 0.0
##   1:     2.3043081: 0.514306 0.448450 0.378584 0.322810 0.279511 0.241953 0.197891 0.170759 0.155017
##   2:     1.6260216: 0.467943 0.373225 0.284612 0.221912 0.181396 0.152708 0.118262 0.101247 0.094134
##   3:     1.1703923: 0.191793 0.127592 0.0759497 0.0571060 0.0632869 0.0790710 0.0838601 0.101302 0.1
##   4:    0.65307252: 0.108561 0.117519 0.113071 0.121188 0.142308 0.167942 0.183282 0.209556 0.243017
##   5:    0.61473080: 0.0849975 0.0985575 0.0981356 0.109684 0.133749 0.161690 0.178731 0.206433 0.241
##   6:    0.54903374: 0.0771497 0.0894130 0.0951599 0.115072 0.138481 0.172313 0.195099 0.221244 0.257
##   7:    0.48101012: 0.0491039 0.0523234 0.0683672 0.105101 0.125059 0.172126 0.206942 0.228035 0.265
##   8:    0.42748744: 0.0569472 0.0689308 0.0792129 0.106539 0.127827 0.155908 0.171689 0.238563 0.278
##   9:    0.39067471: 0.0432457 0.0751630 0.0827439 0.0919493 0.122525 0.121216 0.123807 0.244275 0.28
##  10:    0.32834295: 0.0291569 0.110988 0.108659 0.0741720 0.125518 0.0685919 0.0406312 0.230782 0.26
##  11:    0.23542673:  0.00000 0.0784052 0.100948 0.0941957 0.135231 0.165803 0.129730 0.150292 0.1582
##  12:    0.23481205:  0.00000 0.116951 0.133990 0.0739275 0.147742 0.148366 0.140861 0.113145 0.12667
##  13:    0.15983165: 0.0342118 0.113223 0.132209 0.0557327 0.138497 0.110571 0.231330 0.136870 0.1516
##  14:    0.13680457: 0.0364552 0.105165 0.134288 0.0644952 0.142189 0.113874 0.203597 0.139862 0.1644
##  15:    0.11114307: 0.0512474 0.0962522 0.118456 0.0466778 0.141656 0.0868590 0.248636 0.130676 0.15
##  16:    0.10078679: 0.0646099 0.0895604 0.0913247 0.0239923 0.145908 0.0701817 0.292686 0.118388 0.1
##  17:   0.081048531: 0.0448732 0.0917963 0.0886930 0.0515193 0.147655 0.125649 0.201681 0.135375 0.14
##  18:   0.065580854: 0.0337529 0.0824894 0.0948192 0.0725789 0.165547 0.152005 0.120091 0.171528 0.16
##  19:   0.054505282: 0.0488829 0.0694678 0.0931334 0.0677060 0.151124 0.146395 0.178598 0.178882 0.16
##  20:   0.041705343: 0.0539621 0.0652466 0.101356 0.0665452 0.141473 0.125005 0.166671 0.210480 0.211
##  21:   0.037689601: 0.0488645 0.0746186 0.0762479 0.0876477 0.122173 0.185387 0.130445 0.188038 0.21
##  22:   0.036917554: 0.0556445 0.0837528 0.0596194 0.0832517 0.0992994 0.193577 0.145932 0.186831 0.2
##  23:   0.030831915: 0.0604754 0.0820786 0.0693095 0.0778316 0.0883932 0.182444 0.174667 0.178189 0.2
```

```
##  24:   0.025531470: 0.0586710 0.0788378 0.0818910 0.0779068 0.0930689 0.168029 0.179849 0.179723 0.18
##  25:   0.022391052: 0.0516775 0.0781484 0.0964350 0.0848109 0.106787 0.154179 0.153966 0.182117 0.189
##  26:   0.022195007: 0.0484243 0.0663160 0.111349 0.0901576 0.121882 0.141571 0.141108 0.188181 0.1994
##  27:   0.019657653: 0.0495354 0.0640278 0.109129 0.0904210 0.123416 0.140477 0.139928 0.190459 0.2027
##  28:   0.016077542: 0.0537782 0.0615139 0.0944843 0.0902558 0.123699 0.144710 0.152838 0.189621 0.210
##  29:   0.015336454: 0.0564924 0.0619383 0.0880500 0.0884607 0.126716 0.141199 0.161924 0.186970 0.206
##  30:   0.013283105: 0.0588131 0.0657194 0.0819239 0.0856558 0.130975 0.137984 0.163042 0.185308 0.205
##  31:   0.012598625: 0.0574456 0.0663538 0.0799647 0.0881408 0.136571 0.138510 0.160500 0.184813 0.201
##  32:   0.012529792: 0.0575137 0.0664181 0.0800153 0.0881801 0.136600 0.138539 0.160522 0.184829 0.201
##  33:   0.012331561: 0.0575177 0.0661827 0.0807476 0.0882347 0.136067 0.138064 0.161109 0.184718 0.201
##  34:   0.012093869: 0.0573294 0.0663961 0.0809344 0.0884375 0.135358 0.138061 0.161130 0.184724 0.201
##  35:   0.011886366: 0.0572331 0.0667319 0.0811085 0.0887049 0.134735 0.138196 0.161072 0.184307 0.201
##  36:   0.011404139: 0.0568287 0.0668851 0.0823261 0.0897944 0.132297 0.138475 0.162451 0.181890 0.204
##  37:   0.010637809: 0.0561527 0.0674264 0.0826556 0.0920514 0.128175 0.138972 0.163975 0.181742 0.206
##  38:   0.010166208: 0.0560666 0.0675500 0.0822487 0.0940743 0.123910 0.140990 0.163849 0.180033 0.208
##  39:  0.0094137079: 0.0567254 0.0666405 0.0830968 0.0955443 0.121457 0.138820 0.166073 0.180159 0.209
##  40:  0.0088519526: 0.0564669 0.0666954 0.0831444 0.0976732 0.121169 0.136900 0.165065 0.181553 0.209
##  41:  0.0083657157: 0.0564553 0.0661529 0.0833732 0.0998220 0.119065 0.137060 0.164357 0.183057 0.207
##  42:  0.0082152364: 0.0568699 0.0655762 0.0820222 0.101454 0.118693 0.138046 0.165950 0.182000 0.2055
##  43:  0.0078227322: 0.0565394 0.0655561 0.0827045 0.102520 0.116781 0.140166 0.165249 0.183099 0.2023
##  44:  0.0075888330: 0.0563346 0.0651529 0.0826756 0.102246 0.117223 0.141631 0.164742 0.182456 0.2001
##  45:  0.0072231119: 0.0567442 0.0647313 0.0829702 0.100674 0.119213 0.140837 0.164220 0.182912 0.2005
##  46:  0.0066130780: 0.0568034 0.0650400 0.0827705 0.0989156 0.121429 0.140468 0.162322 0.183776 0.201
##  47:  0.0063968062: 0.0566429 0.0656028 0.0824978 0.0982340 0.122236 0.140894 0.160160 0.184556 0.204
##  48:  0.0061506982: 0.0567870 0.0660762 0.0823192 0.0977331 0.121310 0.140824 0.160871 0.186150 0.204
##  49:  0.0058162108: 0.0566809 0.0667397 0.0822561 0.0977492 0.119990 0.140323 0.162047 0.185052 0.206
##  50:  0.0055504657: 0.0562890 0.0678276 0.0828530 0.0980074 0.118731 0.139887 0.162943 0.184246 0.204
##  51:  0.0054089724: 0.0563132 0.0664158 0.0830700 0.0989595 0.117059 0.146730 0.159011 0.182492 0.207
##  52:  0.0053383198: 0.0572487 0.0652428 0.0845070 0.0971219 0.118247 0.143496 0.159663 0.182306 0.207
##  53:  0.0050013806: 0.0573011 0.0651052 0.0844034 0.0969514 0.119688 0.141419 0.161183 0.182361 0.205
##  54:  0.0041488368: 0.0569276 0.0658913 0.0840297 0.0965070 0.121168 0.140500 0.161791 0.182938 0.203
##  55:  0.0038766920: 0.0568606 0.0663265 0.0835581 0.0963893 0.121189 0.141272 0.160928 0.183978 0.203
##  56:  0.0036144162: 0.0569746 0.0669094 0.0824179 0.0971073 0.120747 0.140971 0.161560 0.182785 0.205
##  57:  0.0033592416: 0.0566842 0.0676193 0.0815518 0.0976831 0.120682 0.140777 0.161747 0.182217 0.205
##  58:  0.0031921637: 0.0560577 0.0679630 0.0813812 0.0994262 0.119042 0.141005 0.161802 0.181728 0.205
##  59:  0.0029008445: 0.0558669 0.0685156 0.0812744 0.0989108 0.118831 0.140516 0.161738 0.182162 0.205
##  60:  0.0024478092: 0.0561433 0.0678468 0.0817179 0.0984519 0.119793 0.141027 0.160976 0.182652 0.204
##  61:  0.0017437352: 0.0567011 0.0668291 0.0825577 0.0990262 0.117379 0.144546 0.159380 0.183695 0.203
##  62:  0.0014310807: 0.0567706 0.0667984 0.0824442 0.0990924 0.117297 0.144944 0.158976 0.184299 0.203
##  63:  0.0012354812: 0.0566700 0.0676537 0.0812963 0.0990581 0.118495 0.143456 0.158822 0.186171 0.204
##  64:  0.0010592017: 0.0564562 0.0681895 0.0810854 0.0985837 0.119118 0.141988 0.159882 0.185352 0.205
##  65:  0.0010052297: 0.0564362 0.0681708 0.0810705 0.0985719 0.119109 0.141981 0.159877 0.185347 0.205
##  66:  0.0010047167: 0.0564295 0.0681615 0.0810624 0.0985688 0.119113 0.141975 0.159882 0.185342 0.205
##  67: 0.00099784861: 0.0564339 0.0681651 0.0810628 0.0985704 0.119119 0.141974 0.159890 0.185340 0.205
##  68: 0.00099710671: 0.0564374 0.0681658 0.0810629 0.0985739 0.119127 0.141973 0.159899 0.185339 0.205
##  69: 0.00099191776: 0.0564330 0.0681616 0.0810574 0.0985700 0.119129 0.141967 0.159904 0.185334 0.205
##  70: 0.00099132473: 0.0564299 0.0681574 0.0810550 0.0985691 0.119129 0.141966 0.159904 0.185334 0.205
##  71: 0.00098919061: 0.0564315 0.0681574 0.0810562 0.0985712 0.119132 0.141968 0.159906 0.185335 0.205
##  72: 0.00098804889: 0.0564284 0.0681535 0.0810526 0.0985694 0.119134 0.141965 0.159909 0.185332 0.205
##  73: 0.00098561435: 0.0564311 0.0681549 0.0810533 0.0985716 0.119136 0.141962 0.159913 0.185331 0.205
##  74: 0.00098338047: 0.0564304 0.0681531 0.0810501 0.0985716 0.119139 0.141956 0.159919 0.185326 0.205
##  75: 0.00095564557: 0.0564481 0.0681553 0.0810061 0.0985817 0.119187 0.141830 0.160040 0.185237 0.205
##  76: 0.00092278480: 0.0564985 0.0681211 0.0810136 0.0985850 0.119215 0.141693 0.160123 0.185196 0.205
##  77: 0.00086380492: 0.0565738 0.0679805 0.0810234 0.0987023 0.119084 0.141320 0.160448 0.185232 0.204
```

```
##  78: 0.00083830692: 0.0567196 0.0679386 0.0810026 0.0984088 0.119254 0.141281 0.161114 0.185139 0.203
##  79: 0.00071808705: 0.0567568 0.0679032 0.0809336 0.0985672 0.119081 0.141468 0.161203 0.184893 0.203
##  80: 0.00057658521: 0.0566817 0.0677755 0.0810344 0.0988739 0.118857 0.141948 0.161034 0.184458 0.203
##  81: 0.00050954918: 0.0564068 0.0680340 0.0811182 0.0989906 0.118607 0.142253 0.161266 0.183779 0.203
##  82: 0.00047309437: 0.0563879 0.0679395 0.0813284 0.0988689 0.118628 0.142185 0.161230 0.183676 0.203
##  83: 0.00042360489: 0.0564491 0.0678776 0.0813960 0.0986882 0.118876 0.141883 0.161331 0.183639 0.204
##  84: 0.00039467900: 0.0565252 0.0678240 0.0813186 0.0986523 0.119258 0.141586 0.161378 0.183679 0.204
##  85: 0.00035284963: 0.0566004 0.0676918 0.0813139 0.0986789 0.119324 0.141310 0.161474 0.183897 0.204
##  86: 0.00029695535: 0.0566978 0.0675309 0.0812148 0.0988653 0.119356 0.141295 0.161625 0.183931 0.204
##  87: 0.00023593121: 0.0566683 0.0675543 0.0812407 0.0988782 0.119387 0.141212 0.161581 0.184017 0.204
##  88: 0.00022052813: 0.0566268 0.0674469 0.0814599 0.0988298 0.119281 0.141406 0.161440 0.184181 0.204
##  89: 0.00018133506: 0.0565879 0.0674065 0.0816663 0.0986437 0.119080 0.141669 0.161483 0.184156 0.203
##  90: 0.00017431264: 0.0565951 0.0674322 0.0817616 0.0984431 0.119077 0.141956 0.161276 0.183959 0.204
##  91: 0.00015122315: 0.0566100 0.0675163 0.0816582 0.0984672 0.119033 0.141921 0.161501 0.183677 0.204
##  92: 0.00012276298: 0.0566125 0.0674910 0.0816446 0.0985313 0.119067 0.141855 0.161500 0.183578 0.204
##  93: 0.00012263571: 0.0566118 0.0674905 0.0816443 0.0985310 0.119067 0.141855 0.161500 0.183578 0.204
##  94: 0.00012223007: 0.0566117 0.0674905 0.0816444 0.0985313 0.119067 0.141855 0.161500 0.183578 0.204
##  95: 0.00012204286: 0.0566109 0.0674900 0.0816441 0.0985314 0.119067 0.141855 0.161500 0.183578 0.204
##  96: 0.00012163276: 0.0566110 0.0674901 0.0816442 0.0985318 0.119068 0.141855 0.161500 0.183579 0.204
##  97: 0.00012153953: 0.0566110 0.0674901 0.0816443 0.0985319 0.119068 0.141855 0.161500 0.183579 0.204
##  98: 0.00012138875: 0.0566106 0.0674898 0.0816442 0.0985319 0.119068 0.141855 0.161500 0.183579 0.204
##  99: 0.00012126005: 0.0566105 0.0674899 0.0816443 0.0985321 0.119068 0.141855 0.161500 0.183579 0.204
## 100: 0.00012112317: 0.0566103 0.0674897 0.0816441 0.0985320 0.119068 0.141855 0.161500 0.183579 0.204
## 101: 0.00012098919: 0.0566102 0.0674896 0.0816442 0.0985322 0.119068 0.141855 0.161500 0.183579 0.204
## 102: 0.00011918918: 0.0566092 0.0674878 0.0816434 0.0985354 0.119071 0.141854 0.161500 0.183579 0.204
## 103: 0.00011868179: 0.0566082 0.0674870 0.0816429 0.0985350 0.119071 0.141854 0.161500 0.183579 0.204
## 104: 0.00011857064: 0.0566084 0.0674872 0.0816431 0.0985353 0.119071 0.141854 0.161500 0.183579 0.204
## 105: 0.00011483281: 0.0566034 0.0674829 0.0816409 0.0985429 0.119077 0.141847 0.161508 0.183570 0.204
## 106: 0.00011167277: 0.0566062 0.0674802 0.0816388 0.0985486 0.119085 0.141840 0.161510 0.183567 0.204
## 107: 0.00010442762: 0.0566059 0.0674661 0.0816301 0.0985827 0.119102 0.141788 0.161541 0.183593 0.204
## 108: 7.9480667e-05: 0.0566180 0.0674077 0.0816241 0.0986286 0.119144 0.141718 0.161562 0.183613 0.204
## 109: 6.4382431e-05: 0.0566138 0.0674202 0.0816110 0.0986220 0.119182 0.141650 0.161605 0.183623 0.204
## 110: 6.3513540e-05: 0.0566144 0.0674208 0.0816116 0.0986224 0.119182 0.141650 0.161605 0.183623 0.204
## 111: 6.3413808e-05: 0.0566142 0.0674207 0.0816115 0.0986223 0.119182 0.141650 0.161605 0.183623 0.204
## 112: 6.3307312e-05: 0.0566143 0.0674208 0.0816116 0.0986224 0.119182 0.141649 0.161605 0.183623 0.204
## 113: 5.4567734e-05: 0.0566065 0.0674305 0.0816088 0.0986253 0.119209 0.141597 0.161628 0.183646 0.204
## 114: 5.3663061e-05: 0.0566056 0.0674277 0.0816151 0.0986090 0.119218 0.141538 0.161661 0.183690 0.204
## 115: 4.9952977e-05: 0.0566047 0.0674350 0.0816132 0.0986006 0.119227 0.141525 0.161656 0.183709 0.204
## 116: 4.1907727e-05: 0.0566088 0.0674354 0.0816252 0.0985793 0.119227 0.141526 0.161629 0.183749 0.204
## 117: 4.1097911e-05: 0.0566004 0.0674507 0.0816332 0.0985646 0.119211 0.141546 0.161593 0.183788 0.204
## 118: 4.1048373e-05: 0.0566003 0.0674506 0.0816331 0.0985647 0.119211 0.141546 0.161593 0.183787 0.204
## 119: 4.1019740e-05: 0.0566004 0.0674507 0.0816332 0.0985648 0.119211 0.141546 0.161593 0.183788 0.204
## 120: 3.8486832e-05: 0.0566033 0.0674407 0.0816351 0.0985770 0.119203 0.141558 0.161573 0.183802 0.204
## 121: 3.6063613e-05: 0.0566056 0.0674350 0.0816368 0.0985867 0.119195 0.141570 0.161549 0.183829 0.204
## 122: 3.5955807e-05: 0.0566050 0.0674363 0.0816391 0.0985883 0.119193 0.141577 0.161543 0.183834 0.204
## 123: 3.3494257e-05: 0.0566033 0.0674340 0.0816409 0.0985899 0.119189 0.141581 0.161542 0.183833 0.204
## 124: 3.3494257e-05: 0.0566033 0.0674340 0.0816409 0.0985899 0.119189 0.141581 0.161542 0.183833 0.204
```

```
R_star_num=Num_sol$par
R_star_num
```

```
##  [1] 0.05660334 0.06743404 0.08164093 0.09858989 0.11918938 0.14158090
##  [7] 0.16154187 0.18383302 0.20431896 0.22684211 0.25037625 0.27463352
## [13] 0.30010289 0.32659832 0.35317907 0.38022182 0.40736568 0.43415422
## [19] 0.46039312 0.48539402 0.50940356 0.54019588 0.57137444 0.59252755
```

```
## [25] 0.62267404 0.65327835 0.68316411 0.71358086 0.74419896 0.77425674
```

Once we got our numerical solution for r_t^*, we can again compute the whole tree.

```
Opt_tree=get_all_Rids(r_stars = R_star_num,vol_ts = vol_ts)
```

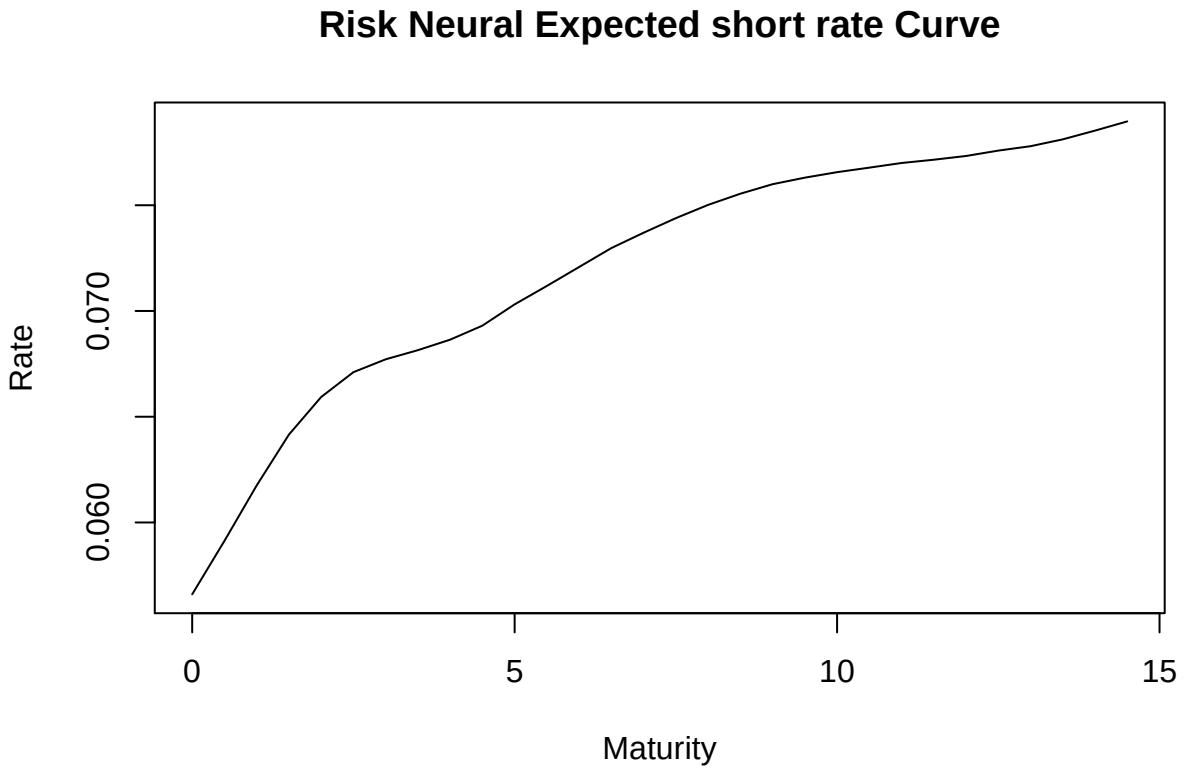
Question 2

Compute the expected value of the short rate tree at each time step, and plot the risk_neural short rate curve.Because the rates follow binary distribution. The martingale probablity of each rate should be

$$C_n^{iu}(\frac{1}{2})^n$$

where n is the number of steps, i is the number of upward movement.

```
#turn all zero values into NA
Opt_tree[Opt_tree==0]=NA
#compute a martinagle probability matrix
Q_prob_mat=Opt_tree
#compute the martingale probability of each node
for (t in 1:ncol(Q_prob_mat)){
  for (i in 1:t){
    Q_prob_mat[i,t]=choose(n = t-1,k = i-1)*(1/2)^(t-1)
  }
}
#compute the martingale probability
Ex_R_q=colSums(Opt_tree*Q_prob_mat,na.rm = T)
#plot the risk_neural short rate curve
plot(y=Ex_R_q,x=seq(0,14.5,0.5),type='l',main = 'Risk Neural Expected short rate Curve',
     ylab = 'Rate',xlab = 'Maturity')
```



Recall the formula of forward rate in lecture1 from discount factor, we can compute the short-term (0.5 year) forward rates at each time step using the following formula:

$$2tf_{0.5} = 2\frac{(1 + r_{t+0.5}/2)^{t+0.5}}{(1 + r_t/2)^t} - 1 = 2\frac{D(0,t)}{D(0,t+0.5)} - 1$$

```
#compute the forward rates and plot it over the martingale expected rate
forward_rates=Ex_R_q
forward_rates[1]=2*(1/True_DTs[1]-1)
forward_rates[2:30]=2*(True_DTs[-length(True_DTs)]/True_DTs[-1]-1)
#plot the short-term rate overlay on the risk-neutral expected short rate
plot(Ex_R_q,type='l',main = 'Martingale expected rate over Forward rate Curve',
     ylab = 'Rate',xlab = 'Maturity',col='red')
lines(x = forward_rates,col='blue')
legend('topleft', legend = c("Q Expected Rates", "Forward Rates"),
col = c("red", "blue"), lty = 1)
```