STAT243 PS1

Wenrui Wang

9/6/2021

```
# 2(a)
## Using function rnorm() to generate random number from normal distribution
# I firstly generated a file with 2000000 numbers and take a look at the size of the file
# and then increment the numbers to approach a file of 100 mb
RandomNum<-matrix(rnorm(5476000),ncol=10)
# Write the random numbers we generate into the file called Random_Num.csv
write.csv(RandomNum,"Random_Num.csv")
```

```
# 2(b)
# Start the clock
ptm<-proc.time()
df=read.csv(file="Random_Num.csv",nrows = 100000)
# Stop the clock
length_of_time<-proc.time()-ptm
# Take a look at the time elapsed
length_of_time
```

```
##    user  system elapsed
##    1.53    0.16    1.73
```

```
# Test whether we can load the data more quickly if we use colClasses
# Start the clock
ptm1<-proc.time()
# Stop the clock
length_of_time1<-proc.time()-ptm1
length_of_time1
```

```
##    user  system elapsed
##       0       0       0
```

```
## Using colClasses enables us to read the data much faster!
```

```
# 2(c)
# Read the data from the middle of the file
nrow(RandomNum)
```

```
## [1] 547600
```

```r
# Start the clock
ptm2=proc.time()
test1<-read.csv(file="Random_Num.csv",skip=273800,nrows=100)
time1=proc.time()-ptm2
time1
```

```
##    user  system elapsed
##    1.28    0.26    1.66
```

```r
# Start the clock
ptm3=proc.time()
test2<-read.csv(file = "Random_Num.csv",nrows = 100)
time2=proc.time()-ptm3
time2
```

```
##    user  system elapsed
##       0       0       0
```

```r
## It shows that we didn't really skip 273800 lines since reading 1000 rows from the
# middle of the file is much slower than reading the first 1000 rows of data
```

```r
# 2(d)
diff<-read.csv(file="Random_Num.csv",nrows=100000,colClasses = c("NULL",rep('double',10)))
# Open a connection to file
con = file("Random_Num.csv","r")
# Read the first half of the file
#rep(read.csv(con,nrows=100),2730) I commented this part of code because it will make pdf extremely lar
# Start the clock
ptm4=proc.time()
read.csv(con,nrows=100)
```

```
##       X          V1           V2          V3          V4          V5
## 1     1 -2.54740462  0.027193094 -0.53905262 -1.41251178  0.60459113
## 2     2  0.85307842 -1.101114821 -0.44163788  0.35635115  0.87877914
## 3     3 -0.93105144  0.021723403 -1.70057974  0.73234019  0.41800833
## 4     4 -0.82801630  0.348219032  1.04047249 -1.50737271  1.16026765
## 5     5 -1.15583515 -0.062661243  1.09352719  0.13479607  0.84200814
## 6     6  0.82301890  0.746080920  1.72617126  0.49958001  1.99291861
## 7     7 -0.54256165  0.646312133  0.21624722  0.02248374 -0.21735672
## 8     8  0.09333548  1.979327187  0.28830334 -2.15183680 -1.64979368
## 9     9  0.13688438 -0.267010052  0.55320681  1.32200504 -0.02802006
## 10   10 -1.75991212  0.072230993  0.16687336 -1.06273863  0.14651027
## 11   11 -1.04856740 -0.788670066 -0.59623236 -0.72535859 -0.27918801
## 12   12  0.92647730 -1.522075187 -0.18622527  1.05288159 -1.32860223
## 13   13  0.93099644  0.798545981 -0.90514043 -1.24834844 -0.37117630
## 14   14  1.33011412 -0.815303173  0.51086814 -0.75071806 -0.58528182
## 15   15  0.17870686 -0.766458805  1.26174245 -0.56486552 -1.47099919
## 16   16  0.17477750 -0.895381145 -0.88731362 -1.28580425  0.93794724
## 17   17  0.72250206 -1.527320047  1.64818663 -0.45160766  0.23132537
## 18   18 -0.57832796 -0.088485356  1.52648136 -0.64653534  0.37082418
## 19   19  0.94443326 -1.077044483 -0.52097789 -0.14618727  0.14315267
## 20   20 -0.60803030  0.729849947 -0.71182428 -1.82424487  0.79015054
## 21   21 -1.49166588 -0.578980293 -0.45472241  0.35650949 -0.59987780
## 22   22 -0.06127188 -0.011977232 -0.57294938  0.87801917  1.38202546
## 23   23 -0.05594300 -0.544507008  1.58742919  1.37549705 -0.29932076
## 24   24 -0.71363277 -2.363334519 -1.53199287 -0.34153245 -0.06808422
## 25   25  1.17987660 -0.961848353 -0.37417040  1.07281603  0.58707524
## 26   26 -0.88748821  0.648816116  0.63791512 -1.85585824  1.30686684
## 27   27 -0.13087444 -0.363061426  0.46750646 -0.10071621  1.46477937
## 28   28  0.97187394  0.019508617 -0.37798333  1.53796449 -0.25718308
## 29   29  0.39371191 -0.585345700  0.91478456 -1.16940119 -1.90017590
## 30   30 -0.29208891  0.426368646  0.78722812  0.27946311 -1.95161866
## 31   31  0.12718083 -1.722088247 -0.97288829  1.32927738 -0.82255472
## 32   32 -0.29486296  0.124452625  1.36871018  0.35436802 -1.97264714
## 33   33 -0.55998802  0.469552339 -0.50890816  0.27248280  1.01696101
## 34   34 -1.53772606 -0.772492947 -0.30919101 -0.26889740 -2.41284206
## 35   35  0.98121728 -0.211497472 -0.31758664  0.18552935  0.94654510
## 36   36  0.15288793 -0.758128133 -1.39481296 -1.30591619  2.01046100
## 37   37 -1.20543103 -1.690316406  1.29827854  1.89196980 -0.07578912
## 38   38 -0.19678659 -0.909864570  0.51912490  1.28445258 -0.52140361
## 39   39 -1.17872434  2.195205995  0.02996695 -0.49342158  1.27171229
## 40   40 -0.55774130  0.042883863 -1.20599550  0.57391926  1.92412894
## 41   41  0.24890255  0.268534975 -1.00017907 -0.39864063  0.38461307
## 42   42 -1.32817892  0.828757505  1.75999962 -1.27193280  0.52213573
## 43   43 -1.19055457 -0.216349607  1.23422677 -0.92201534 -0.19414034
## 44   44  1.80903534  1.208501315 -0.34147151 -0.11243904  0.48384592
## 45   45 -0.52868778  0.778568703 -0.32178740  0.41142112 -0.56142458
## 46   46  1.55745959 -0.222013577  0.79620955 -0.47408049 -0.62152453
## 47   47  1.07510805  0.097224245 -0.98983366  0.36425871  1.21088390
## 48   48 -0.65004033 -0.597707330  0.39798461 -2.10950685  0.18280991
## 49   49 -0.44121418 -0.690873600 -0.58884030  0.46597519 -0.97119765
## 50   50  1.37149684  0.728060321 -0.96640220 -0.41193217  1.03368453
## 51   51 -0.99152154  1.314211542 -0.97064950  0.22249582 -0.95524930
## 52   52 -1.27266026 -0.836527436 -0.10923040 -0.86789691  1.02956982
## 53   53  0.29872808  1.363616557  0.04291611 -0.02039317  0.54706308
## 54   54 -0.09250117 -0.192269340  2.48469346  0.73429265 -0.38151346
## 55   55 -0.39628182 -0.285704194  0.45214256  0.37209735 -0.52059883
## 56   56  0.75159688  0.251259669 -0.50246398  0.16085768 -0.78681816
## 57   57 -0.37672941  0.049122097 -1.39279702  0.47200172  1.01103989
## 58   58 -0.72715342  0.194547525 -0.45783047 -0.08697779 -0.74196625
## 59   59  2.51009356  0.425496416  1.07154957 -0.71236540 -0.55738430
## 60   60 -0.06383062 -0.614014478 -0.48445500  0.32724053  0.88227870
## 61   61  0.87954141 -0.065921659  0.09965447 -2.51637095 -0.94204036
## 62   62 -0.68632267 -1.060838927  1.60100538  1.12859872 -0.07607091
## 63   63  0.09934288 -2.182039251  1.20322731 -0.10978388  0.14819344
## 64   64 -1.47788422  1.342582334 -0.08986303 -1.04144047  0.28862844
## 65   65  0.99755483 -1.611164609  0.49515542 -2.15774495 -1.96424965
## 66   66  1.28258034 -0.714683124  0.97953432 -0.33721982  2.11528534
## 67   67  1.25111701 -0.480799453 -0.62530931 -0.76108892 -0.27968000
## 68   68 -0.99211567  0.833165221  0.96846996  0.02110407  1.21103389
## 69   69  1.17760496 -0.003978889  0.51181333 -0.90591162 -1.51034521
## 70   70 -0.20532382  0.795425826 -0.50019832  0.05044897 -0.90545090
## 71   71 -0.60012199 -0.335608388  1.61489603 -0.36243108  0.91410818
## 72   72  0.56772079  0.684628340  0.66267616 -0.21848527  1.51901936
## 73   73 -0.56698347  1.992137986  0.61937413 -0.29899695  0.99126984
## 74   74  0.83794027  0.790776775  0.48665544 -0.22317430 -0.16195745
## 75   75  0.91687642 -1.101090122  1.25532549  0.35897367 -1.67182660
## 76   76 -0.11042190  0.721520621 -0.37945598  2.07749442 -1.30366103
## 77   77  0.85828603 -0.883340014 -0.41371781  0.89801562  0.52097659
## 78   78  0.73392061  0.569222804 -1.69008739 -0.82272186 -1.38681796
## 79   79  0.11528782 -0.679086969 -1.21554081  0.53991518  0.51842806
## 80   80  0.15191378 -0.108561471 -1.05833932  1.19002426  0.21424333
## 81   81  0.53649276  0.811036859 -1.98129817 -0.18984072  1.71123991
## 82   82 -0.08578883 -0.666275769 -0.70107257 -0.09115800 -1.25765634
## 83   83  0.42736360 -0.860486428 -0.44349622  1.44126424  0.17405965
## 84   84  0.68992788  1.769929283 -0.22717568 -1.10632280 -1.12816370
## 85   85  1.17011605 -0.321598491 -0.04262084 -1.27116328 -0.40947180
## 86   86 -1.89789294  0.339492767 -0.05752372 -0.28231658 -0.62642133
## 87   87  0.07110473  0.774217186  0.72001984 -1.49832510 -0.76159007
## 88   88  0.50366122 -0.068055641 -0.05134680 -0.74297377 -0.32852904
## 89   89 -0.81394613 -0.600206608  0.19619633 -1.40429294 -0.46452231
## 90   90 -0.89758597 -2.741441665  0.54938254  0.64365272  2.08084112
## 91   91  0.36892563  0.241709980 -1.19344451 -0.72548946  0.22352796
## 92   92 -0.89499232  0.977913088  0.77558447  0.58733450 -0.84021679
## 93   93  0.63292908  1.269102699  1.17102925  0.75538133 -1.74264008
## 94   94 -1.33280059  1.341233778 -0.11792111  0.22008997  0.08879045
## 95   95  0.29000955 -0.188288036  0.22383027  1.20219197  0.35369684
## 96   96 -0.85856630 -1.750496452 -0.80665667 -0.12392869 -1.24560027
## 97   97  0.24035931 -1.139367736 -1.07541791  1.34642252 -0.05874671
## 98   98 -0.37851807  0.665482883  1.45382587 -0.65139675 -1.58202755
## 99   99  0.67506886 -0.143920005  0.36332102  0.52580053 -1.94454422
## 100 100  0.06173930 -2.330941411  0.14391651  1.99374240  1.18202858
##              V6          V7          V8           V9         V10
## 1    1.51492225  0.20218842 -0.09237865  0.0113581499  0.77043481
## 2   -0.57175007 -0.22701270 -0.54659885 -0.5348596136  2.21422713
## 3    0.47944474 -0.21068715  0.65713581  1.7167192395  1.66668909
## 4    0.41654020 -0.72572495  0.81710391 -0.0177055982 -1.86608507
## 5   -0.41784954  1.61815437  1.42653962  0.3852076206 -0.53150379
## 6   -0.45838268 -0.86799856  0.86429857  0.2171134188  1.25884626
## 7   -0.20406053  0.16525730  1.46562642  1.3835958691  0.65289366
## 8    1.41486555  0.88451920  0.95294640  1.4298230215 -0.31716719
## 9    0.76586341  0.07215291  0.27787180 -1.5959751437  0.60684911
## 10  -0.48226201 -0.81483524  0.57221922  0.3455778570 -0.12939770
## 11  -0.23371628  0.02506213  0.03958310 -0.1491470423  0.66392380
## 12   0.87760897  0.54029251  0.72575550 -0.3761760912 -1.30269146
## 13  -2.67983790  0.61738751  0.85704297  2.4539263986  1.00216688
## 14   0.50099514  1.44557381 -0.05379517 -0.7337498188  1.71597875
## 15   0.09867591  1.85766902  0.31552081 -0.5585884375 -0.69956429
## 16   0.38755369  1.56289336 -1.05130134  0.7545660129  0.36415253
## 17  -0.33695866 -1.00171506  0.34198121  1.4480456785 -1.91559065
## 18   1.93528351 -1.16055399 -0.66466482 -1.7403792271 -0.62069524
## 19  -1.13427648  1.21523465 -1.62560735 -1.0290193768 -0.94385002
## 20  -1.28082258 -1.94160753 -0.42807467 -1.5432263502 -0.69483455
## 21   1.48117380  1.42846191  1.05654518 -0.2381811879 -0.99751325
## 22  -1.23644562 -0.38472989  1.62068829 -1.3208016086  0.31661525
## 23   0.67518203 -0.56806570  0.05575500  2.7465299750  1.35548927
## 24   0.32807022  0.39356747 -0.95806262  0.7551095204 -2.20099333
## 25   0.90951421 -0.80455488  0.48297717 -1.6718777576  0.99658597
## 26  -0.26446410 -1.80232552  0.99148607  0.8975579719  0.39734447
## 27  -0.63701261 -0.14089247  2.43949306 -0.3089735763  0.69080159
## 28  -0.05525500 -0.40362075  1.62825929 -1.1235309191  0.43826001
## 29   1.99374191  0.93344543  0.05317506  1.1099613660  0.35190074
## 30   0.44568107  0.43669329  0.26957303  0.3113172135 -0.30584118
## 31   1.15691451 -0.36098881  1.93419175  1.5161868260  0.84573764
## 32   0.75866241 -0.17873626  0.62983235 -0.3642240012 -1.44772125
## 33   1.71737428  1.71957729 -0.30201011 -1.1088526982 -0.54266609
## 34  -0.01393651 -0.45447713  0.55457889  0.1485408744 -0.97668750
## 35   0.71525705  0.80240427  1.75375778 -0.8623337247 -0.83639363
## 36  -0.48923546 -0.53220932  1.97621827  2.1598676995  0.24302735
## 37  -0.30476445 -0.48952837 -0.18218866 -0.7288230551 -2.44230926
## 38  -0.10575776 -0.50128211  0.00865904 -0.3602855493 -0.01040070
## 39  -1.02364003 -0.25201941 -0.43709755 -0.9148973318 -2.68049858
## 40   0.85891301  0.13056311 -0.17009152  0.2921397495  2.09241684
## 41  -1.28592129 -2.06515229 -0.29398312 -0.0589332485 -1.29856552
## 42   0.45615533 -1.92112103  0.26479087 -1.6322909854  0.26604925
## 43   1.24185044 -2.52950620  1.54832986 -1.1275992691  0.23456831
## 44   1.85614111 -0.31251579 -1.49316239  1.3705413904  1.81875302
## 45   0.46898466  0.40835593 -0.84513704 -0.2161737482 -0.50323109
## 46  -1.60718395  0.29532724 -1.04129873  0.4377805746  0.42574650
## 47  -1.44193974 -0.42067355  0.63774724 -0.0006173428 -0.54475084
## 48  -0.82263460  0.71982486 -0.21669040 -0.7232867926 -0.66915936
## 49  -0.85562150  0.69145645 -0.36261992 -0.7175840009  0.34682347
## 50   0.42072257  0.43629308 -0.71615883  0.0243732676  0.29912932
## 51   0.24435229  0.40757820 -2.55500814  0.1674771615 -0.55252759
## 52   1.62497541 -0.89676613  0.45925103  0.4076921305  0.93488423
## 53   0.64172436  0.36037126  0.27464336  0.0671713249  0.13831106
## 54  -0.34871124  0.20612776  1.01648972  1.4352791181  0.21279506
## 55   0.21282559  1.72238015  0.22717467  0.2057531592  0.52792006
## 56  -0.34096157 -2.45928585  0.23123671 -1.4541774497  0.01279724
## 57   0.18441826  0.18398967 -1.15612899 -0.8747895955 -0.10921745
## 58  -2.41287214  0.03498479 -1.33677706 -1.1948786001  0.45918732
## 59   1.01267484 -1.07804705 -0.43061613 -0.4103495458  0.81740314
## 60   1.37637394 -0.53228934 -1.27980786  0.4096009265  0.06017297
## 61  -0.45646348 -0.73776726  0.31041536 -1.4218163617 -1.46994471
## 62   0.12050172  0.15953648 -0.28863157  1.9474196435  0.47619449
## 63   1.77597167 -0.08696910  0.82093421 -0.9446884421  0.62145370
## 64   0.13381212  0.58691332 -0.53301370  0.5511538482  0.80868919
## 65  -0.32780500 -0.27827413  0.08585737  0.1601818505 -1.90094334
## 66   0.46150001  0.79862634  0.19236510  1.0981048034  0.52122572
## 67   0.24587592 -1.51322534  0.94587302  0.2862905024  0.61563878
## 68  -2.14317349 -0.44009230  0.42249007  0.9945406972  0.02039152
## 69   1.85505403  0.63790737  0.19770886 -1.1025690671 -0.65728801
## 70  -0.11714888 -1.17769462 -1.78360990  0.1723906108  0.60087476
## 71   0.81701439 -0.66494998 -0.94317109 -1.6187963930 -0.72058811
## 72   1.12208434 -0.14002436 -0.47553393  0.2284186030 -0.26463951
## 73   0.28098515 -0.13632458 -0.10684032  0.7484972359  0.15598476
## 74  -0.09285189 -1.46210328  0.05188315 -0.4483952522  0.45669442
## 75   0.97496080 -0.87733105 -0.88717356  0.9974778949  1.45273016
## 76  -0.51162922 -0.01655325 -2.21743418  0.1776794274  0.57578025
## 77  -1.94441580  1.28132052 -0.99999794  0.5643318595 -0.01691611
## 78  -1.07067207 -0.05018651 -0.42692500  1.8519436256 -0.90670129
## 79  -1.81314897  0.12880495 -0.84422355  0.8932571164 -0.24721367
## 80   0.08660098 -0.24526628 -0.39152384  1.5731677531  1.01731934
## 81   0.75271501  0.58727201  1.59563126  0.5807682038 -1.95580269
## 82   1.65372989 -1.66990136  1.39592729  0.7047413578  1.62758716
## 83   1.35974583 -0.64948929 -0.97645856  0.8860558250  0.99448035
## 84   0.07567335 -1.72262948 -1.32992485 -1.4643850176 -0.80655630
## 85  -0.07431564  1.08066098 -0.23523644 -2.8966479996  0.61361959
## 86   0.99515692 -0.74779861 -1.21892792 -2.7737426606  1.14411921
## 87   0.02109375  0.43839224 -0.48152050 -0.3273075438 -0.96233830
## 88  -0.73457260  0.73832206  0.21344939 -0.8402109861  0.34399763
## 89  -0.32359088 -1.51112275  0.15569424  1.8530747031  0.91652524
## 90  -1.03957583  0.51028121 -1.28744210 -0.4805448961 -0.32788409
## 91  -0.25842241  0.94869050  0.93283724 -1.1666276830  0.64722340
## 92  -2.62635020  0.96506860 -0.02877181  0.0459173234 -0.12013117
## 93   1.13336331 -0.76357386 -0.05926021 -0.1355305752  0.33880814
## 94   0.46478173 -1.30393126  0.26077456  0.7582526549  0.99125244
## 95   1.33482075  0.02444965 -0.80952794 -0.2092627601 -1.77235825
## 96   0.64196837  1.76007190 -0.03458663 -0.5433937545  0.71359262
## 97   0.28169727  0.74675236  2.03232512 -0.7533057475  2.09713326
## 98  -0.11500912 -0.75699809  0.58621646  0.6447012904  0.38665608
## 99   0.37860891  1.24704467  1.95459714  0.5024064722  0.49265870
## 100 -0.08313564  0.30861149 -0.85381285  0.6500514866  0.73164942
```

```
# Stop the clock
time3=proc.time()-ptm4
time3
```

```
##    user  system elapsed
##    0.02    0.00    0.01
```

```
close(con)
## By using a connection, we read the same lines of data from the middle of the data file more quickly t
```

```
# 2(e)
save(RandomNum,file="RandomNum.rData",compress = FALSE)
file.info("RandomNum.rData")$size
```

```
## [1] 43808105
```

```
# 2(f)
Same_value<-matrix(2,547600,10)
save(Same_value,file = "SameNum.rData",compress=TRUE)
file.info("SameNum.rData")$size
```

```
## [1] 63877
```

```
# The size of the file with a single repeated value is significantly smaller
# than the file with same number of random values.
#"Compression" allows me to keep data files on disk compressed saving space and time!
```

```
# 3
## Because it is more readable and more convenient for me to debug the functions.
# Besides that, I pay attention to assigning a meaningful name to each of the functions. If the functio
# if i am not sure about whether the function I wrote is correct, I will test the function and check
# whether the output is valid.
# 4(a)
library(magrittr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rvest)
  url_webscrap<-function(artist,title){
  string1=strsplit(title,split=" ")[[1]]%>%paste(collapse="+")
  first_url<-paste("http://www.mldb.org/search?mq=",string1
                   ,"&si=2&mm=0&ob=1",sep = "")
  html=read_html(first_url)
  all_a_name<-html %>% html_elements("a")%>%html_text()
  all_a_value<-html %>% html_elements("a")%>%html_attr("href")
  song_url<-paste("http://mldb.org/",all_a_value[which(all_a_name==artist)+1],sep="")
  return (song_url)}

# Test the output of the function
m<-url_webscrap("Adele","Someone like you")
m
```

```
## [1] "http://mldb.org/song-248319-someone-like-you.html"
```

```
n<-url_webscrap("Sarah Connor","Just One Last Dance")
n
```

```
## [1] "http://mldb.org/album-20721-key-to-my-soul.html"
```

```
# 4(b) based on the code written in 4(a),we made some revision to get the required function
# Firstly, I just copied the function from 4(a)
  datascrap<-function(title, artist){
  string1=strsplit(title,split=" ")[[1]]%>%paste(collapse="+")
  first_url<-paste("http://www.mldb.org/search?mq=",string1,"&si=2&mm=0&ob=1",sep = "")
  html=read_html(first_url)
  all_a_name<-html %>% html_elements("a")%>%html_text()
  all_a_value<-html %>% html_elements("a")%>%html_attr("href")
  song_url<-paste("http://mldb.org/",all_a_value[which(all_a_name==artist)+1],sep="")
  return (song_url)}

Advanced_search<-function(title,artist){
  string2=strsplit(title,split=" ")[[1]]%>%paste(collapse="+")
  second_url<-paste("http://www.mldb.org/search?mq=",string1,"&si=2&mm=0&ob=1",sep = "")
  song_link<-datascrap(title,artist)
  html2<-read_html(song_link)
  frame1<-html2%>%html_elements("#thelist")%>%html_table()
  song_lyrics<-html2%>%html_elements("p")%>%html_text()
  album_sol=table[[1]][2,2]
  artist_sol=table[[1]][1,2]

  return(c(artist_sol,album_sol,song_lyrics))}
```

```
# 4(C)
# Firstly, I got the code from part(b)
  datascrap<-function(title, artist){

  string1=strsplit(title,split=" ")[[1]]%>%paste(collapse="+")
  first_url<-paste("http://www.mldb.org/search?mq=",string1,"&si=2&mm=0&ob=1",sep = "")
  html=read_html(first_url)
  all_a_name<-html %>% html_elements("a")%>%html_text()
  all_a_value<-html %>% html_elements("a")%>%html_attr("href")
  song_url<-paste("http://mldb.org/",all_a_value[which(all_a_name==artist)+1],sep="")
  # Use the length of all_a_name to verify whether the input is valid
  if(length(all_a_name)<=51){
    stop("The input you provided is invalid")
  }
  # Verify whether the lyrics are returned directly from the initial search
  if(length(all_a_name)<=63){
  html99<-read_html("http://mldb.org/album-20721-key-to-my-soul.html")
  all_a_name<-html99 %>% html_elements("a")%>%html_text()
  all_a_value<-html99 %>% html_elements("a")%>%html_attr("href")
  song_url<-paste("http://mldb.org/",all_a_value[which(all_a_name==title)],sep="")
     return(song_url)}
  else{

  return (song_url)}
  }

  Advanced_search<-function(title,artist){
  string2=strsplit(title,split=" ")[[1]]%>%paste(collapse="+")
  second_url<-paste("http://www.mldb.org/search?mq=",string1,"&si=2&mm=0&ob=1",sep = "")
  song_link<-datascrap(title,artist)
  html2<-read_html(song_link)
  frame1<-html2%>%html_elements("#thelist")%>%html_table()
  song_lyrics<-html2%>%html_elements("p")%>%html_text()
  album_sol=table[[1]][2,2]
  artist_sol=table[[1]][1,2]

  return(c(artist_sol,album_sol,song_lyrics))}
  # Now test whether our function works when the lyrics are returned directly from the initial search
  p<-datascrap("Just One Last Dance","Sarah Connor")
  p
```

```
## [1] "http://mldb.org/song-195467-just-one-last-dance.html"
```

```
  ## It works and returns the lyrics directly
```

```
# 5.
# It is ok to scrape the data from mldb.org since the search directory is not disallowed.
# However, it is not allowed to imiate the search to scrape the data from Google Scholar
# unless the data that you scrape is general data such as https://scholar.google.com/citations?view_op=
```

```
#install.packages("tinytex")
tinytex::install_tinytex()
```