Package *MFA01* Presentation

Wenzhe Ding, Shiying Zhou, Sheng Liu, Haochi Zhang

12/1/2016

About the presentation

In this presentation, we will briefly introduce the following things

- summary of the package tools
- a brief introduction of multiple factor analysis
- showcase of the package functionalities

About MFA01 package

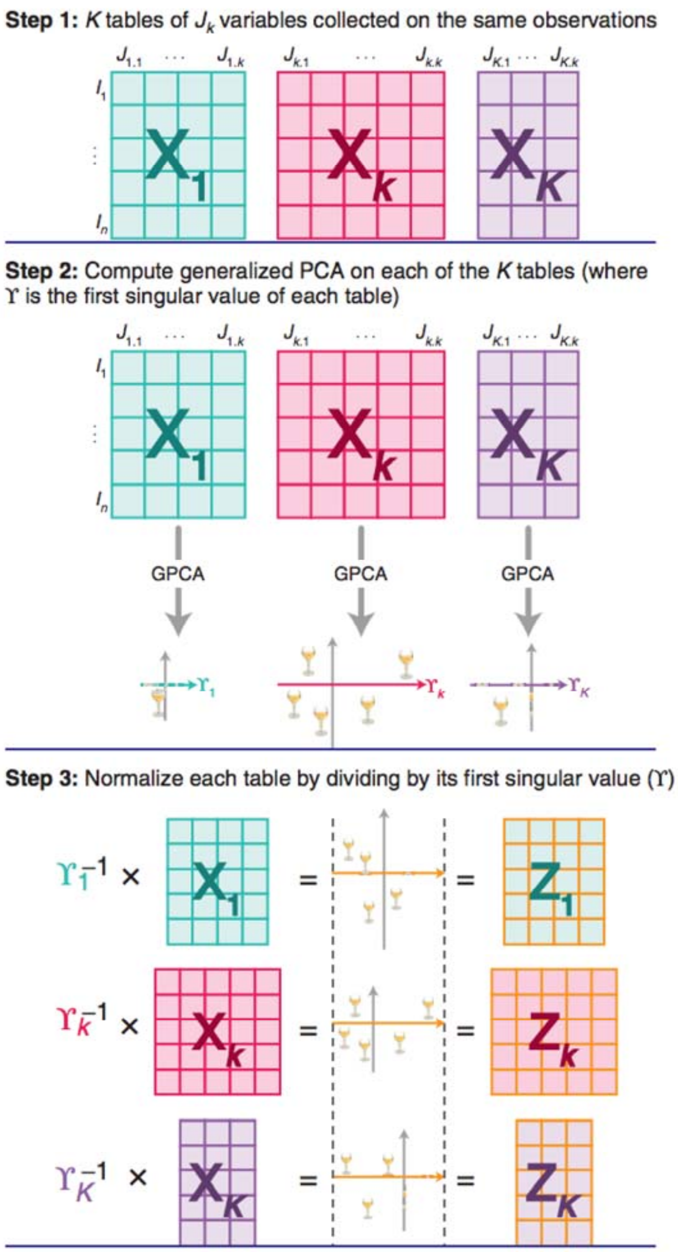
In this package, we created tools for multiple factor analysis (MFA), an extension of principle component analysis (PCA) to process multiple data sets and their correlations. The package include tools to compute the following outputs from MFA

- eigenvalues, common factor scores, partial factor scores, loadings
- visualization tools: generic functions for print, plot and tabularizing results
- auxiliary functions: *summaries of eigenvalues*, *contributions*, R_V *coefficients*, *coefficients to study the between table structure*, L_q *coefficients* and *Bootstrap* method to estimate the stability of the compromise factor scores.
- A shiny application

What is MFA?

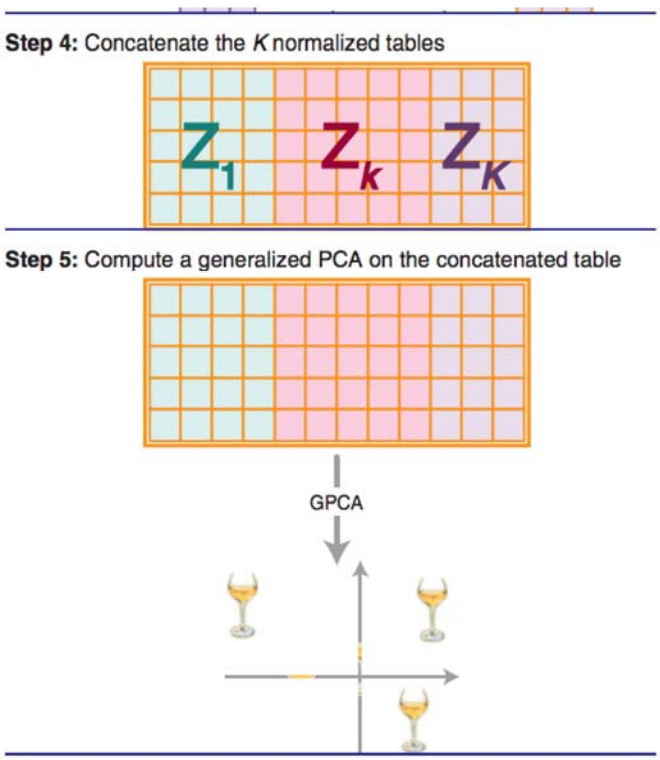
- Multiple factor analysis is a generalization of principle component analysis.
- The goal of MFA is to “analyze several data sets of variables collected on the same set of observations, orâ”as in its dual versionâseveral sets of observations measured on the same set of variables"
- MFA can be summarized into five steps, nicely illustrated by figure 1 by Abdi et al. (2013):
- Step 1: K tables of J_k variables collected on the same observations
- Step 2: Compute generalized PCA on each of the K tables (where Ï is the first singular value of each table)
- Step 3: Normalize each table by dividing by its first singular value (Ï)
- Step 4: Concatenate the K normalized tables
- Step 5: Compute a generalized PCA on the concatenated table

What is MFA?



step 1 to 3

What is MFA?



step 4 to 5

How to install the package?

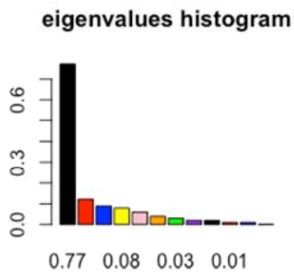
- To install the package, go to the R file 'install.R', use 'setwd()' to change the working directory to 'your_path/MFA01', and run the file.
- Then the MFA package should be installed. To create an 'mfa' object with default data set, simply run

```
## object "mfa"
##
## Number of tables/blocks for analysis: 10
## Weight of tables:
##  0.240983 0.2386034 0.2748748 0.2728329 0.3065003 0.3023908 0.4167984 0.2724079 0.2635522 0.308608
## First two eigenvalues:  0.7702551 0.1229254
## First component of common factor scores:
##  -0.9802057 -0.8088651 -0.7610058 -1.114984 1.372757 1.264015 0.8082827 0.9253423 -0.6689538 0.07316059 -0.4761088 0.3665652
## Facor loadings for the first table (1st component):
##  -0.2943913 -0.2665541 -0.2599726 0.2411168 0.2863237 -0.232907
```

Functionalities

The mfa object computes the eigenvalues of the grand data table, which can be visualized by

```
plot_eig(mfa_obj)
```



Functionalities

A matrix of the common factor score can be accessed by

```
mfa_obj$cfs
```

```
##              [,1]        [,2]        [,3]         [,4]         [,5]
##  [1,] -0.98020575  0.16325474 -0.02833247 -0.122967423 -0.138960264
##  [2,] -0.80886515  0.03262348  0.16181752 -0.376869178 -0.201380272
##  [3,] -0.76100584 -0.45418702 -0.00567849 -0.134631905 -0.061412658
##  [4,] -1.11498367 -0.16586214 -0.22362954  0.567155437  0.006220427
##  [5,]  1.37275684 -0.12838880  0.12163879 -0.315803256 -0.228652248
##  [6,]  1.26401538 -0.10813651 -0.37010578  0.087703704  0.413438110
##  [7,]  0.80828274  0.20466790  0.25317711 -0.085518365  0.247645688
##  [8,]  0.92534231  0.40775212 -0.37972251  0.264344947 -0.505067639
##  [9,] -0.66895382  0.36852275 -0.43311223 -0.342059710  0.247480544
## [10,]  0.07316059 -0.75677932 -0.04550557 -0.003904788  0.043136934
## [11,] -0.47610885  0.51276640  0.40040815  0.173699464  0.217286089
## [12,]  0.36656519 -0.07623359  0.54904503  0.288851074 -0.039734711
##              [,6]         [,7]        [,8]        [,9]        [,10]
##  [1,]  0.21752965 -0.220051492  0.26947251 -0.21850438  0.077474226
##  [2,] -0.13572321  0.147904830 -0.02638970  0.22079673  0.123996699
##  [3,] -0.07115109 -0.279400608 -0.03095015  0.15088378 -0.008565486
##  [4,] -0.30152727  0.002219817  0.01083609 -0.03067018 -0.117655827
##  [5,] -0.06585165 -0.138077283 -0.07701356 -0.08147074 -0.230493449
##  [6,]  0.12748875 -0.156705743  0.08745239  0.16910491  0.072655724
##  [7,] -0.42605557  0.046374939  0.09309193 -0.14630764  0.122837767
##  [8,]  0.03296264  0.058573490 -0.07536367  0.02332136  0.095377250
##  [9,]  0.07862601  0.250187914  0.01424231 -0.00850599 -0.155918796
## [10,]  0.17500807  0.223444581 -0.18009379 -0.18214255  0.104045535
## [11,]  0.16511358 -0.138517707 -0.32629392 -0.01128101  0.006704255
## [12,]  0.20358008  0.204047262  0.24100955  0.11477572 -0.090457898
##               [,11]        [,12]
##  [1,] -0.059739737 1.580398e-16
##  [2,] -0.162685833 1.580398e-16
##  [3,]  0.221691315 1.580398e-16
##  [4,] -0.104911814 1.580398e-16
##  [5,] -0.094639543 1.580398e-16
##  [6,] -0.101440700 1.580398e-16
##  [7,]  0.086496068 1.580398e-16
##  [8,]  0.087305272 1.580398e-16
##  [9,]  0.087365252 1.580398e-16
## [10,] -0.001839931 1.580398e-16
## [11,] -0.014311737 1.580398e-16
## [12,]  0.056711388 1.580398e-16
```

Functionalities

Other elements of the mfa object include:

```
attributes(mfa_obj)$names
```

```
## [1] "assessors"   "index_lists" "weights"     "eigen"       "cfs"
## [6] "fl"          "pfl"         "pfs"
```

Functionalities

To see a summary of what the object returns

```
mfa_obj
```

```
## object "mfa"
##
## Number of tables/blocks for analysis: 10
## Weight of tables:
##  0.240983 0.2386034 0.2748748 0.2728329 0.3065003 0.3023908 0.4167984 0.2724079 0.2635522 0.308608
## First two eigenvalues:  0.7702551 0.1229254
## First component of common factor scores:
##  -0.9802057 -0.8088651 -0.7610058 -1.114984 1.372757 1.264015 0.8082827 0.9253423 -0.6689538 0.07316059 -0.4761088 0.3665652
## Facor loadings for the first table (1st component):
##  -0.2943913 -0.2665541 -0.2599726 0.2411168 0.2863237 -0.232907
```

Eigenvalues

Our “ev.summary”" takes the “mfa” object and returns a table with the singular values, the eigenvalues, cumulative, percentage of intertia, cumulative percentage of inertia, for all the extracted components.

```
setwd("C:/Users/Sheng/Google Drive/Berkeley16Fall/STAT243/Stats-243/Stats-243/final")
library(MFA01)
mfa_obj <- MFA()
data <- get(load("data/wine.rda"))
ev.summary(mfa_obj)
```

```
##    SingularValue Eigenvalue CumulativeEigenvalue Inertia CumulativeInertia
## 1           0.88       0.77                 0.77      62                62
## 2           0.35       0.12                 0.89      10                72
## 3           0.30       0.09                 0.98       7                79
## 4           0.28       0.08                 1.06       6                85
## 5           0.24       0.06                 1.12       5                90
## 6           0.20       0.04                 1.16       3                93
## 7           0.17       0.03                 1.19       2                95
## 8           0.14       0.02                 1.21       2                97
## 9           0.14       0.02                 1.23       2                99
## 10          0.10       0.01                 1.24       1               100
## 11          0.10       0.01                 1.25       1               101
## 12          0.00       0.00                 1.25       0               101
```

Contributions

We include three functions to calculate *contributions*: 1. “ctr.obs”: contribution of an observation to a dimension 1. “ctr.var”: contribution of a variable to a dimension 1. “ctr.table”: contribution of a table to a dimension

For instance, we call “ctr.obs” as follows:

```
ctr.obs(mfa_obj)
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]    [,6]     [,7]
##  [1,] 0.103949 0.018068 7.37e-04 1.66e-02 2.70e-02 0.10059 1.31e-01
##  [2,] 0.070784 0.000722 2.41e-02 1.56e-01 5.67e-02 0.03916 5.90e-02
##  [3,] 0.062656 0.139845 2.96e-05 1.99e-02 5.27e-03 0.01076 2.10e-01
##  [4,] 0.134500 0.018650 4.59e-02 3.53e-01 5.41e-05 0.19326 1.33e-05
##  [5,] 0.203878 0.011175 1.36e-02 1.09e-01 7.31e-02 0.00922 5.14e-02
##  [6,] 0.172858 0.007927 1.26e-01 8.43e-03 2.39e-01 0.03455 6.62e-02
##  [7,] 0.070682 0.028397 5.89e-02 8.02e-03 8.57e-02 0.38586 5.80e-03
##  [8,] 0.092638 0.112712 1.32e-01 7.66e-02 3.57e-01 0.00231 9.25e-03
##  [9,] 0.048415 0.092067 1.72e-01 1.28e-01 8.56e-02 0.01314 1.69e-01
## [10,] 0.000579 0.388254 1.90e-03 1.67e-05 2.60e-03 0.06510 1.35e-01
## [11,] 0.024524 0.178244 1.47e-01 3.31e-02 6.60e-02 0.05795 5.17e-02
## [12,] 0.014537 0.003940 2.77e-01 9.15e-02 2.21e-03 0.08810 1.12e-01
##           [,8]     [,9]    [,10]    [,11]  [,12]
##  [1,] 0.242454 0.213206 0.037223 0.026319 0.0833
##  [2,] 0.002325 0.217703 0.095350 0.195187 0.0833
##  [3,] 0.003198 0.101663 0.000455 0.362450 0.0833
##  [4,] 0.000392 0.004201 0.085848 0.081171 0.0833
##  [5,] 0.019803 0.029640 0.329472 0.066054 0.0833
##  [6,] 0.025535 0.127700 0.032737 0.075888 0.0833
##  [7,] 0.028935 0.095590 0.093576 0.055175 0.0833
##  [8,] 0.018964 0.002429 0.056415 0.056212 0.0833
##  [9,] 0.000677 0.000323 0.150764 0.056290 0.0833
## [10,] 0.108292 0.148150 0.067135 0.000025 0.0833
## [11,] 0.355483 0.000568 0.000279 0.001511 0.0833
## [12,] 0.193941 0.058827 0.050745 0.023719 0.0833
```

Coefficients to study the Between-Table Structure

To evaluate the similarity between two tables we can use the R_V coefficient.

```
table1=mfa_obj$assessors[[1]]
table2=mfa_obj$assessors[[2]]
RV(table1,table2)
```

```
## [1] 0.868
```

Coefficients to study the Between-Table Structure (Continued)

We can also get a matrix of R_V coefficients by the following function.

```
RV_table(data, sets = list(2:3, 4:5, 6:10))
```

```
##       [,1]  [,2]  [,3]
## [1,] 1.000 0.907 0.984
## [2,] 0.907 1.000 0.946
## [3,] 0.984 0.946 1.000
```

L_g Coefficients

We design two functions for L_g coefficients similar to R_V coefficients.

```
Lg(table1,table2)
```

```
## [1] 0.918
```

```
Lg_table(data, sets = list(2:3, 4:5, 6:10))
```

```
##       [,1]  [,2]  [,3]
## [1,] 1.000 0.919 0.988
## [2,] 0.919 1.027 0.962
## [3,] 0.988 0.962 1.006
```

Bootstrap

We also write a function that allows the user to perform bootstrapping in order to estimate the stability of the compromise factor scores.

```
bt <- bootStrap(mfa_obj)
bt$mean
```

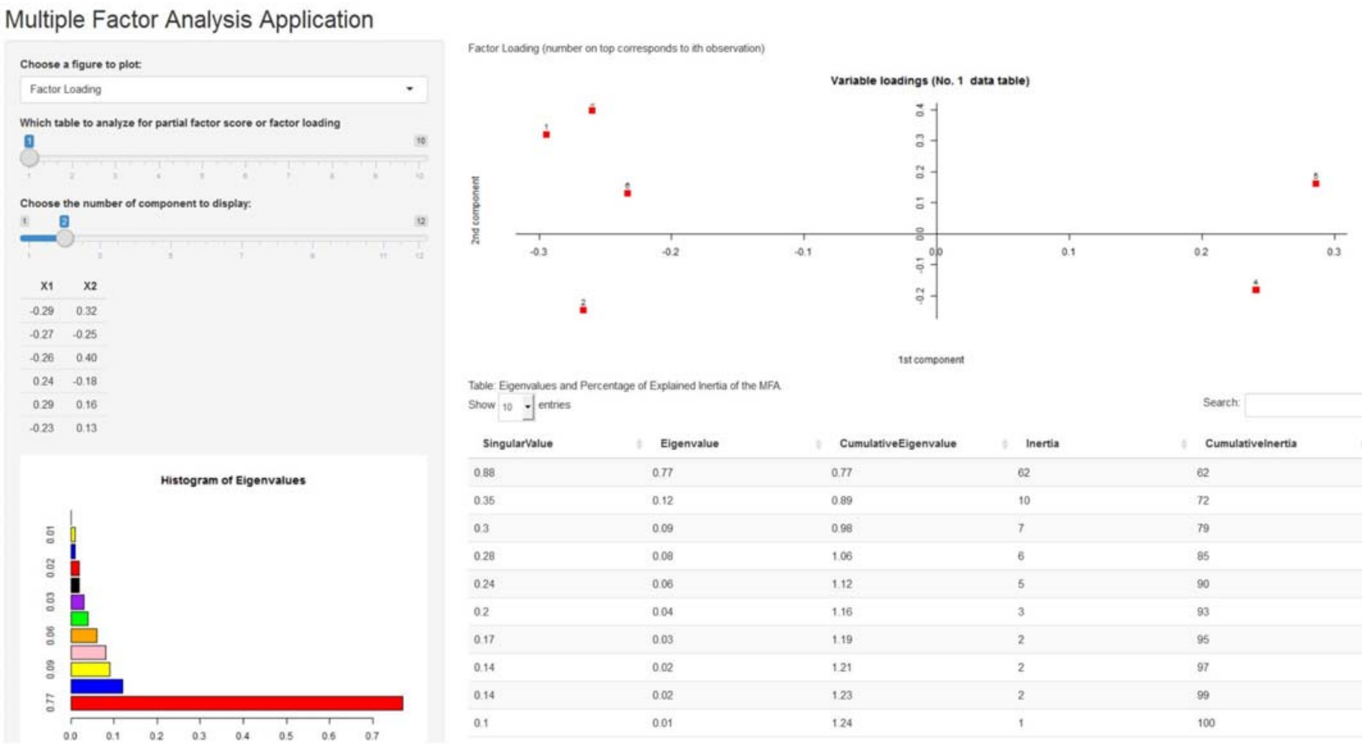
```
##          [,1]    [,2]     [,3]    [,4]    [,5]      [,6]    [,7]    [,8]
##   [1,] -0.8369  0.1136 -0.00903 -0.0847 -0.1265  0.187868 -0.1805  0.2354
##   [2,] -0.6916  0.0218  0.16424 -0.3106 -0.1800 -0.117732  0.1409 -0.0147
##   [3,] -0.6422 -0.3764  0.00302 -0.1287 -0.0605 -0.039700 -0.2478 -0.0164
##   [4,] -0.9404 -0.0920 -0.19393  0.4456 -0.0292 -0.238235  0.0030  0.0234
##   [5,]  1.1602 -0.1322  0.09118 -0.2582 -0.1650 -0.075963 -0.1226 -0.0645
##   [6,]  1.0845 -0.0937 -0.34623  0.0589  0.3783  0.103660 -0.1503  0.0735
##   [7,]  0.6912  0.1786  0.19142 -0.0760  0.2098 -0.382844  0.0393  0.0628
##   [8,]  0.7801  0.3266 -0.30946  0.2548 -0.3869  0.000482  0.0646 -0.0495
##   [9,] -0.5629  0.2776 -0.37118 -0.2603  0.2191  0.046573  0.2262  0.0177
##  [10,]  0.0653 -0.6177 -0.05599 -0.0484  0.0319  0.174356  0.1795 -0.1697
##  [11,] -0.4064  0.4267  0.36770  0.1703  0.1691  0.152271 -0.1038 -0.2776
##  [12,]  0.2992 -0.0329  0.46826  0.2373 -0.0601  0.189266  0.1513  0.1796
##           [,9]   [,10]    [,11]     [,12]
##   [1,] -0.17805  0.0531 -0.06320 -0.016624
##   [2,]  0.18843  0.1076 -0.15323 -0.026391
##   [3,]  0.12647 -0.0171  0.19616 -0.004610
##   [4,] -0.03640 -0.1194 -0.08142  0.006257
##   [5,] -0.07544 -0.1822 -0.08387  0.000254
##   [6,]  0.13833  0.0650 -0.07460  0.024776
##   [7,] -0.11585  0.1118  0.07220  0.005513
##   [8,]  0.01537  0.0728  0.07732 -0.013656
##   [9,] -0.01560 -0.1270  0.06452 -0.026728
##  [10,] -0.17547  0.0896 -0.00035  0.024127
##  [11,]  0.00999  0.0125 -0.00454  0.010913
##  [12,]  0.11822 -0.0667  0.05101  0.016169
```

Bootstrap (Continued)

```
bt$sd
```

```
##         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]
##  [1,] 0.0521 0.0886 0.0580 0.1221 0.0754 0.1150 0.1489 0.1418 0.1585
##  [2,] 0.0826 0.0908 0.0919 0.1369 0.0936 0.1188 0.1256 0.1258 0.0911
##  [3,] 0.0825 0.0703 0.0817 0.1245 0.0829 0.1475 0.1367 0.0998 0.1199
##  [4,] 0.0646 0.1143 0.1024 0.1199 0.1385 0.2012 0.1328 0.1476 0.1128
##  [5,] 0.0701 0.1058 0.0803 0.1189 0.1442 0.2283 0.1424 0.1540 0.1963
##  [6,] 0.0872 0.1045 0.1597 0.1973 0.1408 0.1859 0.2094 0.1830 0.1714
##  [7,] 0.0919 0.0811 0.0770 0.0736 0.0960 0.1390 0.1119 0.0996 0.1230
##  [8,] 0.0540 0.0824 0.1316 0.1669 0.1388 0.1650 0.1683 0.1680 0.1439
##  [9,] 0.0894 0.0984 0.0692 0.1040 0.0613 0.1281 0.0886 0.0590 0.1187
## [10,] 0.0638 0.0970 0.0678 0.0810 0.0949 0.1018 0.0552 0.1052 0.1059
## [11,] 0.0702 0.0659 0.1157 0.0981 0.1034 0.1395 0.1243 0.0898 0.1515
## [12,] 0.0625 0.0985 0.0718 0.0700 0.0666 0.0927 0.0770 0.1083 0.1027
##        [,10]  [,11]  [,12]
##  [1,] 0.1494 0.2074 0.0690
##  [2,] 0.0877 0.2056 0.1055
##  [3,] 0.1021 0.1463 0.0869
##  [4,] 0.1809 0.1750 0.1377
##  [5,] 0.1780 0.2335 0.1055
##  [6,] 0.1586 0.2334 0.1083
##  [7,] 0.1148 0.1885 0.0633
##  [8,] 0.0989 0.1577 0.1048
##  [9,] 0.0864 0.1612 0.1121
## [10,] 0.0985 0.0760 0.1341
## [11,] 0.0859 0.0667 0.0856
## [12,] 0.0640 0.0801 0.0941
```

Shiny App



X1	X2
-0.29	0.32
-0.27	-0.25
-0.26	0.40
0.24	-0.18
0.29	0.16
-0.23	0.13

SingularValue	Eigenvalue	CumulativeEigenvalue	Inertia	CumulativeInertia
0.88	0.77	0.77	62	62
0.35	0.12	0.89	10	72
0.3	0.09	0.98	7	79
0.28	0.08	1.06	6	85
0.24	0.06	1.12	5	90
0.2	0.04	1.16	3	93
0.17	0.03	1.19	2	95
0.14	0.02	1.21	2	97
0.14	0.02	1.23	2	99
0.1	0.01	1.24	1	100