BA\_FinalProject-Arangur

Thorsteinn Nielsen

Priyanandana Varma Mudunuri

Sharang Biradar

Snehal Kasturey

Priyanka Thakkar

December 20, 2016

Reading the datasets.

require('forecast')

require('foreign')

IndexCategories <- read.csv("IndexCategories.csv", header = TRUE)  
  
ses.2000 <- read.spss('ETH\_2000\_HICES\_v01\_M\_v01\_A\_SHIP\_SPSS/ETH\_2000\_E\_P.sav')  
  
ses.2004 <- read.spss('ETH\_2004\_HICES\_v01\_M\_v01\_A\_SHIP\_SPSS/ETH\_2004\_HICES\_v01\_M\_v01\_A\_SHIP\_SPSS/ETH\_2004\_E\_P.sav')  
  
expend201216 <- read.csv('FoodConsumption2012\_16.csv')  
  
a <- read.csv('FinalScoring.csv')  
  
ScoresForForecasting <- read.csv('VariableScoresPredict.csv', header=TRUE,stringsAsFactors=FALSE)

Determining the most important index categories for the Global Food Security Index.

IndexRegression <- lm(Index ~ AFFORDABILITY + AVAILABILITY + QUALITY.AND.SAFETY, data = IndexCategories)  
  
summary(IndexRegression)

##   
## Call:  
## lm(formula = Index ~ AFFORDABILITY + AVAILABILITY + QUALITY.AND.SAFETY,   
## data = IndexCategories)  
##   
## Residuals:  
## 1 2 3 4 5   
## 0.006349 -0.013879 0.011374 -0.001940 -0.001905   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.82076 1.69747 -1.662 0.3449   
## AFFORDABILITY 0.42184 0.02374 17.766 0.0358 \*  
## AVAILABILITY 0.45583 0.01210 37.687 0.0169 \*  
## QUALITY.AND.SAFETY 0.20731 0.04136 5.012 0.1254   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.01923 on 1 degrees of freedom  
## Multiple R-squared: 0.9999, Adjusted R-squared: 0.9996   
## F-statistic: 3235 on 3 and 1 DF, p-value: 0.01292

Calculating household fod consumption as a share of total household consumption and its respective score. Using linear extrapolation between known datapoints (2000, 2004, 2010 and 2012).

food.con.ratio.2000 <- mean((ses.2000$TOTFOOD)/(ses.2000$HHEXP\_N))\*100  
  
food.con.ratio.2004 <- mean(na.exclude((ses.2004$TOTFOOD)/(ses.2004$HHEXP\_N)))\*100  
  
interval1 <- (food.con.ratio.2004 - food.con.ratio.2000)/4  
  
food.con.ratio.2001 <- food.con.ratio.2000 + interval1  
food.con.ratio.2002 <- food.con.ratio.2000 + 2\*interval1  
food.con.ratio.2003 <- food.con.ratio.2000 + 3\*interval1  
  
food.con.ratio.2010 <- 46.1 #From report on food security in Ethiopia  
  
interval2 <- (food.con.ratio.2010 - food.con.ratio.2004)/6  
  
food.con.ratio.2005 <- food.con.ratio.2004 + interval2  
food.con.ratio.2006 <- food.con.ratio.2004 + 2\*interval2  
food.con.ratio.2007 <- food.con.ratio.2004 + 3\*interval2  
food.con.ratio.2008 <- food.con.ratio.2004 + 4\*interval2  
food.con.ratio.2009 <- food.con.ratio.2004 + 5\*interval2  
  
food.con.ratio.2012 <- expend201216$Percentage[1]  
  
food.con.ratio.2011 <- food.con.ratio.2010 + ((food.con.ratio.2012 - food.con.ratio.2010)/2)  
  
food.con.ratio <- c(food.con.ratio.2000, food.con.ratio.2001, food.con.ratio.2002, food.con.ratio.2003,  
 food.con.ratio.2004, food.con.ratio.2005, food.con.ratio.2006, food.con.ratio.2007,  
 food.con.ratio.2008, food.con.ratio.2009, food.con.ratio.2010, food.con.ratio.2011, food.con.ratio.2012)  
  
years <- c('2000','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','2012')  
  
df.food.con.ratio <- data.frame(years,food.con.ratio)  
  
df.food.con.ratio

## years food.con.ratio  
## 1 2000 53.08797  
## 2 2001 51.69791  
## 3 2002 50.30785  
## 4 2003 48.91780  
## 5 2004 47.52774  
## 6 2005 47.28978  
## 7 2006 47.05183  
## 8 2007 46.81387  
## 9 2008 46.57591  
## 10 2009 46.33796  
## 11 2010 46.10000  
## 12 2011 44.18240  
## 13 2012 42.26479

Predicting the unknown scores for Household food expediture as share of total expenditure using linear regression.

y <- expend201216$Score  
x <- expend201216$Percentage  
  
ratio.regression <- lm(y ~ x)  
  
score.extrapolation <- predict(ratio.regression,data.frame(x=food.con.ratio))  
  
df.scoreboard <- data.frame(years, food.con.ratio, score.extrapolation)  
  
df.scoreboard

## years food.con.ratio score.extrapolation  
## 1 2000 53.08797 24.33661  
## 2 2001 51.69791 26.60818  
## 3 2002 50.30785 28.87976  
## 4 2003 48.91780 31.15134  
## 5 2004 47.52774 33.42292  
## 6 2005 47.28978 33.81178  
## 7 2006 47.05183 34.20064  
## 8 2007 46.81387 34.58949  
## 9 2008 46.57591 34.97835  
## 10 2009 46.33796 35.36721  
## 11 2010 46.10000 35.75607  
## 12 2011 44.18240 38.88975  
## 13 2012 42.26479 42.02342

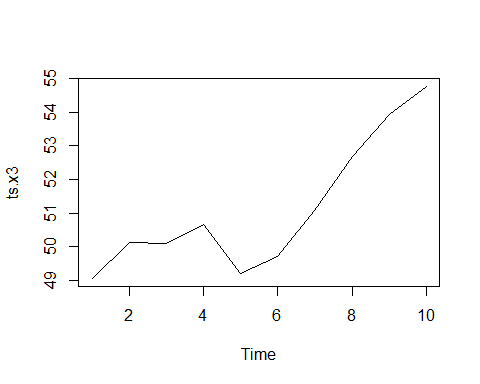
Forecasting the rest of missing values using ARIMA.

Forecast for X3:

data <- ScoresForForecasting  
  
data

## Year X1 X2 X3 X4 X5 X6 X7 X8 X9  
## 1 2000 28.9 35 49.05876 0.1 63.155 57.9 75.40701 80.86086 24.38910  
## 2 2001 30.8 35 50.13538 0.1 65.950 54.8 75.54315 84.66692 28.37728  
## 3 2002 32.7 34 50.09460 0.1 68.745 51.9 75.58177 76.58890 43.13629  
## 4 2003 34.5 53 50.66753 0.1 69.660 49.9 74.30822 69.09691 12.72989  
## 5 2004 36.4 66 49.19991 0.1 72.540 47.8 72.80545 74.81425 28.21101  
## 6 2005 38.3 79 49.72176 0.1 72.724 45.9 72.37256 80.72265 69.14397  
## 7 2006 40.1 75 51.07784 0.2 72.910 43.8 71.86601 80.39543 44.29245  
## 8 2007 42.0 79 52.67388 0.2 73.096 42.4 71.41810 80.69888 56.26154  
## 9 2008 43.9 72 53.94621 0.2 73.282 41.1 71.72427 85.80092 53.23944  
## 10 2009 45.9 69 54.77228 0.3 73.380 39.5 71.49572 84.58543 42.60927  
## X10 X11 X12 X13 X14 Average  
## 1 24.33661 69.84615 59.46970 54.0 18 45.74451  
## 2 26.60818 71.43590 67.91721 54.0 75 51.45243  
## 3 28.87976 73.12821 75.56180 50.0 56 51.17259  
## 4 31.15134 74.61538 84.21053 51.2 88 53.08141  
## 5 33.42292 75.64103 86.13139 37.0 88 54.86185  
## 6 33.81178 76.56410 88.68687 48.0 35 56.43198  
## 7 34.20064 77.38462 95.03386 42.0 60 56.30435  
## 8 34.58949 78.25641 94.31525 37.0 16 54.13640  
## 9 34.97835 78.76923 61.47860 37.0 6 50.95850  
## 10 35.36721 79.28205 59.52381 31.4 9 49.72256

ts.x3<-ts(data$X3)  
plot(ts.x3)



ts.x3.arima<-arima(ts.x3, c(1,2,1))  
x3\_arima\_forecast<-forecast.Arima(ts.x3.arima, h=6)  
summary(x3\_arima\_forecast)

##   
## Forecast method: ARIMA(1,2,1)  
##   
## Model Information:  
##   
## Call:  
## arima(x = ts.x3, order = c(1, 2, 1))  
##   
## Coefficients:  
## ar1 ma1  
## 0.3304 -1.0000  
## s.e. 0.3650 0.4669  
##   
## sigma^2 estimated as 0.8431: log likelihood = -11.48, aic = 28.95  
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.03346203 0.821518 0.6059989 0.04608293 1.200922 0.6247186  
## ACF1  
## Training set -0.0573488  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 11 55.49125 54.25705 56.72544 53.60371 57.37879  
## 12 56.17484 54.03286 58.31681 52.89897 59.45070  
## 13 56.84673 53.90146 59.79200 52.34233 61.35113  
## 14 57.51477 53.83597 61.19357 51.88853 63.14100  
## 15 58.18153 53.81492 62.54814 51.50337 64.85969  
## 16 58.84787 53.82395 63.87179 51.16445 66.53129

Forecast for X5:

ts.x5<-ts(data$X5,start=2000,frequency=1)  
ts.x5.arima<-arima(ts.x5, c(1,2,1))  
x5\_arima\_forecast<-forecast.Arima(ts.x5.arima, h=5)  
summary(x5\_arima\_forecast)

##   
## Forecast method: ARIMA(1,2,1)  
##   
## Model Information:  
##   
## Call:  
## arima(x = ts.x5, order = c(1, 2, 1))  
##   
## Coefficients:  
## ar1 ma1  
## -0.5273 -0.0382  
## s.e. 0.3896 0.4440  
##   
## sigma^2 estimated as 1.149: log likelihood = -12.09, aic = 30.18  
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.428083 0.9592388 0.6141121 -0.5995458 0.8571534 0.5405388  
## ACF1  
## Training set -0.3613124  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2010 73.52777 72.15387 74.90166 71.42658 75.62896  
## 2011 73.64929 71.24682 76.05176 69.97503 77.32356  
## 2012 73.78465 69.95876 77.61054 67.93346 79.63585  
## 2013 73.91272 68.54278 79.28265 65.70011 82.12532  
## 2014 74.04463 66.92635 81.16291 63.15816 84.93110

Forecast for X7:

ts.x7<-ts(data$X7,start=2000,frequency=1)  
ts.x7.arima<-arima(ts.x7, c(1,2,1))  
x7\_arima\_forecast<-forecast.Arima(ts.x7.arima, h=3)  
summary(x7\_arima\_forecast)

##   
## Forecast method: ARIMA(1,2,1)  
##   
## Model Information:  
##   
## Call:  
## arima(x = ts.x7, order = c(1, 2, 1))  
##   
## Coefficients:  
## ar1 ma1  
## 0.4796 -1.0000  
## s.e. 0.3790 0.3416  
##   
## sigma^2 estimated as 0.3385: log likelihood = -7.69, aic = 21.38  
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.0816214 0.5214572 0.3614668 -0.1064696 0.4947802 0.667578  
## ACF1  
## Training set 0.1047943  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2010 71.19432 70.41510 71.97353 70.00261 72.38602  
## 2011 70.85797 69.41297 72.30298 68.64804 73.06791  
## 2012 70.50488 68.43193 72.57783 67.33458 73.67518

Forecast for X9:

ts.x9<-ts(data$X9,start=2000,frequency=1)  
ts.x9.arima<-arima(ts.x9, c(1,2,1))  
x9\_arima\_forecast<-forecast.Arima(ts.x9.arima, h=3)  
summary(x9\_arima\_forecast)

##   
## Forecast method: ARIMA(1,2,1)  
##   
## Model Information:  
##   
## Call:  
## arima(x = ts.x9, order = c(1, 2, 1))  
##   
## Coefficients:  
## ar1 ma1  
## -0.3134 -1.0000  
## s.e. 0.3211 0.3892  
##   
## sigma^2 estimated as 410.5: log likelihood = -36.82, aic = 79.63  
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -1.679988 18.12221 12.55913 -22.55961 42.50367 0.7243768  
## ACF1  
## Training set -0.2475075  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2010 48.99325 21.546383 76.44013 7.016886 90.96962  
## 2011 50.04449 15.158079 84.93089 -3.309669 103.39864  
## 2012 52.76716 9.287541 96.24679 -13.729181 119.26351

Forecast for x10:

ts.x10<-ts(data$X10,start=2000,frequency=1)  
ts.x10.arima<-arima(ts.x10, c(1,2,1))  
x10\_arima\_forecast<-forecast.Arima(ts.x10.arima, h=3)  
summary(x10\_arima\_forecast)

##   
## Forecast method: ARIMA(1,2,1)  
##   
## Model Information:  
##   
## Call:  
## arima(x = ts.x10, order = c(1, 2, 1))  
##   
## Coefficients:  
## ar1 ma1  
## 0.0000 0.0000  
## s.e. 161.5004 161.4999  
##   
## sigma^2 estimated as 0.4431: log likelihood = -8.1, aic = 22.19  
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.1899405 0.5954414 0.1921173 -0.562713 0.5716573 0.1567507  
## ACF1  
## Training set -0.1253541  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 2010 35.75607 34.90302 36.60913 34.45144 37.06071  
## 2011 36.14493 34.23745 38.05242 33.22769 39.06218  
## 2012 36.53379 33.34196 39.72563 31.65231 41.41528

The Weighing Model.

Step 1: Creating a correlation matrix

a2 <- cor(a)  
  
a2

## X1 X2 X3 X4 X5  
## X1 1.00000000 0.5114870 0.966807620 0.90535093 0.827987035  
## X2 0.51148705 1.0000000 0.325825853 0.25814922 0.825915670  
## X3 0.96680762 0.3258259 1.000000000 0.91181281 0.689979943  
## X4 0.90535093 0.2581492 0.911812807 1.00000000 0.609661045  
## X5 0.82798703 0.8259157 0.689979943 0.60966105 1.000000000  
## X6 0.98840506 0.6119261 0.933923738 0.84931423 0.896730990  
## X7 -0.94390316 -0.7491864 -0.849935327 -0.78282544 -0.910192483  
## X8 -0.02317907 0.1532083 0.009746646 -0.04415287 -0.008374347  
## X9 0.82934692 0.3210332 0.781274868 0.84827770 0.611881425  
## X10 0.90234441 0.3077708 0.865952067 0.92631796 0.670240232  
## X11 0.97373280 0.6677760 0.904404979 0.81960421 0.928397137  
## X12 -0.59462380 0.1029669 -0.714258768 -0.51548335 -0.209755175  
## X13 -0.84024249 -0.6823766 -0.762405025 -0.68576197 -0.873542810  
## X14 -0.25106544 -0.3275024 -0.284774557 -0.14279953 -0.183319783  
## X6 X7 X8 X9 X10  
## X1 0.988405057 -0.94390316 -0.023179074 0.82934692 0.90234441  
## X2 0.611926054 -0.74918641 0.153208298 0.32103318 0.30777076  
## X3 0.933923738 -0.84993533 0.009746646 0.78127487 0.86595207  
## X4 0.849314228 -0.78282544 -0.044152869 0.84827770 0.92631796  
## X5 0.896730990 -0.91019248 -0.008374347 0.61188143 0.67024023  
## X6 1.000000000 -0.96749444 0.009928826 0.78085772 0.85217833  
## X7 -0.967494444 1.00000000 -0.031386688 -0.73001342 -0.78842433  
## X8 0.009928826 -0.03138669 1.000000000 -0.08887234 -0.23904504  
## X9 0.780857716 -0.73001342 -0.088872335 1.00000000 0.92371651  
## X10 0.852178327 -0.78842433 -0.239045036 0.92371651 1.00000000  
## X11 0.996220924 -0.97354563 0.010468198 0.75783782 0.83055107  
## X12 -0.536357086 0.42681012 -0.240066427 -0.38342361 -0.45697603  
## X13 -0.878040966 0.89850162 -0.107323871 -0.53490442 -0.63912983  
## X14 -0.274600326 0.31902460 -0.635498148 -0.10715270 0.05871236  
## X11 X12 X13 X14  
## X1 0.9737328 -0.5946238 -0.8402425 -0.25106544  
## X2 0.6677760 0.1029669 -0.6823766 -0.32750238  
## X3 0.9044050 -0.7142588 -0.7624050 -0.28477456  
## X4 0.8196042 -0.5154834 -0.6857620 -0.14279953  
## X5 0.9283971 -0.2097552 -0.8735428 -0.18331978  
## X6 0.9962209 -0.5363571 -0.8780410 -0.27460033  
## X7 -0.9735456 0.4268101 0.8985016 0.31902460  
## X8 0.0104682 -0.2400664 -0.1073239 -0.63549815  
## X9 0.7578378 -0.3834236 -0.5349044 -0.10715270  
## X10 0.8305511 -0.4569760 -0.6391298 0.05871236  
## X11 1.0000000 -0.4811381 -0.8896411 -0.27162524  
## X12 -0.4811381 1.0000000 0.4132391 0.36812487  
## X13 -0.8896411 0.4132391 1.0000000 0.35496778  
## X14 -0.2716252 0.3681249 0.3549678 1.00000000

Step 2: Finding absolute values of correlations

a3 <- abs(cor(a))  
  
a3

## X1 X2 X3 X4 X5 X6  
## X1 1.00000000 0.5114870 0.966807620 0.90535093 0.827987035 0.988405057  
## X2 0.51148705 1.0000000 0.325825853 0.25814922 0.825915670 0.611926054  
## X3 0.96680762 0.3258259 1.000000000 0.91181281 0.689979943 0.933923738  
## X4 0.90535093 0.2581492 0.911812807 1.00000000 0.609661045 0.849314228  
## X5 0.82798703 0.8259157 0.689979943 0.60966105 1.000000000 0.896730990  
## X6 0.98840506 0.6119261 0.933923738 0.84931423 0.896730990 1.000000000  
## X7 0.94390316 0.7491864 0.849935327 0.78282544 0.910192483 0.967494444  
## X8 0.02317907 0.1532083 0.009746646 0.04415287 0.008374347 0.009928826  
## X9 0.82934692 0.3210332 0.781274868 0.84827770 0.611881425 0.780857716  
## X10 0.90234441 0.3077708 0.865952067 0.92631796 0.670240232 0.852178327  
## X11 0.97373280 0.6677760 0.904404979 0.81960421 0.928397137 0.996220924  
## X12 0.59462380 0.1029669 0.714258768 0.51548335 0.209755175 0.536357086  
## X13 0.84024249 0.6823766 0.762405025 0.68576197 0.873542810 0.878040966  
## X14 0.25106544 0.3275024 0.284774557 0.14279953 0.183319783 0.274600326  
## X7 X8 X9 X10 X11 X12  
## X1 0.94390316 0.023179074 0.82934692 0.90234441 0.9737328 0.5946238  
## X2 0.74918641 0.153208298 0.32103318 0.30777076 0.6677760 0.1029669  
## X3 0.84993533 0.009746646 0.78127487 0.86595207 0.9044050 0.7142588  
## X4 0.78282544 0.044152869 0.84827770 0.92631796 0.8196042 0.5154834  
## X5 0.91019248 0.008374347 0.61188143 0.67024023 0.9283971 0.2097552  
## X6 0.96749444 0.009928826 0.78085772 0.85217833 0.9962209 0.5363571  
## X7 1.00000000 0.031386688 0.73001342 0.78842433 0.9735456 0.4268101  
## X8 0.03138669 1.000000000 0.08887234 0.23904504 0.0104682 0.2400664  
## X9 0.73001342 0.088872335 1.00000000 0.92371651 0.7578378 0.3834236  
## X10 0.78842433 0.239045036 0.92371651 1.00000000 0.8305511 0.4569760  
## X11 0.97354563 0.010468198 0.75783782 0.83055107 1.0000000 0.4811381  
## X12 0.42681012 0.240066427 0.38342361 0.45697603 0.4811381 1.0000000  
## X13 0.89850162 0.107323871 0.53490442 0.63912983 0.8896411 0.4132391  
## X14 0.31902460 0.635498148 0.10715270 0.05871236 0.2716252 0.3681249  
## X13 X14  
## X1 0.8402425 0.25106544  
## X2 0.6823766 0.32750238  
## X3 0.7624050 0.28477456  
## X4 0.6857620 0.14279953  
## X5 0.8735428 0.18331978  
## X6 0.8780410 0.27460033  
## X7 0.8985016 0.31902460  
## X8 0.1073239 0.63549815  
## X9 0.5349044 0.10715270  
## X10 0.6391298 0.05871236  
## X11 0.8896411 0.27162524  
## X12 0.4132391 0.36812487  
## X13 1.0000000 0.35496778  
## X14 0.3549678 1.00000000

Step 3: Reversing the correlations

b <- 1-abs(cor(a))  
  
b

## X1 X2 X3 X4 X5 X6  
## X1 0.00000000 0.4885130 0.03319238 0.09464907 0.17201297 0.011594943  
## X2 0.48851295 0.0000000 0.67417415 0.74185078 0.17408433 0.388073946  
## X3 0.03319238 0.6741741 0.00000000 0.08818719 0.31002006 0.066076262  
## X4 0.09464907 0.7418508 0.08818719 0.00000000 0.39033895 0.150685772  
## X5 0.17201297 0.1740843 0.31002006 0.39033895 0.00000000 0.103269010  
## X6 0.01159494 0.3880739 0.06607626 0.15068577 0.10326901 0.000000000  
## X7 0.05609684 0.2508136 0.15006467 0.21717456 0.08980752 0.032505556  
## X8 0.97682093 0.8467917 0.99025335 0.95584713 0.99162565 0.990071174  
## X9 0.17065308 0.6789668 0.21872513 0.15172230 0.38811857 0.219142284  
## X10 0.09765559 0.6922292 0.13404793 0.07368204 0.32975977 0.147821673  
## X11 0.02626720 0.3322240 0.09559502 0.18039579 0.07160286 0.003779076  
## X12 0.40537620 0.8970331 0.28574123 0.48451665 0.79024483 0.463642914  
## X13 0.15975751 0.3176234 0.23759498 0.31423803 0.12645719 0.121959034  
## X14 0.74893456 0.6724976 0.71522544 0.85720047 0.81668022 0.725399674  
## X7 X8 X9 X10 X11 X12  
## X1 0.05609684 0.9768209 0.17065308 0.09765559 0.026267199 0.4053762  
## X2 0.25081359 0.8467917 0.67896682 0.69222924 0.332224030 0.8970331  
## X3 0.15006467 0.9902534 0.21872513 0.13404793 0.095595021 0.2857412  
## X4 0.21717456 0.9558471 0.15172230 0.07368204 0.180395794 0.4845166  
## X5 0.08980752 0.9916257 0.38811857 0.32975977 0.071602863 0.7902448  
## X6 0.03250556 0.9900712 0.21914228 0.14782167 0.003779076 0.4636429  
## X7 0.00000000 0.9686133 0.26998658 0.21157567 0.026454365 0.5731899  
## X8 0.96861331 0.0000000 0.91112766 0.76095496 0.989531802 0.7599336  
## X9 0.26998658 0.9111277 0.00000000 0.07628349 0.242162181 0.6165764  
## X10 0.21157567 0.7609550 0.07628349 0.00000000 0.169448929 0.5430240  
## X11 0.02645437 0.9895318 0.24216218 0.16944893 0.000000000 0.5188619  
## X12 0.57318988 0.7599336 0.61657639 0.54302397 0.518861900 0.0000000  
## X13 0.10149838 0.8926761 0.46509558 0.36087017 0.110358890 0.5867609  
## X14 0.68097540 0.3645019 0.89284730 0.94128764 0.728374762 0.6318751  
## X13 X14  
## X1 0.1597575 0.7489346  
## X2 0.3176234 0.6724976  
## X3 0.2375950 0.7152254  
## X4 0.3142380 0.8572005  
## X5 0.1264572 0.8166802  
## X6 0.1219590 0.7253997  
## X7 0.1014984 0.6809754  
## X8 0.8926761 0.3645019  
## X9 0.4650956 0.8928473  
## X10 0.3608702 0.9412876  
## X11 0.1103589 0.7283748  
## X12 0.5867609 0.6318751  
## X13 0.0000000 0.6450322  
## X14 0.6450322 0.0000000

Step 4: Finding column sums and normalizing the matrix by dividing every entry in each column with the column sum (scale function)

c <- data.frame(b)  
  
d <- colSums(c)  
  
e <- scale(c, center = FALSE, scale = d)  
  
e

## X1 X2 X3 X4 X5 X6  
## X1 0.000000000 0.06827693 0.008300382 0.02013601 0.03618262 0.003386353  
## X2 0.141946685 0.00000000 0.168589992 0.15782418 0.03661833 0.113338648  
## X3 0.009644674 0.09422584 0.000000000 0.01876128 0.06521216 0.019297854  
## X4 0.027502080 0.10368465 0.022052875 0.00000000 0.08210710 0.044008421  
## X5 0.049981623 0.02433087 0.077526377 0.08304221 0.00000000 0.030160154  
## X6 0.003369130 0.05423909 0.016523619 0.03205747 0.02172245 0.000000000  
## X7 0.016299999 0.03505492 0.037526509 0.04620255 0.01889085 0.009493386  
## X8 0.283833809 0.11835170 0.247631573 0.20335058 0.20858668 0.289154500  
## X9 0.049586481 0.09489568 0.054696355 0.03227798 0.08164005 0.064001437  
## X10 0.028375680 0.09674930 0.033521220 0.01567540 0.06936438 0.043171949  
## X11 0.007632432 0.04643324 0.023905342 0.03837809 0.01506153 0.001103695  
## X12 0.117789727 0.12537368 0.071454997 0.10307793 0.16622658 0.135408886  
## X13 0.046420569 0.04439258 0.059415116 0.06685220 0.02660004 0.035618655  
## X14 0.217617111 0.09399152 0.178855644 0.18236411 0.17178722 0.211856062  
## X7 X8 X9 X10 X11 X12  
## X1 0.015458972 0.08569545 0.03219015 0.02151648 0.007515529 0.05364406  
## X2 0.069118333 0.07428812 0.12807294 0.15251905 0.095055402 0.11870578  
## X3 0.041354299 0.08687386 0.04125794 0.02953482 0.027351493 0.03781258  
## X4 0.059848208 0.08385544 0.02861925 0.01623438 0.051614553 0.06411684  
## X5 0.024748842 0.08699425 0.07321048 0.07265606 0.020486895 0.10457433  
## X6 0.008957768 0.08685788 0.04133662 0.03256959 0.001081263 0.06135459  
## X7 0.000000000 0.08497540 0.05092734 0.04661652 0.007569080 0.07585111  
## X8 0.266927075 0.00000000 0.17186524 0.16766141 0.283123238 0.10056319  
## X9 0.074401958 0.07993225 0.00000000 0.01680756 0.069287051 0.08159251  
## X10 0.058305285 0.06675776 0.01438929 0.00000000 0.048482453 0.07185921  
## X11 0.007290202 0.08681056 0.04567885 0.03733473 0.000000000 0.06866180  
## X12 0.157957666 0.06666815 0.11630428 0.11964462 0.148455927 0.00000000  
## X13 0.027970569 0.07831352 0.08773059 0.07951062 0.031575707 0.07764698  
## X14 0.187660824 0.03197736 0.16841703 0.20739416 0.208401408 0.08361702  
## X13 X14  
## X1 0.03598205 0.07949771  
## X2 0.07153805 0.07138410  
## X3 0.05351332 0.07591956  
## X4 0.07077557 0.09098989  
## X5 0.02848185 0.08668875  
## X6 0.02746873 0.07699953  
## X7 0.02286039 0.07228400  
## X8 0.20105670 0.03869105  
## X9 0.10475309 0.09477372  
## X10 0.08127849 0.09991555  
## X11 0.02485604 0.07731533  
## X12 0.13215567 0.06707211  
## X13 0.00000000 0.06846871  
## X14 0.14528007 0.00000000  
## attr(,"scaled:scale")  
## X1 X2 X3 X4 X5 X6 X7   
## 3.441524 7.154876 3.998898 4.700489 4.754022 3.424021 3.628756   
## X8 X9 X10 X11 X12 X13 X14   
## 11.398749 5.301407 4.538641 3.495057 7.556777 4.439922 9.420832

Step 5: Finding the mean values of every row which will represent our weights

f <- 0  
  
for(i in 1:14){  
 f[i]<- mean(e[i,])  
 print(f[i])  
  
}

## [1] 0.03341305  
## [1] 0.09992854  
## [1] 0.04291141  
## [1] 0.05324352  
## [1] 0.05449162  
## [1] 0.03318127  
## [1] 0.037468  
## [1] 0.1843426  
## [1] 0.06418901  
## [1] 0.051989  
## [1] 0.0343187  
## [1] 0.1091136  
## [1] 0.0521797  
## [1] 0.14923

f

## [1] 0.03341305 0.09992854 0.04291141 0.05324352 0.05449162 0.03318127  
## [7] 0.03746800 0.18434262 0.06418901 0.05198900 0.03431870 0.10911359  
## [13] 0.05217970 0.14922997

sum(f)

## [1] 1

Now we multiply the factor scores for each year by their respective weights and get the index values for every year.

g <- 0  
  
for(i in 1:16){  
 g[i]<-sum(f\*a[i,])  
   
}  
  
IndexValues <- g  
IndexValues

## [1] 46.36402 57.29208 54.92833 59.59816 62.73765 60.89159 63.17944  
## [8] 57.71987 52.91232 51.84360 52.12494 52.19797 52.45702 58.56219  
## [15] 64.64577 61.49793