YouthfulCities Interactive Application

Kie Gouveia - 500517428

9 December 2015

Executive Summary

Leveraging the YouthfulCities Ranking Index dataset, there was ample opportunity to explore theoretical machine learning concepts, however, given the practical nature of the dataset, it seemed that a more utilitariamn project would be more interesting and useful to the YouthfulCities team. To that end, this project was designed with the two following objectives in mind:

1. **Data Cleaning and Imputation**
    - Given the complexity of the dataset and underlying research, it seemed likely that a thorough audit of the data would be of great value to Youthful Cities. Moreover, an interactive application can only be truly useful if its foundational data is robust and accurate.
    - Cleaning and imputation amounted to dissecting the dataset, re-developing the indexing algorithm to identify potential anomolies and, in some cases, implementing alternative methods for the YouthCities team to consider.
2. **Developing an Interactive Platform**
    - An interactive application was a specific need expressed by the YouthfulCities team. Therefore, the second, and most important, objective of this project was to develop a platform which the YouthfulCities team might leverage, or use for inspiration, as they seek to make their data as accessible as possible.
    - This tool will allow users to quickly compare three cities based upon criteria that they deem to be relevant and important.
    - This step leveraged RShiny to create a interactive platform which would allow users to choose cities of interest and to re-weight variables according to their individual needs.

The cleaning process involved a full audit of the data, the creation of a codebook in excel to document algorithms and data transformations, and a slight reworking of the indexing algorithm. Transformations in excel are documented in the "excel_data_transformations.txt" and all other transformations are contained in the "yc_index_final.Rmd file" which resulted in the raw scores contained in the "final.ranking.calue.unweight.csv" file. This file is contained in the "app" folder for use by the application.

The interactive application was created as a Shiny Dashboard and is currently being hosted on Shiny.io. You may visit the final project at the following link. Once the webpage has loaded, simply click "Controls" in the left sidebar to get started. YouthfulCities Interactive Ranking Index

All project information is available in a Github repository, located in the following link: YouthfulCities Capstone Repository - Kie

Literature Review

Handbook on Constructing Composite Indicators - Methodology and User Guide

Organization for Economic Co-operation and Development

This handbook provides a complete guide to constructing and utilizing composite indicators. Specifically, the book is concerned with composite indicators that are designed to compare and rank country performance. By outlining technical guidelines and best practices for formulating and leveraging composite indicators, this handbook was an indispensable resource for investigating and optimizing the indices used by Youthful Cities. It also provided insight into the development of the algorithm which leverages user preferences to re-weight characteristics according to personal interests, needs, and preferences.

Composite Indicators of Country Performance: A Critical Assessment

OECD Science, Technology and Industry Working Papers

This paper provides a robust overview of the methodological difficulties associated with the development of composite indicators, specifically those being used to rank countries. Along with simply reviewing the challenges associated with creating composite indicators, this paper also provides steps in constructing composite indicators and makes suggestions to ensure transparency and appropriate use of indices in terms of analytics and policy.

Distinguishing "Missing at Random" and "Missing Completely at Random"
The American Statistician
Daniel F. Heitjan, Srabashi Basu

This article provides a good overview regarding the types of missing variables that may be contained within a dataset and how their type may affect methods of imputation. Among other things, this article touches upon *missing at random* variables and *missing completely at random* variables.

RShiny Website and Tutorials shiny.rstudio.com

The Shiny website has both tutorials and examples which were invaluable in developing the interactive application for this project.

Dataset Overview

According to youthfulcities.com, the YouthfulCities Index is an collaborative effort to analyze cities around the world from a youth perspective. It is designed to look at how youth live, work, and play in their urban setting, with the objective being to decipher how youth can become more engaged in their cities.

Data for the most recent iteration of this index was collected between September 2014 and April 2015, but primarily reflects information that is current as of 2013 - 2014. Data was collected in 55 cities around the world. The index is a composite, made up of 20 attributes, which are further defined and measured based upon 101 indicators.

The stated methodology for building th index is comprised of two steps:

1. *Determining what to measure in each city.* This is accomplished through a separate survey, called the Urban Attitude Survey. This tool is used to determine what characteristic are important to youth in the cities they live in.

2. *Collecting data to measure cities.* Leveraging the information garnered from the Urban Attitudes Survey, primary and secondary data is collected to measures 20 attitudes and 101 indicators which we deemed important to youth. This data, along with weights determined by the relative importance of each indicator to youth, are used to develop a ranking of cities.

For information regarding data processing, including normalization, comparing different years, scale and boundary issues, imputation of missing values, and data credibility, the YouthfulCities webpage offers more information.

The raw data is packaged in the "2015 YouthfulCities Index Ranking Sheet.xlsx" which may be found in the data folder. The document contains 26 sheets of information. Prior to use in R, some preliminary cleaning was done in Excel. These transformations are detailed in the "excel_data_transformation.txt" file and resulted in the spreadsheet "2015_YouthfulCities_Index_Ranking_sheet_Values.xlsx" file in the data folder.

The bulk of the data used in this analysis will be derived from *Raw Data* sheet which aggregates many of the raw data points. However, in reconstructing the index, data is sometimes drawn from other sheets to acquire information such as exchange rate information, population sizes and geographic area. These will be noted as they arise.

The following section contains exploratory analysis to describe the data in more detail

Dataset Overview: Preparing the Data

Analysis will work as long as the working directory is set to the parent directory, "proj_files". This directory should contain a subfolder titled "data".

```
set.seed(101) # to allow for reproducibility of findings.
```

Dataset Overview: Load Necessary Packages

```
# the function below looks to see if a given package has already been installed prior to attaching it.

usePackage <- function(p) {
    if (!is.element(p, installed.packages()[,1]))
        install.packages(p, dep = TRUE)
    library(p, character.only = TRUE)
}

usePackage("openxlsx") # To import excel files
usePackage('VIM')      # Contains many useful imputation-related functions
usePackage("ggplot2")  # For data visualizations
usePackage("plyr")     # For data summarization and manipulation
usePackage("dplyr")    # For data summarization and manipulation
```

Dataset Overview: Data Import

```
file <- "./data/2015_YouthfulCities_Index_Ranking_Sheet_Values.xlsx"

wb <- loadWorkbook(file) # load the excel workbook containing the raw data.

# create a vector of the names corresponding with the excel sheets
sheet.names <- c("Master", "Raw", "Attr.Rank","Attr.Score", "Weighted.Score",
                 "Safety","Affordability","Transit", "Health","Travel","Employment",
                 "Environment","Education","Entrepreneurship","Public.Space",
                 "Financial.Services","Diversity","Digital.Access","Music",
                 "Creative.Arts","Sports","Film","Civic.Engagement","Food.Nightlife",
                 "Fashion","Youth.Population")

# Create a list of the sheets that have been imported from excel.
sheet_list <- lapply(1:length(sheets(wb)), function(x) {
  readWorkbook(wb,
               sheet = x,
               colNames = TRUE,
               rowNames = FALSE)
  })

# The below assigns names to the imported sheets then exports them as csv files
# (which are easier to work with than xlsx)
names(sheet_list)[1:26] <- sheet.names[1:26] # names the components of the sheet list.

outfile <- paste("sheet_list$", sheet.names[1], sep="")

sapply(names(sheet_list), function(x) write.csv
       (sheet_list[[x]],
       file = paste("./data/YCRank-",x,".csv", sep="")
))
# Write the information obtained from the excel sheets into individual csv files

# TODO: make a check to see if the file is there before completing this step.
```

From here we can read in the newly created csv files to dataframes.

```
setwd("./data") # set working directory to data subdirectory.

filenames <- list.files(pattern = "YCRank") # list the files that we just created.

allSheets <- lapply(filenames, function(i){ # read these files into R.
  read.csv(i, header=TRUE, nrows = 250)
  })

# Read in csv files

for (i in 1:length(sheet.names)){
  name <- paste(sheet.names[i])
  assign(name, read.csv(file = paste("YCRank-", sheet.names[i], ".csv", sep = ""),
                        header = FALSE,
                        nrows = 200,
                        stringsAsFactors = FALSE))
  }

setwd("..") # Reset wd
```

Dataset Overview: Preliminary Data Cleaning

We will now take the raw dataframes and convert them to a clean format. This will allow us to perform simple descriptive analysis and will greatly ease calculations later on in the analysis. This manipulation is omitted from the written report, but available in the .rmd file.

Exploratory Analysis

With the data reasonably well structured, we can perform some simple exploratory analysis to verify the structure of the dataset and attempt to identify any anomalies.

```
row.names(Raw.Clean)
```

```
##  [1] "Accra"          "Amsterdam"      "Bangkok"        "Beirut"
##  [5] "Berlin"         "Bogota"         "Boston"         "Buenos.Aires"
##  [9] "Cairo"          "Caracas"        "Casablanca"     "Chicago"
## [13] "Dallas"         "Dar.Es.Salaam"  "Detroit"        "Dubai"
## [17] "Durban"         "Hong.Kong"      "Istanbul"       "Jakarta"
## [21] "Johannesburg"   "Karachi"        "Lagos"          "Lima"
## [25] "London"         "Los.Angeles"    "Madrid"         "Manila"
## [29] "Mexico.City"    "Miami"          "Montreal"       "Moscow"
## [33] "Mumbai"         "Nairobi"        "New.Delhi"      "New.York.City"
## [37] "Osaka"          "Paris"          "Quito"          "Rio.De.Janeiro"
## [41] "Rome"           "San.Francisco"  "Santiago"       "Sao.Paulo"
## [45] "Seoul"          "Shanghai"       "Singapore"      "Sydney"
## [49] "Tehran"         "Tel.Aviv"       "Tokyo"          "Toronto"
## [53] "Vancouver"      "Warsaw"         "Washington"
```

```r
colnames(Raw.Clean)
```

```
##  [1] "road.death"                   "food.safety"
##  [3] "suicides"                     "homicides"
##  [5] "c.tax.rate"                   "rental.housing"
##  [7] "gini"                         "cellular.cost"
##  [9] "food.cost"                    "transit.pass"
## [11] "apt.buy.cost"                 "toothpaste.cost"
## [13] "km.transit"                   "hrs.transit"
## [15] "bike.rental"                  "km.bike.path"
## [17] "commute.time.transit"         "commute.time.foot"
## [19] "commute.time.airport"         "transit.app"
## [21] "pop.density"                  "taxi.rate"
## [23] "health.clinics"               "sex.health.clinic"
## [25] "homeless.shelter"             "smoking.scale"
## [27] "mental.health.facilities"     "doctor.density"
## [29] "city.connections.flight"      "getaway.bus.cost"
## [31] "getaway.bus.time"             "getaway.bus.freq"
## [33] "getaway.plane.cost"           "getaway.plane.time"
## [35] "getaway.plane.freq"           "hostel.stay"
## [37] "unemployment.youth"           "employment.programs"
## [39] "youth.employment.centers"     "new.jobs.2013"
## [41] "water.scale"                  "smart.cities.scale"
## [43] "qty.recycled.materials"       "qty.annual.waste"
## [45] "carbon.emissions"             "num.recycled.materials"
## [47] "registered.vehicles"          "post.secondary.institutions"
## [49] "tuition.cost"                 "undergrad.enrollment"
## [51] "student.housing"              "student.debt"
## [53] "employment.initiatives.scale" "age.register.business"
## [55] "num.incubators"               "early.entrepreneurship"
## [57] "ease.doing.business"          "green.public.space"
## [59] "sports.facilities"            "public.libraries"
## [61] "business.banking"             "personal.banking"
## [63] "chartered.banks"              "online.banking"
## [65] "mobile.banking"               "financial.literacy"
## [67] "voting.languages"             "diversity.food"
## [69] "openness.lgbtq"               "openness.immigrants"
## [71] "diversity.religion"           "gender.gap.index"
## [73] "cellular.comp"                "cellular.package.cost"
##  [75] "free.wifi"                    "mobile.infrastructure"
##  [77] "municipal.open.data"          "gamers.developers.scale"
##  [79] "num.nightclubs"               "music.festivals"
##  [81] "graffiti.street.art"          "number.pro.sports.teams"
##  [83] "num.pro.sports.facilities"    "cost.shorts"
##  [85] "cost.gym"                     "film.festivals"
##  [87] "cost.movie"                   "cinema.seats"
##  [89] "voting.age"                   "councillor.age"
##  [91] "volunteer.opportunities"      "youth.advisory.board"
##  [93] "highschool.volunteer"         "num.restaurants"
##  [95] "last.call.index"              "num.food.festival"
##  [97] "fast.food"                    "young.designer"
##  [99] "fashion.incubators"           "design.schools"
```

As we can see, the rows now list out each city while the columns list each variable by which the cities were measured. Next, let's take a look at the ranges and quartiles of each variable.

```r
summary(Raw.Clean[,1:10]) # for the purposes of the report, we will look at the first ten variables.
```

```
##    road.death     food.safety        suicides       homicides      
##  Min.   : 5.90   Min.   :  0.00   Min.   : 0.90   Min.   :   0.00  
##  1st Qu.:15.60   1st Qu.: 73.00   1st Qu.: 5.35   1st Qu.:  53.25  
##  Median :22.80   Median :100.00   Median : 9.80   Median : 159.00  
##  Mean   :28.85   Mean   : 84.95   Mean   :10.21   Mean   : 535.64  
##  3rd Qu.:40.80   3rd Qu.:100.00   3rd Qu.:12.15   3rd Qu.: 439.00  
##  Max.   :85.90   Max.   :100.00   Max.   :28.90   Max.   :3928.00  
##                                                   NA's   :1        
##    c.tax.rate      rental.housing         gini         cellular.cost     
##  Min.   :0.00000   Min.   :     602   Min.   :0.2830   Min.   :   0.0900  
##  1st Qu.:0.08375   1st Qu.:    2421   1st Qu.:0.3495   1st Qu.:   0.4675  
##  Median :0.12000   Median :    5180   Median :0.4200   Median :   1.0000  
##  Mean   :0.12478   Mean   :  544108   Mean   :0.4321   Mean   : 123.5708  
##  3rd Qu.:0.18000   3rd Qu.:   40434   3rd Qu.:0.4863   3rd Qu.:   2.7750  
##  Max.   :0.23000   Max.   :15968600   Max.   :0.8560   Max.   :2505.0000  
##                                                                           
##    food.cost          transit.pass     
##  Min.   :    1.94   Min.   :    15.0  
##  1st Qu.:    3.58   1st Qu.:    76.5  
##  Median :    9.16   Median :   151.2  
##  Mean   : 1572.61   Mean   : 12699.0  
##  3rd Qu.:   72.27   3rd Qu.:   891.9  
##  Max.   :44247.20   Max.   :306640.0  
## 
```

We can see immediately that we are dealing with variables that have extremely diverse ranges. With this in mind, some sort of feature scaling will need to be done to ensure that variables are contributing equally to the final index. From the excel worksheet, we also know that many of these values are currently measured in different units altogether. For example, between variables, we are looking at units such as kilometers and dollars, and within variables we are sometimes using different currencies. These, and other variables, will need to be standardized.

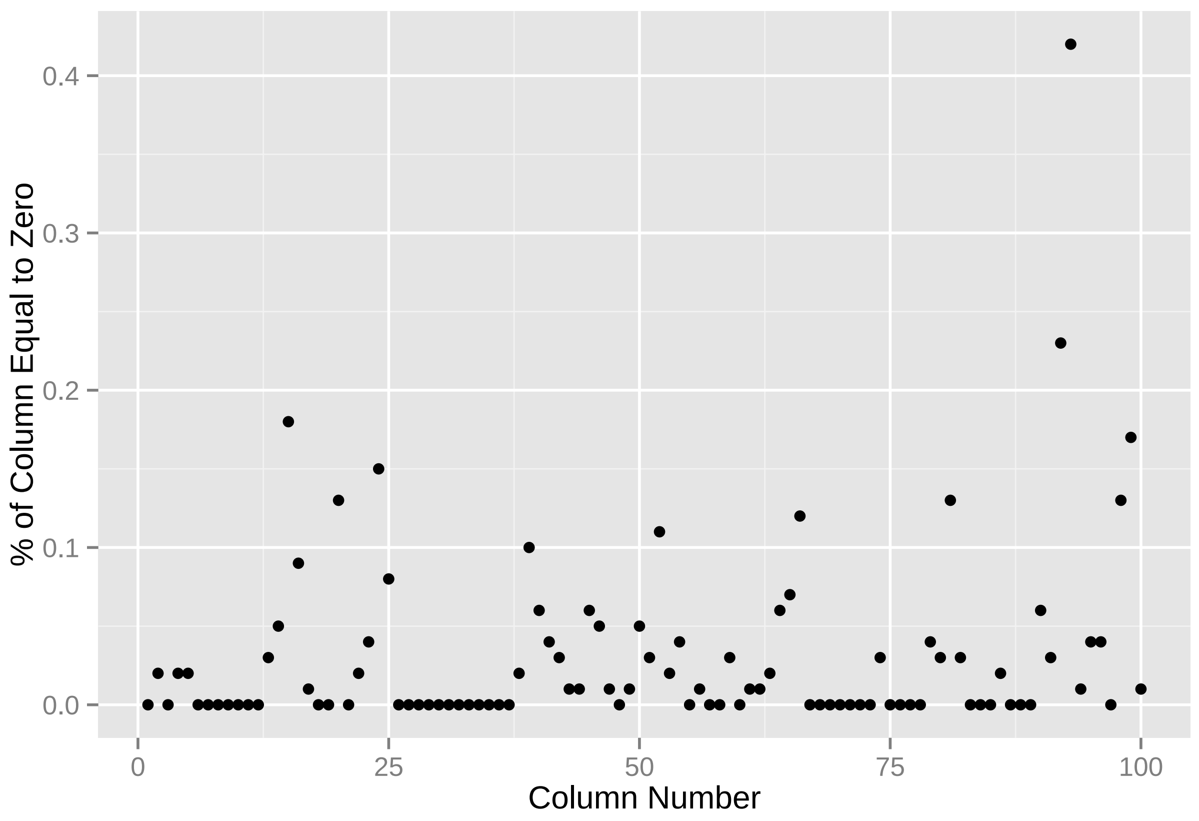
Exploratory Analysis: Zeros and Missing Data

Now, it will be worthwhile to look at the quantity and distribution of missing data.

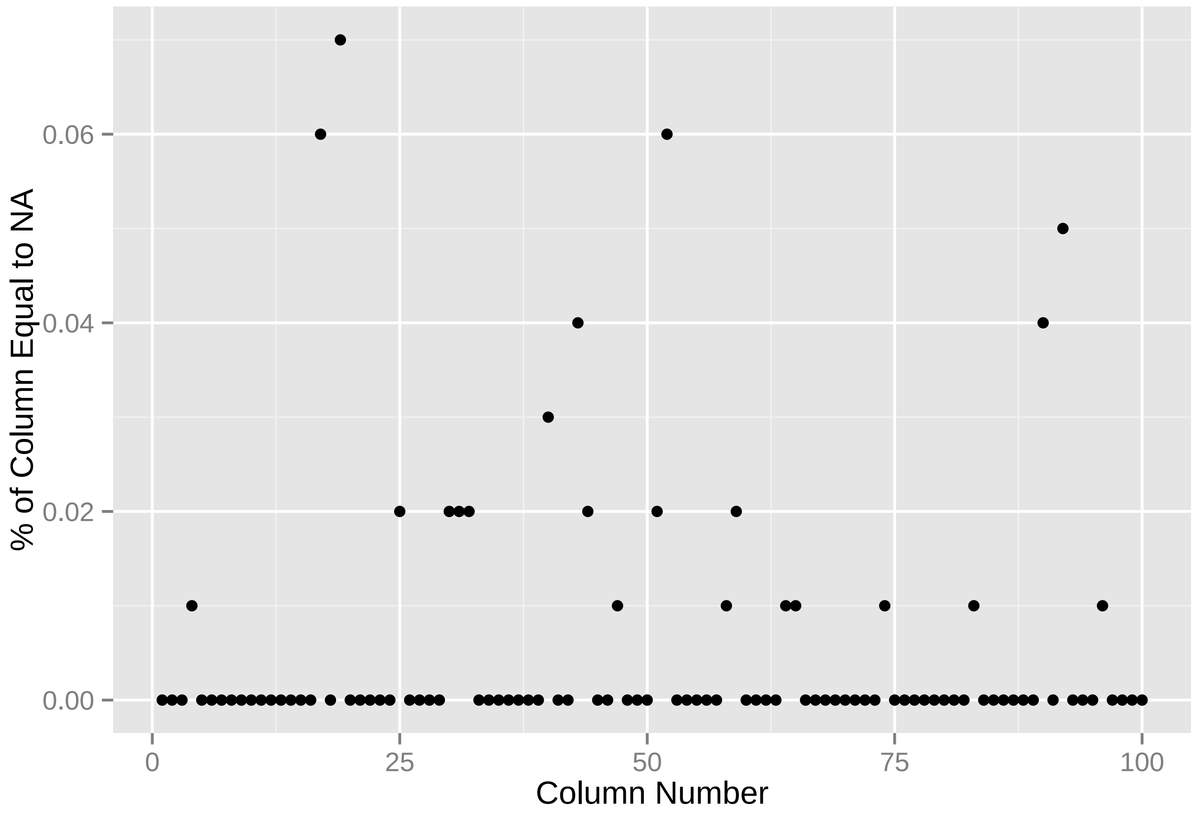
```r
col.num <- 1:ncol(Raw.Clean)

# Which columns have 0's?
zeros <- as.vector(
  do.call(
    cbind, lapply(
      col.num, function(i) length(which(Raw.Clean[, i] == 0)))))

# Which columns have NA's?
NAs <- as.vector(do.call(cbind, lapply(col.num, function(i) sum(is.na(Raw.Clean[, i] == TRUE)))))

missingdata <- as.data.frame(cbind(col.num, zeros, NAs))
```

With the object above, we are able to create plots to visually show the rates of zeros and missing data.

```r
qplot(col.num, zeros/length(col.num), data = missingdata,
      xlab = "Column Number",
      ylab = "% of Column Equal to Zero")
```



```r
qplot(col.num, NAs/length(col.num), data = missingdata,
      xlab = "Column Number",
      ylab = "% of Column Equal to NA")
```



There are a few things to note regarding the charts above. First, rates of zeros and missing data do not seem to corelate with column number. This is a good thing. However, the substantial number of zeros justified a closer look. Without conducting extensive research (that is not within the scope of this project, although that is the recommended course of action), it appears that zeros are being used in some places where they should be labelled *NA*. Because these figures are being used in normalization calculations in the Youthful Cities Index, they are likely distorting final scores.

This tells us that we will have to look at each variable with zeros and replaces them with *NAs* where there is a high likelihood (based on personal judgement alone) that they are, in fact, missing data points and not true zeros. This is carried out in the *Imputation* section.

It is recommended that all of these be double checked by verfying the sources of each data point.

Exploratory Analysis: Correlation Analysis

In constructing indices, it is useful to look at the correlation between variables being used to construct final scores. This is particularly useful to check among variables within a given category. For example, if we look within the *Safety* category and find that two variables are almost perfectly correlated, this could be an indication that one of the variables in not contributing new, meaningful information to the model and thus could safely be removed. For the sake of brevity, we will look at only the first two categories below.

```
cor(Raw.Clean[1:4, 1:4], use = "complete.obs")
```

Safety

```
##             road.death food.safety   suicides  homicides
## road.death   1.0000000  -0.9875198 -0.8376620  0.9072672
## food.safety -0.9875198   1.0000000  0.7411857 -0.9621796
## suicides    -0.8376620   0.7411857  1.0000000 -0.5302811
## homicides    0.9072672  -0.9621796 -0.5302811  1.0000000
```

```
cor(Raw.Clean[5:12, 5:12], use = "complete.obs")
```

Affordability

```
##                 c.tax.rate rental.housing       gini cellular.cost
## c.tax.rate       1.0000000      0.2180718 -0.3532968     0.2211649
## rental.housing   0.2180718      1.0000000  0.6050010     0.9999632
## gini            -0.3532968      0.6050010  1.0000000     0.6065062
## cellular.cost    0.2211649      0.9999632  0.6065062     1.0000000
## food.cost        0.2176966      0.9999039  0.6041189     0.9998084
## transit.pass     0.2186394      0.9999676  0.6044811     0.9999555
## apt.buy.cost     0.2177529      0.9999603  0.6049471     0.9999557
## toothpaste.cost  0.2183412      0.9997149  0.6043569     0.9996086
##                 food.cost transit.pass apt.buy.cost toothpaste.cost
## c.tax.rate      0.2176966    0.2186394    0.2177529       0.2183412
## rental.housing  0.9999039    0.9999676    0.9999603       0.9997149
## gini            0.6041189    0.6044811    0.6049471       0.6043569
## cellular.cost   0.9998084    0.9999555    0.9999557       0.9996086
## food.cost       1.0000000    0.9997630    0.9997454       0.9999441
## transit.pass    0.9997630    1.0000000    0.9999952       0.9994920
## apt.buy.cost    0.9997454    0.9999952    1.0000000       0.9994744
## toothpaste.cost 0.9999441    0.9994920    0.9994744       1.0000000
```

Immediately, we can see that some variables are highly correlated with others within the Affordability category. For instance food.cost, transit.pass, apt.buy.cost and toothpaste.cost are all over 99% correlated (typically, over 80% is considered high). This likely justifies further investigation, potentially looking at collinearity or using a pca analysis. Alternatively, since many of these variables are intrinsically linked to the overall price of goods in a city, a broader index which encapsulates more goods might be used as a replacement for similar variables. Eliminating variables althogether isn't something that will be addressed in this project. Instead, this will be considered as a recommendation.

Approach

The following eight sections detail the approach taken to address the stated objectives:

1. Exploratory Analysis
   - Determined need for feature scaling
   - Looked at missing data patterns and varieties
   - Conducted correlation analysis
2. Assessed current imputation methods
   - Replace 0's with NA's where necessary
   - Compared knn and mean-imputation techniques for accuracy.

3. Imputed data based on findings
    - Imputed missing values using mean imputation
4. Feature scaling and standardization
    - Audited current standardization techniques
    - Adjusted certain formulas where necessary to account for population difference, anomolies etc…
5. Developed weighting strategy for interactive application
6. Developed application user-interface
7. Completed back end development of application
8. Hosted application on shinyapps.io

Sections 2 to 6 are explained below, while section 7 and 8 are contained within the "app.R" file in the app folder.

Imputation

As discovered in the exploratory analysis, many of the variables contain zeros, some of which are representative of the actual value, others which should actually be coded as *NA*. Recoding is essential because the zeros will cause distortions during the normalization process and will vias the data. Without recollecting data common sense must be used in recoding the data. Recoding is also essential because the imputation process that will be used will only deal with *NA* values, not zeros.

This process was done manually, looking at each variable individual, judging the likelihood of zeros being present, and replacing with *NA*s where it was deemed necessary.

The code used in this section may be viewed in **Appendix - I**.

Imputation: Create Lookup Tables

Now we have all of the data in a format which is condusive to manipulation. At this stage data waspulled from several of the original excel tables to create look up tables containing information about category weighting, regions, population, exchange rate, and minimum wage. This information will be drawn upon in later analysis steps.

The code to construct these charts is visible in **Appendix II**

Imputation: Preparing for NA imputation

Prior to imputation, it was necessary to create a dataframe which included potential predictive candidates such as region and population.

```r
# population and region maybe be useful for the imputation algorithm, so they will be
# attached to the raw dataset here.

data.full <- cbind(Raw.Clean, LU.Population[, 2])
colnames(data.full)[101] <- "Population"

data.full <- cbind(data.full, LU.Country.Region[, 2])
colnames(data.full)[102] <- "Region"
```

```
data.full$Population <- as.numeric(as.character((data.full$Population)))
```

Currently, the imputation method being utilized by the YouthfulCities team is a form of mean imputation, where they are drawing the average from similar geographical and socioeconomic regions. Although imputation is not the most important purpose of this project, exploring alternatives was both prudent and a good opportunity to explore alternative methods.

Simply based upon knowledge in the area, it is plausible to assume that data are "not missing at random". That is, there are non-random patterns which are influencing frequencies in missing data. For example, cities in certain regions, loosely defined as "developing regions" are likely to have higher rates of missing data than other regions.

We can verify this quickly:

```
missingRow <- apply(data.full, 1, function(x) sum(is.na(x)))
missingRow <- as.data.frame(cbind("missing" = missingRow, "Region" = as.character(data.full[,102])))

missingRow[, "missing"] <- as.numeric(missingRow[, "missing"])
missingRow[, "Region"] <- as.character(missingRow[, "Region"])

aggregate(missingRow[,"missing"], by=missingRow[c("Region")], FUN=mean)
```

```
##           Region        x
## 1         Africa 7.500000
## 2           Asia 5.923077
## 3         Europe 5.444444
## 4 Latin America 5.222222
## 5   Middle East 5.000000
## 6        USA/Can 3.333333
```

The table above depicts the average number of missing values for cities within each region. Among other things, it demonstrates that cities in Africa have more than 2x more missing values on average than North America. This could result from varying information being collected by governments, data being less readily available, lack of availability in English, or any number of other factors. Regardless, it appears to be a predictor of missing values. It may not be the only one, but it does tell us that the data is not missing at random.

With this in mind, we will attempt to develop an imputation technique leveraging potential drivers of missingness as our predictors.

Imputation: Testing current imputation strategy

Although there are many different procedures for imputation, the analysis below compares just two: mean imputation and k-nearest neighbors. The latter may be a good candidate due to the inclusion of the categorical variable *Region*.

The first step in comparing the two methods, is to input random *NA* values into the current dataset. Once these missing datapoints have been created, we will attempt to use the two algorithms (mean and knn imputation) to predict them. Since we have preserved the full dataset, we will be able to compare the imputed values with the true values to assess the accuracy of each method.

```
# create a function that is designed to introduce NA's to up to 20% of a column

insert_nas <- function(x) {
  len <- length(x)
  n <- sample(1:floor(0.2*len), 1)
  i <- sample(1:len, n)
  x[i] <- NA
  x
}

# Apply the function to each column (except for population and region)
data.NA <- as.data.frame(apply(data.full[1:100], 2, function(x) insert_nas(x)))
data.NA[, c("Population", "Region")] <- data.full[,c("Population", "Region")]
```

Imputation: Mean Imputation

First, we will impute the missing values using *mean imputation.* For this imputation strategy, if there is a missing value, the algorithm will average out the relevant scores for cities in the same region, then impute that value in the place of NA. For example, if the number of homicides is missing for Boston, the algorithm will average the number of homicides reported for every other North American city, then use this value for Boston.

```
m.impute <- data.frame() # create an empty dataframe

for (c in 1:ncol(data.NA)) { # for each column
  for (r in 1:length(data.NA[, c])) { # and for each row in the column
    if (is.na(data.NA[r,c])) {     # if it is NA, continue
      m.impute[r,c] <- as.numeric(mean(data.full[data.full$Region == data.full[r, "Region"], c], na.rm 
      # set it equal to the average of other cities in the region
    } else {  #i f not NA, do not change
      m.impute[r, c] <- as.numeric(data.full[r, c])
    }
  }
}
```

At this point there are 0 missing values in the m.impute dataset.

Imputation: KNN Imputation

Next, we will test k-nearest means method. Since imputation is not the focus of this project, we will simply utilize the default KNN implementation that is packaged with the ‘VIM’ package and test varying numbers of neighbors. The default distance measure is euclidean distance. More intensive experimentation in this area may prove beneficial but determining actual values through research where possible will be the most important course of action for improving overall index accuracy.

```
k.impute <- kNN(data.NA, 1:102, k = 14) # using 14 neighbors was shown to maximize accuracy in repeated
```

Now there are 0 missing values in the k.impute dataset.

Imputation: Model Evaluation

To compare the success of each method, we will look at the normalized root mean squared error. Essentially, this will method will:

1. Take the difference between the actual values and the predicted values.
2. Square the result to avoid negative and positive values cancelling each other out.
3. Sum the errors.
4. Take the square root to put the figures back into relevant units (as opposed to squared units)
5. Then divide by the range to normalize the figure (so that we can easily compare between variables.)

```r
range <- apply(data.NA[ ,1:100], 2, function(x) max(x, na.rm = TRUE) - min(x, na.rm = TRUE))

# mean imputation
mse.m <- (m.impute[ ,1:100] - data.full[ ,1:100])^2
mse.m <- apply(mse.m, 2, function(x) sum(x, na.rm = TRUE))
rmse.m <- sqrt(mse.m)
nrmse.m <- rmse.m/range

#knn imputation
mse.k <- (k.impute[ ,1:100] - data.full[ ,1:100])^2
mse.k <- apply(mse.k, 2, function(x) sum(x, na.rm = TRUE))
rmse.k <- sqrt(mse.k)
nrmse.k <- rmse.k/range

nrmse.m <- sum(nrmse.m)
nrmse.k <- sum(nrmse.k)
```

The normalized root mean square error using **mean imputation** is 35.3295584.

The normalized root mean square error using **knn imputation** is 41.2179971.

A smaller RMSE means that there is less deviation from the real figures, therefore, it appears that mean imputation is the more accurate strategy. Given this finding, for the purposes of this assignment, we will leverage mean imputation to fill in the gaps in our data.

Imputation: Imputing the data

```r
data.impute <- data.frame()

for (c in 1:ncol(data.full)) {

  for (r in 1:length(data.full[, c])) {

    if (is.na(data.full[r,c])) {     # if it is NA, continue

      data.impute[r,c] <- as.numeric(mean(data.full[data.full$Region == data.full[r, "Region"], c], na.
    } else {  #if not NA, keep the same as original dataset
      data.impute[r, c] <- as.numeric(data.full[r, c])
    }
  }
}

# now lets replace row and column names
colnames(data.impute) <- colnames(data.full)
row.names(data.impute) <- row.names(data.full)
```

Feature Scaling / Standardization

The raw YouthfulCities dataset contains many variables which are not directly comparable due to varying ranges and units. With this in mind, it is necessary to normalize the variables. This is something that the raw YouthfulCities dataset had already done, however, because we have changed many zeros to *NA* values, and re-imputed values, it is important to reperform this step.

Additionally, in the data provided, certain variables were normalized using equations which rewarded higher values, even though they should be rewarding lower figures. One example of this is the 'getaway plane cost', which should use a minimizing equation toreward lower costs.

There are also instances where variables could be more meaningful if viewed from a per capita basis. Public Libraries is one example of this.

With this in mind we will define normalization functions, then address each variable one by one, to determine the most appropriate method for normalization.

Since the final index should have a range from 0 to 100, and because we need all variables on the same range (so that they are weighted fairly in the final index value), we will use a simply normalization formula which does the following:

1. Take each value and subtracts the minimum value in the column.
2. Divides that value by the maximum value, minus the minimum value.
3. Finally, we will want to reward certain variable for high values and certain values for lower values. + For example, more health clinics and less homicides are both positive.
   - Therefore, for scores that we would like to minimize, we will invert them by subtracting the value from 1.
   - For certain variables, it will also be necessary to perform other transofrmations. For example, we want to ensure that variable which are in monetary units are denominated in the same currency and reflect the current minimum wage in the country.

Normalize and Reward Smaller Values

```
min.norm <- function(col, df) {
  (1 - ((df[r, col] -
     min(df[, col]) ) /
             (max(df[, col]) -
                      min(df[, col]))))
}

# To construct with weighting add *
# as.numeric(LU.Ind.Cat.Weight[
#   LU.Ind.Cat.Weight$Indicator == col,  "Weight"]
```

Normalize and Reward Greater Values

```
# Normalize Maximize - Reward Larger Values
max.norm <- function(col, df) {
  (df[r, col] -
      min(df[, col]) ) /
              (max(df[, col]) -
                      min(df[, col]))
}

# To construct with weighting add *
# as.numeric(LU.Ind.Cat.Weight[
#   LU.Ind.Cat.Weight$Indicator == col,  "Weight"]
```

Normalize Accounting for Population

```
by.per.capita <- function(col) {
  (data.impute[r, col] /
    data.impute[r, "Population"])* 100000
}

per.capita <- vector()
```

Normalize Accounting for Exchange Rate

```
by.exchange <- function(col) {
  (data.impute[r, col] *
    LU.Exchange.Wage[r, "Exchange"]) / LU.Exchange.Wage[r, "Minimum.Wage"]
}

# a function which runs through each row of a given column dividing exchange rate by minimum wage.
```

Normalize Accounting for Geographic Area

```
by.area <- function(col) {
  (data.impute[r, col] / LU.Exchange.Wage[r, "Area"])
}
```

Transformations were applied to each variable manually. This was a conscious choice which allowed for greater immersion in the data (ultimately helping to identify inconsistencies) and for greater attention to detail in applying appropriate formulas. The calculations may be found in **Appendix III**.

Results

Results: Imputation and Cleaning

With all of the variables normalized, we can assign the dataframe containing our transformed variables to a final dataframe and save the result to a csv titled "final.ranking.values.unweight.csv" for use by the interactive application.

```r
final.scores.unweight <- data.impute.transform[1:100]

write.table(final.scores.unweight, file = "final.ranking.values.unweight.csv",
            sep = " ",
            row.names = TRUE,
            col.names = TRUE)
```

Results: Interactive Application

With respect to objective two, the interactive application, all code may be found in the app.R file in the app folder.

Here, I will review some of the specific feature of the app, as well as the weighting crieria which was used.

Results: New Weighting System In the original YouthfulCities dataset, research was conducted in the form of surveys to determine how important each category was to youth around the world. The research yielded relative weights which were applied to raw scores for each variable. Next, the variable scores for each category were average to determine category scores for each city (for example, Safety in Accra). Finally, the scores for each category was added together to give an overall score.
In summary, you would have an index composed of categories important to youth, which was weighted by the average importance attributed to them by youth around the world.

The goal of the interactive application was to add a new layer of insight to the dataset to increase it's usefulness to users. Specifically, the application has been designed to let users compare up to three cities of their choice, while weighting the categories according to their own needs and preferences. There were a few considerations and objectives in building the application itself:

1. **Usability**
    - The platform would have to be extremely straight forward and (despite containing many variables) avoid overwhelming the user.
2. **Understandability**
    - There were certainly complex weighting systems to choose from which may have eased usability slightly, but to ensure credibility, the system should be designed so that the average user can understand the underlying mechanism being used.
3. **Customizability**
    - Not only should users be able to weight variables as important vs. unimportant, they should be able to exclude variables entirely if they would like to.
4. **Interpretability**
    - Output should be extremely clear and the user should be able to compare cities on the basis of any category that interests them.

Ultimately, the application was designed with two components, "Controls" and "Dashboard".

Results: App-Controls The controls section allows the user to tweak the input parameters. First, they are able to choose three cities to compare. Next, they are able to weight categories to their liking. The weighting system is simple enough that any user can understand it. It offers the following options for each category:

0 Do Not Include - Multiplies the raw score by 0 so that it does not impact the final score.

From here, every increment of 1 increases the weighting factor linearly by 0.2.

1 Unimportant - Multiplies the raw score by 0.2
2 - Multiplies the raw score by 0.4
3 - Multiplies the raw score by 0.6
4 - Multiplies the raw score by 0.8
5 - Default - Multiplies the raw score by 1
6 - Multiplies the raw score by 1.2
7 - Multiplies the raw score by 1.4
8 - Multiplies the raw score by 1.6
9 - Multiplies the raw score by 1.8
10 Very Important - Multiplies the raw score by 2

Results: App-Dashboard The dashboard displays the final index scores for each of the three cities (based on the input paramters) across the top of the page, then presents a chart below that breaks down the scores into weighted category scores for each city that can be easily compared.

Results: App-Next Steps The intent behind this is simply to demonstrate some of the possibilities that are available to the YouthfulCities team. With this foundation in place, it would be trivial to add on visualizations, increase the number of cities available to compare, and add on any number of other features to add value to this already rich dataset.

Works Cited

Freudenberg, M. (2003). Composite Indicators of Country Performance.

Heitjan, Daniel F., and Srabashi Basu. "Distinguishing"Missing at Random" and "Missing Completely at Random"." The American Statistician 50.3 (1996): 207-13. Web. 5 Dec. 2015

Nardo, M., Saisana, M., Saltelli, A., Tarantola, S., Hoffman, A., & Giovannini, E. (2005). Handbook on constructing composite indicators.

Appendices

Appendix I - Imputing Zeros

```r
# food.safety
row.names(Raw.Clean[Raw.Clean$food.safety == 0, ])
  # likely a case of missing data, replace with NA
Raw.Clean$food.safety[Raw.Clean$food.safety == 0] <- NA

# homicides - it is unlikely that there were no homicides, replace 0 with NA
Raw.Clean$homicides[Raw.Clean$homicides == 0] <- NA
row.names(Raw.Clean[is.na(Raw.Clean$homicides), ])

# km.transit
row.names(Raw.Clean[Raw.Clean$km.transit == 0, ])
  # these countries have (limited) transit, replace with NA
Raw.Clean$km.transit[Raw.Clean$km.transit == 0] <- NA

# hrs.transit
```

```r
row.names(Raw.Clean[Raw.Clean$hrs.transit == 0, ])
  # these countries have (limited) transit, replace with NA
Raw.Clean$hrs.transit[Raw.Clean$hrs.transit == 0] <- NA

# bike.rental
row.names(Raw.Clean[Raw.Clean$bike.rental == 0, ])
  # feasible that (some of) these are zero, left unchanged

# km.bike.path
row.names(Raw.Clean[Raw.Clean$km.bike.path == 0, ])
  # feasible that (some of) these are zero, left unchanged

# commute.time.transit
row.names(Raw.Clean[Raw.Clean$commute.time.transit == 0, ])
  # feasible that this is zero due to transit being unfeasible for commute, left unchanged

# commute.time.transit
row.names(Raw.Clean[Raw.Clean$transit.app == 0, ])
  # feasible that this is zero, left unchanged

# taxi.rate
row.names(Raw.Clean[Raw.Clean$taxi.rate == 0, ])
  # these cities have taxis, convert 0's to NA's
Raw.Clean$taxi.rate[Raw.Clean$taxi.rate == 0] <- NA

# health.clinics
row.names(Raw.Clean[Raw.Clean$health.clinics == 0, ])
  # these cities have health clinics, convert 0's to NA's
Raw.Clean$health.clinics[Raw.Clean$health.clinics == 0] <- NA

# sex.health.clinics
row.names(Raw.Clean[Raw.Clean$sex.health.clinic == 0, ])
  # feasible, left unchanged

# sex.homeless.shelter
row.names(Raw.Clean[Raw.Clean$homeless.shelter == 0, ])
  # possible, left unchanged

# employment.programs
row.names(Raw.Clean[Raw.Clean$employment.programs == 0, ])
  # possible, left unchanged

# youth.employment.centers
row.names(Raw.Clean[Raw.Clean$youth.employment.centers == 0, ])
  # several of these definitely have youth employment centers, convert to NA
Raw.Clean$youth.employment.centers[Raw.Clean$youth.employment.centers == 0] <- NA

# new.jobs.2013
row.names(Raw.Clean[Raw.Clean$new.jobs.2013 == 0, ])
  # almost certainly a case of missing data
Raw.Clean$new.jobs.2013[Raw.Clean$new.jobs.2013 == 0] <- NA

# water.scale
```

```r
row.names(Raw.Clean[Raw.Clean$water.scale == 0, ])
  # almost certainly a case of missing data
Raw.Clean$water.scale[Raw.Clean$water.scale == 0] <- NA

# smart.cities.scale
row.names(Raw.Clean[Raw.Clean$smart.cities.scale == 0, ])
  # feasible that there  are not policies in place, left unchanged

# qty.recycled.materials
row.names(Raw.Clean[Raw.Clean$qty.recycled.materials == 0, ])
  # feasible, left unchanged

# qty.annual.waste
row.names(Raw.Clean[Raw.Clean$qty.annual.waste == 0, ])
  # almost certainly a case of missing data
Raw.Clean$qty.annual.waste[Raw.Clean$qty.annual.waste == 0] <- NA

# carbon.emissions
row.names(Raw.Clean[Raw.Clean$carbon.emissions == 0, ])
  # almost certainly a case of missing data
Raw.Clean$carbon.emissions[Raw.Clean$carbon.emissions == 0] <- NA

#num.recycled.materials
row.names(Raw.Clean[Raw.Clean$num.recycled.materials == 0, ])
  # almost certainly a case of missing data
Raw.Clean$num.recycled.materials[Raw.Clean$num.recycled.materials == 0] <- NA
  #feasible, left unchanged

#registered.vehicles
row.names(Raw.Clean[Raw.Clean$registered.vehicles == 0, ])
  # almost certainly a case of missing data
Raw.Clean$registered.vehicles[Raw.Clean$registered.vehicles == 0] <- NA

# tuition.cost
row.names(Raw.Clean[Raw.Clean$tuition.cost == 0, ])
  # almost certainly a case of missing data
Raw.Clean$registered.vehicles[Raw.Clean$tuition.cost == 0] <- NA

# undergrad.enrollment
row.names(Raw.Clean[Raw.Clean$undergrad.enrollment == 0, ])
  # almost certainly a case of missing data
Raw.Clean$undergrad.enrollment[Raw.Clean$undergrad.enrollment == 0] <- NA

# student.housing
row.names(Raw.Clean[Raw.Clean$student.housing == 0, ])
  # possibly a case of missing data
Raw.Clean$student.housing[Raw.Clean$student.housing == 0] <- NA

# student.debt
row.names(Raw.Clean[Raw.Clean$student.debt == 0, ])
  # possibly a case of missing data
Raw.Clean$student.debt[Raw.Clean$student.debt == 0] <- NA
```

```r
# employment.initiatives.scale
row.names(Raw.Clean[Raw.Clean$employment.initiatives.scale == 0, ])
  # possible, left unchanged

# age.register.business
row.names(Raw.Clean[Raw.Clean$age.register.business == 0, ])
  # almost certainly a case of missing data
Raw.Clean$age.register.business[Raw.Clean$age.register.business == 0] <- NA

# early.entrepreneurship
row.names(Raw.Clean[Raw.Clean$early.entrepreneurship == 0, ])
  # possible, left unchanged

# sports.facilities
row.names(Raw.Clean[Raw.Clean$sports.facilities == 0, ])
  # almost certainly a case of missing data
Raw.Clean$sports.facilities[Raw.Clean$sports.facilities == 0] <- NA

# business.banking
row.names(Raw.Clean[Raw.Clean$business.banking == 0, ])
  # almost certainly a case of missing data
Raw.Clean$business.banking[Raw.Clean$business.banking == 0] <- NA

# personal.banking
row.names(Raw.Clean[Raw.Clean$personal.banking == 0, ])
  # almost certainly a case of missing data
Raw.Clean$personal.banking[Raw.Clean$personal.banking == 0] <- NA

# chartered.banks
row.names(Raw.Clean[Raw.Clean$chartered.banks == 0, ])
  # almost certainly a case of missing data
Raw.Clean$chartered.banks[Raw.Clean$chartered.banks == 0] <- NA

# online.banking
row.names(Raw.Clean[Raw.Clean$online.banking == 0, ])
  # almost certainly a case of missing data
Raw.Clean$online.banking[Raw.Clean$online.banking == 0] <- NA

# mobile.banking
row.names(Raw.Clean[Raw.Clean$mobile.banking == 0, ])
  # almost certainly a case of missing data
Raw.Clean$mobile.banking[Raw.Clean$mobile.banking == 0] <- NA

# financial.literacy
row.names(Raw.Clean[Raw.Clean$financial.literacy == 0, ])
  # possible, left unchanged

# cellular.package.cost
row.names(Raw.Clean[Raw.Clean$cellular.package.cost == 0, ])
  # almost certainly a case of missing data
Raw.Clean$cellular.package.cost[Raw.Clean$cellular.package.cost == 0] <- NA

# num.nightclubs
```

```r
row.names(Raw.Clean[Raw.Clean$num.nightclubs == 0, ])
  # almost certainly a case of missing data
Raw.Clean$num.nightclubs[Raw.Clean$num.nightclubs == 0] <- NA

# music.festivals
row.names(Raw.Clean[Raw.Clean$music.festivals == 0, ])
  # possibly a case of missing data
Raw.Clean$music.festivals[Raw.Clean$music.festivals == 0] <- NA

# graffiti.street.art
row.names(Raw.Clean[Raw.Clean$graffiti.street.art == 0, ])
  # feasible, left unchanged

# number.pro.sports.teams
row.names(Raw.Clean[Raw.Clean$number.pro.sports.teams == 0, ])
  # possibly a case of missing data
Raw.Clean$number.pro.sports.teams[Raw.Clean$number.pro.sports.teams == 0] <- NA

# film.festivals
row.names(Raw.Clean[Raw.Clean$film.festivals == 0, ])
  # likely a case of missing data
Raw.Clean$film.festivals[Raw.Clean$film.festivals == 0] <- NA

# councillor.age
row.names(Raw.Clean[Raw.Clean$councillor.age == 0, ])
  # certainly a case of missing data
Raw.Clean$councillor.age[Raw.Clean$councillor.age == 0] <- NA

# volunteer.opportunities
row.names(Raw.Clean[Raw.Clean$volunteer.opportunities == 0, ])
  # certainly a case of missing data
Raw.Clean$volunteer.opportunities[Raw.Clean$volunteer.opportunities == 0] <- NA

# youth.advisory.board
row.names(Raw.Clean[Raw.Clean$youth.advisory.board == 0, ])
  # feasible, left unchanged

# highschool.volunteer
row.names(Raw.Clean[Raw.Clean$highschool.volunteer == 0, ])
  # left unchanged

# num.restaurants
row.names(Raw.Clean[Raw.Clean$num.restaurants == 0, ])
  # certainly a case of missing data
Raw.Clean$num.restaurants[Raw.Clean$num.restaurants == 0] <- NA

# last.call.index
row.names(Raw.Clean[Raw.Clean$last.call.index == 0, ])
  # potentially no legislated last call, left unchanged

# num.food.festival
row.names(Raw.Clean[Raw.Clean$num.food.festival == 0, ])
  # certainly a case of missing data
```

```r
Raw.Clean$num.food.festival[Raw.Clean$num.food.festival == 0] <- NA

# young.designer
row.names(Raw.Clean[Raw.Clean$young.designer == 0, ])
  # possible, left unchanged

# fashion.incubators
row.names(Raw.Clean[Raw.Clean$fashion.incubators == 0, ])
  # possible, left unchanged

# design.schools
row.names(Raw.Clean[Raw.Clean$design.schools == 0, ])
  # possible, left unchanged
```

Appendix II - Constructing Look Up Tables

```r
# Create Lookup Table for Indicator Category and Weight
LU.Ind.Cat.Weight <- Raw[,1:4]
colnames(LU.Ind.Cat.Weight) <- LU.Ind.Cat.Weight[1, ]
LU.Ind.Cat.Weight <- LU.Ind.Cat.Weight[-1, ]
LU.Ind.Cat.Weight <- LU.Ind.Cat.Weight[, -1]

# Create Country Region look up chart
LU.Country.Region <- Attr.Score[, 2:3]
colnames(LU.Country.Region) <- LU.Country.Region[1, ]
LU.Country.Region <- LU.Country.Region[-1, ]

# Population
LU.Population <- Education[-1, ]
LU.Population <- LU.Population[-1, ]
LU.Population <- LU.Population[, -1]
colnames(LU.Population) <- LU.Population[1, ]
LU.Population <- LU.Population[-1, ]
LU.Population <- LU.Population[1:55, ]
colnames(LU.Population)[1] <- c("Country")
LU.Population <- LU.Population[,c(1,5)]

# Exchange Rate and Minimum Wage
LU.Exchange.Wage <- Affordability[1:57,c(2,11,13)]
Area <- Transit[1:57, 23]
LU.Exchange.Wage <- cbind(LU.Exchange.Wage, Area)
LU.Exchange.Wage <- LU.Exchange.Wage[-1, ]
LU.Exchange.Wage <- LU.Exchange.Wage[-1, ]
colnames(LU.Exchange.Wage) <- c("Country", "Exchange", "Minimum.Wage", "Area")


LU.Exchange.Wage$Exchange <- as.numeric(as.character(LU.Exchange.Wage$Exchange))
LU.Exchange.Wage$Minimum.Wage <- as.numeric(as.character(LU.Exchange.Wage$Minimum.Wage))

LU.Exchange.Wage[, 2:ncol(LU.Exchange.Wage)] <-sapply(LU.Exchange.Wage[, 2:ncol(LU.Exchange.Wage)],
                                        function(x) as.numeric(as.character(x)))
```

Appendix III - Normalization

```r
## SAFETY ##

# Minimize - Road.Death
t <- vector()
data.impute.transform <- data.impute

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("road.death", data.impute)
  norm.road.death <- t
}

data.impute.transform[, "road.death"] <- norm.road.death * 100


# MAXIMIZE - food.safety

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("food.safety", data.impute)
  norm.food.safety <- t
}

data.impute.transform[, "food.safety"] <- norm.food.safety*100


# Minimize - Suicide

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("suicides", data.impute)
  norm.suicides <- t
}

data.impute.transform[, "suicides"] <- norm.suicides*100

# Minimize - Homicides

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("homicides")
  homicides.per.capita <- t
}

data.impute.transform[, "homicides"] <- homicides.per.capita

# minimze
for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("homicides", data.impute.transform)
  norm.homicides <- t
}

data.impute.transform[, "homicides"] <- norm.homicides * 100
```

```
### AFFORDABILITY ###

# Consumer Tax Rate
for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("c.tax.rate", data.impute)
  norm.c.tax.rate <- t
}

data.impute.transform[, "c.tax.rate"]  <- norm.c.tax.rate * 100


# Exchange Minimized- Rental Housing

  # multiply each cost by respective exhange rate.
for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("rental.housing")
  rental.housing.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "rental.housing"] <- rental.housing.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("rental.housing", data.impute.transform)
  norm.rental.housing <- t

}

data.impute.transform[, "rental.housing"] <- norm.rental.housing *100



# Exchange Minimized - Gini

data.impute.transform[ ,"gini"] <- data.impute[ ,"gini"] *
  100 *
  as.numeric(LU.Ind.Cat.Weight[
  LU.Ind.Cat.Weight$Indicator == "gini",  "Weight"])


# Exchange Minimized- Cellular Cost

  # multiply each cost by respective exhange rate.
for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("cellular.cost")
  cellular.cost.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "cellular.cost"] <- cellular.cost.exchange

  # normalize new column
```

```r
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("cellular.cost", data.impute.transform)
  norm.cellular.cost <- t

}

data.impute.transform[, "cellular.cost"] <- norm.cellular.cost *100



# Exchange Minimized - Food - 12 Eggs

  # multiply each cost by respective exhange rate.
for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("food.cost")
  food.cost.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "food.cost"] <- food.cost.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("food.cost", data.impute.transform)
  norm.food.cost <- t

}

data.impute.transform[, "food.cost"] <- norm.food.cost *100


# Exchange Minimized - Monthly Transit Pass

  # multiply each cost by respective exhange rate.
for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("transit.pass")
  transit.pass.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "transit.pass"] <- transit.pass.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("transit.pass", data.impute.transform)
  norm.transit.pass <- t

}

data.impute.transform[, "transit.pass"] <- norm.transit.pass *100
```

```r
# Exchange Minimized - Apartment Purchase COst

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("apt.buy.cost")
  apt.buy.cost.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "apt.buy.cost"] <- apt.buy.cost.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("apt.buy.cost", data.impute.transform)
  norm.apt.buy.cost <- t

}

data.impute.transform[, "apt.buy.cost"] <- norm.apt.buy.cost *100


# Exchange Minimized - Toothpaste Cost

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("toothpaste.cost")
  toothpaste.cost.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "toothpaste.cost"] <- toothpaste.cost.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("toothpaste.cost", data.impute.transform)
  norm.toothpaste.cost <- t

}

data.impute.transform[, "toothpaste.cost"] <- norm.toothpaste.cost *100


### TRANSIT ###


# MAXIMIZE - km.transit

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("km.transit", data.impute)
  norm.km.transit <- t
}
```

```r
data.impute.transform[, "km.transit"] <- norm.km.transit*100


# MAXIMIZE - hrs.transit

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("hrs.transit", data.impute)
  norm.hrs.transit <- t
}

data.impute.transform[, "hrs.transit"] <- norm.hrs.transit*100


### POPULATION ADJUSTED = Public Bike Rental ###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("bike.rental")
  bike.rental.per.capita <- t
}

data.impute.transform[, "bike.rental"] <- bike.rental.per.capita


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("bike.rental", data.impute.transform)
  norm.bike.rental <- t
}

data.impute.transform[, "bike.rental"] <- norm.bike.rental * 100


### AREA ADJUSTED = kms bike path ###

# by area
for(r in 1:nrow(data.impute)){
  t[r] <- by.area("km.bike.path")
  bike.rental.area <- t
}

data.impute.transform[, "km.bike.path"] <- bike.rental.area


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("km.bike.path", data.impute.transform)
  norm.km.bike.path <- t
}

data.impute.transform[, "km.bike.path"] <- norm.km.bike.path * 100
```

```
### Minimize - Commuter Time Transit

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("commute.time.transit", data.impute)
  norm.commute.time.transit <- t
}

data.impute.transform[, "commute.time.transit"] <- norm.commute.time.transit*100


### Minimize - Commuter Time Foot

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("commute.time.foot", data.impute)
  norm.commute.time.foot <- t
}

data.impute.transform[, "commute.time.foot"] <- norm.commute.time.foot*100



### Minimize - Commuter Time Airport

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("commute.time.airport", data.impute)
  norm.commute.time.airport <- t
}

data.impute.transform[, "commute.time.airport"] <- norm.commute.time.airport*100


### Maximize - Transit App

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("transit.app", data.impute)
  norm.transit.app <- t
}

data.impute.transform[, "transit.app"] <- norm.transit.app*100


### Maximize - Density

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("pop.density", data.impute)
  norm.pop.density <- t
}

data.impute.transform[, "pop.density"] <- norm.pop.density*100



# Exchange Minimized - Taxi Stand Rate
```

```
  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("taxi.rate")
  taxi.rate.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "taxi.rate"] <- taxi.rate.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("taxi.rate", data.impute.transform)
  norm.taxi.rate <- t

}

data.impute.transform[, "taxi.rate"] <- norm.taxi.rate *100
```

```
### Health

### MAXIMIZE Health Clinics by pop###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("health.clinics")
  health.clinics.per.capita <- t
}

data.impute.transform[, "health.clinics"] <- health.clinics.per.capita


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("health.clinics", data.impute.transform)
  norm.health.clinics <- t
}

data.impute.transform[, "health.clinics"] <- norm.health.clinics * 100

### MAXIMIZE Sexual Health Clinics by pop###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("sex.health.clinic")
  sex.health.clinic.per.capita <- t
}

data.impute.transform[, "sex.health.clinic"] <- sex.health.clinic.per.capita


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("sex.health.clinic", data.impute.transform)
  norm.sex.health.clinic <- t
}

data.impute.transform[, "sex.health.clinic"] <- norm.sex.health.clinic * 100


### MAXIMIZE homeless shelter by pop###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("homeless.shelter")
  homeless.shelter.per.capita <- t
}

data.impute.transform[, "homeless.shelter"] <- homeless.shelter.per.capita


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("homeless.shelter", data.impute.transform)
  norm.homeless.shelter <- t
}

data.impute.transform[, "homeless.shelter"] <- norm.homeless.shelter * 100


### Maximize - Urban Smoking Scale

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("smoking.scale", data.impute)
  norm.smoking.scale <- t
}

data.impute.transform[, "smoking.scale"] <- norm.smoking.scale*100


### MAXIMIZE mental health facilities by pop###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("mental.health.facilities")
  mental.health.facilities.per.capita <- t
}

data.impute.transform[, "mental.health.facilities"] <- mental.health.facilities.per.capita


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("mental.health.facilities", data.impute.transform)
  norm.mental.health.facilities <- t
}
```

```r
data.impute.transform[, "mental.health.facilities"] <- norm.mental.health.facilities * 100


### Maximize - Doctor Density

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("doctor.density", data.impute)
  norm.doctor.density <- t
}

data.impute.transform[, "doctor.density"] <- norm.doctor.density*100


### Travel ###

### Maximize - Cities Connected

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("city.connections.flight", data.impute)
  norm.city.connections.flight<- t
}

data.impute.transform[, "city.connections.flight"] <- norm.city.connections.flight*100


# Exchange Minimized - Getaway Bus Cost

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("getaway.bus.cost")
  getaway.bus.cost.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "getaway.bus.cost"] <- getaway.bus.cost.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("getaway.bus.cost", data.impute.transform)
  getaway.bus.cost <- t

}

data.impute.transform[, "getaway.bus.cost"] <- getaway.bus.cost *100


# Minimize - Getaway Bus Time

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("getaway.bus.time", data.impute)
  norm.getaway.bus.time <- t
}
```

```r
data.impute.transform[, "getaway.bus.time"] <- norm.getaway.bus.time*100

# Minimize - Getaway Bus Freq

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("getaway.bus.freq", data.impute)
  norm.getaway.bus.freq <- t
}

data.impute.transform[, "getaway.bus.freq"] <- norm.getaway.bus.freq*100


# Exchange Minimized - Getaway Plane Cost

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("getaway.plane.cost")
  getaway.plane.cost.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "getaway.plane.cost"] <- getaway.plane.cost.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("getaway.plane.cost", data.impute.transform)
  getaway.plane.cost <- t

}

data.impute.transform[, "getaway.plane.cost"] <- getaway.plane.cost *100


# Minimize - Getaway Plane Time

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("getaway.plane.time", data.impute)
  norm.getaway.plane.time <- t
}

data.impute.transform[, "getaway.plane.time"] <- norm.getaway.plane.time*100

# Minimize - Getaway Bus Freq

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("getaway.plane.freq", data.impute)
  norm.getaway.plane.freq <- t
}

data.impute.transform[, "getaway.plane.freq"] <- norm.getaway.plane.freq*100
```

```r
# Exchange Minimized - Hostel Stay

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("hostel.stay")
  hostel.stay.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "hostel.stay"] <- hostel.stay.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("hostel.stay", data.impute.transform)
  norm.hostel.stay <- t

}

data.impute.transform[, "hostel.stay"] <- norm.hostel.stay *100


### EMPLOYMENT ####

# Minimize - Youth Unemployment

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("unemployment.youth", data.impute)
  norm.unemployment.youth <- t
}

data.impute.transform[, "unemployment.youth"] <- norm.unemployment.youth*100


# Maximize - Employment Initiatives

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("employment.programs", data.impute)
  norm.employment.programs <- t
}

data.impute.transform[, "employment.programs"] <- norm.employment.programs*100

### MAXIMIZE Youth Employment Centers pop###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("youth.employment.centers")
  youth.employment.centers.per.capita <- t
}

data.impute.transform[, "youth.employment.centers"] <- youth.employment.centers.per.capita
```

```r
# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("youth.employment.centers", data.impute.transform)
  norm.youth.employment.centers <- t
}

data.impute.transform[, "youth.employment.centers"] <- norm.youth.employment.centers * 100


### MAXIMIZE Number of New Jobs Created 2013###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("new.jobs.2013")
  new.jobs.2013.per.capita <- t
}

data.impute.transform[, "new.jobs.2013"] <- new.jobs.2013.per.capita


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("new.jobs.2013", data.impute.transform)
  norm.new.jobs.2013 <- t
}

data.impute.transform[, "new.jobs.2013"] <- norm.new.jobs.2013 * 100
```

```r
## ENVIRONMENT ###

# maximze water scale
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("water.scale", data.impute.transform)
  norm.water.scale <- t
}

data.impute.transform[, "water.scale"] <- norm.water.scale * 100


# maximze smart cities scale
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("smart.cities.scale", data.impute.transform)
  norm.smart.cities.scale <- t
}

data.impute.transform[, "smart.cities.scale"] <- norm.smart.cities.scale * 100


### MAXIMIZE Quantity of Recycled Materials###

# per capita
# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("qty.recycled.materials")
  qty.recycled.materials.per.capita <- t
}

data.impute.transform[, "qty.recycled.materials"] <- qty.recycled.materials.per.capita


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("qty.recycled.materials", data.impute.transform)
  norm.qty.recycled.materials <- t
}

data.impute.transform[, "qty.recycled.materials"] <- norm.qty.recycled.materials * 100

# Minimize Quantity of Recycled Materials###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("qty.annual.waste")
  qty.annual.waste.per.capita <- t
}

data.impute.transform[, "qty.annual.waste"] <- qty.annual.waste.per.capita


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("qty.annual.waste", data.impute.transform)
  norm.qty.annual.waste <- t
}

data.impute.transform[, "qty.annual.waste"] <- norm.qty.annual.waste * 100


# Minimize Carbon Emissions ###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("carbon.emissions")
  carbon.emissions.per.capita <- t
}

data.impute.transform[, "carbon.emissions"] <- carbon.emissions.per.capita


# maximze
for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("carbon.emissions", data.impute.transform)
  norm.carbon.emissions <- t
}

data.impute.transform[, "carbon.emissions"] <- norm.carbon.emissions * 100
```

```
# Maximze Number of Recycled Materials
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("num.recycled.materials", data.impute.transform)
  norm.num.recycled.materials <- t
}


data.impute.transform[, "num.recycled.materials"] <- norm.num.recycled.materials * 100



# Minimize Regstered Vehicles ###

# per capita
for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("registered.vehicles")
  registered.vehicles.per.capita <- t
}

data.impute.transform[, "registered.vehicles"] <- registered.vehicles.per.capita


# minimize
for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("registered.vehicles", data.impute.transform)
  norm.registered.vehicles <- t
}

data.impute.transform[, "registered.vehicles"] <- norm.registered.vehicles * 100



### Education

# Maximize post secondary institutions per capita

for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("post.secondary.institutions")
  post.secondary.institutions.per.capita <- t
}

data.impute.transform[, "post.secondary.institutions"] <- post.secondary.institutions.per.capita


# maximize
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("post.secondary.institutions", data.impute.transform)
  norm.post.secondary.institutions <- t
}

data.impute.transform[, "post.secondary.institutions"] <- norm.post.secondary.institutions * 100
```

```
# Exchange Minimized - Tuition

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("tuition.cost")
  tuition.cost.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "tuition.cost"] <- tuition.cost.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("tuition.cost", data.impute.transform)
  norm.tuition.cost <- t

}

data.impute.transform[, "tuition.cost"] <- norm.tuition.cost *100


# Maximize post secondary enrollment per capita

for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("undergrad.enrollment")
  undergrad.enrollment.per.capita <- t
}

data.impute.transform[, "undergrad.enrollment"] <- undergrad.enrollment.per.capita


# maximize
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("undergrad.enrollment", data.impute.transform)
  norm.undergrad.enrollment <- t
}

data.impute.transform[, "undergrad.enrollment"] <- norm.undergrad.enrollment * 100


# Exchange Minimized - Student Housing Cost

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("student.housing")
  student.housing.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "student.housing"] <- student.housing.exchange
```

```
  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("student.housing", data.impute.transform)
  norm.student.housing <- t

}

data.impute.transform[, "student.housing"] <- norm.student.housing * 100


# Exchange Minimized - Student Debt

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("student.debt")
  student.debt.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "student.debt"] <- student.debt.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("student.debt", data.impute.transform)
  norm.student.debt <- t

}

data.impute.transform[, "student.debt"] <- norm.student.debt * 100


### Entrepreneurship

# max employment initiatives scale
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("employment.initiatives.scale", data.impute.transform)
  norm.employment.initiatives.scale <- t
}

data.impute.transform[, "employment.initiatives.scale"] <- norm.employment.initiatives.scale * 100

# minimize age register business
for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("age.register.business", data.impute.transform)
  norm.age.register.business <- t
}

data.impute.transform[, "age.register.business"] <- norm.age.register.business * 100

# max age register business
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("num.incubators", data.impute.transform)
```

```r
  norm.num.incubators <- t
}

data.impute.transform[, "num.incubators"] <- norm.num.incubators * 100

# max early entrepreneurship
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("early.entrepreneurship", data.impute.transform)
  norm.early.entrepreneurship <- t
}

data.impute.transform[, "early.entrepreneurship"] <- norm.early.entrepreneurship * 100

# min ease of doing business
for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("ease.doing.business", data.impute.transform)
  norm.ease.doing.business <- t
}

data.impute.transform[, "ease.doing.business"] <- norm.ease.doing.business * 100


### PUBLIC SPACE


# Maximize green space per capita

for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("green.public.space")
  green.public.space.per.capita <- t
}

data.impute.transform[, "green.public.space"] <- green.public.space.per.capita


# maximize
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("green.public.space", data.impute.transform)
  norm.green.public.space <- t
}

data.impute.transform[, "green.public.space"] <- norm.green.public.space * 100




# Maximize sports facilities per capita

for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("sports.facilities")
  sports.facilities.per.capita <- t
}
```

```r
data.impute.transform[, "sports.facilities"] <- sports.facilities.per.capita


# maximize
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("public.libraries", data.impute.transform)
  norm.public.libraries <- t
}

data.impute.transform[, "public.libraries"] <- norm.public.libraries * 100


# Maximize sports facilities per capita

for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("public.libraries")
  public.libraries.per.capita <- t
}

data.impute.transform[, "public.libraries"] <- public.libraries.per.capita


# maximize
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("public.libraries", data.impute.transform)
  norm.public.libraries <- t
}

data.impute.transform[, "public.libraries"] <- norm.public.libraries * 100


### Financial Services

# Maximize business banking availability


for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("business.banking", data.impute.transform)
  norm.business.banking <- t
}

data.impute.transform[, "business.banking"] <- norm.business.banking * 100


# Maximize personal banking availability

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("personal.banking", data.impute.transform)
  norm.personal.banking <- t
}

data.impute.transform[, "personal.banking"] <- norm.personal.banking * 100
```

```
# Maximize number of chartered banks per capita

for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("chartered.banks")
  chartered.banks.per.capita <- t
}

data.impute.transform[, "chartered.banks"] <- chartered.banks.per.capita


# maximize
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("chartered.banks", data.impute.transform)
  norm.chartered.banks <- t
}

data.impute.transform[, "chartered.banks"] <- norm.chartered.banks * 100


# Maximize online banking

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("online.banking", data.impute.transform)
  norm.online.banking <- t
}

data.impute.transform[, "online.banking"] <- norm.online.banking * 100


# Maximize mobile banking

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("mobile.banking", data.impute.transform)
  norm.mobile.banking <- t
}

data.impute.transform[, "mobile.banking"] <- norm.mobile.banking * 100

# Maximize financial literacy

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("financial.literacy", data.impute.transform)
  norm.financial.literacy <- t
}

data.impute.transform[, "financial.literacy"] <- norm.financial.literacy * 100


### DIVERSITY

# Maximize voting languages

for (r in 1:nrow(data.impute)) {
```

```r
  t[r] <- max.norm("voting.languages", data.impute.transform)
  norm.voting.languages <- t
}

data.impute.transform[, "voting.languages"] <- norm.voting.languages * 100

# Maximize food diversity

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("diversity.food", data.impute.transform)
  norm.diversity.food <- t
}

data.impute.transform[, "diversity.food"] <- norm.diversity.food * 100


# Maximize LGBT Openness

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("openness.lgbtq", data.impute.transform)
  norm.openness.lgbtq <- t
}

data.impute.transform[, "openness.lgbtq"] <- norm.openness.lgbtq * 100



# Maximize Immigrant Openness

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("openness.immigrants", data.impute.transform)
  norm.openness.immigrants <- t
}

data.impute.transform[, "openness.immigrants"] <- norm.openness.immigrants * 100


# Maximize Religious Diversity

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("diversity.religion", data.impute.transform)
  norm.diversity.religion <- t
}

data.impute.transform[, "diversity.religion"] <- norm.diversity.religion * 100


# Maximize Global Gender Gap

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("gender.gap.index", data.impute.transform)
  norm.gender.gap.index <- t
}
```

```r
data.impute.transform[, "gender.gap.index"] <- norm.gender.gap.index * 100
```

```r
### Digital Access

# Maximize Cellular Comp
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("cellular.comp", data.impute.transform)
  norm.cellular.comp <- t
}

data.impute.transform[, "cellular.comp"] <- norm.cellular.comp * 100


# Exchange Minimized - Cellular Package Cost

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("cellular.package.cost")
  cellular.package.cost.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "cellular.package.cost"] <- cellular.package.cost.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("cellular.package.cost", data.impute.transform)
  norm.cellular.package.cost <- t

}

data.impute.transform[, "cellular.package.cost"] <- norm.cellular.package.cost * 100


# Maximize free.wifi
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("free.wifi", data.impute.transform)
  norm.free.wifi <- t
}

data.impute.transform[, "free.wifi"] <- norm.free.wifi * 100

# Maximize mobile infrastructure
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("mobile.infrastructure", data.impute.transform)
  norm.mobile.infrastructure <- t
}

data.impute.transform[, "mobile.infrastructure"] <- norm.mobile.infrastructure * 100

# Maximize Municipal open data
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("municipal.open.data", data.impute.transform)
  norm.municipal.open.data <- t
}

data.impute.transform[, "municipal.open.data"] <- norm.municipal.open.data * 100


# Maximize Gamers and Developer Friendliness
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("gamers.developers.scale", data.impute.transform)
  norm.gamers.developers.scale <- t
}

data.impute.transform[, "gamers.developers.scale"] <- norm.gamers.developers.scale * 100



### Music

# Maximize nightclubs
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("num.nightclubs", data.impute.transform)
  norm.num.nightclubs <- t
}

data.impute.transform[, "num.nightclubs"] <- norm.num.nightclubs * 100

# Maximize music festivals
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("music.festivals", data.impute.transform)
  norm.music.festivals <- t
}

data.impute.transform[, "music.festivals"] <- norm.music.festivals * 100


### Creative Arts

# Maximize Grafitti and street Art

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("graffiti.street.art", data.impute.transform)
  norm.graffiti.street.art <- t
}

data.impute.transform[, "graffiti.street.art"] <- norm.graffiti.street.art * 100

### Sports

#Maximize Number of sports teams

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("number.pro.sports.teams", data.impute.transform)
  norm.number.pro.sports.teams <- t
}

data.impute.transform[, "number.pro.sports.teams"] <- norm.number.pro.sports.teams * 100

#Maximize Number of sports facilities

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("num.pro.sports.facilities", data.impute.transform)
  norm.num.pro.sports.facilities <- t
}

data.impute.transform[, "num.pro.sports.facilities"] <- norm.num.pro.sports.facilities * 100


# Exchange Minimized - Cost of Shorts

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("cost.shorts")
  cost.shorts.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "cost.shorts"] <- cost.shorts.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("cost.shorts", data.impute.transform)
  norm.cost.shorts <- t

}

data.impute.transform[, "cost.shorts"] <- norm.cost.shorts * 100


# Exchange Minimized - Cost of Gym

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("cost.gym")
  cost.gym.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "cost.gym"] <- cost.gym.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("cost.gym", data.impute.transform)
  norm.cost.gym <- t
```

```r
}

data.impute.transform[, "cost.gym"] <- norm.cost.gym * 100


### Film

# Maximize FIlm Festivals
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("film.festivals", data.impute.transform)
  norm.film.festivals <- t
}

data.impute.transform[, "film.festivals"] <- norm.film.festivals * 100

# Exchange Minimized - Cost of a Movie Ticket

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("cost.movie")
  cost.movie.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "cost.movie"] <- cost.movie.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("cost.movie", data.impute.transform)
  norm.cost.movie <- t

}

data.impute.transform[, "cost.movie"] <- norm.cost.movie * 100


# Maximize cinema seats per capita

for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("cinema.seats")
  cinema.seats.per.capita <- t
}

data.impute.transform[, "cinema.seats"] <- cinema.seats.per.capita


# maximize
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("cinema.seats", data.impute.transform)
  norm.cinema.seats <- t
}
```

```r
data.impute.transform[, "cinema.seats"] <- norm.cinema.seats * 100
```

```r
### Civic Engagement

# Minimize Voting Age

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("voting.age", data.impute.transform)
  norm.voting.age <- t
}

data.impute.transform[, "voting.age"] <- norm.voting.age * 100

# Minimize Councillor Age

for (r in 1:nrow(data.impute)) {
  t[r] <- min.norm("councillor.age", data.impute.transform)
  norm.councillor.age <- t
}

data.impute.transform[, "councillor.age"] <- norm.councillor.age * 100

# Maximize Volunteer Opportunities

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("volunteer.opportunities", data.impute.transform)
  norm.volunteer.opportunities <- t
}

data.impute.transform[, "volunteer.opportunities"] <- norm.volunteer.opportunities * 100


# Maximize Political Influence

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("youth.advisory.board", data.impute.transform)
  norm.youth.advisory.board <- t
}

data.impute.transform[, "youth.advisory.board"] <- norm.youth.advisory.board * 100

# Maximize Political Influence

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("highschool.volunteer", data.impute.transform)
  norm.highschool.volunteer<- t
}

data.impute.transform[, "highschool.volunteer"] <- norm.highschool.volunteer * 100


### Food
```

```r
# Maximize restaurants per capita

for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("num.restaurants")
  num.restaurants.per.capita <- t
}

data.impute.transform[, "num.restaurants"] <- num.restaurants.per.capita


# maximize
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("num.restaurants", data.impute.transform)
  norm.num.restaurants <- t
}

data.impute.transform[, "num.restaurants"] <- norm.num.restaurants * 100


# Maximize Last Call

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("last.call.index", data.impute.transform)
  norm.last.call.index<- t
}

data.impute.transform[, "last.call.index"] <- norm.last.call.index * 100


# Maximize Food Festivals

for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("num.food.festival", data.impute.transform)
  norm.num.food.festival<- t
}

data.impute.transform[, "num.food.festival"] <- norm.num.food.festival * 100


# Exchange Minimized - Cost of Fast Food

  # multiply each cost by respective exhange rate.

for(r in 1:nrow(data.impute)){
  t[r] <- by.exchange ("fast.food")
  fast.food.exchange <- t
}

  # input into new dataframe
data.impute.transform[, "fast.food"] <- fast.food.exchange

  # normalize new column
for (r in 1:nrow(data.impute.transform)) {
  t[r] <- min.norm("fast.food", data.impute.transform)
  norm.fast.food <- t

}

data.impute.transform[, "fast.food"] <- norm.fast.food * 100


# Fashion

# Maximize young designer showcase
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("young.designer", data.impute.transform)
  norm.young.designer<- t
}

data.impute.transform[, "young.designer"] <- norm.young.designer * 100


# Maximize young designer showcase
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("fashion.incubators", data.impute.transform)
  norm.fashion.incubators<- t
}

data.impute.transform[, "fashion.incubators"] <- norm.fashion.incubators * 100


# Maximize fashion schools per capita

for(r in 1:nrow(data.impute)){
  t[r] <- by.per.capita("design.schools")
  design.schools.per.capita <- t
}

data.impute.transform[, "design.schools"] <- design.schools.per.capita


# maximize
for (r in 1:nrow(data.impute)) {
  t[r] <- max.norm("design.schools", data.impute.transform)
  norm.design.schools <- t
}

data.impute.transform[, "design.schools"] <- norm.design.schools * 100
```