Food Contamination Source Detection

Mohammad Mahdi Hajiha

9/7/2021

R Markdown

We are given a set of indices related to the customer areas that the sick people come from. Using a proximity matrix of customer areas to stores and assignment matrix of food product to store, we can determine the likelihood of each food product being contaminated.

```
library(zipcodeR)
names(zip_code_db)
```

```
##  [1] "zipcode"                 "zipcode_type"
##  [3] "major_city"              "post_office_city"
##  [5] "common_city_list"        "county"
##  [7] "state"                   "lat"
##  [9] "lng"                     "timezone"
## [11] "radius_in_miles"         "area_code_list"
## [13] "population"              "population_density"
## [15] "land_area_in_sqmi"       "water_area_in_sqmi"
## [17] "housing_units"           "occupied_housing_units"
## [19] "median_home_value"       "median_household_income"
## [21] "bounds_west"             "bounds_east"
## [23] "bounds_north"            "bounds_south"
```

We pick 22 zipcodes in North West Arkansas area:

```
ziplist = c(72768, 72715, 72714, 72751, 72736, 72739, 72712, 72719, 72756,
            72722, 72734, 72718, 72758, 72745, 72761, 72762, 72764, 72738,
            72704, 72703, 72701, 72730)
```

Initialize the coordinates of customer areas

```
l = length(ziplist)
x_w = rep(0, l)
x_e = rep(0, l)
y_n = rep(0, l)
y_s = rep(0, l)
centroid_lat = rep(0, l)
centroid_lng = rep(0, l)
x_c = rep(0, l)
y_c = rep(0, l)
pop = rep(0, l)

for(i in 1:l){
  x_w[i] = zip_code_db[which( zip_code_db$zipcode == ziplist[i]) , "bounds_west"]
  x_e[i] = zip_code_db[which( zip_code_db$zipcode == ziplist[i]) , "bounds_east"]
  y_n[i] = zip_code_db[which( zip_code_db$zipcode == ziplist[i]) , "bounds_north"]
  y_s[i] = zip_code_db[which( zip_code_db$zipcode == ziplist[i]) , "bounds_south"]
  centroid_lat[i] =  zip_code_db[which( zip_code_db$zipcode == ziplist[i]) , "lat"]
  centroid_lng[i] =  zip_code_db[which( zip_code_db$zipcode == ziplist[i]) , "lng"]
  pop[i] =  zip_code_db[which( zip_code_db$zipcode == ziplist[i]) , "population"]
  x_c[i] = (x_w[i] + x_e[i])/2
  y_c[i]= (y_n[i] + y_s[i])/2
}
```

Make the plot of areas with demographic information:

```
##   region state.name county.name county.fips.numeric  cbsa cbsa.title
## 1  70560  louisiana      iberia               22045 10020       <NA>
## 2  70510  louisiana   vermilion               22113 10020       <NA>
## 3  70592  louisiana   lafayette               22055 10020       <NA>
## 4  70548  louisiana   vermilion               22113 10020       <NA>
## 5  70560  louisiana   vermilion               22113 10020       <NA>
## 6  70578  louisiana   lafayette               22055 10020       <NA>
##   metropolitan.micropolitan.statistical.area
## 1                                       <NA>
## 2                                       <NA>
## 3                                       <NA>
## 4                                       <NA>
## 5                                       <NA>
## 6                                       <NA>
```

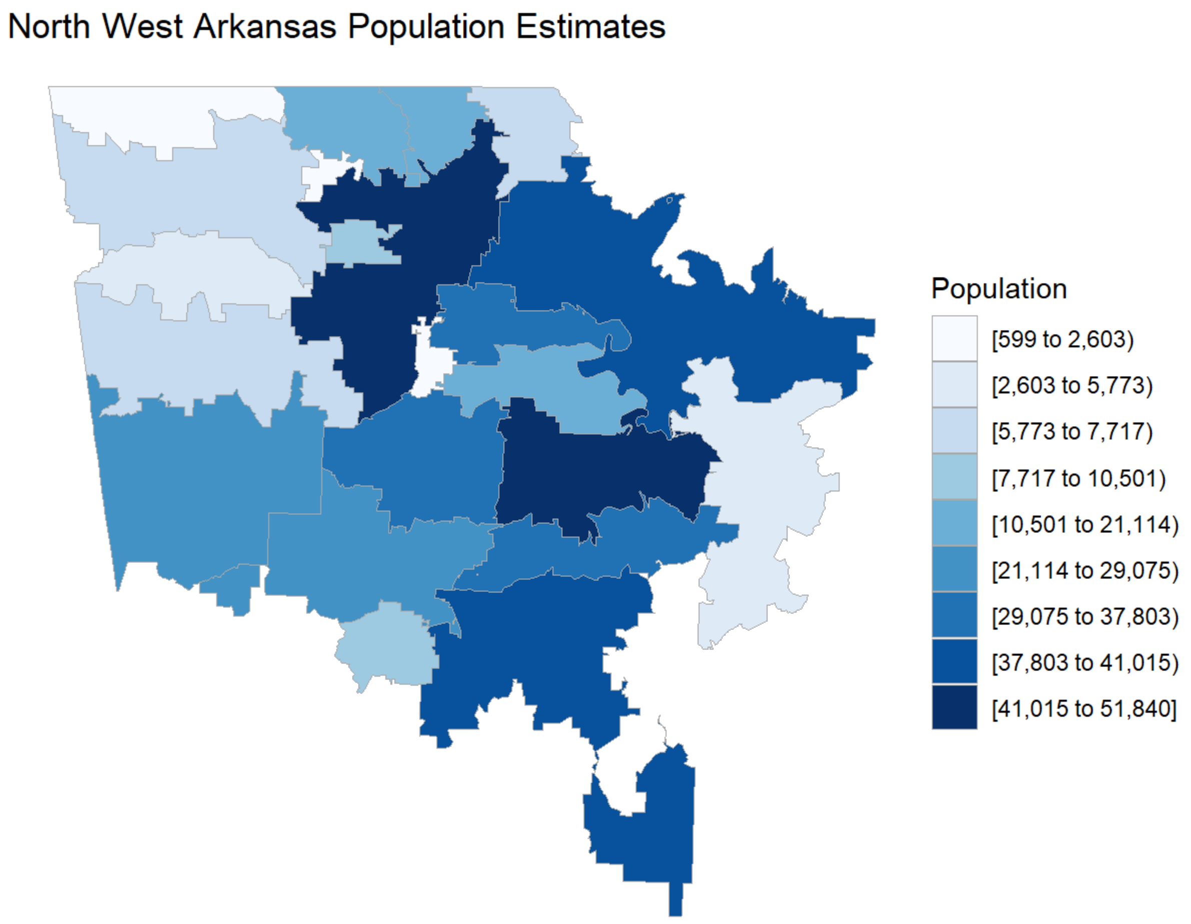
```
## Loading required package: acs
```

```
## Loading required package: stringr
```

```
## Loading required package: XML
```

```
##
## Attaching package: 'acs'
## The following object is masked from 'package:base':
##
##     apply
```

```
## [1] "region" "value"
```



capture the number of customer areas:

```
c =length(ziplist)
```

Capture the coordinates of stores:

```
storeCoords = matrix(0,nrow = 30,ncol = 2)
allcoords = c(
  36.123113936797765, -94.15307469647306, "Walmart",
  36.05917620090397, -94.16826492730598, "Walmart",
  36.07860311276757, -94.21012387382375,"Walmart",
  36.0532891862925, -94.2011288941917, "Walmart",
  36.09149062952714, -94.12312024842625,"Walmart",
  36.056603242256315, -94.20649856879572, "Aldi",
  36.17225183961867, -94.1433068000788,"Aldi",
  36.11653202083311, -94.14676948792697,  "Whole food market",
  36.07967017893204, -94.1766693920096,   "Harps",
  36.11072729048959, -94.14885855743809,  "Harps",
  36.08820124556599, -94.11920377153957,  "Harps",
  36.12893900174774, -94.14404422000965,  "Natural grocers",
  36.070540587548656, -94.15667332507864,  "Ozark",
  36.180635870886285, -94.2113265981114,  "Harps",
  336.11504675016247, -94.18164681776115,  "Sams",
  36.16764239653776, -94.09203057512475, "Walmart",
  36.1729319621903, -94.14525137904437, "Walmart",
  36.19871276156632, -94.18440075684978, "Walmart",
  36.19942621910992, -94.13875630348161, "Walmart",
  36.18341440041083, -94.20844677512713,  "Harps",
  36.34828713792211, -94.11730563548043, "Harps",
  36.33946293745903, -94.19822769787933, "India Grand Bazaar",
  36.31051761784318, -94.1768467608696, "The Fresh",
  36.25983261092982, -94.15320784627708,"Harps",
  36.1746511629422, -94.13827457961328 , "10 Box Cost Plus",
  36.44687878360732, -94.12063892054967, "Walmart",
  36.51404180637726, -94.27831861754763, "Walmart",
  36.39388474218325, -94.22126399551749, "Walmart",
  36.3575222045497, -94.17103692298501, "Walmart",
  36.359404428633866, -94.25306224670877,"Walmart",
  36.36983483658389, -94.22493045395112,  "Walmart",
  36.375533984665815, -94.20860291693954, "Walmart",
  36.39107173696156, -94.21967792034948, "Walmart",
  36.35610476391975, -94.17004867223065, "Walmart",
  36.33969355181811, -94.22457553732445, "Walmart",
  36.3338455744834, -94.14927338581889, "Walmart",
  36.33516119589057, -94.12653090690505, "Walmart",
  36.31524731730122, -94.12308843755933, "Walmart",
  36.31538062431988, -94.12391215644486 , "Walmart",
  36.335426880918554, -94.12686027334658,"Walmart",
  36.09173250177016, -94.12246822755881, "Walmart",
  36.09279208919376, -94.12356178550766, "Walmart",
  36.197443497744096, -94.1828910011279, "Walmart",
  36.169083802742456, -94.14610757020229, "Walmart",
  36.175589906710556, -94.16300771787593, "Walmart",
  36.3334966835299, -94.14927167095931, "Walmart",
  36.33915902329997, -94.22364179375515, "Walmart",
  36.39203324501347, -94.21998391382736, "Walmart",
  36.378318650218205, -94.20997603313656, "Walmart",
  36.376971684909655, -94.18784720259255, "Walmart",
  36.35742290461202, -94.17025459791934,"Walmart",
  36.362035280295125, -94.2525641647124, "Walmart",
  36.371423712447154, -94.22544331542541, "Walmart",
  36.334069382157, -94.14884020397395, "Walmart",
  36.335650688700774, -94.12658186343708, "Walmart",
  36.280420898632435, -94.15183320285055,"Walmart",
  36.29862175739072, -94.18631524857899, "Walmart",
  36.19637683645089, -94.18486532026792, "Walmart",
  36.16630196649756, -94.09090719949677, "Walmart",
  36.480824051630286, -94.24769454519118,"Harps",
  36.44575605972363, -94.23602228357758, "Allens",
  36.36103452772716, -94.22358354424347 , "Aldi",
  36.3585135798278, -94.23116152926738,  "India Plaza",
  36.35426777330645, -94.21271137257163,  "Home Market Foods",
  36.364881637452996, -94.21567409469245,  "Diamond's Food",
  36.337010601010476, -94.19442161242993,  "World Food Mart",
  36.34071897185698, -94.11664029566482,  "Harps",
  36.33774393226712, -94.14206803769827, "A Chau Oriental",
  36.336022700809515, -94.14261610592402, "10 Box Cost Plus",
  36.334389817016046, -94.15636785863205, "Tropical Food Market",
  36.33390357532949, -94.13609658172112 , "Aldi",
  36.29833652057349, -94.1863719620193,  "Walmart",
  36.16553352450921, -94.1422924915573, "Asian Amigo",
  36.04188639443438, -94.25672013420402,"Walmart",
  35.99532534999367, -94.30892919812285  , "Harps"
)

storeCoords = matrix(allcoords, nrow = length(allcoords)/3, ncol = 3, byrow = T)
NumStores = nrow(storeCoords)
storesY = as.numeric(storeCoords[,1])
storesX = as.numeric(storeCoords[,2])
storeCoordinates = data.frame(storesX, storesY)
```

calculate the store attraction factor:

```
A = rep(0.25, length(storesX))

for(i in 1:length(A)){
  if(storeCoords[,3][i] == "Walmart"  ){
    A[i] = 1
  }
  if(storeCoords[,3][i] == "Sams" ){
    A[i] = 1
  }
  if(storeCoords[,3][i] == "Whole Food Market" ){
    A[i] = 0.4
  }
  if(storeCoords[,3][i] ==  "Natural grocers"){
    A[i] = 0.4
  }
  if(storeCoords[,3][i] == "Harps"){
    A[i] = 0.5
  }
  if(storeCoords[,3][i] == "Aldi"){
    A[i] = 0.75
  }

}
```

Compute the distances:

```
distances = matrix(0, nrow =  c, ncol = nrow(storeCoords))
for( i in 1: c){
  for(j in 1:NumStores){
    distances[i, j] = sqrt((x_c[i]-storeCoordinates$storesX[j])^2 + (y_c[i]-storeCoordinates$sto
resY[j])^2)
  }
}
```

Compute the proximity matrix:

```
P = matrix(0, nrow =  c, ncol = NumStores)
gamma = 1
for (i in 1:c){
  for( j in 1:NumStores){
     P[i,j] =  (A[j]*distances[i,j]**(-gamma)) /
     (t(A) %*% distances[i,]**(-gamma))
  }
}
```

normalize the result:

```
for (i in 1:c){
  S = sum( P[i,])
  for( j in 1:NumStores){
    P[i,j] =  P[i,j] / S
   }
}
#make sure the row sums are 1:
rowSums(P)
```

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

##Sampling from population

Initialize the food vector and make one food product contaminated, sat the first one:

```
NumFoodProducts = 10
theta = rep(0, NumFoodProducts)
theta[1] = 1
```

Randomly assign food products to 1/10 of stores:

```
set.seed(123)
  R = matrix(0, nrow = NumFoodProducts, ncol = NumStores)
  for(i in 1:nrow(R)){
    R[i,sample(1:NumStores, size =floor(NumStores*1/10)  , replace = F) ]  = 1
  }
```

Calculate the population density

```
popDensity = pop/sum(pop)
```

Compute binomial probabilities for the individuals of each region becoming sick:

```
p = rep(0,c)
for (r in 1:c){
  outerSum = rep(0, NumFoodProducts)
  for(j in 1:NumFoodProducts){
    innerSum = rep(0,NumStores)
    for ( k in 1:NumStores){
      innerSum[k] = P[r,k]*R[j,k]
    }
    outerSum[j] = theta[j] * sum(innerSum)
  }
  p[r] =  sum(outerSum) * popDensity[r]
}
```

Take the sample using binomial probabilities:

```
data.list = list()
for(r in 1:c){
  tmp =    rbinom(n = 1, prob =p[r], size = pop[r] )
  if(tmp ==0){
  }
  else{
    tmp1 = rep(r, tmp)
    tmp1 = data.frame(tmp1)
    data.list[[r]] = tmp1
  }
}
data = do.call(rbind, data.list)
nrow(data)   #about more than 6k people sampled
```

```
## [1] 3217
```

```
colnames(data) = "sampled_poeple"
head(data)
```

```
##   sampled_poeple
## 1              1
## 2              2
## 3              2
## 4              2
## 5              2
## 6              2
```

See the distribution of samples:

```
table(data)
```

```
## data
##   1   2   3   4   5   7   8   9  10  11  13  14  15  16  17  18  19  20  21  22 
##   1  65  31   8   4 436  19 389   4  19 246  30 102 465 639   2 125 214 409   9
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Make the likelihood function and test it:

```
theta = theta[1:(NumFoodProducts-1)]
likelihood = function(theta){
  lambda = 0
  theta_k = 1-sum(theta)
  theta_complete = c(theta, theta_k)
  regions = data$sampled_poeple
  pb = rep(0, length(regions))
  index = 1
  for( index in 1:length(regions)){
    r = regions[index]
    pb[index] = theta_complete %*% t(P[r,]%*%t(R))
  }
  # return(pb)
  return( sum(log(pb)))
}

#theta = rep(x = 1/NumFoodProducts,NumFoodProducts)
likelihood(theta = theta)
```

```
## [1] -7263.723
```

Then, compute the gradient of the likelihood:

```
grad_likelihood_2 = function(theta){
  theta_k = 1-sum(theta)
  theta_complete = c(theta, theta_k)
  lambda = 0
  regions = data$sampled_poeple
  pb = rep(0, length(regions))
  #index = 1
  #a = matrix(0,nrow =length(regions), ncol = NumFoodProducts)
  Sum = array(data = 0, dim = c(NumFoodProducts-1,1))
  for( index in 1:length(regions)){
    r = regions[index]
    A = t(P[r,]%*%t(R))
    pb[index] = theta_complete%*%A
    Sum = Sum+ A[1:(NumFoodProducts-1)]/pb[index]
    # for(kk in 1:NumFoodProducts){
    #   Sum[kk] = Sum[kk] + A[kk]/pb[index]
    # }
  }
  return(Sum)
}
grad_likelihood_2(theta = theta)
```

```
##           [,1]
##  [1,] 3217.000
##  [2,] 2245.369
##  [3,] 2867.038
##  [4,] 2381.824
##  [5,] 2398.830
##  [6,] 3241.604
##  [7,] 3493.789
##  [8,] 1865.094
##  [9,] 2929.529
```

Doing the experiment for 10 food sizes from 10 to 100:

We have included the code for sampling sick people and optimizing theta in an R file called experiment. So in each experiment we just need to run that file. The number of time the algorithms is able to identify the true contaminated products per each rank in each experiment is written to a matrix called “result” with rows equal to the number of food product sizes and the columns equal to number of experiments (30 here). The computation time is stored in a matrix called “TIME”.

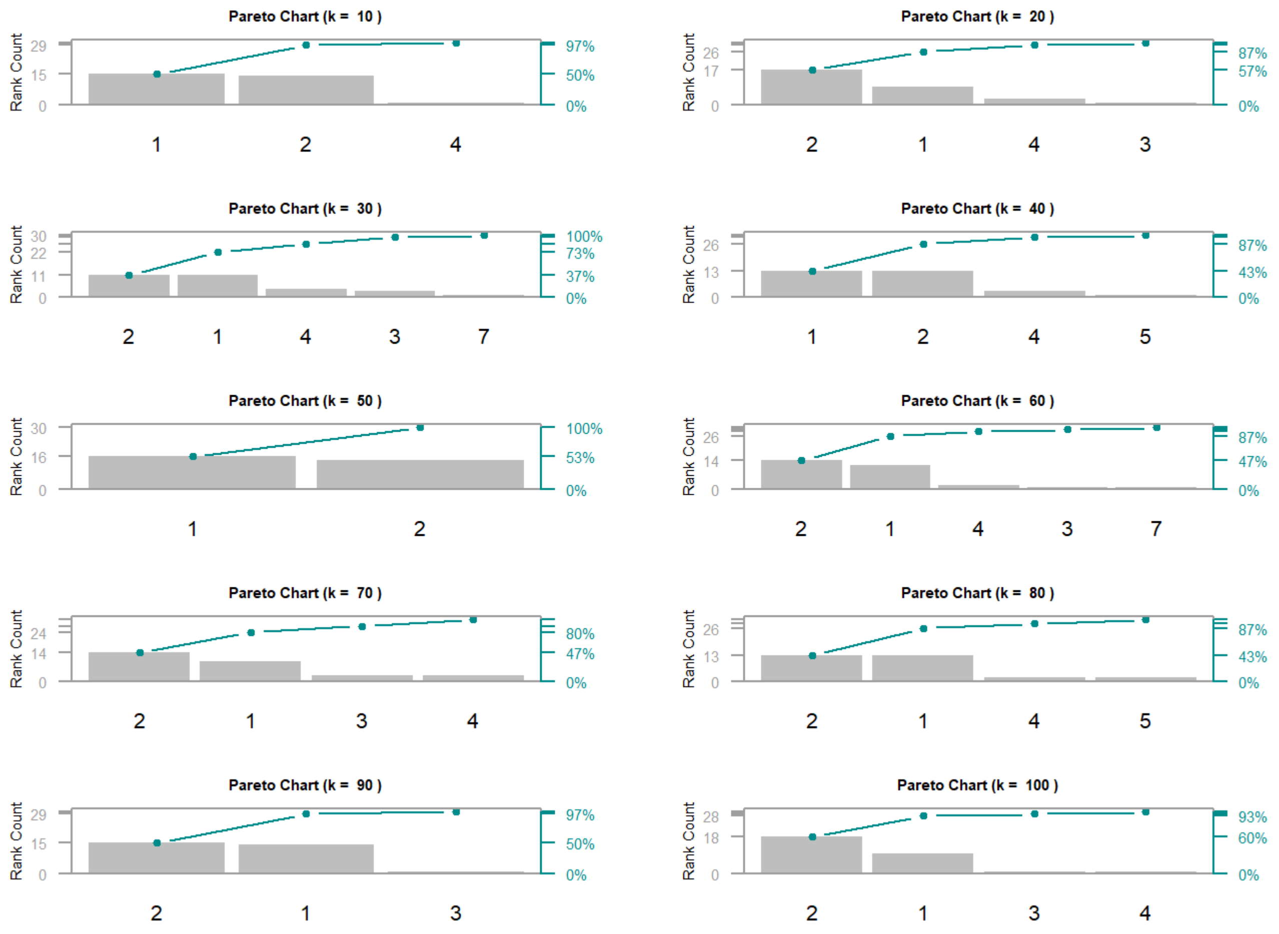
```
set.seed(123)
numSamples = 30
numFoodProductSizes = 10
#Initialize the result and TIME matrices:
TIME = matrix(0, nrow =numFoodProductSizes, ncol = numSamples )
result = matrix(0, nrow =numFoodProductSizes, ncol = numSamples )
#Initialize number of food products:
NumFoodProducts = 10
#Initialize a list to store the results of each experiment:
output = list()
#Run a for loop for the number of times that is equal to the row of "TIME" matrix (10 here)
for(ex0 in 1: nrow(TIME)){
  R = matrix(0, nrow = NumFoodProducts, ncol = NumStores)
  #generate food to store assignment matrix:
  for(i in 1:nrow(R)){
    R[i,sample(1:NumStores, size =floor(NumStores*1/10)  , replace = F) ]  = 1
  }

  #run the experiments a number of times that is equal to the columns of "TIME" (30 here):
  for(ex1 in 1:ncol(TIME)){
    source("experiment.R")
    #calculate the last element of the likelihhod vector based on the defined properties that
the sum is equal to 1:
    output[[ex0]] = c(res$solution,1-sum(res$solution))
    print(ex1)
    #compute the rank of true contaminated food product by our likelihhod approach:
    result[ex0, ex1] =    NumFoodProducts -  sum((output[[ex0]][1] -output[[ex0]]) >= 0) +1
    TIME[ex0, ex1] = (end - start)
  }
  #increase the number of food products for the next set of simulation experiments;
  NumFoodProducts  = NumFoodProducts + 10
  print(ex0)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 1
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 2
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 3
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 4
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 5
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 6
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 7
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 8
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 9
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 10
```

Plotting the results:

```
#Initialize a data frame called "d" to plot the results:
d = list()
def_par = par()
#We have done 10 experiments; we can plot them in a 5 by 2 grid.
par(mfrow=c(5,2))
suppressPackageStartupMessages(library(dplyr))
kk=1
## New margins
par(mar=c(3,4,2,4))
for(kk in 1:nrow(result)){
  d[[kk]] =  data.frame(c(unique(result[kk,])), rep(NA, length(unique(result[kk,]))  ))
  colnames(d[[kk]]) = c("category", "Freq")


  for(i in d[[kk]]$category){
    d[[kk]]$Freq[ which( d[[kk]]$category==i )     ] =  length(which(result[kk,] == i))
  }

  dd = d[[kk]]
  dd <- arrange(dd, desc(Freq)) %>%
    mutate(
      cumsum = cumsum(Freq),
      freq = round(Freq / sum(Freq), 3),
      cum_freq = cumsum(freq)
    )

  ## Saving Parameters



  ## bar plot, pc will hold x values for bars
  pc = barplot(dd$Freq ,
               width = 0.5, space = 0.1, border = NA, axes = F,
               ylim = c(0, 1.05 * max(dd$cumsum, na.rm = T)),
             #  ylab = "Cummulative Counts" , cex.names = 0.7,
               names.arg = dd$category,
               main = paste("Pareto Chart (k = ",kk*10, ")"),
               cex.main = 0.7)
  title(ylab="Rank Count", line=1.8, cex.lab=0.7, family="Calibri Light")

  ## Cumulative counts line
  lines(pc, dd$cumsum, type = "b", cex = 0.7, pch = 19, col="cyan4")

  ## Framing plot
  box(col = "grey62")

  ## adding axes
  axis(side = 2, at = c(0, dd$cumsum), las = 1, col.axis = "grey62", col = "grey62", cex.axis =
0.7)
  axis(side = 4, at = c(0, dd$cumsum), labels = paste(c(0, round(dd$cum_freq * 100)) ,"%",sep=""
),
       las = 1, col.axis = "cyan4", col = "cyan4", cex.axis = 0.7)
  ## restoring default paramenter
  #par(def_par)

}
```

```
## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database

## Warning in title(ylab = "Rank Count", line = 1.8, cex.lab = 0.7, family =
## "Calibri Light"): font family not found in Windows font database
```


Pareto Chart (k = 10)
Rank Count
Pareto Chart (k = 20)
Rank Count
Pareto Chart (k = 30)
Rank Count
Pareto Chart (k = 40)
Rank Count
Pareto Chart (k = 50)
Rank Count
Pareto Chart (k = 60)
Rank Count
Pareto Chart (k = 70)
Rank Count
Pareto Chart (k = 80)
Rank Count
Pareto Chart (k = 90)
Rank Count
Pareto Chart (k = 100)
Rank Count