Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence
MSDS 7333 – Quantifying the World – Case Study 6
June 19th, 2018

Abstract

Indoor Positining Sytems analysis is a fast growing field requires the ability to predict the location of an object within a building. As the growth of IOT continues to rise research on IPS location prediction is becoming more relevant. In this paper, we use K nearest neighbors (kNN) alogritm to predict the location of a signal within a building. We then test another method using weighted distance to see if that method preforms better and finally we compare two different MAC addresses to see if there is impact on results.

Introduction

Indoor positioning system (IPS) analysis is a growing field for analysts and data scientist to take time and understand. Gartner, Inc., a research and industry advisory firm predicts by 2020 that the market for indoor location platforms and services could grow to $12.5B by 2020 [1]. Statistical methods are deployed to track people and objects inside retail stores, healthcare facilities' and warehouses with the use of IPS to understand a variety of behaviors and offer insights to the users of IPS data.

Global Positioning Systems (GPS) do not reliably work inside buildings, but with the growth of Wireless Fidelity (Wi-Fi), signals from Wi-Fi can be used to build indoor positioning systems that offer almost real-time answers to where objects, or people are. One method used to predict where an item of interest is located is K-Nearest Neighbors (K-NN).

The motivation behind this analysis is supplied by a case study in the book "Data Science in R: A Case Study Approach to Computational Reasoning and Problem Solving by Deborah Nolan and Duncan Temple Long. The data is presented to us in raw form where we will have to understand how the data was obtained and formatted so we can create a structure to the data to make in palatable for analysis through a statistical analysis language called R to predict location with K-NN.

Literature Review

Research on IPS like the industry has expanded quickly. One of the key issues raised in research is the importance of the reduction of distance measurement error [2]. One reason for the increased research in IPS is the importance of automation in global supply chains. Goods pass through warehouse owned by customer, manufacturing and shipping company. In the case of some warehouses autonomous systems ferry goods around the warehouse. Navigational capability has two essential components that need to be taken into account. The first is the location of objects, such as obstacles or barriers. The second is information about the building itself [3].

[Paragraph about location prediction lit]

Background

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

The data for this analysis is from the Community Resource for Archiving Wireless Data at Dartmouth (CRAWDAD). The "offline" is a referenced data set that houses signal strengths measured with a hand-held device on a grid of 166 points spaced 1 meter apart located in the hallways of a building at the University of Mannheim in Germany. The floor plan measures 15 by 36 meters. A floor plan is given in Figure 1:

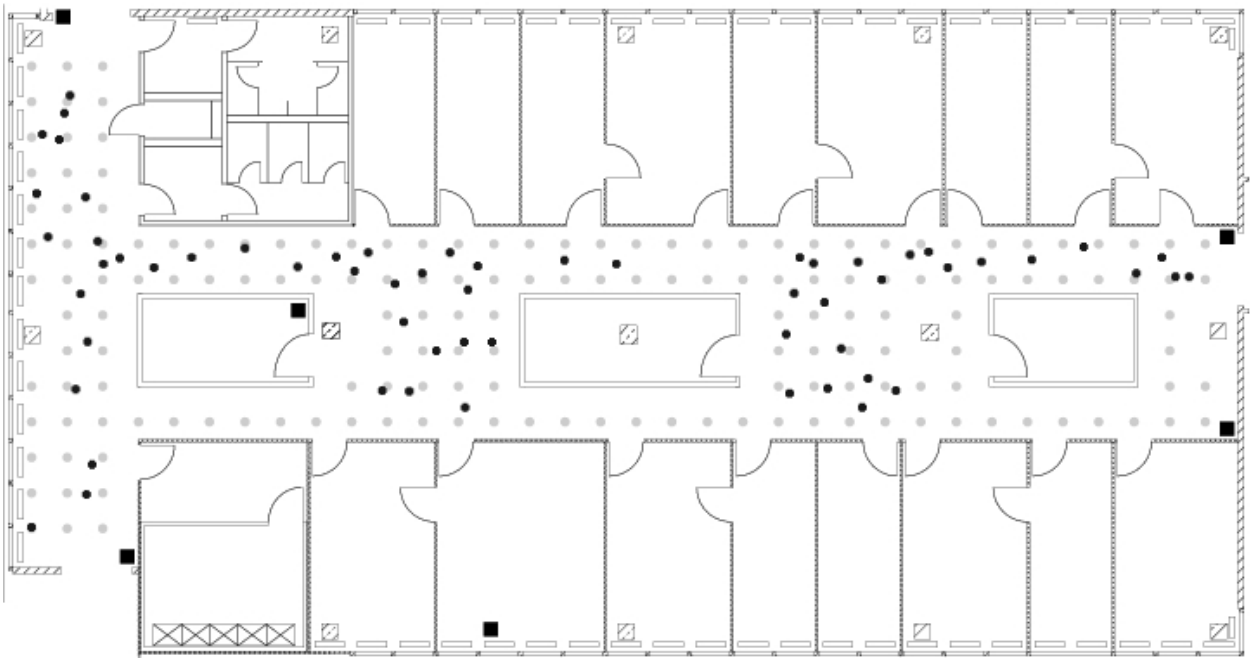


Figure 1: *There are 6 fixed access points which are denoted by black square markers. The "offline" training data were collected at the locations marked with the grey dots. We can see that the grey dots are spaced a meter apart.*

The offline dataset once cleaned contains around 915K observations with nine variables. The observations represent recordings of signal strength which were recorded at eight orientations in 45-degree increments (0, 45, 90, and so on). A full list of variables can be found in Table 1:

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

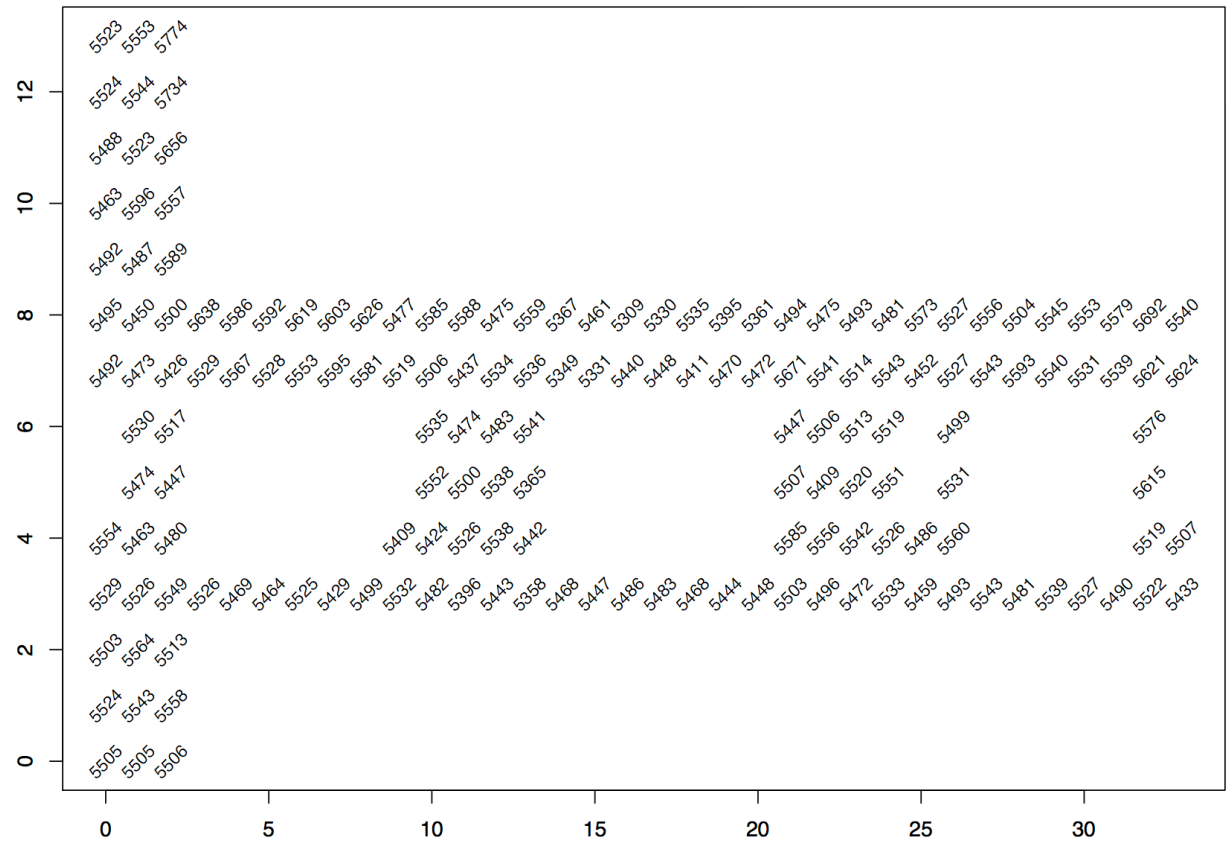
MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

Variable	Description	Data Type
time	Time of Observation	Time/Date
posX	X coordinate inside building	Discrete
posY	Y coordinate inside building	Discrete
orientation	Angle of observation	Continuous
mac	MAC address of device	Continuous
signal	Signal strength	Continuous
rawTime	Number of milliseconds after 1/1/1970	Continuous
angle	45 degree cutoff of orientation	Discrete

Table 1 - Variable Descriptions and Types

Exploratory data analysis (EDA) of this involved some spatial analysis to look at number of recordings at each position and exploring the signal strength within the building. In a perfect world there would be 110 signals measured with eight angles for each of the six access points for a total 5,280 recordings, we come up with about 5,500 recordings at each location in Figure 2.



Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence
MSDS 7333 – Quantifying the World – Case Study 6
June 19th, 2018

Figure. 2. Counts of signals at each position. *Each location in the building is the total number of signals detected from all access points for the offline data. In a perfect world there is 110 signals measured with 8 angles for each 6-access point for a total of 5,280 recordings. These data include a 7th MAC address and not all signals were detected, so there are about 5,500 recordings at each location.*

The distribution of signal strength should tell us that signal strength decays linearly with log distance. A simple triangulation should give us the location of a device [4]. We compared the distribution of signal strength at different orientations and at different access points. Signal strength does vary with the orientation for both close and distant access points. The weakest signals have a smaller standard deviation and it would appear that when signal strength increases so does standard deviation. We can observe this is Figure 3.

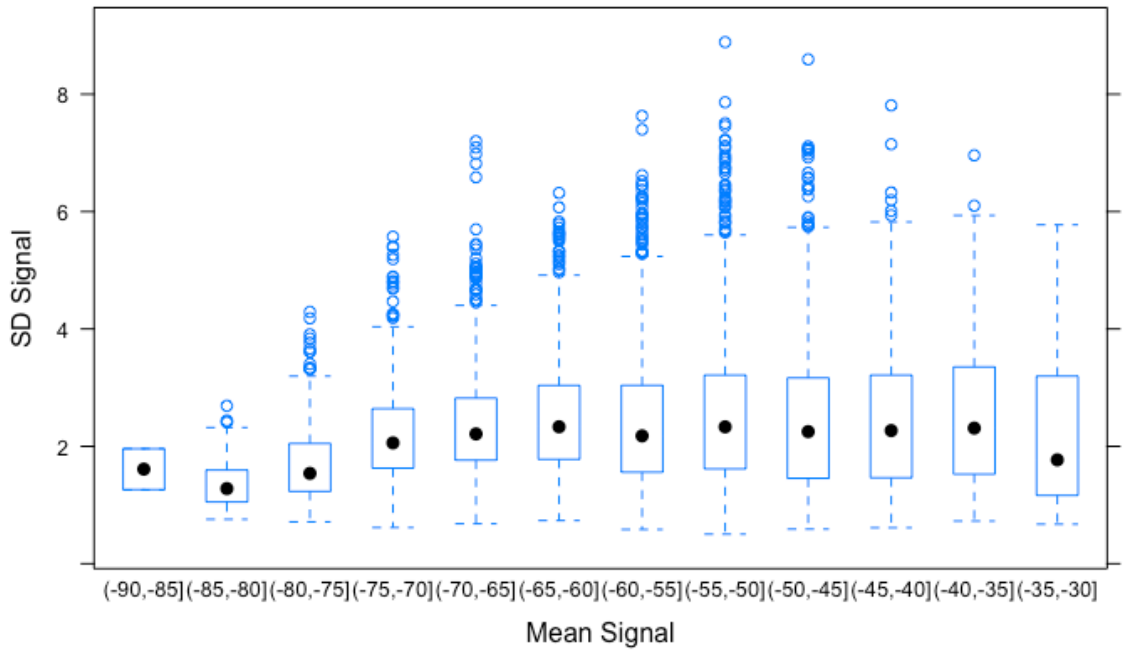


Figure 3. Box Plot of Standard Deviation of Signal and Mean Signal Strength. *As signal strength increases so does variability in signal.*

Methods

There are many statistical methods that can be utilized to estimate the location of a device from the strength of the signal. For this analysis we are going to use a common method called K-Nearest Neighbors, or k-NN to predict which Media Access Control (MAC) address yields the best prediction of location. When test k-NN against a method that uses the weights on the received signal strength to estimate distance. K-NN is an easy to understand method and is known as a lazy learning. This means it does not take much training time. Nearest neighbors can also be useful when data may be unusual. One of the drawbacks of

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence
MSDS 7333 – Quantifying the World – Case Study 6
June 19th, 2018

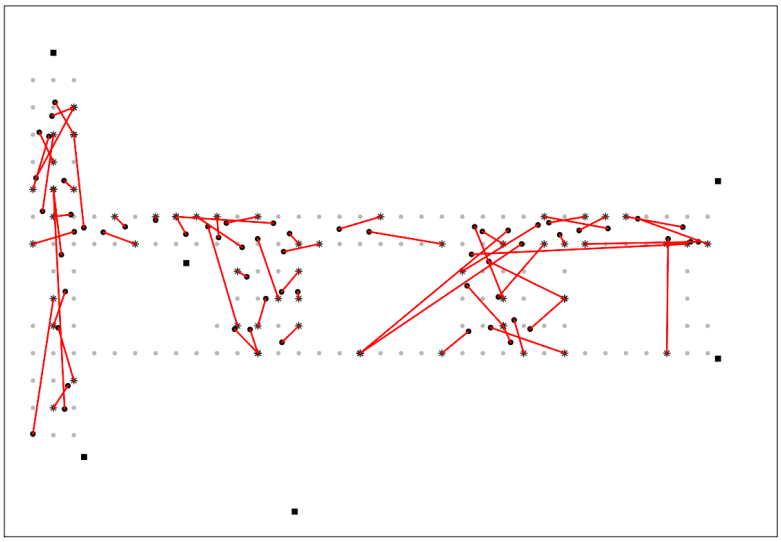
k-NN is that is computationally expensive. Also k-NN results can be decreased when data presents high dimensionality [4].

First, through a function designed by authors Nolan and Lang we experimented with one neighbor and then three. This then worked into an iteration of walking through 20 neighbors. These results were run through cross-validation to prevent over fitting of the model. We choose v-fold cross-validation. V-fold cross-validation divides our training data into v non-overlapping subsets. We then assess the predictive ability of the model using the subset that was left out. This type of cross-validation is also known as leave-one-out cross-validation.

The evaluation of the method was exercised with the measurement of the sum of squared errors as a function of k, or nearest neighbors used for prediction. With k-NN we want to minimize the sum of squared errors.

Results

When k-NN was run with k = 1 neighbors sum of squared errors (SSE) in prediction was 659, and when k = 3 our SSE dropped to 307. Even though we know we reduced the SSE, it's easier to explain with a visualization. In figure 4



Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence
MSDS 7333 – Quantifying the World – Case Study 6
June 19th, 2018

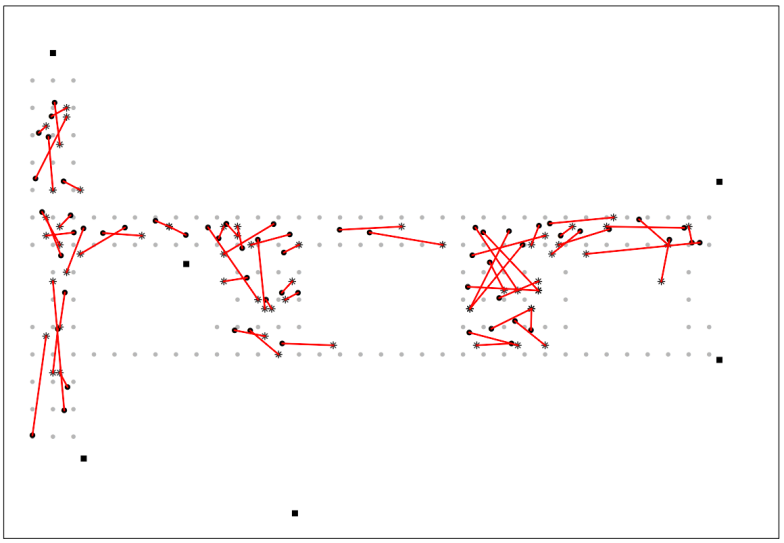


Figure 4: Floor plan with single NN(top) showing Predicted and Actual Locations. *The red line segments connect the test locations in BLACK dots to their predicted locations ASTERISKS. The bottom plot shows K = 3 which decrease our SSE.*

We created a function for K to run from 1 to 20 to see if we could the optimal number of neighbors. With this function we created a visualization in Figure 5 that shows the relationship between SSE and number of neighbors.

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

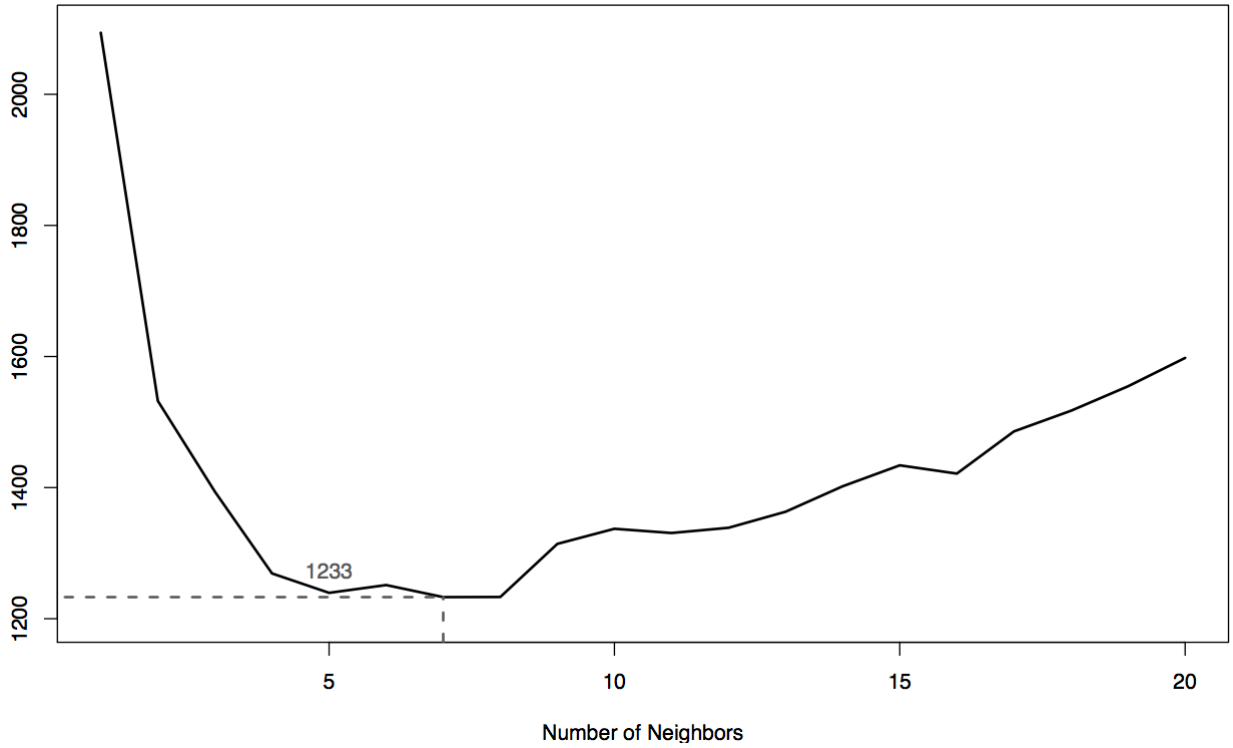


Figure 5: Cross validation selection of k. The line displays the relationship of the sum of squared errors as a function of the number of neighbors used in the prediction of location with a new location.

We decided that our best nearest neighbor number was five. That took our error down to the 276. While this may not be the optimized lowers amount of SSE we could generate, we could risk over fit, if we continue to turn the model.

The cased asked us to evaluate the MAC 00:0f:a3:39:e1:c0 and eliminate 00:0f:a3:39:dd:cd, we did see an increase when MAC MAC 00:0f:a3:39:e1:c0 was used, but deleting 00:0f:a3:39:dd:cd gave a negative impact on prediction accuracy.

Future Work, Discussion Conclusions, and Next Steps

We found that k-NN works pretty well in predicting the location inside of a building that is equipped with Wi-Fi. With k-NN using 5 neighbors we could get within 3 feet of accuracy. Prediction within IPS has some very important uses to many different industries. Interested parties can look for items that may be in the way of flow. A flow could be a group people at an airport, packages moving in a warehouse, or at times we may want to find ways to slow the flow of people moving through high margin areas of retail stores.

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

In companies such as UPS/FedEx and DHL their customers entrust them with biological samples. Some biological samples can be very hard to execute and could be painful to the patient. That is why such care must be given to these samples as they travel to test facilities. Package carriers do use and could use more IOT devices to track the location of those sensitive packages in real time. Inside those warehouses hundreds of people and vehicles, and belts are moving to carry thousands of packages per hour around the building, with humans being in control in many of these warehouses it becomes easy to misplace a package even with the strictest rules and precautions. IPS and IOT allow for the entire journey of that sample to be recorded for patient, medical facility, shipper and regulatory bodies.

References

1. Gartner, Market Guide for Indoor Location Application Platforms
2. Khanzada, Baloch, "Distance Measurement Error Reduction Analysis for the Indoor Positioning System", Mehran University Research Journal of Engineering & Technology (2012)
3. Jeamwatthanachai, Wald & Wills, "Map Data Representation for Indoor Navigation, International Conference on Information Society (2016).
4. Pros and Cons of k-NN, https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/#pros-and-cons-of-knn
5.

Appendix – R Code

```
library(codetools)
library(lattice)
library(fields)

# Use URL to bring in text data
url <- "http://rdatasciencecases.org/Data/offline.final.trace.txt"
# Read in entire document
txt <- readLines(url)
# Each line in the offline file has been read into R as a string in the character
vector txt
# Use the function substr() to locate lines/strings that start with '#' and count how
many we have
sum(substr(txt, 1, 1) == "#")
length(txt)

strsplit(txt[4], ";")[[1]]

unlist(lapply(strsplit(txt[4], ";")[[1]],
  function(x)
    sapply(strsplit(x, "=")[[1]], strsplit, ",")))

# create a spilt using ;,=,','
tokens <- strsplit(txt[4], "[;=,]")[[1]]

tokens[1:10]
# Extract values of variables
tokens[c(2, 4, 6:8, 10)]
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
tmp <- matrix(tokens[ - (1:10 ) ], ncol = 4, byrow = TRUE)
mat <- cbind(matrix(tokens[c(2, 4, 6:8, 10)], nrow = nrow(tmp),
  ncol = 6, byrow = TRUE), tmp)
# Check
dim(mat)

processLine =
function(x)
{
  tokens = strsplit(x, "[;=,]")[[1]]
  tmp = matrix(tokens[ - (1:10) ], ncol = 4, byrow = TRUE)
  cbind(matrix(tokens[c(2, 4, 6:8, 10)], nrow = nrow(tmp),
               ncol = 6, byrow = TRUE), tmp)
}

tmp = lapply(txt[4:20], processLine)
sapply(tmp, nrow)

offline = as.data.frame(do.call("rbind", tmp))
# Check it
dim(offline)

lines <- txt[ substr(txt, 1, 1) != "#" ]
tmp = lapply(lines, processLine)

options(error = recover, warn = 2)
tmp = lapply(lines, processLine)

processLine = function(x)
{
  tokens = strsplit(x, "[;=,]")[[1]]

  if (length(tokens) == 10)
    return(NULL)

  tmp = matrix(tokens[ - (1:10) ], , 4, byrow = TRUE)
  cbind(matrix(tokens[c(2, 4, 6:8, 10)], nrow(tmp), 6,
               byrow = TRUE), tmp)
}

options(error = recover, warn = 1)
tmp <- lapply(lines, processLine)
offline <- as.data.frame(do.call("rbind", tmp),
                         stringsAsFactors = FALSE)

dim(offline)

names(offline) = c("time", "scanMac", "posX", "posY", "posZ",
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
                    "orientation", "mac", "signal",
                    "channel", "type")

numVars = c("time", "posX", "posY", "posZ",
            "orientation", "signal")
offline[ numVars ] =  lapply(offline[ numVars ], as.numeric)

offline = offline[ offline$type == "3", ]
offline = offline[ , "type" != names(offline) ]
dim(offline)

offline$rawTime = offline$time
offline$time = offline$time/1000
class(offline$time) = c("POSIXt", "POSIXct")

summary(offline[, numVars])

summary(sapply(offline[ , c("mac", "channel", "scanMac")], as.factor))

offline <- offline[ , !(names(offline) %in% c("scanMac", "posZ"))]

plot(ecdf(offline$orientation),
  main = "Orientation Distribution",
  xlab = "Orientation",
  ylab = "Empirical CDF")

roundOrientation = function(angles) {
  refs = seq(0, by = 45, length  = 9)
  q = sapply(angles, function(o) which.min(abs(o - refs)))
  c(refs[1:8], 0)[q]
}
# We now use our function to created the rounded angles
offline$angle <- roundOrientation(offline$orientation)

with(offline, boxplot(orientation ~ angle,
                      xlab = "nearest 45 degree angle",
                      ylab = "orientation"))

subMacs <- names(sort(table(offline$mac), decreasing = TRUE))[1:7]
offline <- offline[ offline$mac %in% subMacs, ]

macChannel <- with(offline, table(mac, channel))
apply(macChannel, 1, function(x) sum(x > 0))

offline <- offline[ , "channel" != names(offline)]

locDF <- with(offline,
            by(offline, list(posX, posY), function(x) x))
# Check
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
length(locDF)

sum(sapply(locDF, is.null))

locDF <- locDF[ !sapply(locDF, is.null) ]
# Check
length(locDF)

locCounts <- sapply(locDF, nrow)

# If we want to keep the position information with the lcoation, we do this with
locCounts <- sapply(locDF,
                    function(df)
                      c(df[1, c("posX", "posY")], count = nrow(df)))
# Confirm we have a matrix
class(locCounts)
# Confirm 3 rows
dim(locCounts)
# Examine a few of the counts
locCounts[ , 1:8]

pdf(file = "Geo_XYByCount.pdf", width = 10)
oldPar = par(mar = c(3.1, 3.1, 1, 1))

locCounts = t(locCounts)
plot(locCounts, type = "n", xlab = "", ylab = "")
text(locCounts, labels = locCounts[,3], cex = .8, srt = 45)

par(oldPar)
dev.off()

readData =
  function(filename = "http://rdatasciencecases.org/Data/offline.final.trace.txt",
           subMacs = c("00:0f:a3:39:e1:c0", "00:0f:a3:39:dd:cd", "00:14:bf:b1:97:8a",
                       "00:14:bf:3b:c7:c6", "00:14:bf:b1:97:90", "00:14:bf:b1:97:8d",
                       "00:14:bf:b1:97:81"))
  {
    txt = readLines(filename)
    lines = txt[ substr(txt, 1, 1) != "#" ]
    tmp = lapply(lines, processLine)
    offline = as.data.frame(do.call("rbind", tmp),
                            stringsAsFactors= FALSE)

    names(offline) = c("time", "scanMac",
                       "posX", "posY", "posZ", "orientation",
                       "mac", "signal", "channel", "type")

     # keep only signals from access points
    offline = offline[ offline$type == "3", ]
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
    # drop scanMac, posZ, channel, and type - no info in them
    dropVars = c("scanMac", "posZ", "channel", "type")
    offline = offline[ , !( names(offline) %in% dropVars ) ]

    # drop more unwanted access points
    offline = offline[ offline$mac %in% subMacs, ]

    # convert numeric values
    numVars = c("time", "posX", "posY", "orientation", "signal")
    offline[ numVars ] = lapply(offline[ numVars ], as.numeric)

    # convert time to POSIX
    offline$rawTime = offline$time
    offline$time = offline$time/1000
    class(offline$time) = c("POSIXt", "POSIXct")

    # round orientations to nearest 45
    offline$angle = roundOrientation(offline$orientation)

    return(offline)
  }

offlineRedo = readData()
# Check to see if this fucntion matches what we did.
identical(offline, offlineRedo)

bwplot(signal ~ factor(angle) | mac, data = offline,
        subset = posX == 2 & posY == 12
                  & mac != "00:0f:a3:39:dd:cd",
        layout = c(2,3))

pdf(file = "Geo_DensitySignalByMacAngle.pdf", width = 8, height = 12)
oldPar = par(mar = c(3.1, 3, 1, 1))

densityplot( ~ signal | mac + factor(angle), data = offline,
             subset = posX == 24 & posY == 4 &
                         mac != "00:0f:a3:39:dd:cd",
             bw = 0.5, plot.points = FALSE)

par(oldPar)
dev.off()

offline$posXY <- paste(offline$posX, offline$posY, sep = "-")

byLocAngleAP <- with(offline,
                     by(offline, list(posXY, angle, mac),
                        function(x) x))
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
signalSummary =
  lapply(byLocAngleAP,
         function(oneLoc) {
           ans = oneLoc[1, ]
           ans$medSignal = median(oneLoc$signal)
           ans$avgSignal = mean(oneLoc$signal)
           ans$num = length(oneLoc$signal)
           ans$sdSignal = sd(oneLoc$signal)
           ans$iqrSignal = IQR(oneLoc$signal)
           ans
           })

offlineSummary = do.call("rbind", signalSummary)

breaks = seq(-90, -30, by = 5)
bwplot(sdSignal ~ cut(avgSignal, breaks = breaks),
       data = offlineSummary,
       subset = mac != "00:0f:a3:39:dd:cd",
       xlab = "Mean Signal", ylab = "SD Signal")

with(offlineSummary,
     smoothScatter((avgSignal - medSignal) ~ num,
                   xlab = "Number of Observations",
                   ylab = "mean - median"))
abline(h = 0, col = "#984ea3", lwd = 2)

lo.obj =
  with(offlineSummary,
       loess(diff ~ num,
             data = data.frame(diff = (avgSignal - medSignal),
                               num = num)))

lo.obj.pr = predict(lo.obj, newdata = data.frame(num = (70:120)))
lines(x = 70:120, y = lo.obj.pr, col = "#4daf4a", lwd = 2)

oneAPAngle <- subset(offline, mac == subMacs[5] & angle == 0)

oneAPAngle = subset(offlineSummary,
                    mac == subMacs[5] & angle == 0)

smoothSS = Tps(oneAPAngle[, c("posX","posY")],
               oneAPAngle$avgSignal)

vizSmooth = predictSurface(smoothSS)

plot.surface(vizSmooth, type = "C")

points(oneAPAngle$posX, oneAPAngle$posY, pch=19, cex = 0.5)
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
surfaceSS = function(data, mac, angle = 45) {
  require(fields)
  oneAPAngle = data[ data$mac == mac & data$angle == angle, ]
  smoothSS = Tps(oneAPAngle[, c("posX","posY")],
                 oneAPAngle$avgSignal)
  vizSmooth = predictSurface(smoothSS)
  plot.surface(vizSmooth, type = "C",
               xlab = "", ylab = "", xaxt = "n", yaxt = "n")
  points(oneAPAngle$posX, oneAPAngle$posY, pch=19, cex = 0.5)
}

parCur = par(mfrow = c(2,2), mar = rep(1, 4))

mapply(surfaceSS, mac = subMacs[ rep(c(5, 1), each = 2) ],
       angle = rep(c(0, 135), 2),
       data = list(data = offlineSummary))

par(parCur)

offlineSummary <- subset(offlineSummary, mac != subMacs[2])

AP <- matrix( c( 7.5, 6.3, 2.5, -.8, 12.8, -2.8,
                 1, 14, 33.5, 9.3,  33.5, 2.8),
            ncol = 2, byrow = TRUE,
            dimnames = list(subMacs[ -2 ], c("x", "y") ))

AP

diffs <- offlineSummary[ , c("posX", "posY")] -
            AP[ offlineSummary$mac, ]

offlineSummary$dist <- sqrt(diffs[ , 1]^2 + diffs[ , 2]^2)

xyplot(signal ~ dist | factor(mac) + factor(angle),
       data = offlineSummary, pch = 19, cex = 0.3,
       xlab ="distance")

pdf(file="Geo_ScatterSignalDist.pdf", width = 7, height = 10)
oldPar = par(mar = c(3.1, 3.1, 1, 1))

xyplot(signal ~ dist | factor(mac) + factor(angle),
       data = offlineSummary, pch = 19, cex = 0.3,
       xlab ="distance")
par(oldPar)
dev.off()

macs <- unique(offlineSummary$mac)
online <- readData("http://rdatasciencecases.org/Data/online.final.trace.txt",
subMacs = macs)
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
online$posXY <- paste(online$posX, online$posY, online$posY, sep = "-")
# Check
length(unique(online$posXY))

tabonlineXYA <- table(online$posXY, online$angle)
tabonlineXYA[1:6, ]

keepVars = c("posXY", "posX","posY", "orientation", "angle")
byLoc = with(online,
             by(online, list(posXY),
                function(x) {
                  ans = x[1, keepVars]
                  avgSS = tapply(x$signal, x$mac, mean)
                  y = matrix(avgSS, nrow = 1, ncol = 6,
                        dimnames = list(ans$posXY, names(avgSS)))
                  cbind(ans, y)
                }))

onlineSummary = do.call("rbind", byLoc)

dim(onlineSummary)

names(onlineSummary)

m <- 3; angleNewObs <- 230
refs <- seq(0, by = 45, length = 8)
nearestAngle <- roundOrientation(angleNewObs)

if (m %% 2 == 1) {
  angles = seq(-45 * (m-1) /2, 45 * (m-1) /2, length = m)
} else {
  m = m+1
  angles = seq(-45 * (m-1) /2, 45 * (m-1) /2, length = m)
  if (sign(angleNewObs - nearestAngle) > -1)
    angles = angles[ -1 ]
  else
    angles = angles[ -m ]
}

## m odd and even are handled seperatly. Map angles to refs
## Adjustments
angles <- angles + nearestAngle
angles[angles < 0] = angles[ angles < 0 ] + 360
angles[angles > 360] = angles[ angles > 360 ] – 360

offlineSubset <-
  offlineSummary[ offlineSummary$angle %in% angles, ]
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
reshapeSS <- function(data, varSignal = "signal",
                      keepVars = c("posXY", "posX","posY")) {
  byLocation <-
    with(data, by(data, list(posXY),
                  function(x) {
                    ans = x[1, keepVars]
                    avgSS = tapply(x[ , varSignal ], x$mac, mean)
                    y = matrix(avgSS, nrow = 1, ncol = 6,
                               dimnames = list(ans$posXY,
                                               names(avgSS)))
                    cbind(ans, y)
                  }))

  newDataSS <- do.call("rbind", byLocation)
  return(newDataSS)
}

Summarize and reshape offlineSubset
```{r}
trainSS <- reshapeSS(offlineSubset, varSignal = "avgSignal")
```

Reshape function with the same logic as above
```{r tidy=TRUE}
trainSS = reshapeSS(offlineSubset, varSignal = "avgSignal")

selectTrain = function(angleNewObs, signals = NULL, m = 1){
  # m is the number of angles to keep between 1 and 5
  refs = seq(0, by = 45, length  = 8)
  nearestAngle = roundOrientation(angleNewObs)

  if (m %% 2 == 1)
    angles = seq(-45 * (m - 1) /2, 45 * (m - 1) /2, length = m)
  else {
    m = m + 1
    angles = seq(-45 * (m - 1) /2, 45 * (m - 1) /2, length = m)
    if (sign(angleNewObs - nearestAngle) > -1)
      angles = angles[ -1 ]
    else
      angles = angles[ -m ]
  }
  angles = angles + nearestAngle
  angles[angles < 0] = angles[ angles < 0 ] + 360
  angles[angles > 360] = angles[ angles > 360 ] - 360
  angles = sort(angles)

    offlineSubset = signals[ signals$angle %in% angles, ]
  reshapeSS(offlineSubset, varSignal = "avgSignal")
}
```
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```

```
train130 <- selectTrain(130, offlineSummary, m = 3)
head(train130)
# Should be 166
length(train130[[1]])

findNN <- function(newSignal, trainSubset) {
  diffs = apply(trainSubset[, 4:9], 1,
    function(x) x - newSignal)
  dists = apply(diffs, 2, function(x) sqrt(sum(x^2)))
  closest = order(dists)
  return(trainSubset[closest, 1:3])
}

predXY <- function(newSignals, newAngles, trainData,
  numAngles = 1, k = 3){

  closeXY = list(length = nrow(newSignals))

  for (i in 1:nrow(newSignals)) {
    trainSS = selectTrain(newAngles[i], trainData, m = numAngles)
    closeXY[[i]] =
      findNN(newSignal = as.numeric(newSignals[i,]), trainSS)
  }

  estXY <- lapply(closeXY,
    function(x) sapply(x[, 2:3],
      function(x) mean(x[1:k])))
  estXY = do.call("rbind", estXY)
  return(estXY)
}

findNN <- function(newSignal, trainSubset) {
  diffs = apply(trainSubset[, 4:9], 1,
    function(x) x - newSignal)
  dists = apply(diffs, 2, function(x) sqrt(sum(x^2)))
  closest = order(dists)
  return(trainSubset[closest, 1:3])
}

predXY <- function(newSignals, newAngles, trainData,
  numAngles = 1, k = 3){

  closeXY = list(length = nrow(newSignals))

  for (i in 1:nrow(newSignals)) {
    trainSS = selectTrain(newAngles[i], trainData, m = numAngles)
    closeXY[[i]] =
```
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
      findNN(newSignal = as.numeric(newSignals[i, ]), trainSS)
  }

  estXY <- lapply(closeXY,
    function(x) sapply(x[ , 2:3],
      function(x) mean(x[1:k])))
  estXY = do.call("rbind", estXY)
  return(estXY)
}

estXYk3 <- predXY(newSignals = onlineSummary[ , 6:11],
                  newAngles = onlineSummary[ , 4],
                  offlineSummary, numAngles = 3, k = 3)

estXYk1 <- predXY(newSignals = onlineSummary[ , 6:11],
                  newAngles = onlineSummary[ , 4],
                  offlineSummary, numAngles = 3, k = 1)

calcError <- function(estXY, actualXY)
               sum( rowSums( (estXY - actualXY)^2) )

## Apply the functions to our two sets
actualXY <- onlineSummary[ , c("posX", "posY")]
sapply(list(estXYk1, estXYk3), calcError, actualXY)

floorErrorMap <- function(estXY, actualXY, trainPoints = NULL, AP = NULL){

  plot(0, 0, xlim = c(0, 35), ylim = c(-3, 15), type = "n",
    xlab = "", ylab = "", axes = FALSE)
  box()
  if ( !is.null(AP) ) points(AP, pch = 15)
  if ( !is.null(trainPoints) )
    points(trainPoints, pch = 19, col="grey", cex = 0.6)

  points(x = actualXY[, 1], y = actualXY[, 2],
    pch = 19, cex = 0.8 )
  points(x = estXY[, 1], y = estXY[, 2],
    pch = 8, cex = 0.8 )
  segments(x0 = estXY[, 1], y0 = estXY[, 2],
    x1 = actualXY[, 1], y1 = actualXY[ , 2],
    lwd = 2, col = "red")
}

trainPoints <- offlineSummary[ offlineSummary$angle == 0 &
    offlineSummary$mac == "00:0f:a3:39:e1:c0" ,
  c("posX", "posY")]

pdf(file="GEO_FloorPlanK3Errors.pdf", width = 10, height = 7)
oldPar = par(mar = c(1, 1, 1, 1))
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
floorErrorMap(estXYk3, onlineSummary[ , c("posX","posY")],
  trainPoints = trainPoints, AP = AP)
par(oldPar)
dev.off()

pdf(file="GEO_FloorPlanK1Errors.pdf", width = 10, height = 7)
oldPar = par(mar = c(1, 1, 1, 1))
floorErrorMap(estXYk1, onlineSummary[ , c("posX","posY")],
  trainPoints = trainPoints, AP = AP)
par(oldPar)
dev.off()

v <- 11
permuteLocs <- sample(unique(offlineSummary$posXY))
permuteLocs <- matrix(permuteLocs, ncol = v,
                      nrow = floor(length(permuteLocs)/v))
## This will generate a warning message with the call to matrix() because
## v does not divide evenly into 166, so permuteLocs does not contain all
## 166 locations. Each subset of 1s5 locations is used as the "online" data.
onlineFold <- subset(offlineSubset, posXY %in% permuteLocs[ , 1])

reshapeSS <- function(data, varSignal = "signal",
  keepVars = c("posXY", "posX","posY"),
  sampleAngle = FALSE,
  refs = seq(0, 315, by = 45)) {
  byLocation =
    with(data, by(data, list(posXY),
      function(x) {
        if (sampleAngle) {
          x = x[x$angle == sample(refs, size = 1), ]}
        ans = x[1, keepVars]
        avgSS = tapply(x[ , varSignal ], x$mac, mean)
        y = matrix(avgSS, nrow = 1, ncol = 6,
          dimnames = list(ans$posXY,
            names(avgSS)))
        cbind(ans, y)
      }))

  newDataSS = do.call("rbind", byLocation)
  return(newDataSS)
}

offline <- offline[ offline$mac != "00:0f:a3:39:dd:cd", ]

keepVars <- c("posXY", "posX","posY", "orientation", "angle")

onlineCVSummary <- reshapeSS(offline, keepVars = keepVars,
  sampleAngle = TRUE)
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
# This chunk of code takes a few minutes
onlineFold <- subset(onlineCVSummary,
  posXY %in% permuteLocs[ , 1])

offlineFold <- subset(offlineSummary,
  posXY %in% permuteLocs[ , -1])

estFold <- predXY(newSignals = onlineFold[ , 6:11],
  newAngles = onlineFold[ , 4],
  offlineFold, numAngles = 3, k = 3)

actualFold <- onlineFold[ , c("posX", "posY")]
calcError(estFold, actualFold)

K <- 20
err <- rep(0, K)

for (j in 1:v) {
  onlineFold = subset(onlineCVSummary,
    posXY %in% permuteLocs[ , j])
  offlineFold = subset(offlineSummary,
    posXY %in% permuteLocs[ , -j])
  actualFold = onlineFold[ , c("posX", "posY")]

  for (k in 1:K) {
    estFold = predXY(newSignals = onlineFold[ , 6:11],
      newAngles = onlineFold[ , 4],
      offlineFold, numAngles = 3, k = k)
    err[k] = err[k] + calcError(estFold, actualFold)
  }
}

pdf(file = "Geo_CVChoiceOfK.pdf", width = 10, height = 6)
oldPar = par(mar = c(4, 3, 1, 1))
plot(y = err, x = (1:K),  type = "l", lwd= 2,
  ylim = c(1200, 2100),
  xlab = "Number of Neighbors",
  ylab = "Sum of Square Errors")

rmseMin <- min(err)
kMin <- which(err == rmseMin)[1]
segments(x0 = 0, x1 = kMin, y0 = rmseMin, col = gray(0.4),
  lty = 2, lwd = 2)
segments(x0 = kMin, x1 = kMin, y0 = 1100,  y1 = rmseMin,
  col = grey(0.4), lty = 2, lwd = 2)

#mtext(kMin, side = 1, line = 1, at = kMin, col = grey(0.4))
text(x = kMin - 2, y = rmseMin + 40,
  label = as.character(round(rmseMin)), col = grey(0.4))
```

Predicting Location via Indoor Positioning Systems

Michael Crowder, Gerardo Garza, Brian Kolovich and Brandon Lawrence

MSDS 7333 – Quantifying the World – Case Study 6

June 19th, 2018

```
par(oldPar)
dev.off()

estXYk5 <- predXY(newSignals = onlineSummary[ , 6:11],
                  newAngles = onlineSummary[ , 4],
                  offlineSummary, numAngles = 3, k = 5)
# Add up errors in prediction
calcError(estXYk5, actualXY)
```