Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
MSDS 7333 - Quantifying the World - Case Study # 2
01/22/2019

Abstract

The purpose of this study was to build a model to determine the location of a device as a function of the strength of the signals between the device and an access point. The model to predict the location of a new unknown device is based on the signal strength for the device within a building at the University of Mannheim. The design of the study examines nearly one million measurements of signal strength recorded from six stationary Wi-Fi access points. We used a nearest neighbor method to predict the location and test our model on a new set of data.

Introduction

With the advent of modern technology, nearly everyone has been exposed to some form of Real Time Location Services (RTLS). An everyday example of this technology is global positioning system (GPS) devices which can track our movements, providing the user with real time location determination. While GPS systems work well where there is a direct line-of-sight to a GPS satellite, there are areas where GPS technology is not a viable alternative; indoors, crowded city with large building, underground, etc. In these types of places, alternative methods of RTLS must be utilized.

The ubiquity of wireless local area networks (LANs) and handled mobile devices has given us a tremendous advantage, producing meaningful data points, which can be utilized as part of an Indoor Positioning System (IPS). IPS' have the ability ascertain the location of people and equipment within an indoor environment, utilizing a variety of methods and techniques (signal strength, signal direction/angle, etc.). We analyze the data to determine the location of people and equipment, as well as determine potential issues with the equipment used to ascertain these data points.

The authors of *Predicting Location via Indoor Positioning Systems (PLIPS)*, Nolan and Lang (NTL) [1], explore a data set of an IPS and detail a study that takes data in by way of scanning devices and the data is represented in milliseconds since midnight, January 1, 1970 UTC. The data is analyzed using the k-nearest neighbors' method to determine potential issues regarding the use and non-use of the data.

Building an IPS usually requires a data set where we can measure known locations in relation to some form of received information. Signal strengths from these know locations are read by the fixed access points. The data set used is this study is from the University of Mannheim where nearly one million observations (measurements of signal strength) were recorded access points. The goal is to utilize the dataset collected and predict the location of hand-held devices. This case study will conduct a more thorough analysis into this RTLS application dataset, to determine if NTL's RTLS model can be improved. Three additional areas will be investigated.

Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
MSDS 7333 - Quantifying the World - Case Study # 2
01/22/2019

A) Inclusion of previously removed observations
B) Use of weighted distance for location predictions
C) Optimizing of number included signal-angles included in the analysis

Predictive models were created, utilizing the same k-nearest neighbor (KNN) predictive analytics model.   The model results are discussed in this case study.

Background

In our modern economy we have been able to take advantage of the growth of wireless networking and its ability to provide information to analyze indoor positioning systems (IPS). Our analysis focuses on determining the unknown location of a hand-held device by measuring its signal strength. We will be utilizing the University of Mannheim data set and previous analysis performed by NTL.

To fabricate an indoor situating framework requires a reference set of information where the signal quality between a hand-held device, for example, a phone or PC and permanent access switches are estimated at known areas within the building. With this preparation information, we can construct a model for the area of a device as a component of the quality of the signs between the devices and each passageway.

To properly utilize the NTL's PLIPS dataset, the first step is to determine how the data were collected and formatted. Afterwards, we clean the data and prepare it for analysis in a manner that we can accurately begin to model our data so that we may best examine signal strength.

The data is composed of two different data sets, 'Offline' and 'Online'. The offline data set is our baseline dataset, which is composed of signal strengths measured using a hand-held device on a grid of 166 points spaced 1 meter apart in the hallway of a one floors of the University of Mannheim's buildings, see Fig. 1.

Utilizing the 'offline' dataset, we can manufacture a model for the area utilizing the strengths of the signals received by each access points corresponding relative to a known location (Grey Dots).   Since variations in signal strength will be evident if the hand-held device is pointed in different directions, readings are also taken at each reference location with the hand-held device positioned at different angles. This is repeated at each reference location on the grid.
A model is developed to map signal strength vs location and angle.

Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud

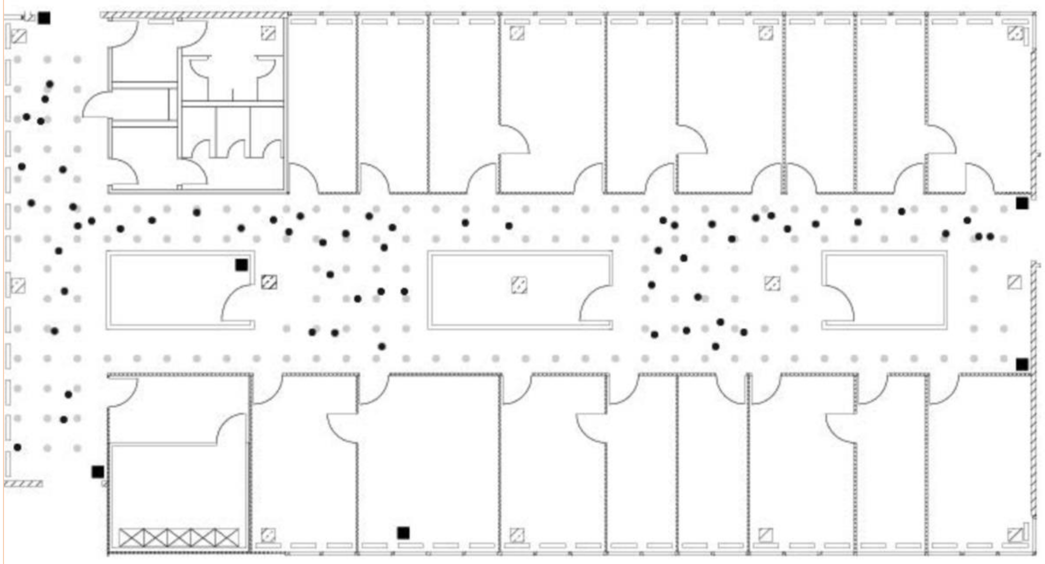
MSDS 7333 - Quantifying the World - Case Study # 2
01/22/2019

Methods

Data Formatting

The data is composed of two different data sets, offline and online. The offline data set is composed of signal strengths measured using a hand-held device on a grid of 166 points spaced 1 meter apart in the hallway of a one floor building. For the online data, 60 locations and orientations are chosen at random and 110 signals are measured from an access point. Both the offline [3] and online dataset [3] are in text files and have similar formatting. The first step to parse the data into a usable format for our analysis. Figure 2 shows the structure of the raw dataset. The first 4 rows are displayed.

Fig 1. Floor Plan of the Test Environment [1]



Legend: Black Square Markers: Fixed Access points: Black Square Markers
Grey Dots: Offline/training signal reference data locations
(Note: Dots are spaced one meter apart)

Fig 2. Data Structure (Raw)

```
[1] "# timestamp=2006-02-11 08:31:58"
[2] "# usec=250"
[3] "# minReadings=110"
[4] "t=1139643118358;id=00:02:2D:21:0F:33;pos=0.0,0.0,0.0;degree=0.0;00:14:bf:b1:97:8a=-
38,2437000000,3;00:14:bf:b1:97:90=-56,2427000000,3;00:0f:a3:39:e1:c0=-
53,2462000000,3;00:14:bf:b1:97:8d=-65,2442000000,3;00:14:bf:b1:97:81=-
65,2422000000,3;00:14:bf:3b:c7:c6=-66,2432000000,3;00:0f:a3:39:dd:cd=-
75,2412000000,3;00:0f:a3:39:e0:4b=-78,2462000000,3;00:0f:a3:39:e2:10=-
87,2437000000,3;02:64:fb:68:52:e6=-88,2447000000,1;02:00:42:55:31:00=-84,2457000000,1"
```

Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
MSDS 7333 - Quantifying the World - Case Study # 2
01/22/2019

Each data entry contains a time stamp, access point MacID, Position information X,Y,Z, Angle of hand-held device x,y,z, Signal Strength, Channel, and Type. Several comment rows were also observed in the dataset (lines 1,2,3 Fig. 2).

We parse the data and build a matrix, so that all unnecessary/redundant information is removed, i.e. comments, separating each observation into individual entries by MAC. At this point, we have 1.18 Million observations with the following variables (see Table 1).

Table 1: Data Fields

time	scanMac	posX	posY	posZ	orientation	mac	signal	channel	type
num	char	num	num	num	num	char	num	char	char

We then further formatted the variables, giving them proper names and formats. We also removed unnecessary data points (posZ) and signals that are not from the access points (type=1). At this point, we have approximately 978K observations and 8 variables.

We will utilize only the following five variables: time, id, position, orientation, Mac ID, and signal strength (see Table 2).

Table 2: Variables Utilized

Variable	Description
time	Timestamp in milliseconds since midnight, January 1, 1970 UTC
posX,posY	The physical coordinate of the scanning device
orientation	Orientation of the user carrying the scanning device in degrees
MAC	MAC address of a responding peer with the corresponding values for signal strength in dBm (Decibel-milliwatts), the channel frequency and its mode (access point = 3, device in adhoc mode = 1)
Signal	Signal Strength (dBm)

Evaluating Degrees

Reviewing the parsed dataset, we observed 203 values for orientation verses the 8 values that were listed in the dataset documentation. The original documentation stated that the signals were intended to be at 8 different orientations at 45° apart (0°,45°,90°…,315°). Figure 3a & 3b shows the orientation values are distributed in clusters around the anticipated 8 orientations. To simplify the analysis, the signal orientations were rounded to the nearest multiple of 45°, to reduce the signal orientations to the anticipated 8 orientations.

Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud

MSDS 7333 - Quantifying the World - Case Study # 2

01/22/2019

MAC Addresses

We want to ascertain whether there is a one-to-one relationship mapping between the MAC address of the access points and channel. In order to draw this conclusion, we must look at the relationship between the MAC address and channel. From floor plan of the test environment (see Figure 1), we are under the impression that there are 6 access points. Our dataset includes 12 MAC addresses and 8 channels. According to the documentation, there

Figure 3a: Signal Density Plot

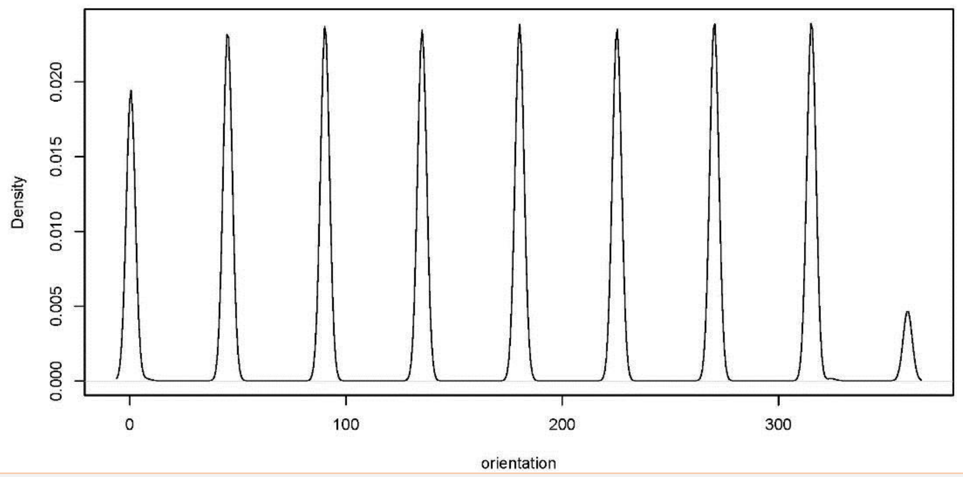
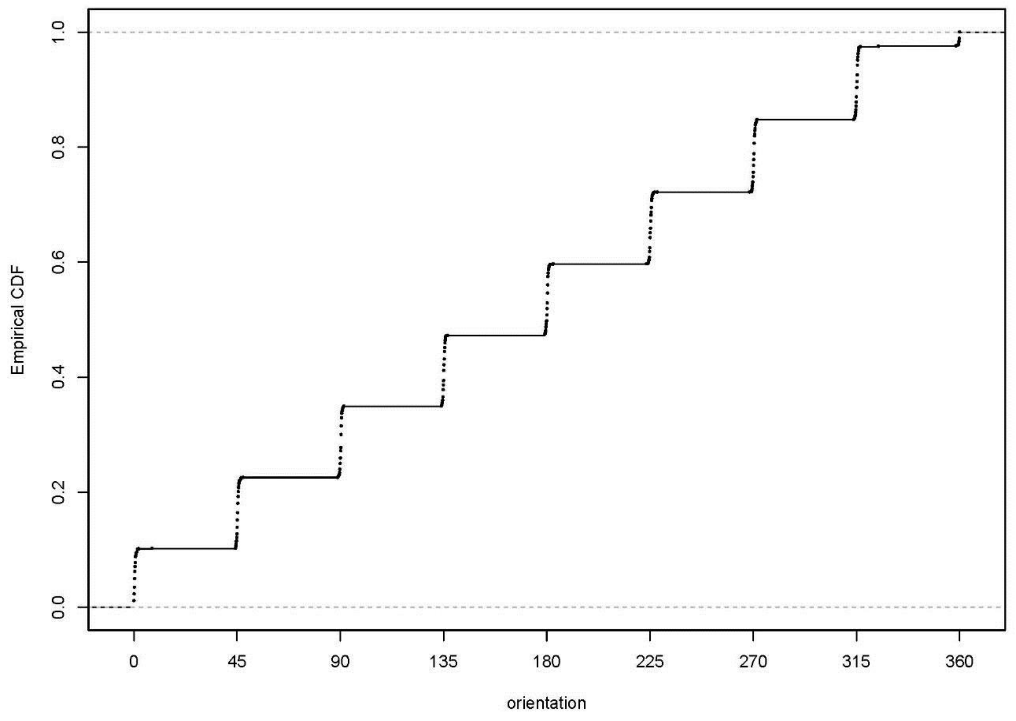


Figure 3b: Cumulative Signal Density Plot (203 orientations)

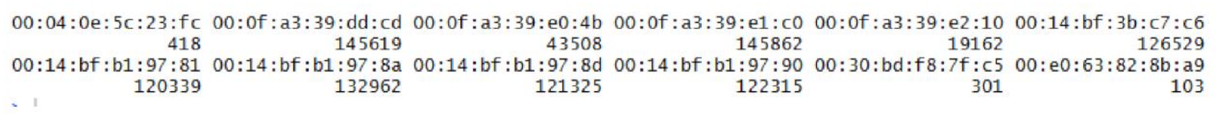


are additional access points that are not seen on the floor plan because they are not part of the testing data. After reviewing the dataset, it was determined that 5 MAC address had minimal number of signals, so they were eliminated (Table 3). Mac ID's eliminated: 00:04:0e:5c:23:fc, 00:0f:a3:39: e0:4b, 00:0f:a3:39: e2:10, 00:30:bd:f8:7f:c5, and 00:e0:63:82:8b:a9. This will leave us with 7 MAC IDs, one more than anticipated. We will expound on this issue in the Results Section.

Position (posX, posY)

From Figure 1, there are a total of 476 possible unique locations for a data location (34) in the X and (14) in the Y direction. Since not all locations were observed, we will query the dataset to determine the number of non-null data locations. There were 166 locations that had observations, which corresponds to the dataset literature.

Table 3: Cumulative number Signals for Each MAC ID



```
00:04:0e:5c:23:fc 00:0f:a3:39:dd:cd 00:0f:a3:39:e0:4b 00:0f:a3:39:e1:c0 00:0f:a3:39:e2:10 00:14:bf:3b:c7:c6
              418            145619             43508            145862             19162            126529
00:14:bf:b1:97:81 00:14:bf:b1:97:8a 00:14:bf:b1:97:8d 00:14:bf:b1:97:90 00:30:bd:f8:7f:c5 00:e0:63:82:8b:a9
           120339            132962            121325            122315               301               103
```

Signal Strength Evaluation

The premise of the analysis is notion that the location of a handheld device can be determined by the strength of the signal a receiver receives from that device.
We want to understand the properties and values of signal strength and how it will help us in our analysis and interpretation. One of the first things that we need to determine if how signal strength is distributed in from different access points. We want to determine if signal strength behaves differently in different locations and how the location, orientation, and access point affect this distribution.

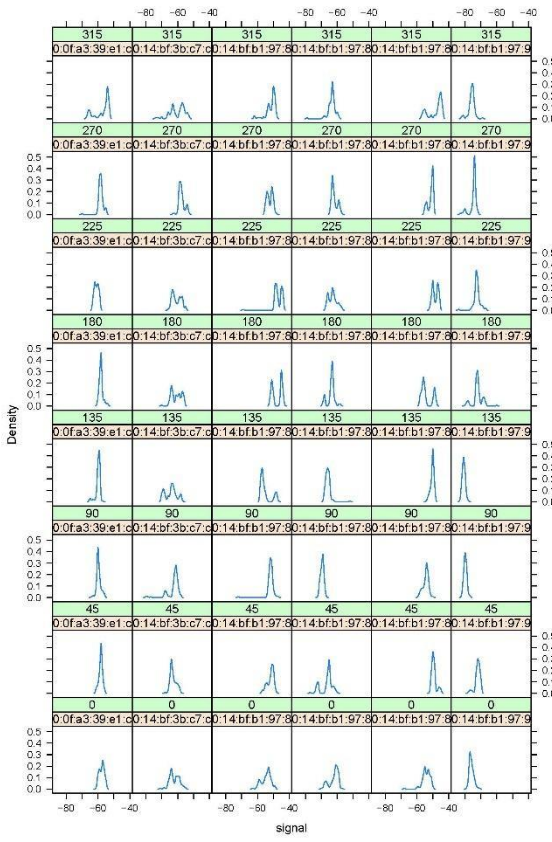
The density curves in the chart below (Fig. 4) represent the density and signal strength recorded at each MAC id at each angle. We use this chart to understand if our data is normally distributed or skewed, according to the central limit theorem.

Signal strength degradation plays an instrumental part in the RTLS. Figure 5 is a signal strength heat map with contour lines for 2 access points, depicting the signal strength received by an access point from each of the 166 observation locations. Just by looking at the above heat maps, it is easy to determine the location of the 2 access points in question.

Results

Although we have several tools in our arsenal to predict location, k-NN (k-nearest neighbors) stands out as being the simplest and perhaps most effective method. In this project we employ k-NN as follows: using our training data, we can determine access points from known locations throughout the building. Each of these points emits a signal, which is measured. When we get a new set of signal strengths from another location, we find the observation in our training data that is closest. This collection allows us to predict the position of the new observation as the position of the closest training observation.

Figure 4: Signal Density Plot

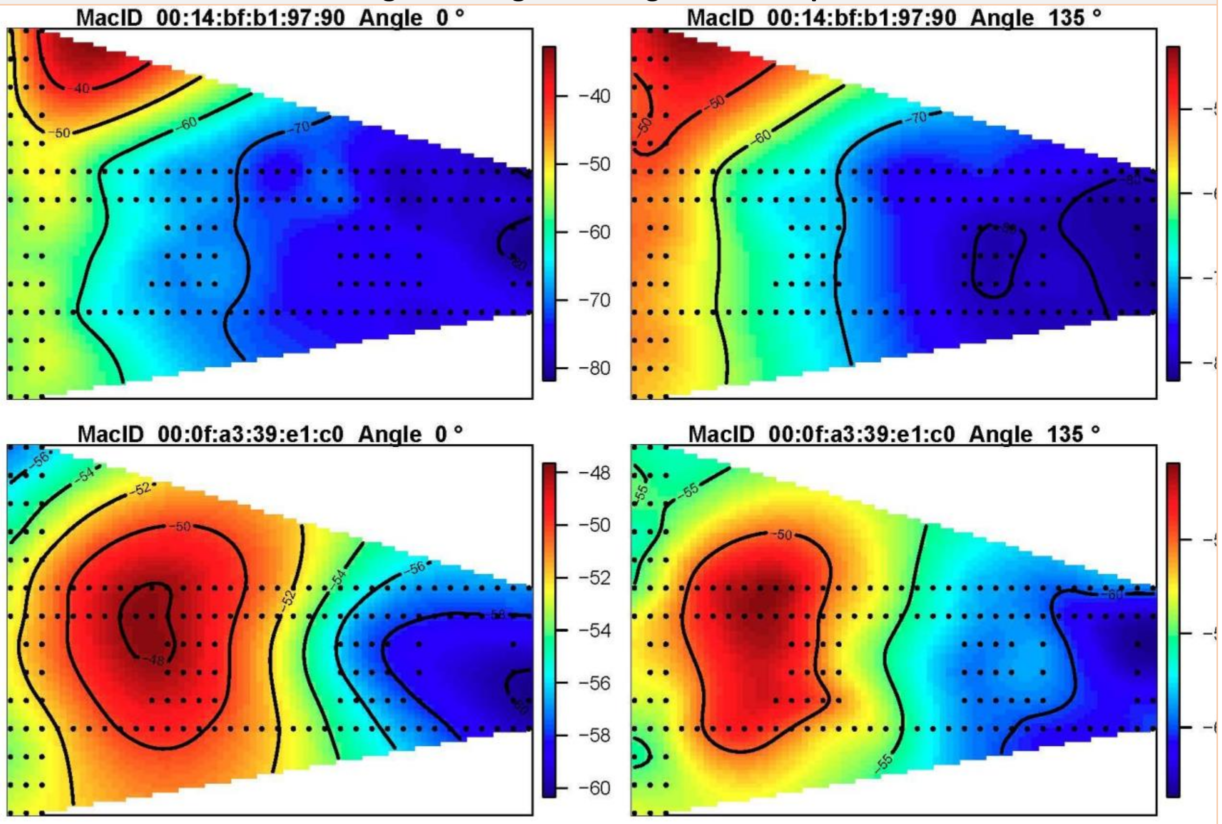


From the dataset literature, the case study should only have 6 LAN Wi-Fi MAC addresses. The dataset contains 7 MAC addresses. Looking at the MAC addresses information, we can determine that 5 of the access points are Linksys routers (00:14:bf..). The two remaining access point MAC addresses Alpha Networks (00:0f:a3..). From this information, we cannot determine which one of the non-Linksys routers should be considered one of the 6 original routers.   Utilizing a heat map of the 2 MAC IDs in question, to attempt to preliminarily determine where they were located, it appears that both have the same relative location. Because of this, In NTL's RTLS model and analysis, NTL arbitrarily chose to eliminate MAC ID 00:0f:a3:39:dd:cd.

KNN Analysis

This section will expand NTL's KNN analysis, to determine which MAC address should be used in the analysis, determine if estimated 'Online' observations location are improved using KNN weighted or unweighted signal strengths. We will also analyze if different numbers of 'included signal angles' improves accuracy.

Figure 5: Signal Strength Heat Map



NTL's KNN eliminated MAC ID 00:0f:a3:39:dd:cd from is training 'offline' dataset. NTL utilize 11-fold cross validation, using 3 included angles, and averaging the NN distances to determine the target's location to determine the optimal k value for the nearest neighbor analysis.

Table 4 and Figures 6a & 6b shows the parameters and results of NTL's analysis. NTL determined that 5 nearest neighbors was optimal for analysis. Note: Actual Min. SSE will vary depending on the actual folds that were picked and will not match up exactly with text

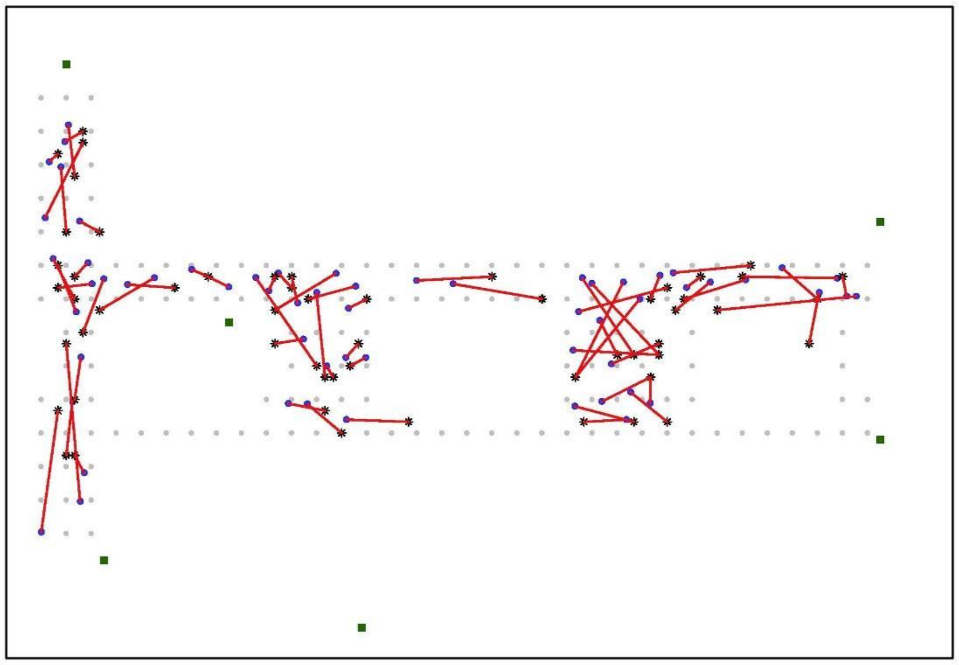
Table 4: KNN Parameters

Parameter Scenarios	SubMac ID Removed	KNN Distance Weighted	# of Signal Angles
SM2_WF_SA3	2	FALSE	3

SubMac ID Removed: 2 = 00:0f:a3:39:dd:cd

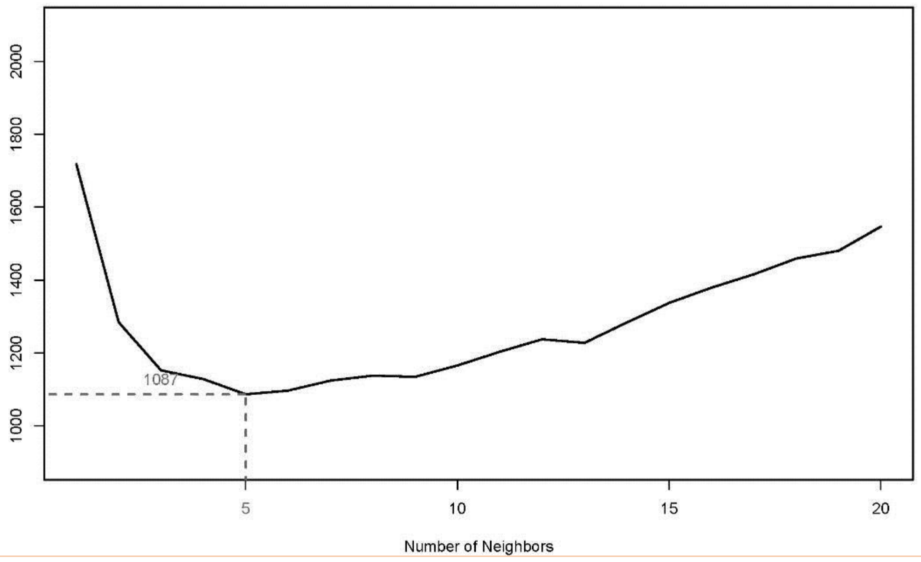
Results		
Optimal # KNN	Min. SSE	SSE*
5	1087	275.5083

Figure 6A: Floor Plan Predicted vs Actual Locations (NTL's Model)



Legend: Green Square Markers: Fixed Access points
Blue Round Markers: Actual position of online observation
Grey Asterisks: Predicted position of online observation
Grey Dots: Offline/training signal reference data locations

Fig 6b: Optimal Number of Nearest Neighbors Determination (NTLs Model)



Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
MSDS 7333 - Quantifying the World - Case Study # 2
01/22/2019

To determine if NTL's model is optimal for this dataset, we built training and prediction KNN model, utilizing average signal strengths to predict locations of our hand-held devices. Our prediction model observed what our error model rates would be for nearest neighbor values of k = 1 to 12, using an 11-fold cross validation approach, picking the optimal k value. Our model allows us to manipulate an additional 3 key KNN variables:

A) Mac ID removal (3 options: Removal of MacID: 1 = 00:0f:a3:39:e1:c0
Removal of MacID: 2 = 00:0f:a3:39:dd:cd
Removal of no Mac IDs)

B) Change Number of Included training dataset angles:
(4 options: 1, 2, 3, or 4 angles)

C) Weighting of Nearest Neighbor distance: (Yes or No)

MacIDs

The results from Table 5 show the models that did not eliminate any MacID's performed better than models that eliminated either of them. Also, models that removed MacID 1 performed better than those the removed MacID2 (NTL's choice).

Table 5: KNN Model Results

	Parameter Scenarios	SubMac ID Removed	KNN Distance Weighted	# of Signal Angles	Results: Optimal # K	Min. SSE	SSE*
	SM2_WF_SA4	2	FALSE	4	5	1075.6	287.07
NTL Case	SM2_WF_SA3	2	FALSE	3	5	1087	275.51
	SM2_WF_SA2	2	FALSE	2	6	1106.5	296.55
	SM2_WF_SA1	2	FALSE	1	7	1344	373.06
	SM2_WT_SA4	2	TRUE	4	5	1071.5	286.63
	SM2_WT_SA3	2	TRUE	3	5	1073.2	273.12
	SM2_WT_SA2	2	TRUE	2	6	1076.7	293.78
	SM2_WT_SA1	2	TRUE	1	7	1320.7	366.37
	SM1_WF_SA4	1	FALSE	4	3	908.2	283.17
	SM1_WF_SA3	1	FALSE	3	4	899.5	243.43
	SM1_WF_SA2	1	FALSE	2	6	1038.3	255.75
	SM1_WF_SA1	1	FALSE	1	10	1217.2	299.55
	SM1_WT_SA4	1	TRUE	4	3	907.7	287.7
	SM1_WT_SA3	1	TRUE	3	6	890.3	242.4
	SM1_WT_SA2	1	TRUE	2	9	1005.7	246
	SM1_WT_SA1	1	TRUE	1	12	1167.7	279.65
	SM0_WF_SA4	0	FALSE	4	4	881.5	244.34

	SM0_WF_SA3	0	FALSE	3		4	870.6	230.32
	SM0_WF_SA2	0	FALSE	2		6	982.8	225.85
	SM0_WF_SA1	0	FALSE	1		9	1113.2	280.45
	SM0_WT_SA4	0	TRUE	4		4	869	242.63
Optimal	SM0_WT_SA3	0	TRUE	3		6	863.7	207.25
	SM0_WT_SA2	0	TRUE	2		6	957	215.68
	SM0_WT_SA1	0	TRUE	1		9	1091.3	269.89

* SSE after running complete training dataset with optimal parameters for use case

Signal Angle Inclusion

The number of signal angles included influences the prediction capability of the model. While the inclusion of more angles intuitively seems to be a better choice (more data points) the variation in signal strengths associated with those extra points dilutes the ability of the model to differentiate between locations. It appears that 3 or 4 signal angle inclusion is optimal for this dataset. It is noted that as the number of included angles is decreased, there tends to be an increase of optimal k (neighbors) needed for the model.

Weighted Distance

NTL's initial KNN model's prediction location used an average of the k-nearest neighbors' location to predict the location of the hand-held devices. Another technique is to apply weighted distance metric which is inversely proportional to the difference in signal strengths (d_i) This would likely increase the accuracy to the predictions. The equation is shown below:

$$\frac{1/d_i}{\sum_{i=1}^{k} 1/d_1} \quad [1]$$

Reviewing the results from Table 5, k-neighbor analysis using weighted distance did improve the accuracy of the models modestly by 5.8% on average.

Optimal Model

Reviewing the results from Table 5 and Fig 7a and 7b, the model with lowest sum square error (SSE) was model SM0_WT_SA3, with an SSE of 863.7. The SSE for the complete model, utilizing the optimal parameters is 207.25. The parameters are:

- No Mac IDs removed
- Weighted Distances
- Three (3) signal angles
- Six (6) Nearest Neighbors (k)

From an SSE point of view, this is a 33% increase in accuracy over NTL's model

Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
MSDS 7333 - Quantifying the World - Case Study # 2
01/22/2019

Future Work / Discussions

Practicality of using K-nearest neighbors

Nearest neighbor methods are computationally intensive. Having to find the k-NN for every new signal requires processing the entire training dataset, or at least in this case, the summary dataset to determine the k nearest neighbors.

Figure 7a: Floor Plan Predicted vs Actual Locations (Optimal Model)

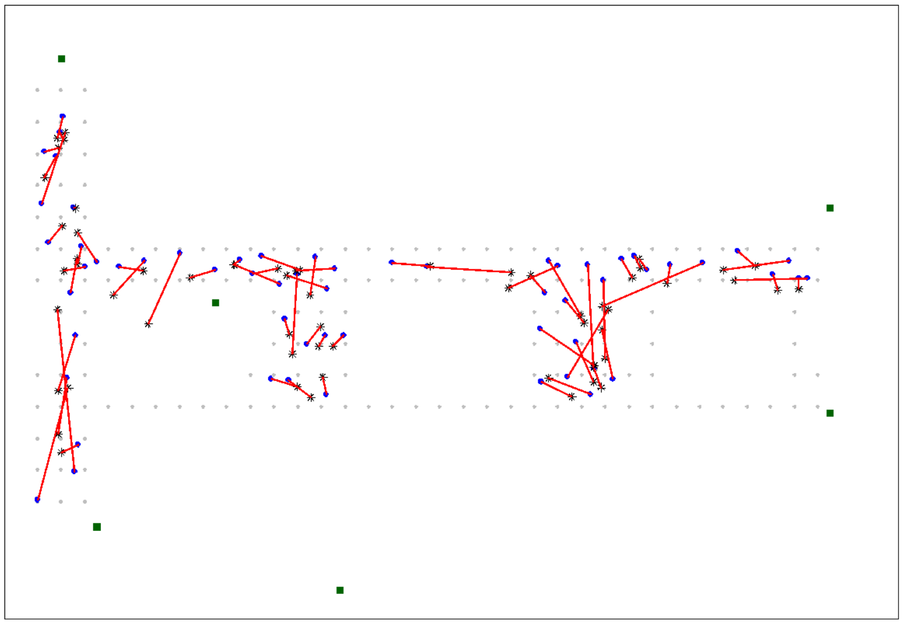
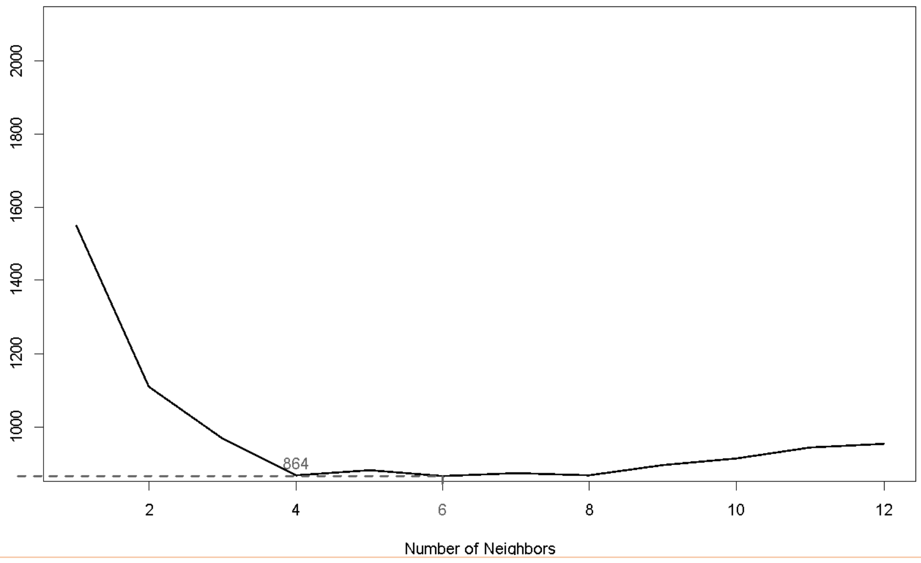


Fig 7b: Optimal Number of Nearest Neighbors Determination (Optimal Model)



There is no practical way to pre-process the training data or store the results from the training data (Algorithm/formula, variables), so that the task of processing the training dataset does not have to be repeated performed. There are a few tricks to help speed the calculations of finding the KNN but no method of eliminating this task.

The nearest neighbor (NN) methodology cannot be employed unless there is a training dataset 'base case' established. RTLS systems based on NN methodology are not plug-and-play straight out of the box and require a decent level of technical expertise to set them up. This may limit their market.

Alternative methodologies

Given that KNN methodology has its drawbacks, it would be prudent to investigate RTLS alternative methodologies. Some of the methodologies previously employed in RTLS systems, either to increase accuracy, reduce setup requirements, and/or improve overall operations are:

- Least Squares
- Clustering Analysis (Bayesian)
- Gaussian
- others…..

Degradation of Signal

Signal strength degradation plays an instrumental part in the RTLS. By default, signal strength degradation is not a 'good' quality to have when working with wireless sensors. Most applications want a robust signal with little to no degradation in strength. The opposite is true for RTLS. A substantial and predictable signal strength degradation is advantageous. If the signal strength diminishes minimally over a long range, then it is difficult to determine where a signal is located based on this measurement. The location determination error may be too great to be practically usable.

This was evident with this use case as well. Looking at the floor map, depicting estimated vs actual location, you can see a definite trend that the size of location errors being more pronounced in the 'open' hallways, where signal strength tends to not degrade as rapidly as other areas of the map, thus, KNN has a difficult time predicting the location of the signal/object.

Noise (obstructions), if predictable or constant, can help improve location determinization, by degrading the signals more rapidly than would otherwise be observed

Gather additional information / Improve Hardware

If improvements in performance were desired over the current system, there are several ways to accomplish this. One would be to gather additional data, maybe upgrade the

router system to include directional antenna, to gather the angle-of-arrival (AoA) of incoming signal. This would improve the systems' location determination ability.
Under special situations, there may be other type of data that can be gathered: Time-of-flight (ToF), time-of-arrival (ToA). These are not readily deployed due to the nature of the hardware and synchronicity that is required.

Conclusions

We conclude that including the data from all the seven Wi-Fi access points(routers) utilizing our average weighted signal strength KNN model gives best location predictability. Signal strength degradation plays an instrumental part in this Real time location system (RTLS) implemented within the defined floorplan of a building. Looking at the floor map, depicting estimated vs actual location, you can see a definite trend that the size of location errors being more pronounced in the 'open' hallways, where signal strength tends to not degrade as rapidly as other areas of the map, thus, KNN has a difficult time predicting the location of the signal/object. Noise (obstructions), if predictable or constant, can help improve location determinization, by degrading the signals more rapidly than would otherwise be observed.

Real time location systems (RTLS) have a wide range of practical applications. Indoor positioning can help with identifying mobile entities within a building ensuring the safety and security. It can be used to track young children(infants/toddlers) inside home or child care facility or to track mentally ill and /or restrained patients. It may also deter theft by tracking the position of the assets within the building or being notified if it is outside the confines of the building. Such RTLS systems with the use of tracking tags can be used in industrial setting for indoor supply chain management, inventory management etc.

References

[1] Nolan, Deborah and Lang, Duncan Temple; from: "Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving" Chapman & Hall/CRC ©2015
[2] "Geo Location in R"; from: http://rdatasciencecases.org/GeoLoc/code.R
[3] "Predicting Location via Indoor Positioning Systems"; from: http://rdatasciencecases.org/Data.html

Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud

MSDS 7333 - Quantifying the World - Case Study # 2

01/22/2019

Appendix A

R Code

```
#  MSDS 7333 - Quantifying the World - Case Study #2
#  Predicting Location via Indoor Positioning System
#  Team Members:
#         Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
#  Date: 01/22/2019
#  Case Study from: Data Science in R: Nolan,Temple,Lang (Ch 1)
#  Initial source code: http://rdatasciencecases.org/code.html
#  Initial Dataset: http://rdatasciencecases.org/Data.html

options(digits = 2)
# Setting working directory to folder where source
# code is located
library(lattice)
library(fields)
library(rstudioapi)
current_path <- getActiveDocumentContext()$path
setwd(dirname(current_path ))

# Data Exploration
# Loading Offline Data text file
txt = readLines("Data/offline.final.trace.txt")
length(txt)    # Loaded 151392 lines/strings
head(txt,4)

# Locating and counting the lines/strings that
# begin with '#' Comment Lines
sum(substr(txt, 1, 1) == "#") # Total of 5312

# Parsing 'txt' character strings; Eliminating the comment lines
# beginning with '#'
lines = txt[ substr(txt, 1, 1) != "#" ] # 146080 strings


# From documnentation, we expect there to be 146,080 lines
# (166 locations x 8 angles x 110 recodings) = 146,080
# This leaves 5312 comment lines (151392 - 146080)

    #------------------ sample Parsing
    # We split the string on several
    # different characters ( ; = , ). Using ( txt[4] )
    # as sample. Saving strings to new variable ' tokens'
    tokens = strsplit(txt[4], "[;=,]")[[1]]

    # The first 10 parts are
    tokens[1:10]
    tokens[c(2, 4, 6:8, 10)]

    # Displaying tokens variables minus the identification components located
    # in the first 10 variables.  MACID; Signal Strength, Freq., Mode
    tokens[ - ( 1:10 ) ]

    # convert tokens to martrix w/o ID components. Matrix 4 columns X # of rows
    tmp = matrix(tokens[ - (1:10) ], ncol = 4, byrow = TRUE)
    # combines tokens' tmp matrix, adding on the relavent ID components from tokens
    mat = cbind(matrix(tokens[c(2, 4, 6:8, 10)],
                  nrow = nrow(tmp),
                  ncol = 6, byrow = TRUE), tmp)

    # Creates a 11 X 10 matrix of the results of txt[4].
    dim(mat)

      #-------------------- processingLine()  Parsing function
      # Now that we know the parsing works properly, we can create a function to
      # to preform the same parsing activity to all observations

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

       # Adds error statement if tokens length is == 10 (no sensor data)
       # Will return a null value instead or warning message
       if (length(tokens) == 10)
         return(NULL)

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

    # Trying the Parsing function on the offline dataset (txt)
```

```
options(error = recover, warn = 1)
tmp = lapply(lines, processLine)
typeof(tmp) # Creates a list
tmp[1]

# Listing the number of rows in for each  of the first 17 'tmp' list
sapply(tmp[c(1:17)], nrow) # 11 10 10 11  9 10  9  9 10 11 11  9  9  9  8 10 14

# Create a dataframe, called offline' with the lists from tmp
offline = as.data.frame(do.call("rbind", tmp),stringsAsFactors = FALSE)
dim(offline) #1181628 rows 10 columns

#####offlineTest = offline
####identical(offline,offlineTest)

# Changing names of columns in offline dataFrame to more
# descriptive names
names(offline) = c("time", "scanMac", "posX", "posY", "posZ",
                "orientation", "mac", "signal",
                "channel", "type")

# Creating list numVars: of 'numerical' variables columns.
#   Time, Position, Orientation, Signal Strength
numVars = c("time", "posX", "posY", "posZ",
         "orientation", "signal")


# Converting Offline dataFrame variables to numeric variables from character
# using NumVars list
offline[ numVars ] =  lapply(offline[ numVars ], as.numeric)
summary(offline)
#dim(offline) #1181628 rows x 10 columns

# Keeping only 'Access point'  type = 3 (adhoc type=1 are eliminated)
offline = offline[ offline$type == "3", ] # 203,185 observations eliminated
offline = offline[ , "type" != names(offline) ] # Removing 'type' from Dataframe
#dim(offline) #978443 rows x 9 columns

# Creating a new variable 'rawTime'
offline$rawTime = offline$time
offline$time = offline$time/1000
# Creating list of classes for offline$time
```

```
class(offline$time) = c("POSIXt", "POSIXct")
# Converts time variable into 2 variable types time1-POSIXt and time2-POSIXct
unlist(lapply(offline, class))
offline$time[1]

# Displaying summary of offline dataframe (variable columns only)
summary(offline[, numVars])

#Displaying summary of offline dataframe (character variables)
# Also converting 'mac','channel',and 'scanMac' columns to factors
summary(sapply(offline[ , c("mac", "channel", "scanMac")],
            as.factor))

# Eliminating the scanMac and z position 'posZ' variables/columns from dataFrame
offline = offline[ , !(names(offline) %in% c("scanMac", "posZ"))]
#dim(offline) #978443 rows x 8 columns

# Determining the number of unique orientations within
# the dataset
length(unique(offline$orientation)) #203 unique orientations

####testing = offline
####identical(offline, testing)
#------------- FIG 1.2-Test  Plotting Empirical Cumulative Distribution
#?ecdf()
plot(ecdf(offline$orientation))
# Creating PDF of ECDF graph
pdf(file = "Fig 1.2 Geo_ECDFOrientation.pdf", width = 10, height = 7)
oldPar = par(mar = c(4, 4, 1, 1))
plot(ecdf(offline$orientation), pch = 19, cex = 0.3,
   xlim = c(-5, 365), axes = FALSE,
   xlab = "orientation", ylab = "Empirical CDF", main = "")
box()
axis(2)
axis(side = 1, at = seq(0, 360, by = 45))
par(oldPar)
dev.off()

#------------- Plotting Geo Density Orientation
# Creating PDF of Geo Density graph
pdf(file = "Fig 1.2b Geo_DensityOrientation.pdf", width = 10, height = 5)
oldPar = par(mar = c(4, 4, 1, 1))
```

```
plot(density(offline$orientation, bw = 2),
   xlab = "orientation", main = "")
par(oldPar)
dev.off()

#---------- Orientation Angle (round to 8 values 0,45,90...270,315)
# Currently 203 unique orientations. ROund to nearest value.
roundOrientation = function(angles) {
 refs = seq(0, by = 45, length  = 9)
 q = sapply(angles, function(z) which.min(abs(z - refs)))
 c(refs[1:8], 0)[q]
}

offline$angle = roundOrientation(offline$orientation)

#------------- FIG. 1.3  Plotting BoxPlots Rounded Orientation Angle
# Creating PDF of BoxPlot Angle graph
pdf(file = "Fig 1.3 Geo_BoxplotAngle.pdf", width = 10)
oldPar = par(mar = c(4, 4, 1, 1))
boxplot(offline$orientation ~ offline$angle,
        xlab = 'nearest 45 degree angle',
        ylab = 'orientation')
par(oldPar)
dev.off()


# 12 mac addresses; 8 channels
# For additional info about access points from
# mac IDs http://coffer.com/mac_find/
c(length(unique(offline$mac)), length(unique(offline$channel)))

# Create a table showing the mac ID and the number of occurrances
# Create a list of the top 7 mac IDs(access Points) in
# decreasing order.
# Removing the mac IDs with the lowest number of
# readings (to eliminate access points that are not
# included in the tests)
table(offline$mac)
subMacs = names(sort(table(offline$mac), decreasing = TRUE))[1:7]
offline = offline[ offline$mac %in% subMacs, ]

# Each mac-ID is associated with a certain 'channel'
# thus, we can eliminate the 'channel' variable from
# the dataset
macChannel = with(offline, table(mac, channel))
apply(macChannel, 1, function(x) sum(x > 0))

offline = offline[ , "channel" != names(offline)]
summary(offline)
offlineExplore = offline

#---------------- Exploring position of Hand Held

# Determining the number of locations we have. Create
# a list of dataframes containing all unique
# posx (34),posy (14) combinations [476 total]

locDF = with(offline,
        by(offline, list(posX, posY), function(x) x))
typeof(locDF)
locDF[3]
length(locDF) # 476 locations
sum(sapply(locDF, is.null)) # 310 are empty

# This leaves 166 locations that were observed. Matches
# the literature.  We eliminate the non-observed locatoins
# from location dataframe list (locDF)
locDF = locDF[ !sapply(locDF, is.null) ]
length(locDF) # List of 166 unique location dataframes


# Determine the observations for each observed
# unique location
# Creates matrix with location and count of number
# of observations
locCounts = sapply(locDF,
               function(df)
                 c(df[1, c("posX", "posY")], count = nrow(df)))
class(locCounts) # matrix
dim(locCounts) # 3 166

locCounts[ , 1:8] # approx 5500+ observations for each
# observed locations (8 orient X 7 acc
# pts x 110 replications = 6160 possible
# observations)

#-------------FIG 1.5 Plotting X,Y Dot Plot with # of Observations
#
# Creating PDF of Dot plot displaying the number of observations
# at each x,y  location
pdf(file = "Fig 1.5 Geo_XYByCount.pdf", width = 10)
oldPar = par(mar = c(3.1, 3.1, 1, 1))
# locCounts matrix transposed
locCounts = t(locCounts)
plot(locCounts, type = "n", xlab = "", ylab = "")
text(locCounts, labels = locCounts[,3], cex = .8, srt = 45)

par(oldPar)
dev.off()

#-------------FIG 1.6 Creating a grid of boxplots of signal strength by
# orientation/angle, subdivided by macID
pdf(file = "Fig 1.6 Geo_BoxplotSignalByMacAngle.pdf", width = 7)
oldPar = par(mar = c(3.1, 3, 1, 1))

bwplot(signal ~ factor(angle) | mac, data = offline,
    subset = posX == 2 & posY == 12 #Chose to look at pos 2,12 for
    # reference. May choose to look at
    # other locations if desired
    & mac != "00:0f:a3:39:dd:cd",  # eliminated mac id'ed as extra address
    layout = c(2,3))

par(oldPar)
dev.off()

# Signal strenght ranges from min -98 to max -25. More negative the signal
# the weaker the signal
summary(offline$signal)


#-------------FIG 1.7 Creating signal strength density curves matrix
# for each macID and orientation
pdf(file = "Fig 1.7 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 &  #Chose to look at pos 24,4 for
           # reference. May choose to look at
           # other locations if desired
           mac != "00:0f:a3:39:dd:cd",
          bw = 0.5, plot.points = FALSE)

    par(oldPar)
    dev.off()



#-------------- Read Data Function ----------
#--------------------------------------------
# Function reads data and parses it according to
# our requirements.  Duplicates all functions previously executed on dataset
# Puts it in a convieneient function

readData =
 function(filename = 'Data/offline.final.trace.txt', subMacs)
 {
  # Loading Data Set text file
  txt = readLines(filename)
  lines = txt[ substr(txt, 1, 1) != "#" ]

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

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

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

  tmp = lapply(lines, processLine)
  DataFunc = as.data.frame(do.call("rbind", tmp),
              stringsAsFactors= FALSE)

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

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

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

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

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

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

  # round orientations to nearest 45
  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]
  }

  DataFunc$angle = roundOrientation(DataFunc$orientation)

  return(DataFunc)
 }
#----------- readData() Function End ------



#Read original dataset, using readData function
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")
offline = readData(subMacs=subMacs)
```

```
        # Test to see if the original 'stepwise' data parsing
        # and function data parsing dataset are identical
#       identical(offline, offlineExplore) # TRUE


# Creation of new variable posXY which combines both x and y IDs
offline$posXY = paste(offline$posX, offline$posY, sep = "-")

# Creating a list of dataframes for each unique location id (166),
# angle(8), and macIDs(7) = Total 9296 list of dataframes
byLocAngleAP = with(offline,
             by(offline, list(posXY, angle, mac),
              function(x) x))

# ------------------- signal summary function ---
# Signal summary function: Calculates summary statistics on each
# of the dataframes
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
         })

# Creates matrix from unique values from LocAngleAP (9 variables) and
# append signal summary data (median,mean,length,sd,IQR)
offlineSummary = do.call("rbind", signalSummary)    # 9296 x 14


#-------------FIG 1.8 Creating boxplots of standard deviation of
# signal strength for a given range of average signal strengths
win.metafile('Fig 1.8 Geo_BoxplotSignalSDByAvg.wmf')
#pdf(file = "Fig 1.8 Geo_BoxplotSignalSDByAvg.pdf", width = 10)
oldPar = par(mar = c(3.1, 3, 1, 1))

   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")

par(oldPar)
dev.off()

#-------------FIG 1.9 Skewness of signal strength - Scatter plot (smooth)
# looking at mean-median vs # of observations
# No apparent evidence of skewing since mean and median differ by less than
# 1 - 2 dBm. Green trend line confirms this
pdf(file = "Fig 1.9 Geo_ScatterMean-Median.pdf", width = 10)
oldPar = par(mar = c(4.1, 4.1, 1, 1))

   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)))
   # Predicted difference (mean-median) trend line from # observation = 70-120
   lo.obj.pr = predict(lo.obj, newdata = data.frame(num = (70:120)))
   lines(x = 70:120, y = lo.obj.pr, col = "#4daf4a", lwd = 2)

par(oldPar)
dev.off()


#------------------ surfaceSS() Signal strength heat/contour Function
surfaceSS = function(data, mac, angle = 45) {
   require(fields)
   oneAPAngle = data[ data$mac == mac & data$angle == angle, ]
   smoothSS = Tps(oneAPAngle[, c("posX","posY")],
               oneAPAngle$avgSignal)
   vizSmooth = predictSurface(smoothSS)
   plot.surface(vizSmooth, type = "C", main = paste("MacID ",mac," Angle ",angle,"°"),
             xlab = "", ylab = "", xaxt = "n", yaxt = "n")
   points(oneAPAngle$posX, oneAPAngle$posY, pch=19, cex = 0.5)
```

```
}
#--------------------

#-------------FIG 1.10 Signal Strength Heat Map for various Access Points and Angles
pdf(file="Fig 1.10 MedSignal_2AccessPts_2Angles.pdf", width = 9, height = 6)
parCur = par(mfrow = c(2,2), mar = rep(1, 4)) # 2 x 2 plot
  # arbitrarily choosing macID's "00:14:bf:b1:97:90" & "00:0f:a3:39:e1:c0"
  # Angles: 0 & 135
  # using mapply to make 4 separate calls to the surfaceSS function
  mapply(surfaceSS, mac = subMacs[ rep(c(5, 1), each = 2) ],
       angle = rep(c(0, 135), 2),
       data = list(data = offlineSummary))

par(parCur)
dev.off()

# We can identify the access point from the 'dark-red' region on the map
# The affect of orientation can also be visualized
# Corridor affect can also be noted (signal is stronger relative to the
# distance where signals are not blocked by walls)
#------------------------
#------------------------

# We find 2 heat maps that have similar signals, corresponding to an access
# point at roughly x=7.5 & y=6.3. [i.e subMac[2] "00:0f:a3:39:dd:cd"
# and subMac[1] "00:0f:a3:39:e1:c0"].
# The text recommended removing the observations (subMac[2]) out of the analysis;
"00:0f:a3:39:dd:cd".
# offlineSummaryTest = subset(offlineSummary, mac != subMacs[2])



# Multiple options are available for parameters required to conduct the k-Nearest Neighbor
# Calculations.
# To assist in determining optimal results, we have created code loops to allow for rapid
# preformance of calculations with minimal inputs
# Parameters that can be manipulated:
# Number of Signal angles used in development of training dataset
# SubMac IDs to be eliminated within training dataset
# Location Estimation (Weighted distance or not) Boolean Value

#-------------------------------------------
```

Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud

MSDS 7333 - Quantifying the World - Case Study # 2
01/22/2019

```
#-------------Parmeter Settings -------------

# WARNING WARNING.   If code is run as-is it will take a
# about 60 minutes to complete. To run the code in less time,
# reduce the number of options for the key parameters below.

signalAngles = c(1,2,3,4)
SubMacRemoved = c(0,1,2)
weighted = c(FALSE,TRUE)

#-------------------------------------------
#-------------------------------------------

# Creation of a summary dataframe which will include key values:
# MacID's (Removed) ; If Distance is Weighted; # of Angles included in Analysis
# Minimum RMSE for Analysis(using crossvalidated model) 'rmseMin'
# RMSE for Analysis(using full training model, incorporation optimal parameter values)
'CalcErrorOptimal'
# RMSE for all values of k neighbors, for crossvalidated model (err)
Summary = as.data.frame(matrix(ncol=7))
colnames(Summary) <- c('macid','DisWeighted','Angles','val','rmseMin','CalcErrorOPtimal','err')



# SubMac Removal for analysis loop
for (mr in SubMacRemoved){
  # Signal Angles to include in Analysis
  for (Angles in signalAngles){
    # NN weighted or not in Analysis
    for (DisWeighted in weighted){
if (mr == 0){
  offlineSummaryMacRemoved = offlineSummary
  # Create a matrix with the relative locations of th 7 access points
  # using the heat maps to determine which MacID goes with a given location.
  AP = matrix( c( 7.5, 6.3, 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, c("x", "y") ))
  macid = 'NA'

} else {
   offlineSummaryMacRemoved = subset(offlineSummary, mac != subMacs[as.integer(mr)])
   #identical(offlineSummaryTest, offlineSummaryMacRemoved)

   # Create a matrix with the relative locations of th 6 access points
   # using the heat maps to determine which MacID goes with a given location.
   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[ -(as.integer(mr)) ], c("x", "y") ))
   macid = subMacs[as.integer(mr)]}
# Dispaly of Access Pt location and ID
AP

# Look at signal strength vs distance from access pt. Compute distances
# from taget emitting signal to access point.
diffs = offlineSummaryMacRemoved[ , c("posX", "posY")] -
  AP[ offlineSummaryMacRemoved$mac, ]


# Using Eclidean distances for distance between handheld and access pts
offlineSummaryMacRemoved$dist = sqrt(diffs[ , 1]^2 + diffs[ , 2]^2)

#-------------FIG 1.11 Scatter Plot Matrix signal strength vs distance
# for each angle (8) and access points (6/7) = 48/56.
#pdf(file=paste("Fig 1.11
Geo_ScatterSignalDist_MacRemoved_",mr,'_weighted_',DisWeighted,'.pdf'), width = 7, height =
10)
win.metafile(file=paste("Fig 1.11 Geo_ScatterSignalDist_MacRemoved_",mr,'.wmf'), width = 7,
height = 10)

oldPar = par(mar = c(3.1, 3.1, 1, 1))
library(lattice)
xyplot(signal ~ dist | factor(mac) + factor(angle),
     data = offlineSummaryMacRemoved, pch = 19, cex = 0.3,
     par.strip.text = list(cex = 0.7),
     xlab =paste("distance   Mac ID Removed ",subMacs[mr]))
par(oldPar)
dev.off()


#---------------------------------------------------
#--------------
#--------------
```

Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
MSDS 7333 - Quantifying the World - Case Study # 2
01/22/2019

```
#--------------  Working with Online dataset
#--------------
#--------------
#---------------------------------------------------

# using readData function to read in online data and parse the data
# accordingly.  Only using SubMac ID identified from offline results
# Creating location ID posXY variable, same as offline
macs = unique(offlineSummaryMacRemoved$mac)
online = readData("Data/online.final.trace.txt", subMacs = macs)
# (Book Base Case) 34778 observations x 8 variables
online$posXY = paste(online$posX, online$posY, sep = "-")

length(unique(online$posXY)) # 60 unique test positions [Note: 60 posX; 53 posY] (Book Base
Case)

# Create a table with # of online readings at each unique location and angle
tabonlineXYA = table(online$posXY, online$angle)
tabonlineXYA[1:7, ]

# Rearranging the dataset:
# Creating dataset with single entry for each unique position and orientatoin
# Each obseravation will keep: Position variables orientation / angle
# Have separate variables for each MacID
# and create an average Signal Strength variable from all observations
# in online dataset (avg roughly ±110 observations each loc/angle/macID)
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 = (nrow(AP)),
                        dimnames = list(ans$posXY, names(avgSS)))
                cbind(ans, y)
              }))

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

dim(onlineSummary) # Matrix OnlineSummary 60 X 11 (Book Base Case)
names(onlineSummary)
```

Predicting Location via Indoor Positioning System

Jeffery Lancon, Manisha Pednekar, Andrew Walch, David Stroud
MSDS 7333 - Quantifying the World - Case Study # 2
01/22/2019

```
#------------ Nearest Neighbor (1.5.2)
#----------------------------------------------

# Selecting the number of different angle observations that will be used
# during the NearestNeighbor calculations. Example:  Say we have an
# observation at 120°.  We can choose to include only training dataset
# with 135° orientation[closest angle multiple] (m=1), or we can choose
# flanking angles 90°, 135° m=2, or for 3 we choose the closest angle
# and the flanking angles: 90°,135°,180° (m=3).
# The signal strength for each location is averaged across all included
# angles

#------------------ ReShape Dataset Function-Signal Strength
# Similar to previous code that reshaped the Online dataset.
# but creating a function to do it.
# Rearranging the dataset:
# Creating dataset with single entry for each unique position and orientatoin
# Each obseravation will keep: Position variables orientation / angle
# Have separate variables for each MacID
# and create an average Signal Strength variable from all observations
# added the option to select one angle at random for each location.
# Using Boolean operation to make selection

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 = (nrow(AP)),
                               dimnames = list(ans$posXY,
                                               names(avgSS)))
                    cbind(ans, y)
                  }))

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



#------------------ Selection of Traning Dataset Function --
# Similar to previous code that selected the number of different
# angle observations that will be used for the training dataset
# but creating a function to do it
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")
}
#--------
        # #------  Test Code ----
        # # Testing function with an observed angle of 130°; including
        # # 3 (flanking) observation angles. Signal strengths are averaged
        # # for the included angle observations.
        #   train130 = selectTrain(130, offlineSummaryMacRemoved, m = 3)
        #   dim(train130) #166 x 9
        #   head(train130)

#------------------ Nearest Neighbor Function --
# Finding the nearest neighbor for newly observed signal,
# given the training data subset (angles included in training dataset).
# Calculates distance and returns an ordered listing of distances from
# the training observations in order of closeness to newly observed signal
# Also includes the distance values

findNN = function(newSignal, trainSubset) {
  diffs = apply(trainSubset[ , 4:(3+nrow(AP))], 1,
              function(x) x - newSignal)
  dists = apply(diffs, 2, function(x) sqrt(sum(x^2)) )
  closest = cbind(trainSubset[,1:3],dists)
  closest = closest[order(closest$dists),]
  return(closest)
}
#----------

#------------------ KNN Prediction Function --
# Predicting the XY corrdinates for newly observed signals,
# given the subset of training dataset(# of angles,
# and # of nearest neighbors to use.
predXY = function(newSignals, newAngles, trainData,
                 numAngles = 1, k = 3,weighted = FALSE){

  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)
  }

  # Determining predicted location using k nearest neighbors with distance
  # that is inversely proportional to the observed distance.
  if (weighted == TRUE) {
    estX <- matrix(ncol = 1,nrow = length(closeXY))
    estY <- matrix(ncol = 1,nrow = length(closeXY))
    for (i in 1:length(closeXY)){
    numerX=rep(0,k)
    numerY=rep(0,k)
    denom = 0
    for (m in 1:k){
      denom = denom + (1/closeXY[[i]][m,4])}
      for (j in 1:k) {
```

```
      numerX[j] = closeXY[[i]][j,2]*((1/closeXY[[i]][j,4])/denom)
      numerY[j] = closeXY[[i]][j,3]*((1/closeXY[[i]][j,4])/denom)
     }
   estX[i,1] = sum(numerX)
   estY[i,1] = sum(numerY)
    }
  estXY = cbind(estX,estY)

 } else {
 # Determining predicted location using k nearest neighbors using
 # average distances
    estXY = lapply(closeXY,
              function(x) sapply(x[ , 2:3],
                          function(x) mean(x[1:k])))
    estXY = do.call("rbind", estXY)
 }
 return(estXY)
}
#-----
   # # ---------- Test Code ----
   # # Using training Dataset (offlineSummaryMacRemoved) to predict the
   # # location of online dataset (onlineSummary)
   # # Using 3 angles nearest and flanking above and below
   # # and non-weighted distance, using 3 nearest neighbors
   # estXYk3 = predXY(newSignals = onlineSummary[ , 6:ncol(onlineSummary)],
   #             newAngles = onlineSummary[ , 4],
   #             offlineSummaryMacRemoved, numAngles = 3, k = 3,weighted = FALSE)
   #
   # # Same code as above with the exception of
   # # using 1 nearest neighbors and weighted distance
   # estXYk1 = predXY(newSignals = onlineSummary[ , 6:ncol(onlineSummary)],
   #             newAngles = onlineSummary[ , 4],
   #             offlineSummaryMacRemoved, numAngles = 3, k = 1,weighted = TRUE)
   #
   # estXYk5 = predXY(newSignals = onlineSummary[ , 6:ncol(onlineSummary)],
   #             newAngles = onlineSummary[ , 4],
   #             offlineSummaryMacRemoved, numAngles = 3, k = 5,weighted = DisWeighted)


   # #------------------ Floor Map Function --
   # # Creating a floor map with to compare actual online signal XY location
   # # and comparing it to the predicted XY location
   # 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()
   #     # Drawing of Access Points
   #     if ( !is.null(AP) ) points(AP, pch = 15, col = 'DarkGreen')
   #       if ( !is.null(trainPoints) )
   #         # Drawing of 'training points' locations
   #         points(trainPoints, pch = 19, col="grey", cex = 0.6)
   #
   #         #Drawing of Actual point location
   #         points(x = actualXY[, 1], y = actualXY[, 2],
   #               pch = 19, cex = 0.8, col = 'blue' )
   #         #Drawing of Estimated point location
   #         points(x = estXY[, 1], y = estXY[, 2],
   #               pch = 8, cex = 0.8 )
   #         #Drawing line linking the est and actual points
   #         segments(x0 = estXY[, 1], y0 = estXY[, 2],
   #               x1 = actualXY[, 1], y1 = actualXY[ , 2],
   #               lwd = 2, col = "red")
   # }
   # #---
   #
   # # creating variable to locate all the training Points
   # trainPoints = offlineSummaryMacRemoved[ offlineSummaryMacRemoved$angle == 0 &
   #                               offlineSummaryMacRemoved$mac == "00:14:bf:3b:c7:c6" ,
   #                         c("posX", "posY")]
   #
   # #-------------FIG 1.12A Geo Floor Map (actual vs predicted locations)
   # # Using KNN = 3, Number of traning dataset angles = 3
   # #pdf(file=paste("Fig 1.12A
GEO_FloorPlan_K3_Errors_MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',Angles,
".pdf"), width = 10, height = 7)
   # win.metafile(file=paste("Fig 1.12A
GEO_FloorPlan_K3_Errors_MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',Angles,
".pdf"), width = 10, height = 7)
   #
   # oldPar = par(mar = c(1, 1, 1, 1))
   #     floorErrorMap(estXYk3, onlineSummary[ , c("posX","posY")],
   #                 trainPoints = trainPoints, AP = AP)
   # par(oldPar)
```

```
    # dev.off()
    #
    # #-------------FIG 1.12B Geo Floor Map (actual vs predicted locations)
    # # Using KNN = 1, Number of traning dataset angles = 3
    # #pdf(file=paste("Fig 1.12B
GEO_FloorPlan_K1_Errors_MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',Angles,
".pdf"), width = 10, height = 7)
    # win.metafile(file=paste("Fig 1.12B
GEO_FloorPlan_K1_Errors_MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',Angles,
".wmf"), 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()
    #
    # #-------------FIG 1.12C Geo Floor Map (actual vs predicted locations)
    # # Using KNN = 5, Number of traning dataset angles = 3
    # #pdf(file=paste("Fig 1.12C
GEO_FloorPlan_K5_Errors_MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',Angles,
".pdf"), width = 10, height = 7)
    # win.metafile(file=paste("Fig 1.12C
GEO_FloorPlan_K5_Errors_MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',Angles,
".wmf"), 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()


#------------------ Sum Square Error Function --
# creating function to calculate the SS error for KNN prediction models
calcError =
  function(estXY, actualXY)
    sum( rowSums( (estXY - actualXY)^2) )
#-----------

# # Calculating Sum of squared Error for model
actualXY = onlineSummary[ ,c('posX','posY')]
```

```
# SSError <- sapply(list(estXYk1,estXYk3,estXYk5),calcError,actualXY)
# # SS Error for K=1 659; SS Error for K=3 307, SS Error for K=5 276,


# --------------- Cross Validation ----------------
# ------------------------------------------------

# Number of folds for cross validation
# For all of our models, we will use 11 fold CV
v = 11

set.seed(123) # setting seed value, so that results are the same
# for multiple runs. If not, results would vary every time
# code is re-run

        # -------------- Test Code --------
        # Taking a random sample, without replacement, of the PosXY locations
        permuteLocs = sample(unique(offlineSummaryMacRemoved$posXY))
        # Creating a matrix with v columns and unique(posXY values) / V rows
        permuteLocs = matrix(permuteLocs, ncol = v,
                        nrow = floor(length(permuteLocs)/v))

        # Created a subset of offlineSummary matrix according to the split
        # created in permueLocs.
        # Picked PermuteLocs[ ,1] arbitrarity for reference
        onlineFold = subset(offlineSummaryMacRemoved, posXY %in% permuteLocs[ , 1]) #
approx 7900 observations
        #---

#list(unique(offline$mac))
# Confirming that subMac[2] "00:0f:a3:39:dd:cd" is removed from offline dataset
#offlineMacRemoved = offline[ offline$mac != subMacs[as.integer(mr)], ] #reduced to 769332
obs

if (mr == 0){
  offlineMacRemoved = offline
} else {
  offlineMacRemoved = offline[ offline$mac != subMacs[as.integer(mr)], ]}


keepVars = c("posXY", "posX","posY", "orientation", "angle")
```

```
# Create a new dataframe using random sample angles for observations,
# using the reshapeSS function: 166 observations x 11 variables
onlineCVSummary = reshapeSS(offlineMacRemoved, keepVars = keepVars,
                          sampleAngle = TRUE)

        # # ----- Test Code ---
        # # Create onlineFold, using XY locaitons chose by permuteLocs.
        # #  Chose the 1st fold for reference: 15 obs x 11 variables
        # onlineFold = subset(onlineCVSummary,
        #                   posXY %in% permuteLocs[ , 1])
        #
        # # Create offlineFold, using XY locaitons chose by permuteLocs.
        # #  Chose the last fold for reference: 7200 obs x 15 variables
        # offlineFold = subset(offlineSummaryMacRemoved,
        #                    posXY %in% permuteLocs[ , -1])
        #
        # # Using predXY function to predict xy locations, using
        # # cross validation folds of datasets
        # estFold = predXY(newSignals = onlineFold[ , 6:ncol(onlineFold)],
        #                newAngles = onlineFold[ , 4],
        #                offlineFold, numAngles = 3, k = 3,weighted = DisWeighted)
        #
# Calculating SS Error for the folded subset (permuteLocs[ , 1])
# of the dataset
#actualFold = onlineFold[ , c("posX", "posY")]
#calcError(estFold, actualFold) #Calculated SS Error 133 (will vary)

v = 11 #repeated from earlier, for clarity: # of Folds
K = 12 # Max Number of nearest neighbors
err = rep(0, K)


# loop through all the cross validation folds, using K values from
# 1 to 20, to find and calculate the SS Errors for each
# Same as before: using numAngles = 3.
# Attempting to find the 'optimal' KNN value for the given dataset
# and parameters
# Note:  Takes Several minutes to run this loop************


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

  for (k in 1:K) {
    estFold = predXY(newSignals = onlineFold[ , 6:ncol(onlineFold)],
                newAngles = onlineFold[ , 4],
                offlineFold, numAngles = Angles, k = k,weighted = DisWeighted)
    err[k] = err[k] + calcError(estFold, actualFold)
  }
}
# Identifying Optimal number of nearest neighbors
val <- match(min(err),err)


#-------------FIG 1.13 RMSE vs K neighbors-Line graph
# Number of traning dataset angles = 3
#pdf(file = paste("Fig 1.13
Geo_CVChoiceOfK_MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',Angles,".pdf"),
width = 10, height = 6)
win.metafile(file = paste("Fig 1.13
Geo_CVChoiceOfK_",val,"MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',Angles,".
wmf"), width = 10, height = 6)

oldPar = par(mar = c(4, 3, 1, 1))
plot(y = err, x = (1:K),  type = "l", lwd= 2,
    ylim = c((ymin=900), 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 = ymin-100,  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))
par(oldPar)
```

```
dev.off()
#-------

val <- match(min(err),err)
# Finding RMSE for optimal values of KNN k='val'
# Optimal KNN might vary if seed value changes

# Rerun prediction model using full training set, incorporation optimal parameters (# of Nearest
Neighbors)
estXYkBest = predXY(newSignals = onlineSummary[ , 6:ncol(onlineSummary)],
            newAngles = onlineSummary[ , 4],
            offlineSummaryMacRemoved, numAngles = Angles, k = val,weighted = DisWeighted)


#------------------ Floor Map Function --
# Creating a floor map with to compare actual online signal XY location
# and comparing it to the predicted XY location
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()
  # Drawing of Access Points
  if ( !is.null(AP) ) points(AP, pch = 15, col = 'DarkGreen')
  if ( !is.null(trainPoints) )
    # Drawing of 'training points' locations
    points(trainPoints, pch = 19, col="grey", cex = 0.6)

  #Drawing of Actual point location
  points(x = actualXY[, 1], y = actualXY[, 2],
       pch = 19, cex = 0.8, col = 'blue' )
  #Drawing of Estimated point location
  points(x = estXY[, 1], y = estXY[, 2],
       pch = 8, cex = 0.8 )
  #Drawing line linking the est and actual points
  segments(x0 = estXY[, 1], y0 = estXY[, 2],
         x1 = actualXY[, 1], y1 = actualXY[ , 2],
         lwd = 2, col = "red")
}
#---

# creating variable to locate all the training Points
```

```
trainPoints = offlineSummaryMacRemoved[ offlineSummaryMacRemoved$angle == 0 &
                          offlineSummaryMacRemoved$mac == "00:14:bf:3b:c7:c6" ,
                         c("posX", "posY")]

#-------------FIG 1.12 Geo Floor Map (actual vs predicted locations)
# Using optimal KNN = val, Number of traning dataset angles = per model, Weighted Distance =
per model, MacID removed = per model
#pdf(file=paste("Fig 1.12
GEO_FloorPlan_Errors_K_",val,"_MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',A
ngles,".pdf"), width = 10, height = 7)
win.metafile(file=paste("Fig 1.12
GEO_FloorPlan_Errors_K_",val,"_MacRemoved_",mr,'_weighted_',DisWeighted,'_SigAngles_',A
ngles,".wmf"), width = 10, height = 7)

oldPar = par(mar = c(1, 1, 1, 1))
floorErrorMap(estXYkBest, onlineSummary[ , c("posX","posY")],
        trainPoints = trainPoints, AP = AP)
par(oldPar)
dev.off()

#calcError(estXYkBest, actualXY)# Error - varies
#CalcError for optimal number of k-nearest neighbors
CalcErrorOPtimal <- calcError(estXYkBest, actualXY)
paste("Minimum Calc Error: ",CalcErrorOPtimal)
paste("K value: ",val)
paste('MacID Removed: ',macid)
paste('Weighted: ',DisWeighted)
paste('Signal Angles: ',Angles)
paste("Minimum RMSE: ",rmseMin)

#SummaryTemp = data.frame(macid,DisWeighted,Angles,val,rmseMin,CalcErrorOPtimal)
#SummaryTemp$err = list(err)
#Summary = rbind(Summary,SummaryTemp)

#Summary$err[[2]][2]
} # Closing Bracket for Weighted Loop
} # Closing Bracket for Signal Angles Loop
} # Closing bracket for SubMacID Removal loop

# Write Summary CSVs
Summary <- Summary[-1,]
```

```
row.names(Summary) <-
c('SM0_WT_SA1','SM0_WF_SA1','SM0_WT_SA2','SM0_WF_SA2','SM0_WT_SA3','SM0_WF_SA3'
,'SM0_WT_SA4','SM0_WF_SA4',

'SM1_WT_SA1','SM1_WF_SA1','SM1_WT_SA2','SM1_WF_SA2','SM1_WT_SA3','SM1_WF_SA3','
SM1_WT_SA4','SM1_WF_SA4',

'SM2_WT_SA1','SM2_WF_SA1','SM2_WT_SA2','SM2_WF_SA2','SM2_WT_SA3','SM2_WF_SA3','
SM2_WT_SA4','SM2_WF_SA4')
colnames(Summary) <-
c('MacID_Elim','Wt_Dist','Num_Sig_Angles','Opt_Near_Neig','RMSE_CV_Model','RMSE_Full_Mo
del','RMSE_All_Vals_kNN')
write.csv(Summary[,1:6], file = "SummaryDataResults.csv", row.names = TRUE)

# Writing Summary Data (RMSE for all values of K per parameter setting)
write.csv(Summary[,7], file = "SummaryDataResults2.csv", row.names = TRUE)
```