Geolocation

Predicting Location via Indoor Positioning Systems

The Raw Data

```
file_offline <- file.path(data.dir, "offline.final.trace.txt")
file_online <- file.path(data.dir, "online.final.trace.txt")

raw_offline <- read_lines(file_offline)
raw_online <- read_lines(file_online)
```

Sanity Check

```
# number of comment lines in the data
sum(substr(raw_offline, 1, 1) == "#")
```

```
[1] 5312
```

```
# total number of lines in the data
length(raw_offline)
```

```
[1] 151392
```

Data Pre-processing

Generate read data function for preprocessing training and test data.

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

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

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

```
                ncol = 6, byrow = T), tmp)
}

validLines <- function(data) {
   substr(data, 1, 1) != "#"
}

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]
}

readData <- function(file, submacs = macs) {

   lines <- read_lines(file)

   valid_lines <- lines[ validLines(lines) ]

   processed_lines <- lapply(valid_lines, processLine)

   data <- as.data.table(do.call("rbind", processed_lines),
                         stringsAsFactors = F)

   names(data) <- c("time", "scanMac", "posX", "posY", "posZ",
                       "orientation", "mac", "signal",
                       "channel", "type")

   numVars <- c("time", "posX", "posY", "posZ",
                "orientation", "signal")

   data[, (numVars) := lapply(.SD, as.numeric), .SDcols = numVars]

   data <- data[ data$type == 3, ]
   data[, type := NULL]

   data[, rawTime := time]
   data[, time := time/1000]
   class(data$time) = c("POSIXt", "POSIXct")

   # drop scanMac & posZ
   data[, `:=`(scanMac = NULL, posZ = NULL)]

   data$angle = roundOrientation(data$orientation)
```

```
  data$channel = NULL

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

  return(data[mac %in% submacs])
}
```

Test Pre-processor

```
lines <- processLine(raw_offline[4:20])
lines
```

```
      [,1]            [,2]                [,3]  [,4]  [,5]  [,6]
 [1,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
 [2,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
 [3,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
 [4,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
 [5,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
 [6,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
 [7,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
 [8,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
 [9,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
[10,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
[11,] "1139643118358" "00:02:2D:21:0F:33" "0.0" "0.0" "0.0" "0.0"
      [,7]                [,8]  [,9]         [,10]
 [1,] "00:14:bf:b1:97:8a" "-38" "2437000000" "3"
 [2,] "00:14:bf:b1:97:90" "-56" "2427000000" "3"
 [3,] "00:0f:a3:39:e1:c0" "-53" "2462000000" "3"
 [4,] "00:14:bf:b1:97:8d" "-65" "2442000000" "3"
 [5,] "00:14:bf:b1:97:81" "-65" "2422000000" "3"
 [6,] "00:14:bf:3b:c7:c6" "-66" "2432000000" "3"
 [7,] "00:0f:a3:39:dd:cd" "-75" "2412000000" "3"
 [8,] "00:0f:a3:39:e0:4b" "-78" "2462000000" "3"
 [9,] "00:0f:a3:39:e2:10" "-87" "2437000000" "3"
[10,] "02:64:fb:68:52:e6" "-88" "2447000000" "1"
[11,] "02:00:42:55:31:00" "-84" "2457000000" "1"
```

```
offline_valid <- raw_offline[validLines(raw_offline)]

head(offline_valid)
```

```
[1] "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=-38,2437
[2] "t=1139643118744;id=00:02:2D:21:0F:33;pos=0.0,0.0,0.0;degree=0.0;00:14:bf:b1:97:8a=-38,2437
[3] "t=1139643119002;id=00:02:2D:21:0F:33;pos=0.0,0.0,0.0;degree=0.0;00:14:bf:b1:97:8a=-38,2437
```

```
[4] "t=1139643119263;id=00:02:2D:21:0F:33;pos=0.0,0.0,0.0;degree=0.0;00:14:bf:b1:97:8a=-38,2437
[5] "t=1139643119538;id=00:02:2D:21:0F:33;pos=0.0,0.0,0.0;degree=0.0;00:14:bf:b1:97:8a=-46,2437
[6] "t=1139643119818;id=00:02:2D:21:0F:33;pos=0.0,0.0,0.0;degree=0.0;00:14:bf:b1:97:8a=-37,2437
```

```r
length(offline_valid)
```

```
[1] 146080
```

```r
tmp <- lapply(offline_valid[1:17], processLine)

sapply(tmp, nrow)
```

```
 [1] 11 10 10 11  9 10  9  9 10 11 11  9  9  9  8 10 14
```

Dry Run

```r
offline_test <- as.data.table(do.call("rbind", tmp))

offline_test
```

```
                V1                V2  V3  V4  V5  V6                V7  V8
  1: 1139643118358 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 00:14:bf:b1:97:8a -38
  2: 1139643118358 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 00:14:bf:b1:97:90 -56
  3: 1139643118358 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 00:0f:a3:39:e1:c0 -53
  4: 1139643118358 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 00:14:bf:b1:97:8d -65
  5: 1139643118358 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 00:14:bf:b1:97:81 -65
 ---
166: 1139643122647 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 00:0f:a3:39:e0:4b -79
167: 1139643122647 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 00:0f:a3:39:e0:4b -80
168: 1139643122647 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 00:0f:a3:39:e2:10 -83
169: 1139643122647 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 00:0f:a3:39:dd:cd -65
170: 1139643122647 00:02:2D:21:0F:33 0.0 0.0 0.0 0.0 02:00:42:55:31:00 -84
             V9 V10
  1: 2437000000   3
  2: 2427000000   3
  3: 2462000000   3
  4: 2442000000   3
  5: 2422000000   3
 ---
166: 2462000000   3
167: 2462000000   3
168: 2437000000   3
169: 2412000000   3
170: 2457000000   1
```

Process

```r
# Enter Debug Context

# options(error = recover, warn = 2)

offline_data <- lapply(offline_valid, processLine)
offline <- as.data.table(do.call("rbind", offline_data),
                         stringsAsFactors = F)

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

numVars <- c("time", "posX", "posY", "posZ",
             "orientation", "signal")

dim(offline)
```

```
[1] 1181628      10
```

```r
offline[, (numVars) := lapply(.SD, as.numeric), .SDcols = numVars]

offline <- offline[ offline$type == 3, ]
offline[, type := NULL]

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

# options(error = recover, warn = 1)
```

Data Cleaning

```r
summary(offline)
```

```
      time                          scanMac              posX
 Min.   :2006-02-11 02:31:58   Length:978443      Min.   : 0.00
 1st Qu.:2006-02-11 08:21:27   Class :character   1st Qu.: 2.00
 Median :2006-02-11 14:57:58   Mode  :character   Median :12.00
 Mean   :2006-02-16 09:57:37                      Mean   :13.52
 3rd Qu.:2006-02-19 09:52:40                      3rd Qu.:23.00
 Max.   :2006-03-09 15:41:10                      Max.   :33.00
      posY             posZ    orientation        mac
 Min.   : 0.000   Min.   :0   Min.   :  0.0   Length:978443
```

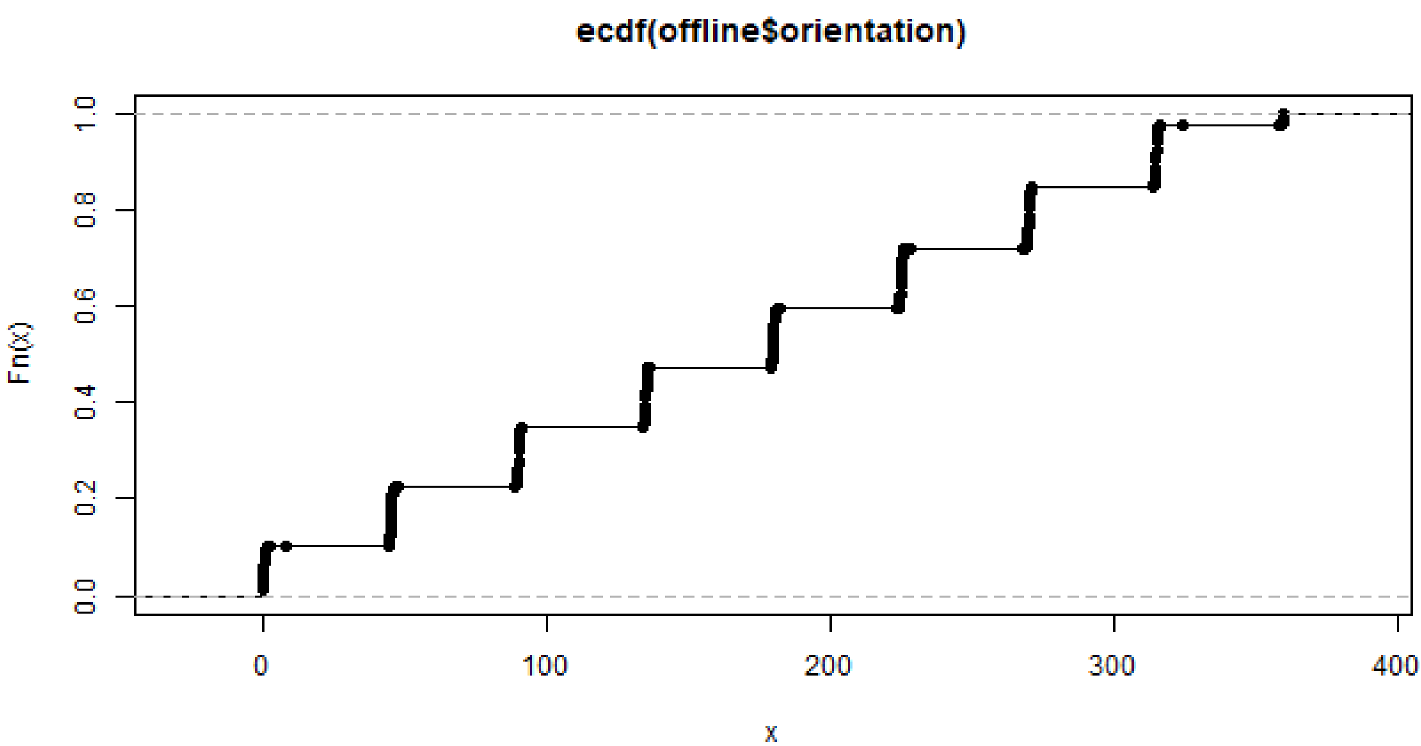
```
 1st Qu.: 3.000   1st Qu.:0   1st Qu.: 90.0   Class :character
 Median : 6.000   Median :0   Median :180.0   Mode  :character
 Mean   : 5.897   Mean   :0   Mean   :167.2
 3rd Qu.: 8.000   3rd Qu.:0   3rd Qu.:270.0
 Max.   :13.000   Max.   :0   Max.   :359.9
     signal          channel             rawTime
 Min.   :-99.0   Length:978443      Min.   :1.140e+12
 1st Qu.:-69.0   Class :character   1st Qu.:1.140e+12
 Median :-60.0   Mode  :character   Median :1.140e+12
 Mean   :-61.7                      Mean   :1.140e+12
 3rd Qu.:-53.0                      3rd Qu.:1.140e+12
 Max.   :-25.0                      Max.   :1.142e+12
```

Orientation Exploration

```r
length(unique(offline$orientation))
```

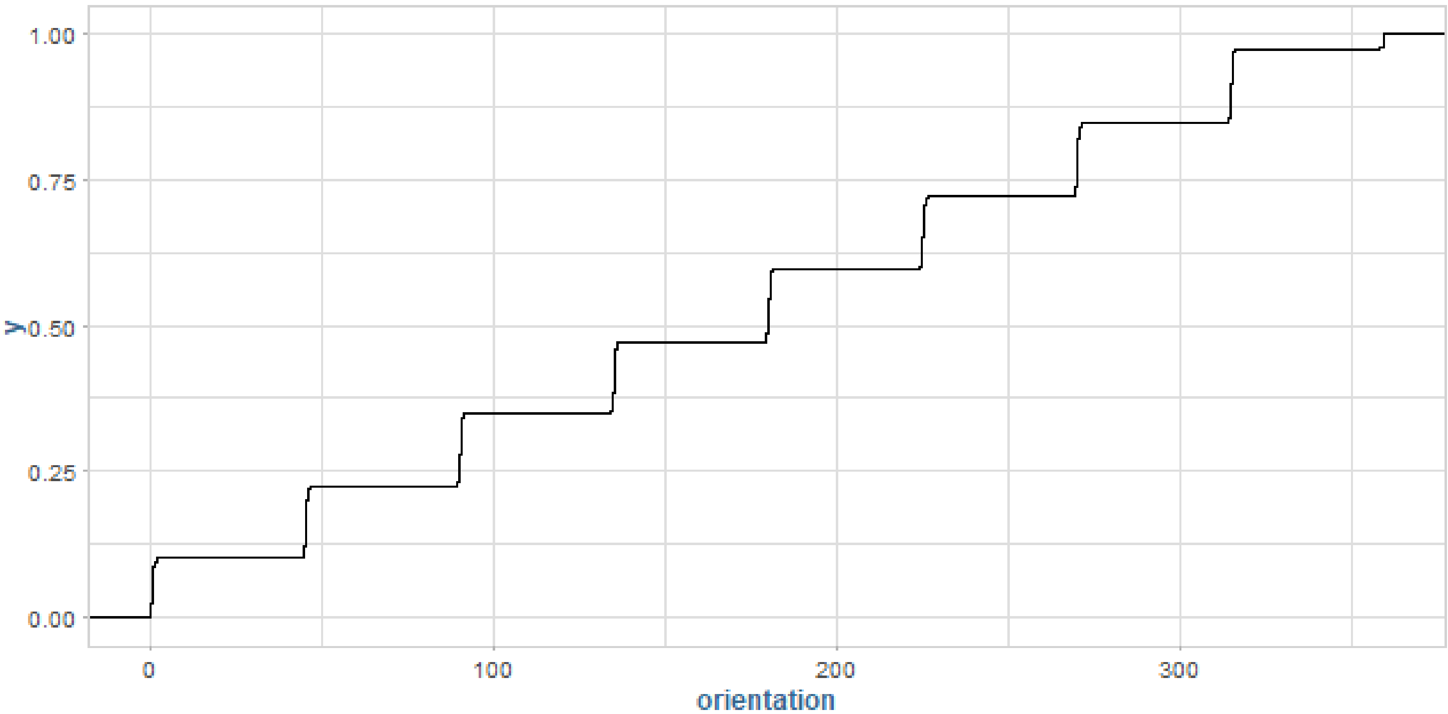
```
[1] 203
```

```r
plot(ecdf(offline$orientation))
```

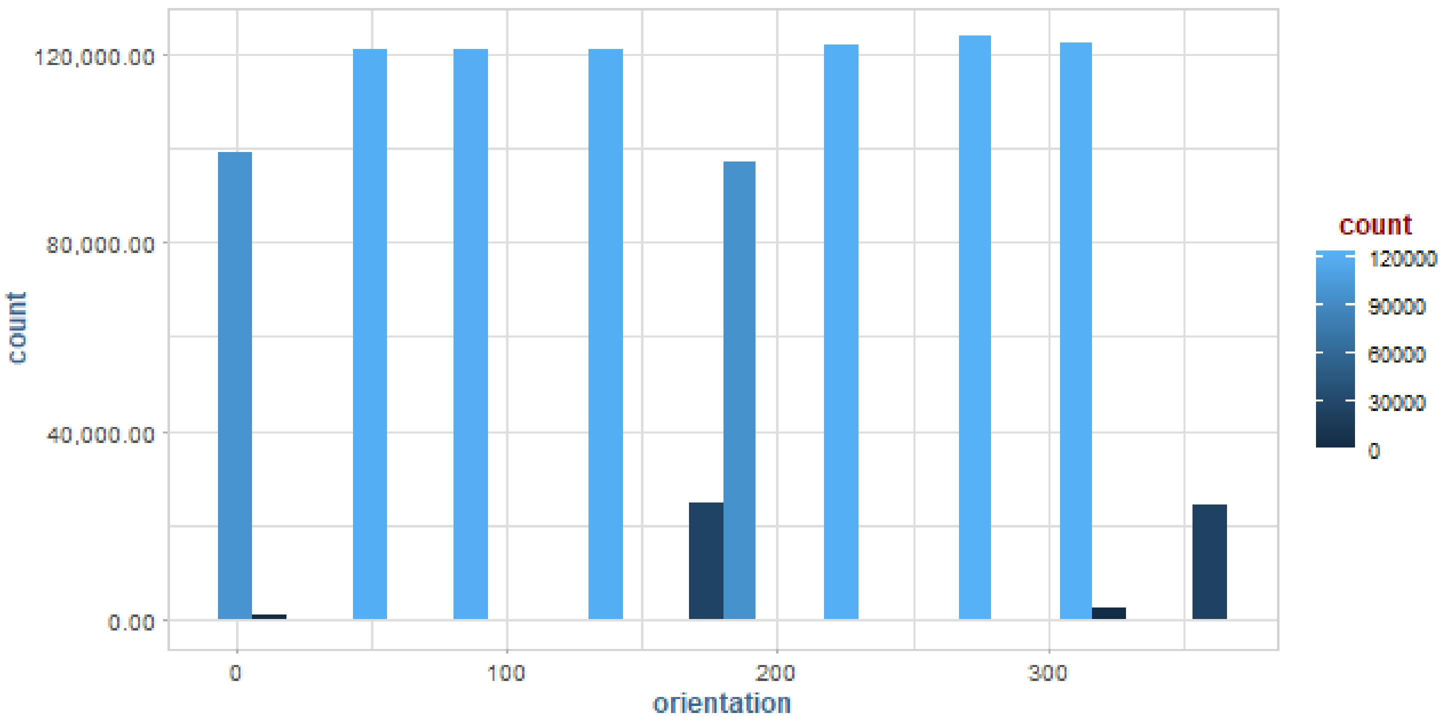



```r
ggplot(offline, aes(orientation)) +
  stat_ecdf() +
```

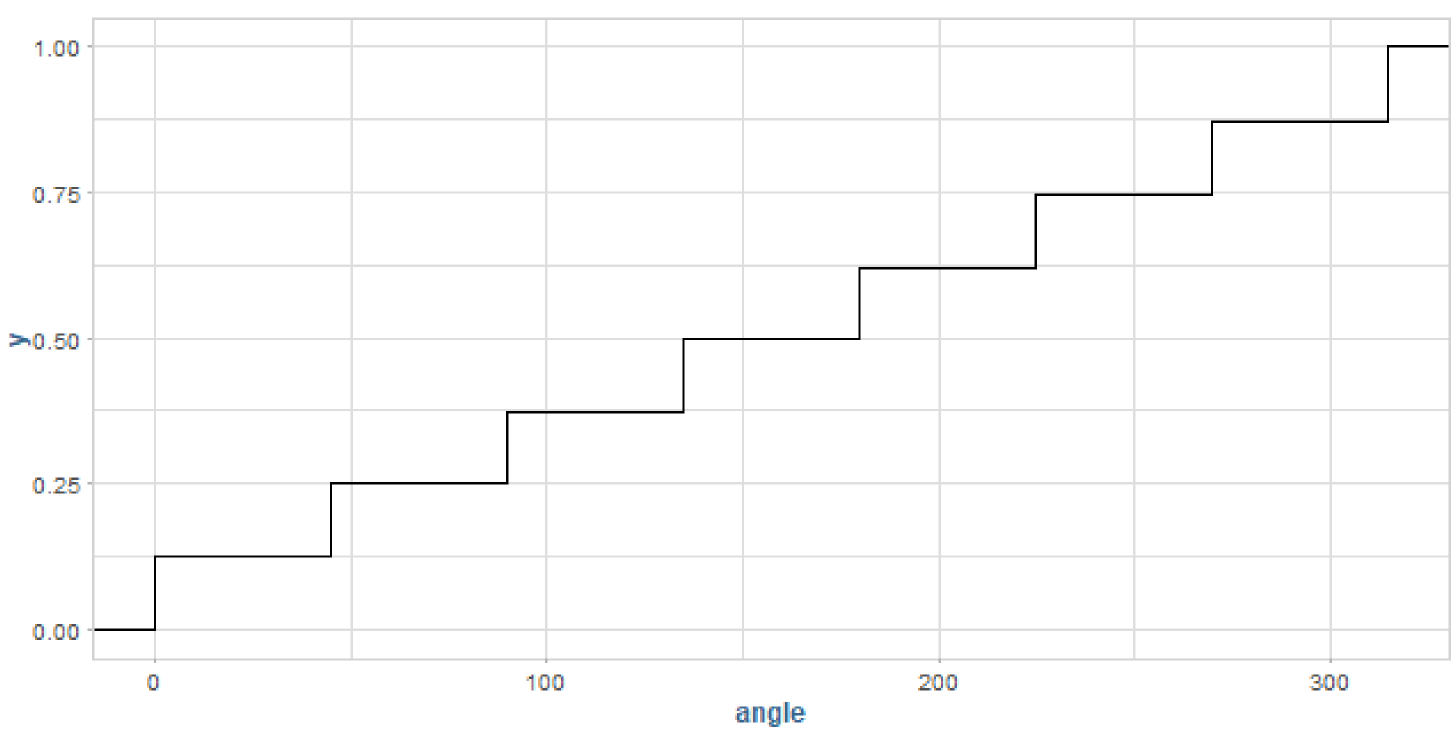
```
  labs("Orientation")
```



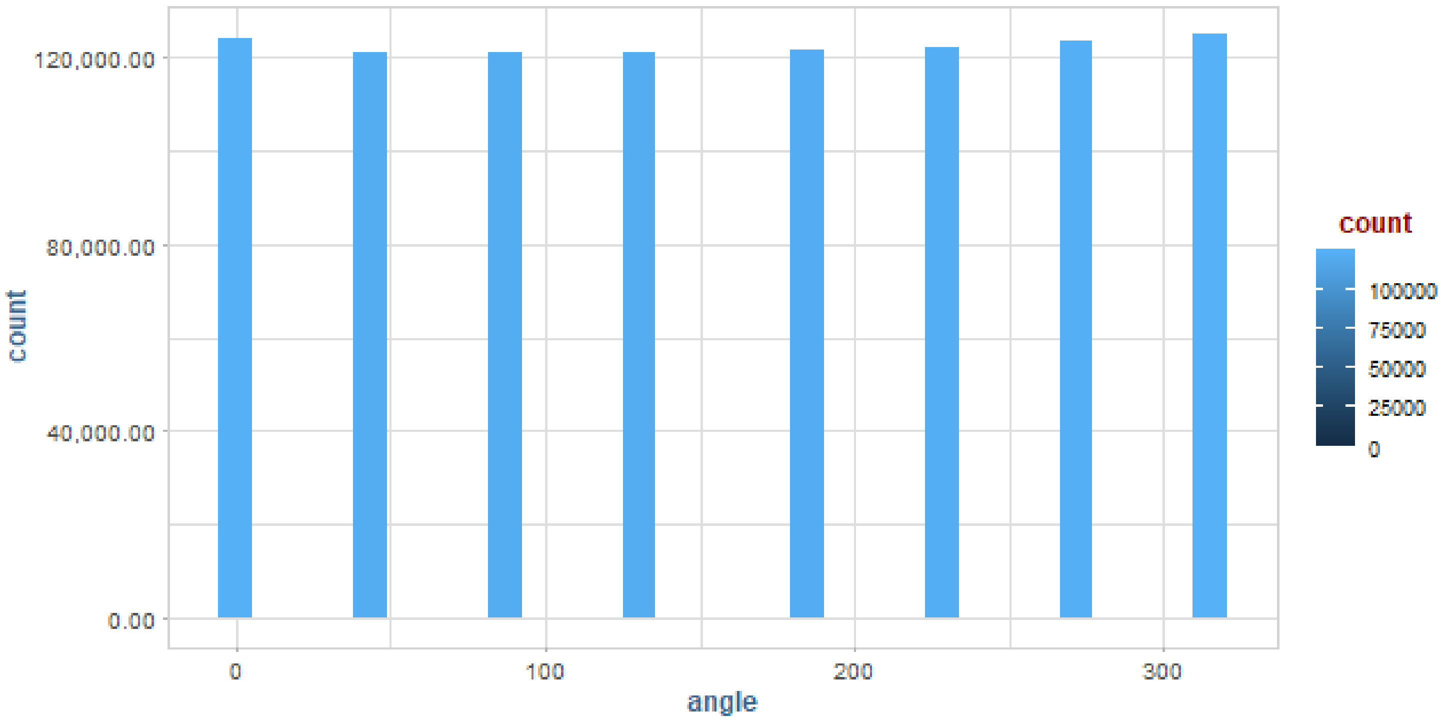

```
ggplot(offline, aes(orientation, fill = ..count..)) +
  geom_histogram(bins = 30) +
  scale_y_continuous(labels = comma) +
  labs("Orientation Value Clusters")
```



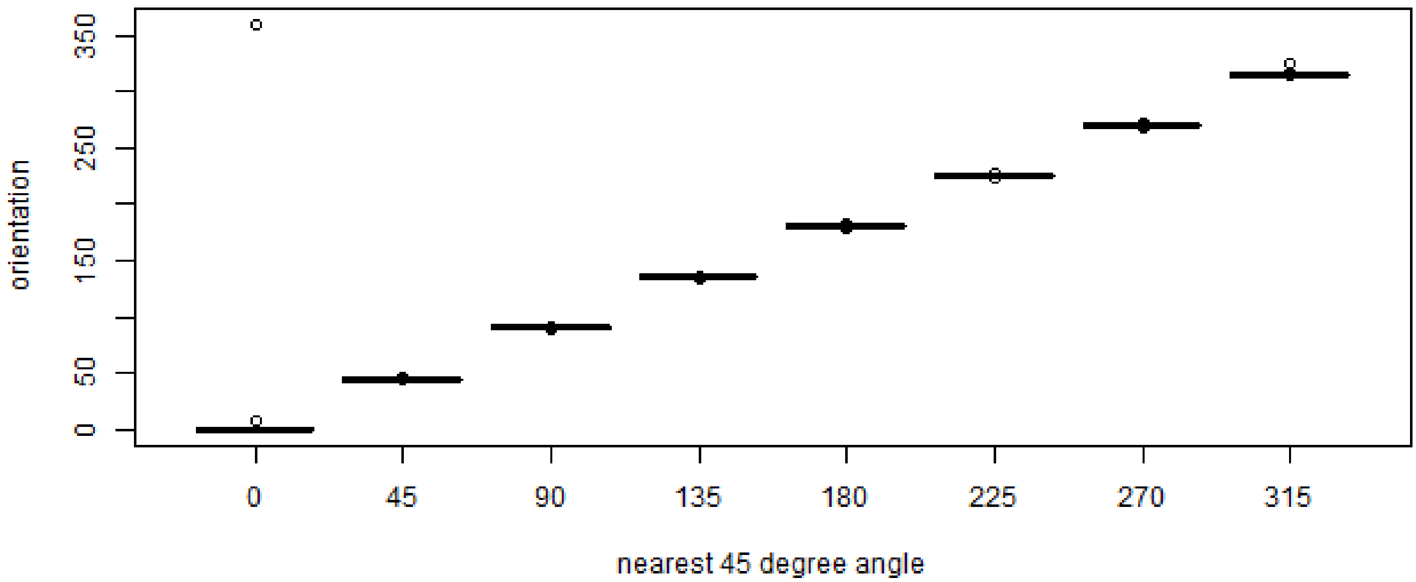

```r
offline$angle <- roundOrientation(offline$orientation)

# angle = cleaned orientation column
ggplot(offline, aes(angle)) +
   stat_ecdf()
```



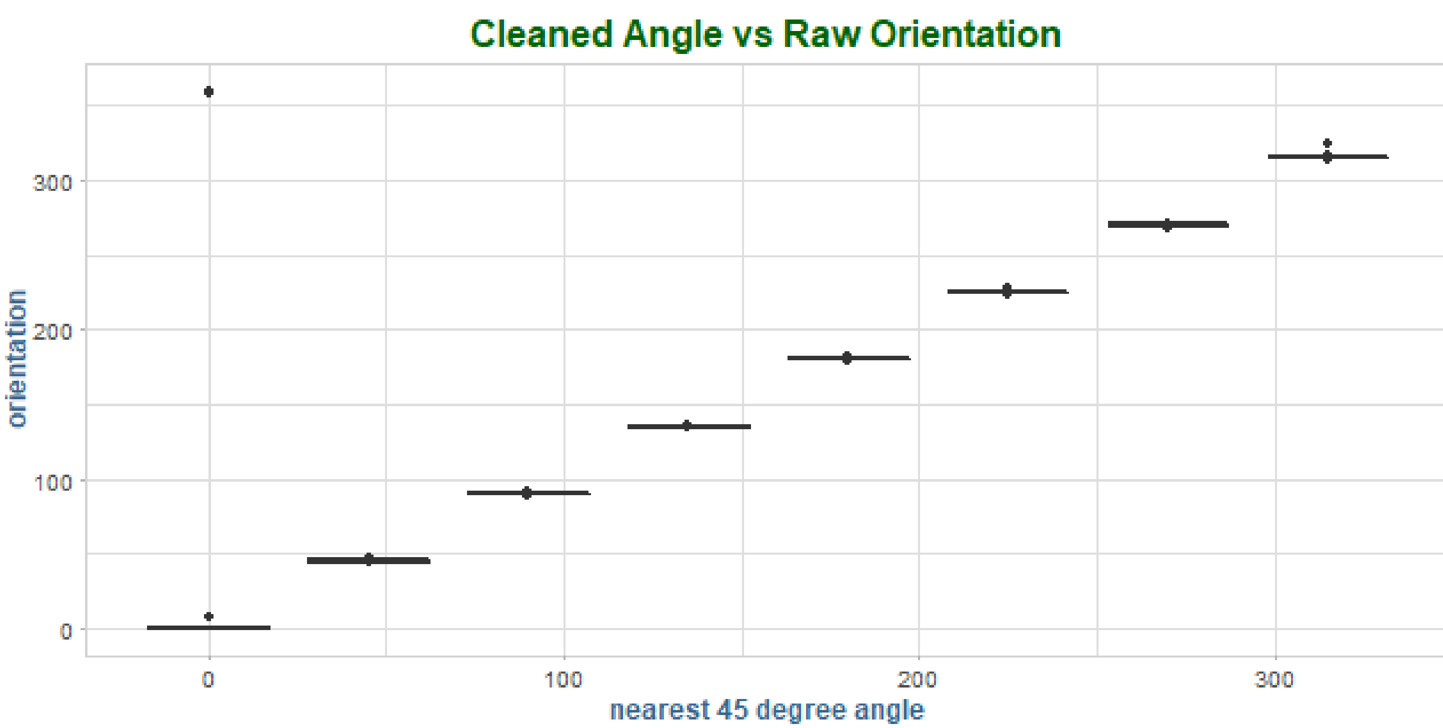
```r
ggplot(offline, aes(angle, fill = ..count..)) +
  geom_histogram(bins = 30) +
  scale_y_continuous(labels = comma) +
  labs("Cleaned Up Angles")
```




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




```r
ggplot(offline, aes(angle, orientation, group = angle)) +
  geom_boxplot() +
  labs(title = "Cleaned Angle vs Raw Orientation",
       x = "nearest 45 degree angle",
       y = "orientation")
```



MAC Address

```r
c(length(unique(offline$mac)), length(unique(offline$channel)))
```

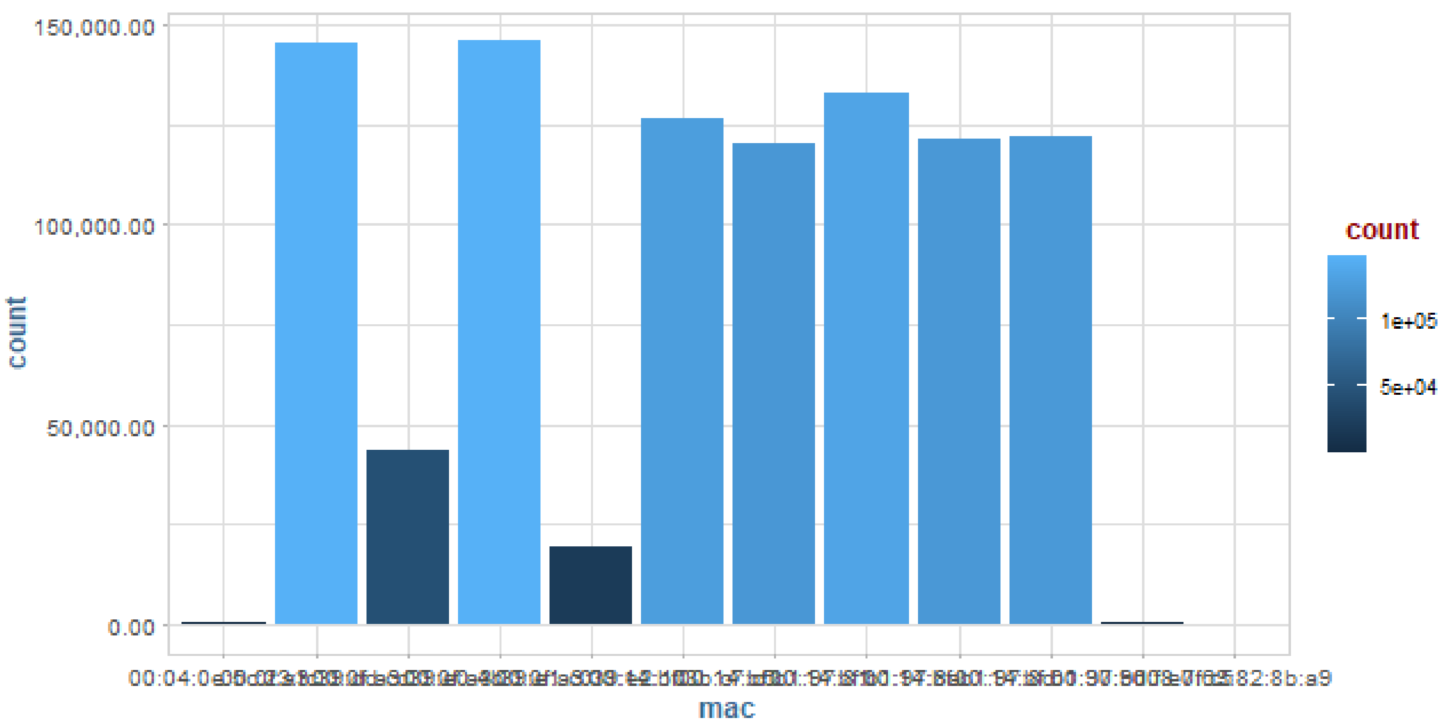
```
[1] 12  8
```

```r
table(offline$mac)
```

```

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

```r
ggplot(offline, aes(mac, fill = ..count..)) +
   geom_histogram(stat = "count", bins = 30) +
   scale_y_continuous(labels = comma)
```

```
Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```r
# keep only the top 7 MAC address data points

dim(offline)
```

```
[1] 978443     11
```

```
offline_macs <- names(sort(table(offline$mac), decreasing = T)[1:7])

dim(offline)
```

```
[1] 978443     11
```

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

```
00:04:0e:5c:23:fc 00:0f:a3:39:dd:cd 00:0f:a3:39:e0:4b 00:0f:a3:39:e1:c0
                1                 1                 1                 1
00:0f:a3:39:e2:10 00:14:bf:3b:c7:c6 00:14:bf:b1:97:81 00:14:bf:b1:97:8a
                1                 1                 1                 1
00:14:bf:b1:97:8d 00:14:bf:b1:97:90 00:30:bd:f8:7f:c5 00:e0:63:82:8b:a9
                1                 1                 1                 1
```

```
# mac and channel are 1:1, we can remove channel
#offline$channel := NULL
```

Exploring the Position of the Hand-Held Device

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

length(locDF)
```

```
[1] 476
```

```
sum(sapply(locDF, is.null))
```

```
[1] 310
```

```
locDF <- locDF[ !sapply(locDF, is.null)]

length(locDF)
```

```
[1] 166
```

```
locCounts <- sapply(locDF, nrow)

locCounts <- sapply(locDF,
                    function(df)
                       c(df[1, c("posX", "posY")], count = nrow(df)))

class(locCounts)
```

```
[1] "matrix"
```

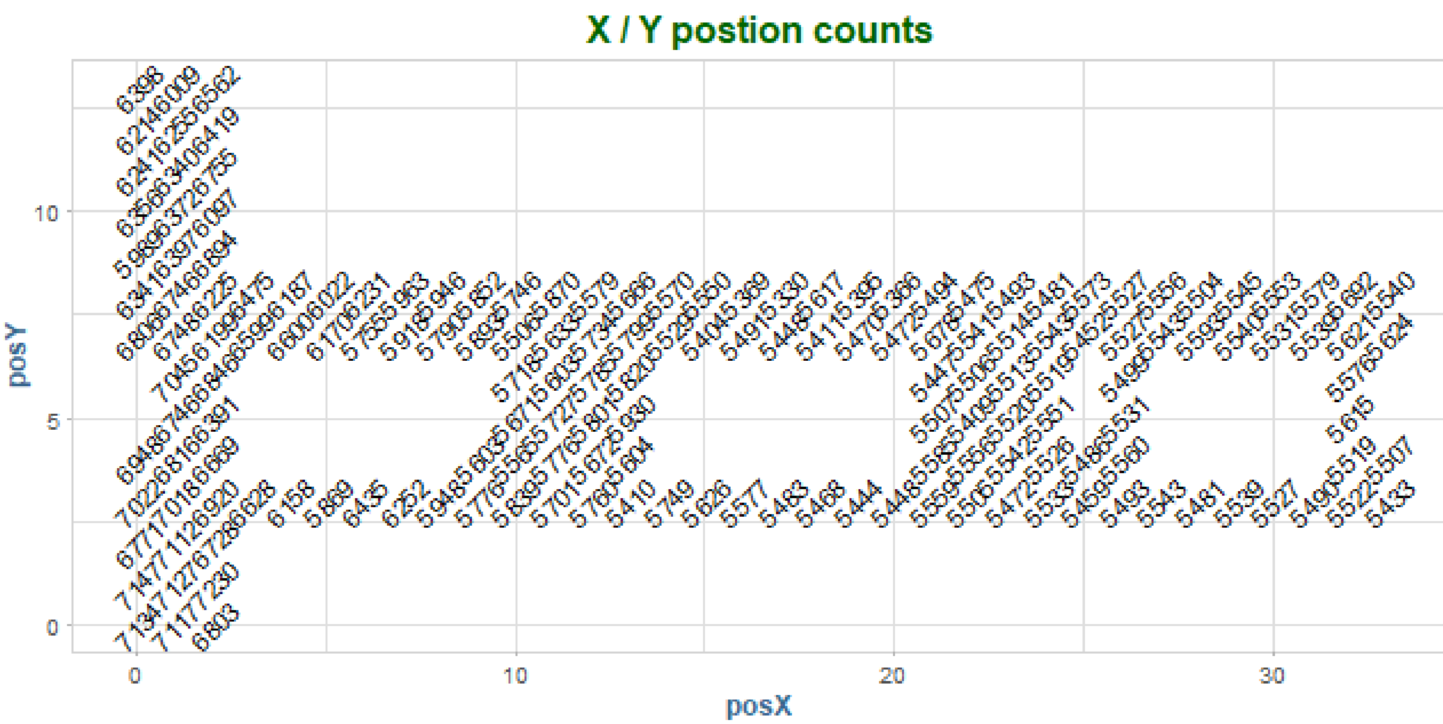
```r
dim(locCounts)
```

```
[1]   3 166
```

```r
locCounts[ , 1:8]
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
posX  0    1    2    0    1    2    0    1
posY  0    0    0    1    1    1    2    2
count 7134 7117 6803 7147 7127 7230 6771 7112
```

```r
locCountsDF <- as.data.table(t(locCounts))

locCountsDF$posX <- unlist(locCountsDF$posX)
locCountsDF$posY <- unlist(locCountsDF$posY)

ggplot(locCountsDF, aes(posX, posY, label = count)) +
  geom_text(angle = 45) +
  labs(title = "X / Y postion counts")
```

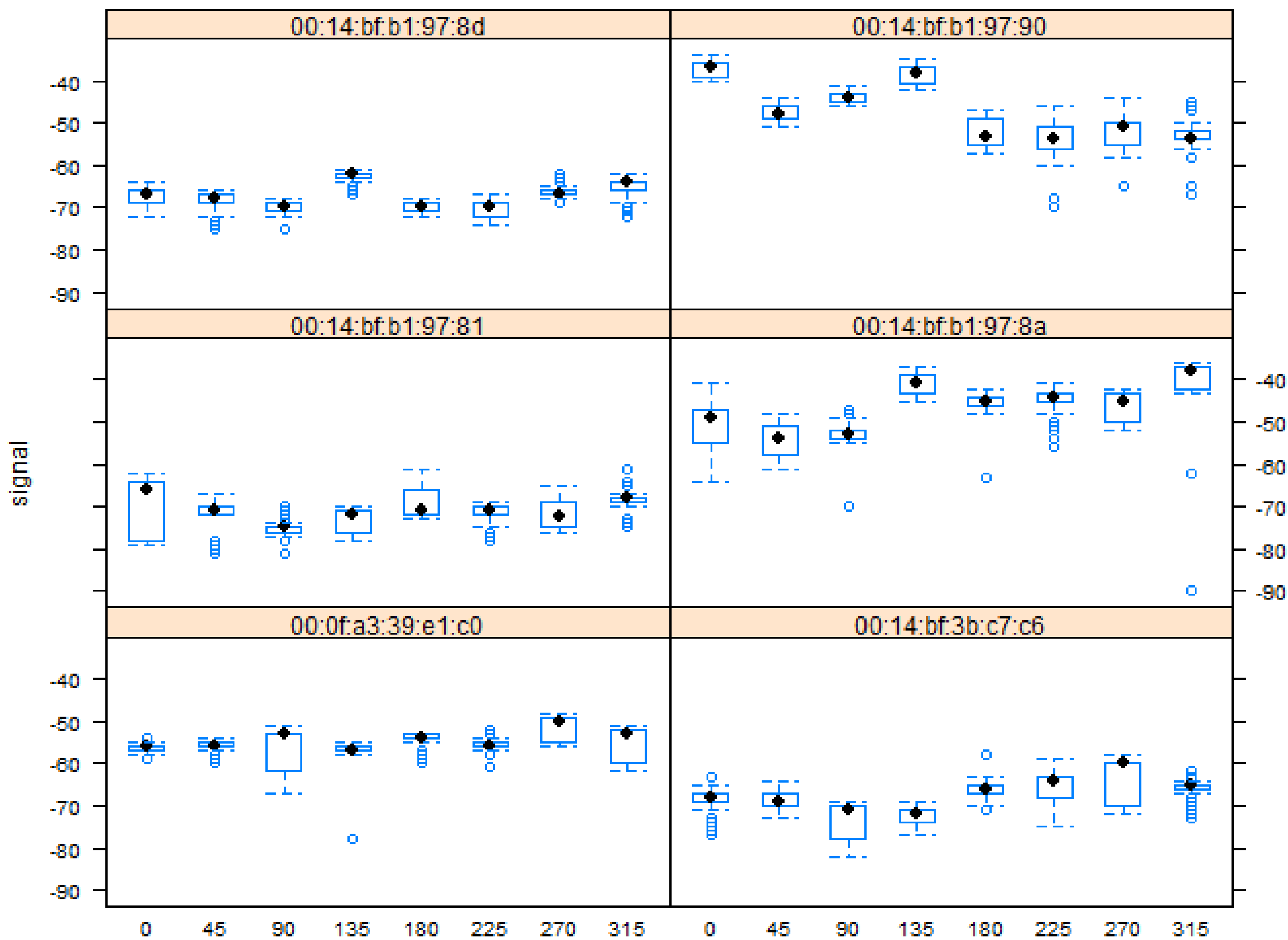


Final Data Prep

```r
offline <- readData(file_offline, offline_macs)
```

Signal Strength

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

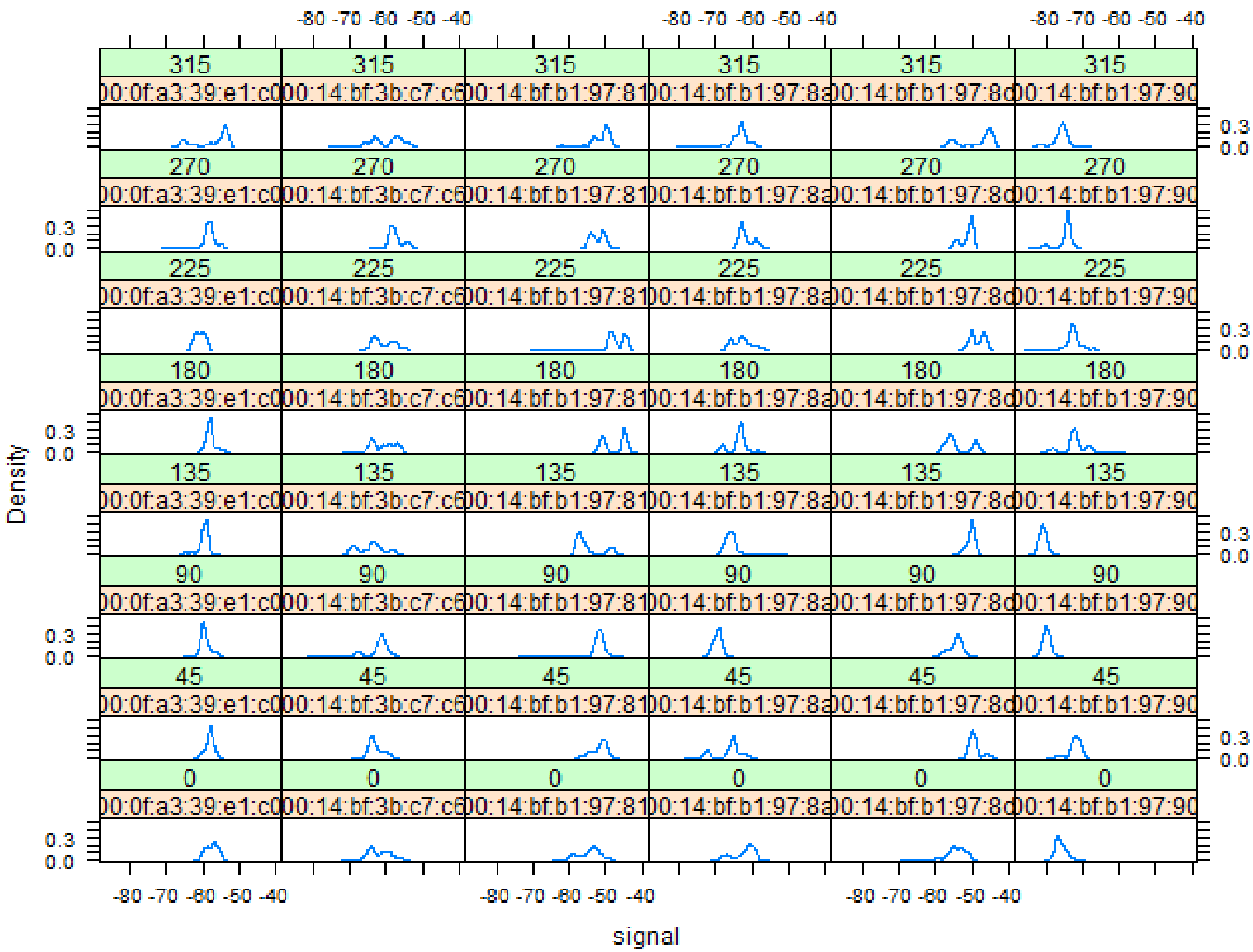



```
summary(offline$signal)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -98.00  -67.00  -59.00  -59.92  -53.00  -25.00
```

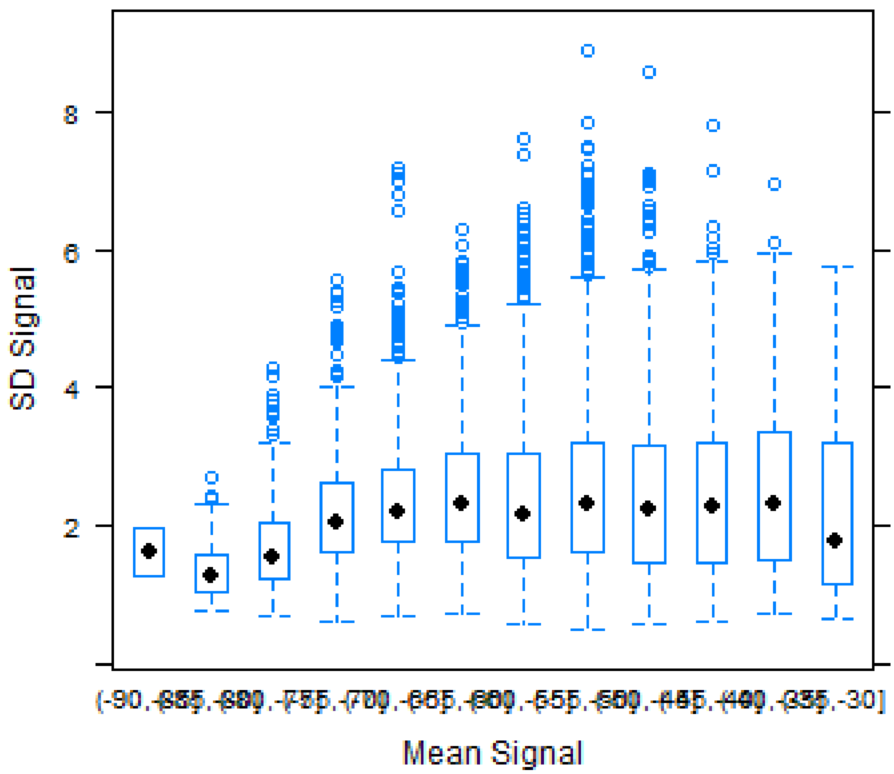
```
densityplot(~ signal | mac + factor(angle), data = offline,
            subset = posX == 24 & posY == 4 &
               mac != "00:0f:a3:39:dd:cd",
```

```
             bw = 0.5, plot.points = F)
```



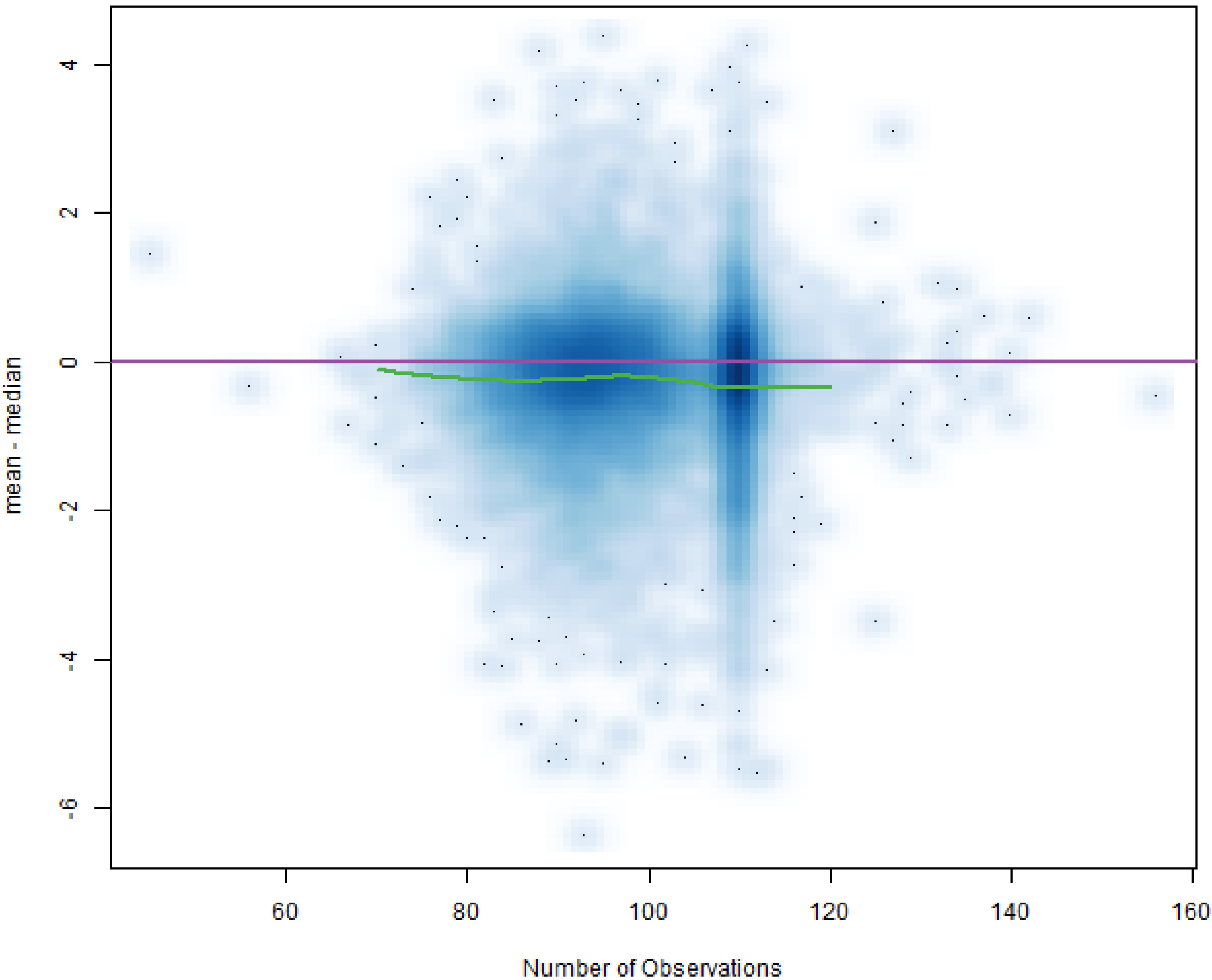

```
byLocAngleAP <- with(offline,
                     by(offline, list(posXY, angle, mac),
                        function(x) x))
signalSummary <-
   lapply(byLocAngleAP,
          function(oneLoc) {
             ans = oneLoc[1, ]
             ans$medSignal = median(oneLoc$signal)
             ans$avgSignal = mean(oneLoc$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)))
```

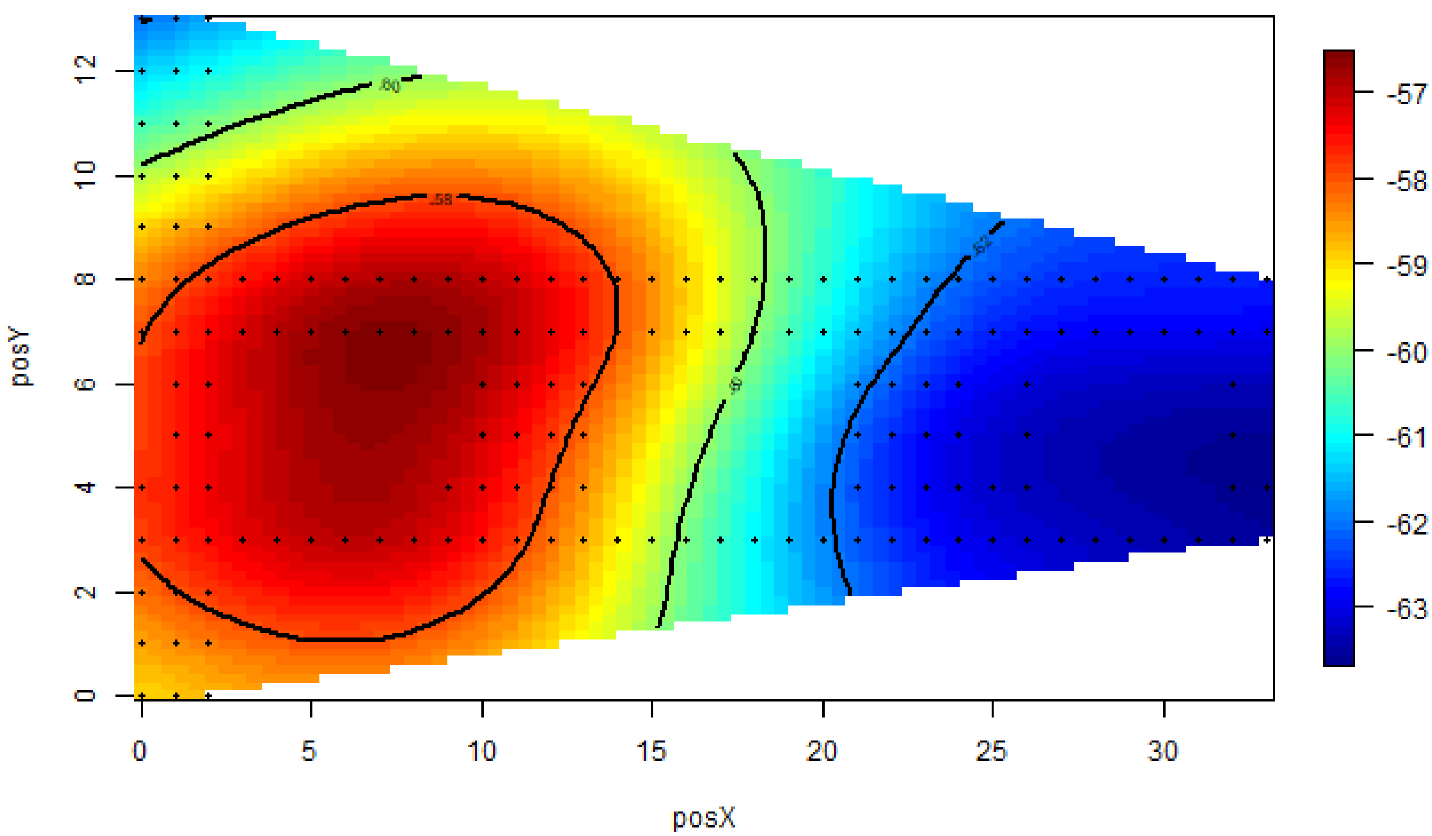
```r
lo.obj.pr <- predict(lo.obj, newdata = data.frame(num = (70:120)))
lines(x = 70:120, y = lo.obj.pr, col = "#4daf4a", lwd = 2)
```



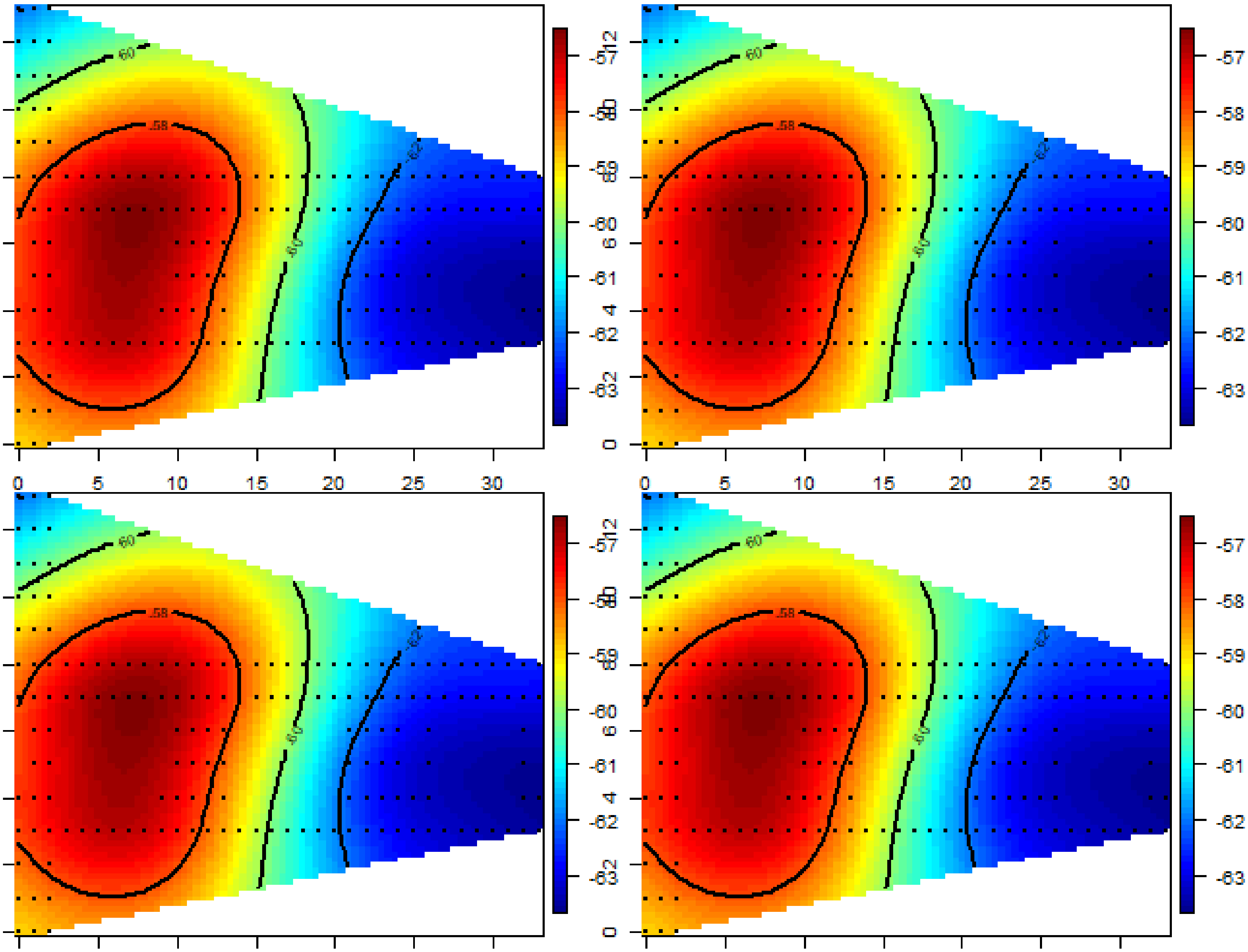
Signal and Distance

```r
subMacs <- names(sort(table(offline$mac), decreasing = T))[1:7]

surfaceSS <- function(data, mac, angle) {
```

```
  oneAPAngle <- subset(offlineSummary, mac == mac & angle == angle)

  smothSS <- Tps(oneAPAngle[, c("posX", "posY")],
                 oneAPAngle$avgSignal)

  vizSmooth <- predictSurface(smothSS)

  plot.surface(vizSmooth, type = "C")
  points(oneAPAngle$posX, oneAPAngle$posY, pch=19, cex = 0.5)
}

surfaceSS(offlineSummary, subMacs[5], 0)
```




```
parCur <- par(mfrow = c(2,2), mar = rep(1, 4))
mapply(surfaceSS, mac = subMacs[ rep(c(5, 1), each = 2)],
       data = list(data = offlineSummary))
```



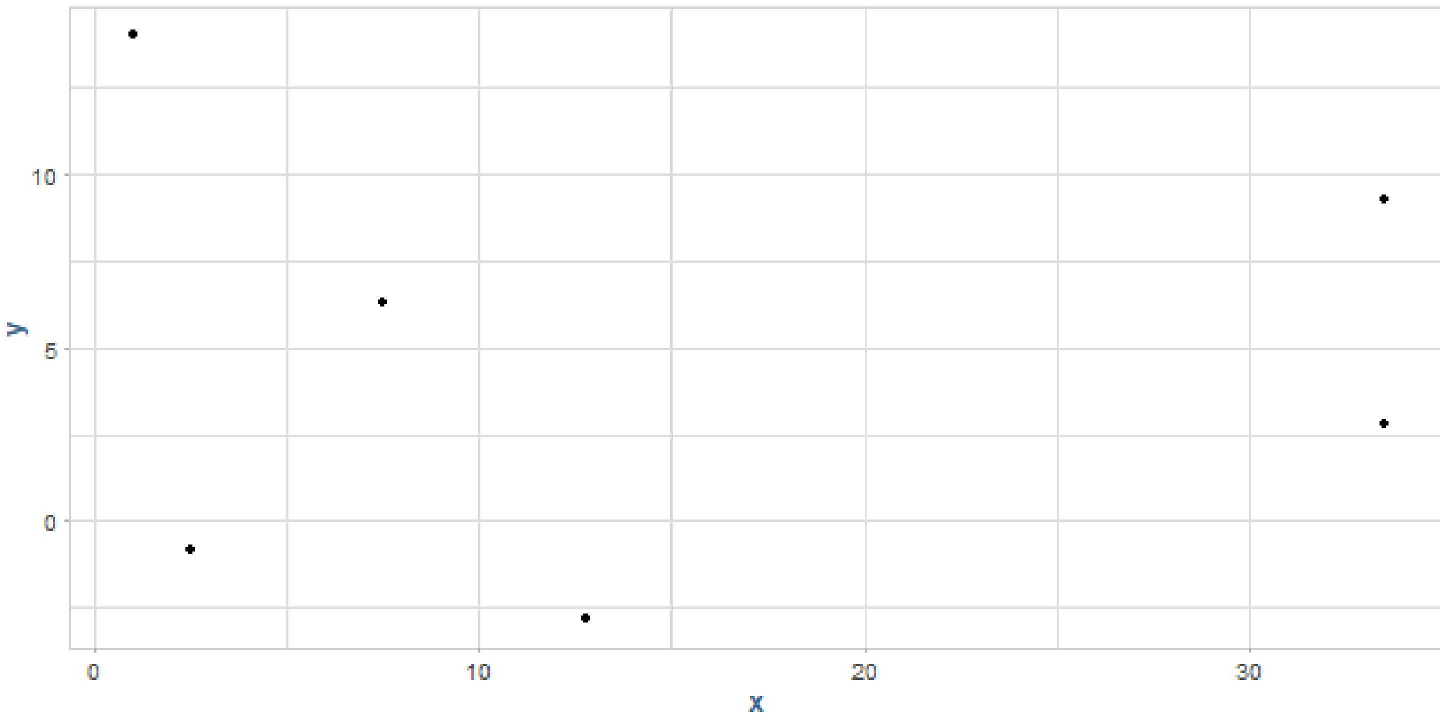

```
$`00:14:bf:b1:97:90`
NULL

$`00:14:bf:b1:97:90`
NULL

$`00:0f:a3:39:e1:c0`
NULL

$`00:0f:a3:39:e1:c0`
NULL
```

```r
par(parCur)
```

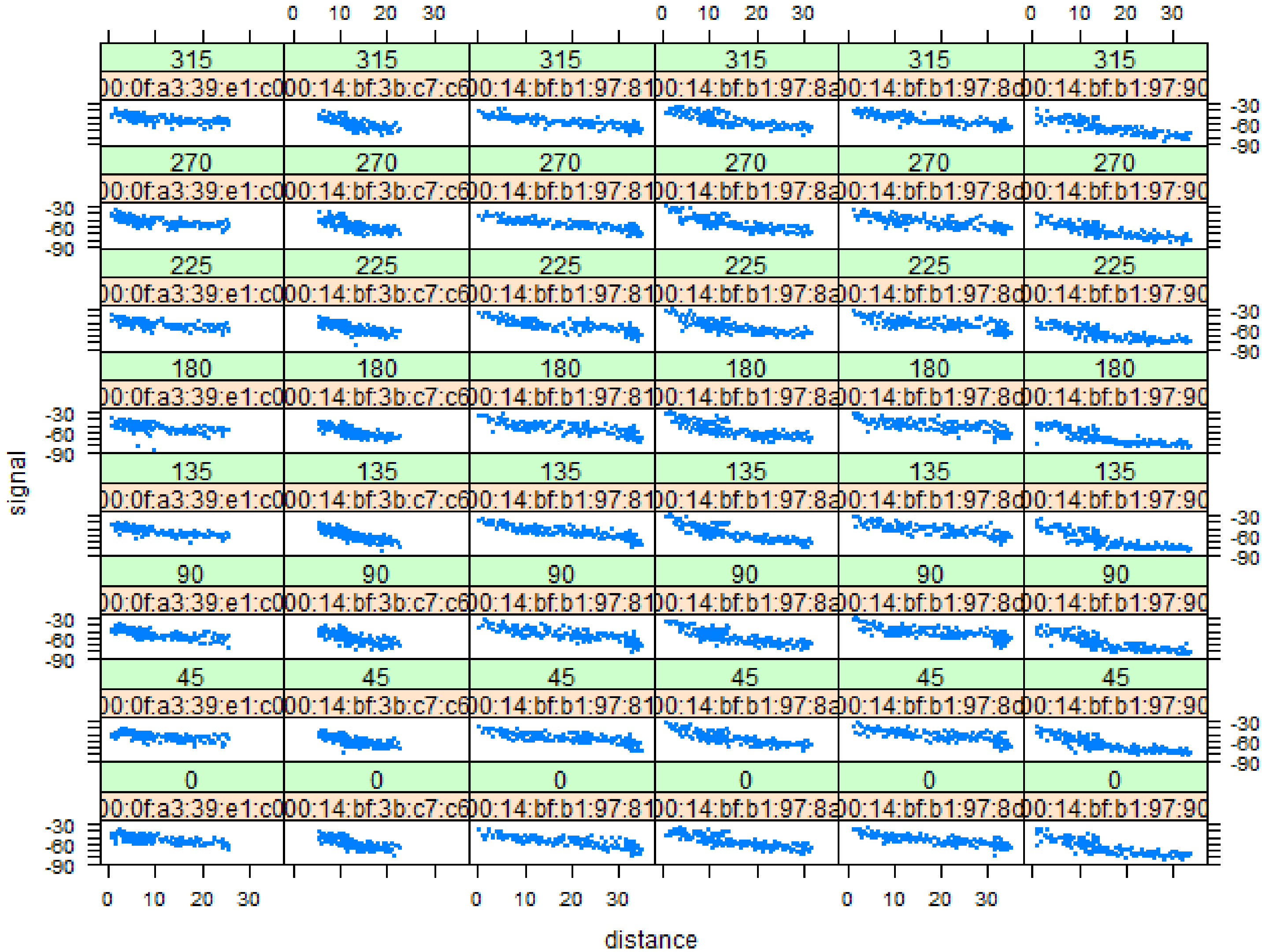
Exclude one of two similar access points

```
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 = T,
              dimnames = list(subMacs[ -2 ], c("x", "y") ))

ggplot(data.table(mac = rownames(AP), AP), aes(x, y)) +
   geom_point()
```




```
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")
```



Nearest Neighbor Methods to Predict Location

```
macs <- unique(offlineSummary$mac)

online <- readData(file_online, submacs = macs)

length(unique(online$posXY))
```

```
[1] 60
```

```
tabonlineXYA = table(online$posXY, online$angle)
tabonlineXYA[1:6, ]
```

```
               0  45  90 135 180 225 270 315
  0-0.05       0   0   0 593   0   0   0   0
  0.15-9.42    0   0 606   0   0   0   0   0
  0.31-11.09   0   0   0   0   0 573   0   0
  0.47-8.2   590   0   0   0   0   0   0   0
  0.78-10.94 586   0   0   0   0   0   0   0
  0.93-11.69   0   0   0   0 583   0   0   0
```

```
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)
```

```
[1] 60 11
```

```
names(onlineSummary)
```

```
 [1] "posXY"             "posX"              "posY"
 [4] "orientation"       "angle"             "00:0f:a3:39:e1:c0"
 [7] "00:14:bf:3b:c7:c6" "00:14:bf:b1:97:81" "00:14:bf:b1:97:8a"
[10] "00:14:bf:b1:97:8d" "00:14:bf:b1:97:90"
```

Choice of Orientation

```
reshapeSS <- function(data,
                      varSignal = "signal",
                      keepVars = c("posXY", "posX", "posY"),
                      sampleAngle = F) {

   if(sampleAngle)
      data <- data[angle == sample(data$angle, size = 1), ]

   byLocation <- with(data,
                 by(data, list(posXY),
                    function(x) {
```

```
                        ans <- x[1, ..keepVars]
                        avgSS <- tapply(x$signal, x$mac, mean)
                        y = matrix(avgSS, nrow = 1, dimnames = list(ans$posXY, names(avgSS)))
                        cbind(ans, y)
                     }))

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

   col_names <- colnames(newDataSS)
   to_change <- !(col_names %in% keepVars)

   n_cols <- length(col_names)
   start <- length(keepVars)

   colnames(newDataSS)[to_change] <- sapply(col_names[to_change], function(col) {
      n <- nchar(col)
      substr(col, n - 2, n)
   })

   newDataSS[, start:ncol(newDataSS)] <- round(newDataSS[, start:ncol(newDataSS)])

   return(newDataSS)
}

selectTrain <- function(angleNewObs, signals, m) {

   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

   signals <- signals[angle %in% angles, ]

   reshapeSS(signals, varSignal = "avgSignal")
}

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)

      sigValues <- suppressWarnings({ as.numeric(newSignals[i, ]) })

      closeXY[[i]] <- findNN(newSignal = sigValues,
                             trainSS)
   }

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

   return(estXY)
}

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

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

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

actualXY <- onlineSummary[, c("posX", "posY")]
sapply(list(estXYk1, estXYk3), calcError, actualXY)
```

```
[1] 420.7603 388.1070
```

```
v <- 11;

permuteLocs <- sample(unique(offlineSummary$posXY))
permuteLocs <- matrix(permuteLocs, ncol = v,
                      nrow = floor(length(permuteLocs)/v))
```

```
Warning in matrix(permuteLocs, ncol = v, nrow = floor(length(permuteLocs)/v)):
data length [166] is not a sub-multiple or multiple of the number of rows [15]
```

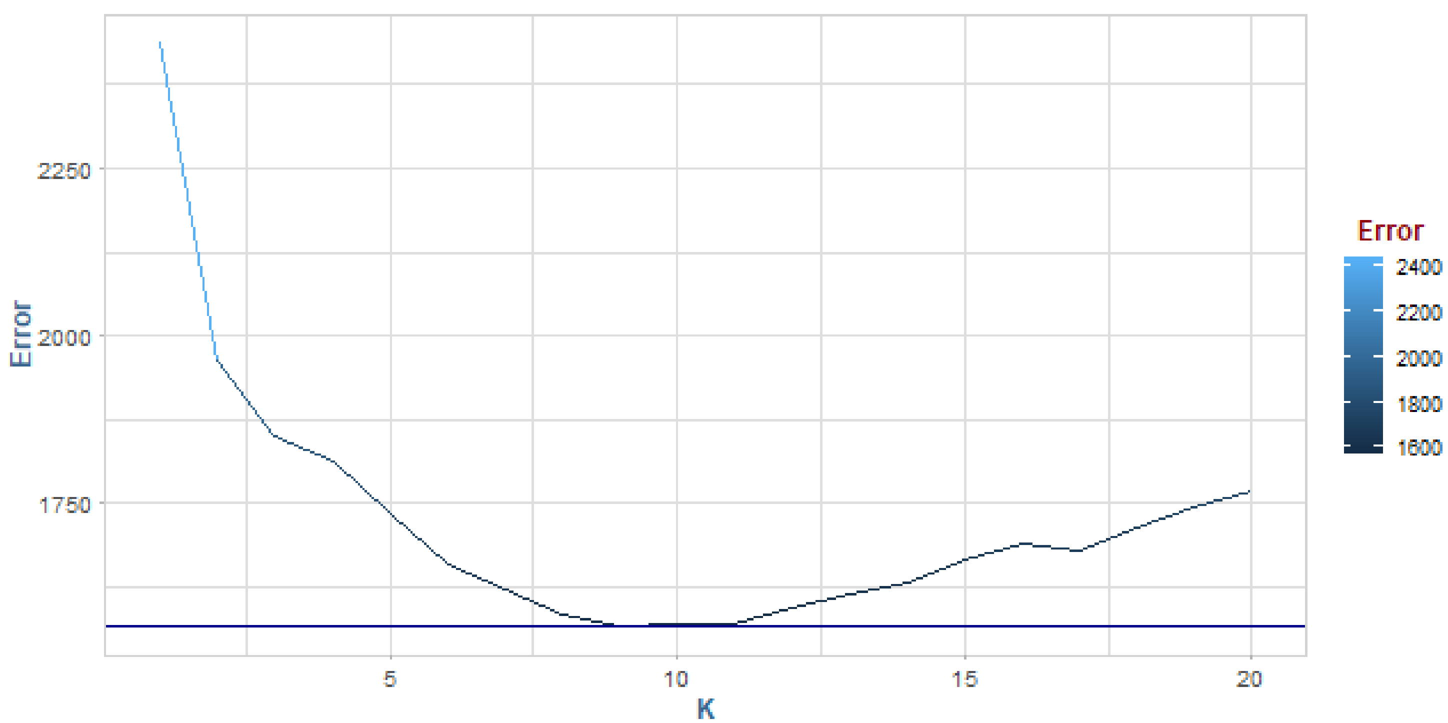
```
onlineFold <- subset(offlineSummary, posXY %in% permuteLocs[, 1])

onlineCVSummary <- reshapeSS(offline,
                             keepVars = c("posXY", "posX", "posY", "angle"),
                             sampleAngle = T)

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)
```

```
[1] 234192
```

```
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)
  }
}
```

```
k_values <- data.table(K = 1:K, Error = err, Min = which.min(err))

ggplot(k_values, aes(K, Error)) +
  geom_line(aes(col = Error)) +
  geom_hline(data = k_values[K == Min], aes(yintercept = Error), col = "darkblue") +
  labs("K-NN Training Error")
```

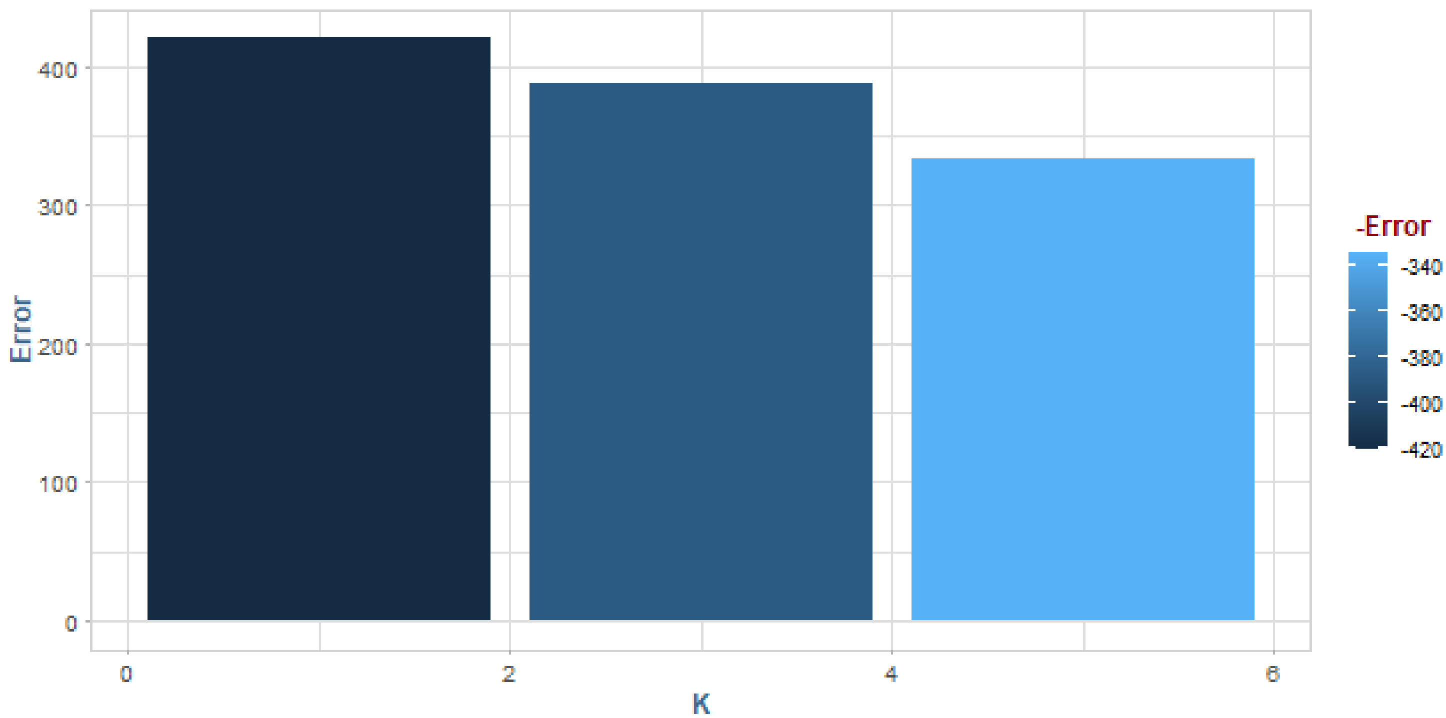



```
estXYk5 <- predXY(newSignals = onlineSummary[, 6:11],
                  newAngles = onlineSummary[, 4],
                  offlineSummary, numAngles = 3, k = 5)
```

```
calcError(estXYk5, actualXY)
```

```
[1] 334.3683
```

```
error_table <- data.table(K = c(1, 3, 5),
                          Error = c(calcError(estXYk1, actualXY),
                                    calcError(estXYk3, actualXY),
                                    calcError(estXYk5, actualXY)))

ggplot(error_table, aes(K, Error, fill = -Error)) +
   geom_bar(stat = "identity")
```



Further Analysis

1.)

Write the code to read the raw training data into the data structure in the first approach described in section 1.2. That is, the data structure is a data frame with a column for each MAC address that detected a signal. For the column name, use the last two characters of the MAC address, or some other unique identifier.

```
file <- file_offline

lines <- read_lines(file)

valid_lines <- lines[ validLines(lines) ]
```

```r
parseLine <- function(line) {

   pairs <- strsplit(line, ";")
   tokens <- lapply(pairs, function(x)strsplit(x, "="))

   mat <- matrix(unlist(tokens), ncol = 2, byrow = T)

   colnames <- mat[, 1]
   values <- mat[, 2]

   dt <- data.table(t(values))
   colnames(dt) <- colnames

   dt
}

rows <- lapply(valid_lines, parseLine)

offline_alt <- rbindlist(rows, fill = T)

head(offline_alt)
```

```
                t                id         pos degree 00:14:bf:b1:97:8a
1: 1139643118358 00:02:2D:21:0F:33 0.0,0.0,0.0    0.0  -38,2437000000,3
2: 1139643118744 00:02:2D:21:0F:33 0.0,0.0,0.0    0.0  -38,2437000000,3
3: 1139643119002 00:02:2D:21:0F:33 0.0,0.0,0.0    0.0  -38,2437000000,3
4: 1139643119263 00:02:2D:21:0F:33 0.0,0.0,0.0    0.0  -38,2437000000,3
5: 1139643119538 00:02:2D:21:0F:33 0.0,0.0,0.0    0.0  -46,2437000000,3
6: 1139643119818 00:02:2D:21:0F:33 0.0,0.0,0.0    0.0  -37,2437000000,3
   00:14:bf:b1:97:90 00:0f:a3:39:e1:c0 00:14:bf:b1:97:8d 00:14:bf:b1:97:81
1:  -56,2427000000,3  -53,2462000000,3  -65,2442000000,3  -65,2422000000,3
2:  -56,2427000000,3  -54,2462000000,3  -70,2442000000,3  -66,2422000000,3
3:  -57,2427000000,3  -54,2462000000,3  -70,2442000000,3  -66,2422000000,3
4:  -52,2427000000,3  -54,2462000000,3  -74,2442000000,3  -64,2422000000,3
5:  -57,2427000000,3  -55,2462000000,3              <NA>  -66,2422000000,3
6:              <NA>  -54,2462000000,3  -67,2442000000,3  -65,2422000000,3
   00:14:bf:3b:c7:c6 00:0f:a3:39:dd:cd 00:0f:a3:39:e0:4b 00:0f:a3:39:e2:10
1:  -66,2432000000,3  -75,2412000000,3  -78,2462000000,3  -87,2437000000,3
2:  -67,2432000000,3  -73,2412000000,3  -79,2462000000,3  -83,2437000000,3
3:  -69,2432000000,3  -65,2412000000,3  -78,2462000000,3  -83,2437000000,3
4:  -68,2432000000,3  -78,2412000000,3  -78,2462000000,3  -83,2437000000,3
5:  -67,2432000000,3  -66,2412000000,3  -80,2462000000,3  -83,2437000000,3
6:  -67,2432000000,3  -67,2412000000,3  -79,2462000000,3  -89,2437000000,3
   02:64:fb:68:52:e6 02:00:42:55:31:00 00:14:bf:3b:c7:c6 00:0f:a3:39:e0:4b
```

```
1:  -88,2447000000,1  -84,2457000000,1              <NA>              <NA>
2:              <NA>  -85,2457000000,1              <NA>              <NA>
3:  -90,2447000000,1              <NA>              <NA>              <NA>
4:  -87,2447000000,1  -84,2457000000,1              <NA>              <NA>
5:              <NA>  -87,2457000000,1              <NA>              <NA>
6:  -90,2447000000,1  -86,2457000000,1              <NA>              <NA>
   00:0f:a3:39:e0:4b 00:0f:a3:39:e0:4b 00:0f:a3:39:dd:cd 00:14:bf:b1:97:90
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:14:bf:b1:97:8d 00:0f:a3:39:e0:4b 00:0f:a3:39:e0:4b 00:14:bf:b1:97:8a
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:04:0e:5c:23:fc 00:0f:a3:39:e2:10 00:04:0e:5c:23:fc 00:04:0e:5c:23:fc
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:04:0e:5c:23:fc 00:0f:a3:39:e2:10 00:0f:a3:39:e2:10 00:0f:a3:39:e2:10
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:04:0e:5c:23:fc 00:04:0e:5c:23:fc 02:64:fb:68:52:e6 00:0f:a3:39:e1:c0
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:14:bf:b1:97:81 00:14:bf:b1:97:8d 00:14:bf:3b:c7:c6 02:64:fb:68:52:e6
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
```

```
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:14:bf:3b:c7:c6 00:0f:a3:39:dd:cd 00:14:bf:b1:97:8a 00:0f:a3:39:e2:10
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   02:64:fb:68:52:e6 00:14:bf:b1:97:81 00:14:bf:b1:97:81 00:30:bd:f8:7f:c5
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:30:bd:f8:7f:c5 00:04:0e:5c:23:fc 00:0f:a3:39:e0:4b 00:30:bd:f8:7f:c5
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:0f:a3:39:e0:4b 00:30:bd:f8:7f:c5 00:14:bf:b1:97:8a 00:14:bf:b1:97:90
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:14:bf:3b:c7:c6 00:14:bf:b1:97:81 00:14:bf:b1:97:81 00:14:bf:3b:c7:c6
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:0f:a3:39:e2:10 00:14:bf:b1:97:8a 00:e0:63:82:8b:a9 00:e0:63:82:8b:a9
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
```

```
   00:e0:63:82:8b:a9 00:0f:a3:39:e1:c0 02:37:fd:3b:54:b5 02:37:fd:3b:54:b5
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   02:00:42:55:31:00 00:0f:a3:39:dd:cd 02:37:fd:3b:54:b5 00:0f:a3:39:dd:cd
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   02:2e:58:22:f1:ac 02:2e:58:22:f1:ac 02:00:42:55:31:00 02:42:1c:4e:b5:c0
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:14:bf:b1:97:8a 00:14:bf:b1:97:90 02:0a:3d:06:94:88 00:0f:a3:39:dd:cd
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   02:5c:e0:50:49:de 02:5c:e0:50:49:de 00:0f:a3:39:dd:cd 00:0f:a3:39:dd:cd
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   02:4f:99:43:30:cd 02:5c:e0:50:49:de 02:b7:00:bb:a9:35 02:b7:00:bb:a9:35
1:              <NA>              <NA>              <NA>              <NA>
2:              <NA>              <NA>              <NA>              <NA>
3:              <NA>              <NA>              <NA>              <NA>
4:              <NA>              <NA>              <NA>              <NA>
5:              <NA>              <NA>              <NA>              <NA>
6:              <NA>              <NA>              <NA>              <NA>
   00:14:bf:b1:97:90
1:              <NA>
2:              <NA>
```

```
3:              <NA>
4:              <NA>
5:              <NA>
6:              <NA>
```

2.)

Compare the size of two data structures: the data frame created in Section 1.2 and the data frame created in the previous problem.

- Which uses less memory?

```
object.size(offline)
```

```
69549552 bytes
```

```
object.size(offline_alt)
```

```
114691968 bytes
```

- What are the dimensions of each?

```
dim(offline)
```

```
[1] 914951      9
```

```
dim(offline_alt)
```

```
[1] 146080     90
```

- How might this change with different numbers of devices in the building?

Columns will expand out drastically with more devices.

- Different number of signals from the less commonly detected devices?

The more devices we have the more the columns in the alternative approach will be filled with NAs, thus bloating the object size significantly.

3.)

Compare the total time it takes to read the raw data and create the data frame, for the two approaches described in Section 1.2.

```
file <- file_offline

fn_read_data <- function(file, processFun, sample = F, samp.size = 100 ) {
   start_time <- Sys.time()

   lines <- read_lines(file)
   valid_lines <- lines[ validLines(lines) ]
```

```r
   n <- length(valid_lines)

   if(sample) {
      samp <- sample(n, samp.size, replace = F)

      valid_lines <- valid_lines[samp]
   }

   res <- processFun(valid_lines) # throw away

   return(Sys.time() - start_time)
}

fn_read_data(file_offline, processLine)
```

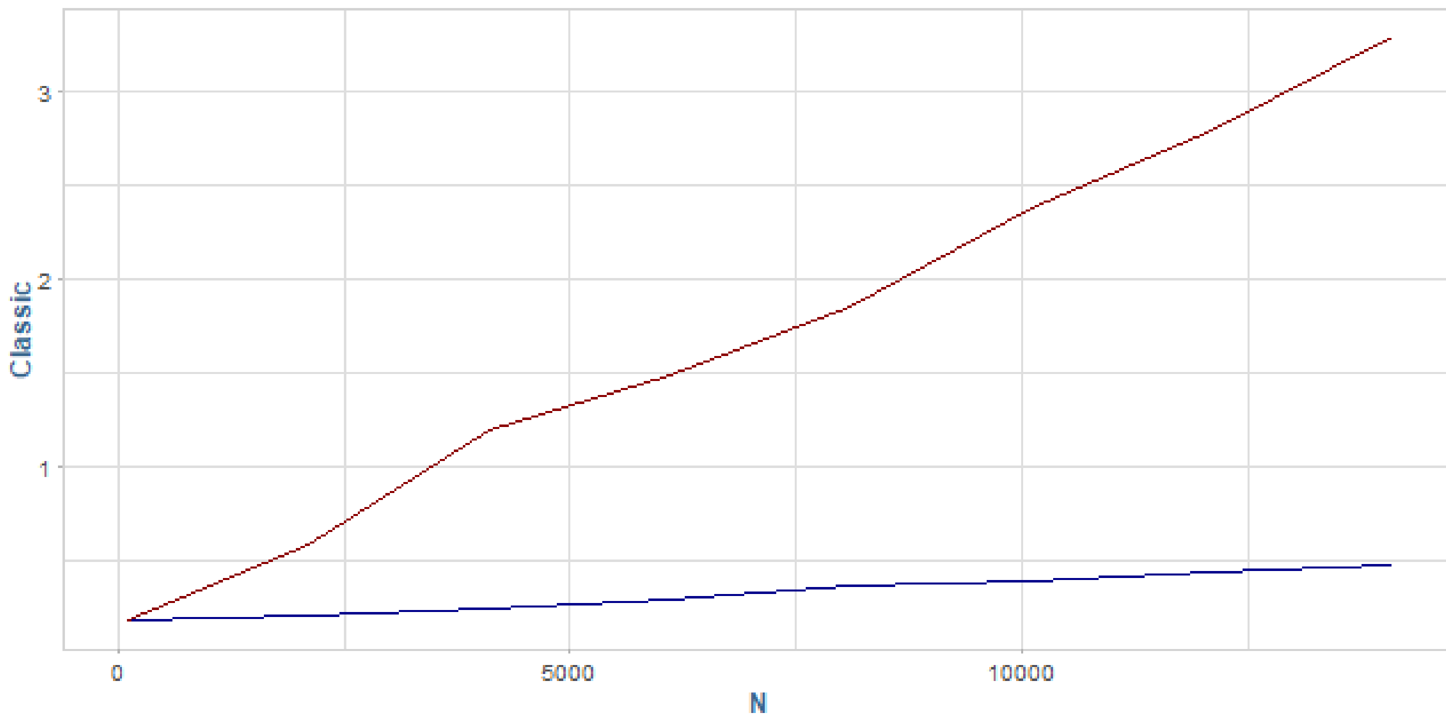
```
Time difference of 3.322008 secs
```

```r
fn_read_data(file_offline, parseLine)
```

```
Time difference of 38.95947 secs
```

Do this for different size subsets of the data (chosen at random) and draw a curve of the time against input size for each of the approaches.

```r
sample_sizes <- seq( from = 100, to = 15000, by = 2000)

times_classic <- sapply(sample_sizes, function(samp) {
   fn_read_data(file_offline, processLine, sample = T, samp.size = samp)
})

times_alt <- sapply(sample_sizes, function(samp) {
   fn_read_data(file_offline, parseLine, sample = T, samp.size = samp)
})

results <- data.table(N = sample_sizes, Classic = times_classic, alt = times_alt)

ggplot(results) +
   geom_line(aes(N, Classic), col = "darkblue") +
   geom_line(aes(N, alt), col = "darkred")
```



Also, comment on the memory and speed for the two approaches.

The alt approach is substantially worse in all aspects.

4.)

Examine the time variable in the offline data. Any chance over time in the characteristics of the signal caused by, e.g., reduced battery power in the measuring devices as time goes by, or measurments taken on different days may be made by different people with different levels of accuracy. Also, examination of time can give insights into how the experiment was carried out.

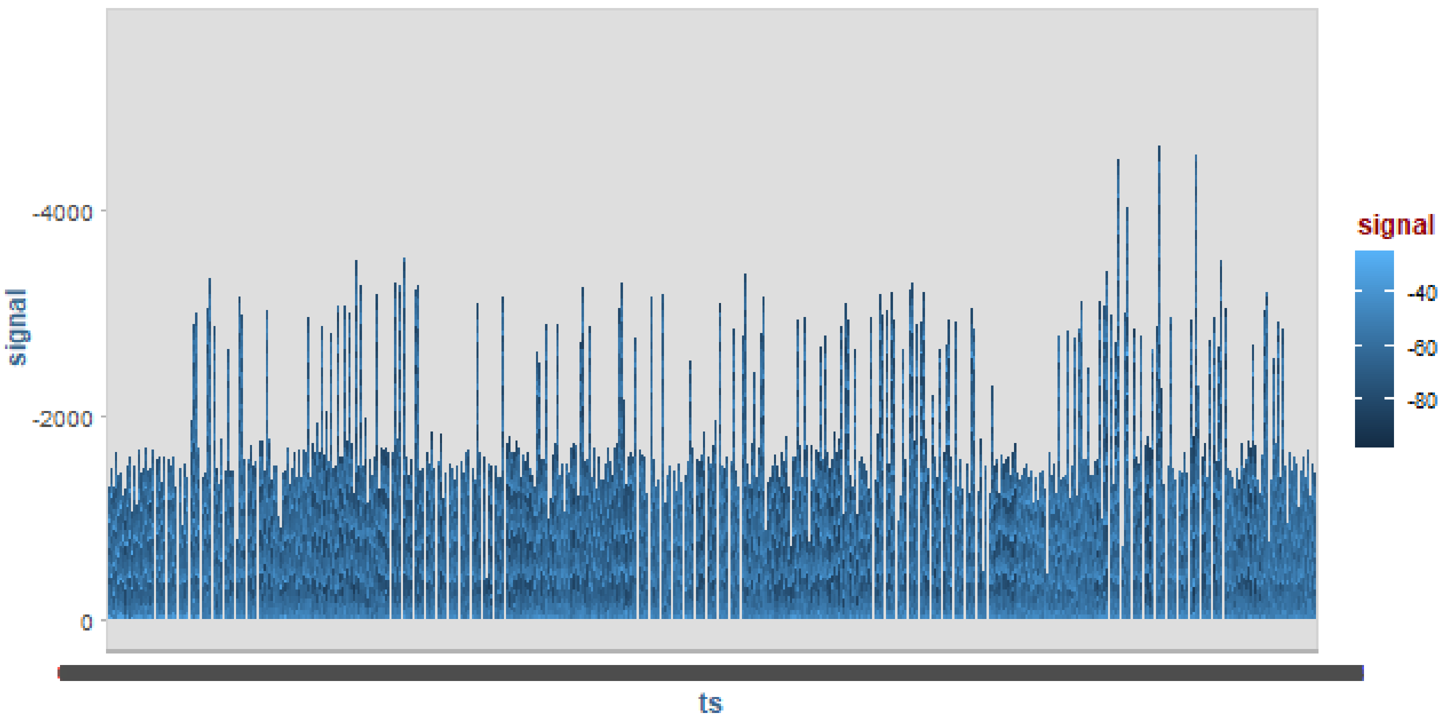
Were the positions close to each other measured at similar times?

Do you see any change in the signal strength variable or mean over time?

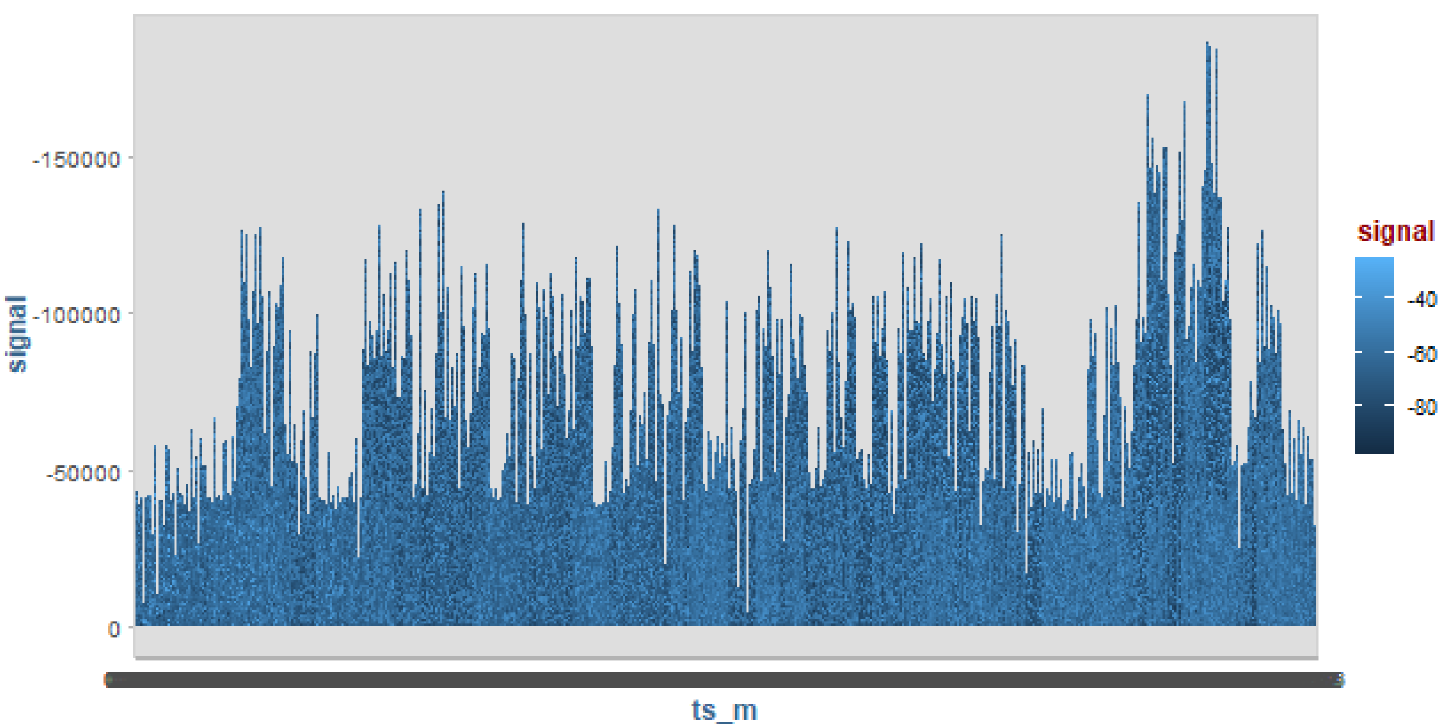
Try controlling for other variables that might affect this relationship.

```
offline_copy <- offline
offline_copy$ts <- strftime(offline_copy$time, format="%H:%M:%S")
offline_copy$ts_m <- strftime(offline_copy$time, format="%H:%M")
offline_copy$ts_h <- strftime(offline_copy$time, format="%H")
```

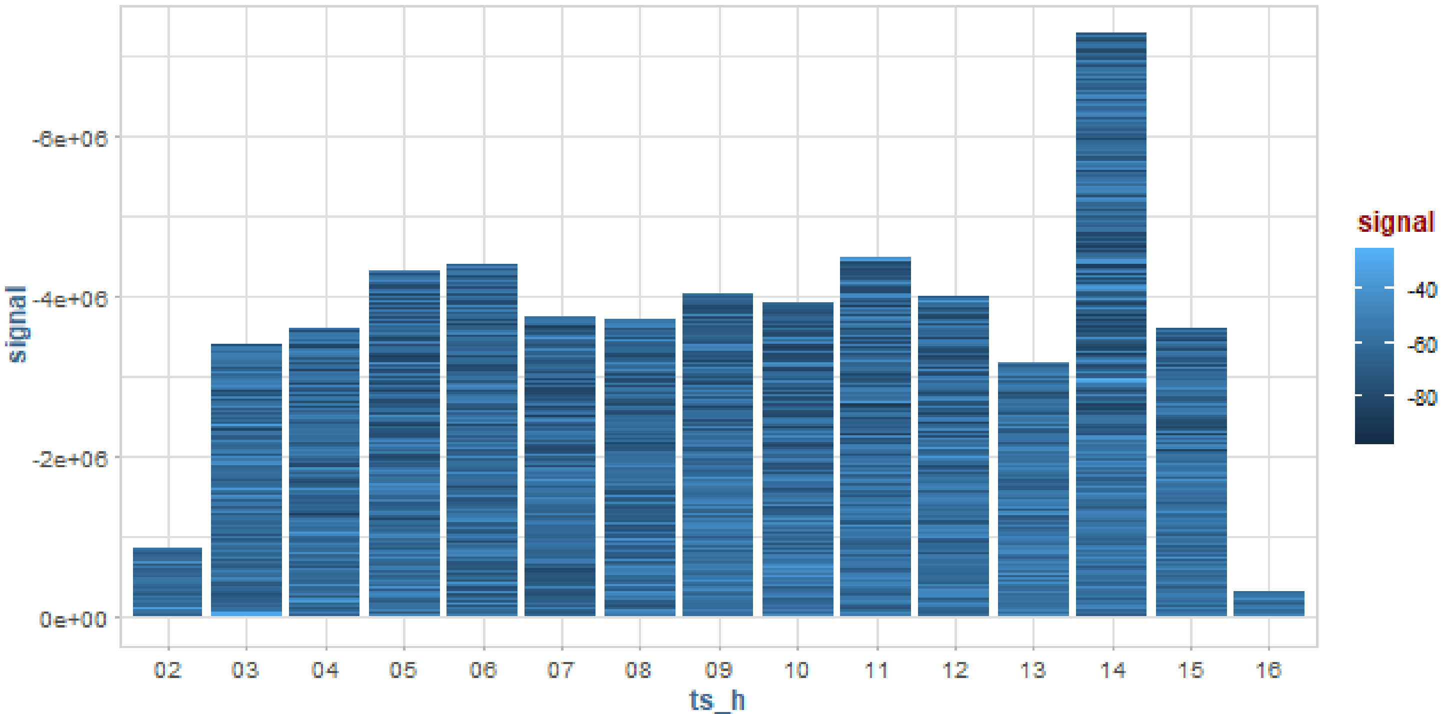
```
ggplot(offline_copy, aes(ts, signal, fill = signal)) +
   geom_bar(stat = "identity") +
   scale_y_reverse()
```




```r
ggplot(offline_copy, aes(ts_m, signal, fill = signal)) +
  geom_bar(stat = "identity") +
  scale_y_reverse()
```



```
ggplot(offline_copy, aes(ts_h, signal, fill = signal)) +
  geom_bar(stat = "identity") +
  scale_y_reverse()
```



5.)

Write the readData function described in Section 1.3.4. The arguments to this function are the file name and the MAC address to retain, subKMacs. Determine whether these parameters should have default values or not. The return value is data frame described in section 1.3. Use the findGlobals() function available in *codetools* to check that the function is not relying on any global variables.

```
readData <- function(file, submacs = macs) {

  lines <- read_lines(file)

  valid_lines <- lines[ validLines(lines) ]

  processed_lines <- lapply(valid_lines, processLine)

  data <- as.data.table(do.call("rbind", processed_lines),
                        stringsAsFactors = F)

  names(data) <- c("time", "scanMac", "posX", "posY", "posZ",
                   "orientation", "mac", "signal",
```

```
                    "channel", "type")

   numVars <- c("time", "posX", "posY", "posZ",
                "orientation", "signal")

   data[, (numVars) := lapply(.SD, as.numeric), .SDcols = numVars]

   data <- data[ data$type == 3, ]
   data[, type := NULL]

   data[, rawTime := time]
   data[, time := time/1000]
   class(data$time) = c("POSIXt", "POSIXct")

   # drop scanMac & posZ
   data[, `:=`(scanMac = NULL, posZ = NULL)]

   data$angle = roundOrientation(data$orientation)

   data$channel = NULL

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

   return(data[mac %in% submacs])
}
```

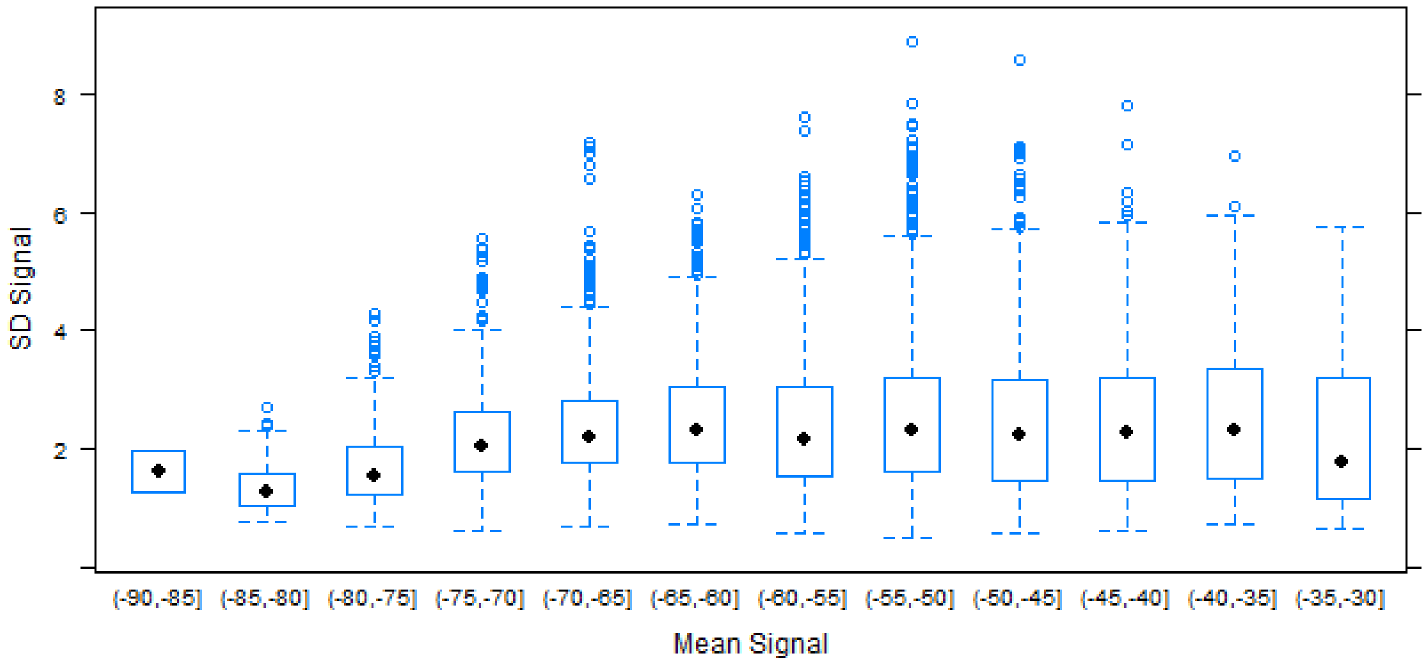
6.)

In section 1.4.1 we calculated measures of center and location for the signal strenghts at each location x angle x access point combination (see fig 1.9 for example).

Another possible summary statistic we can calculate is the Kolmogorov-Smirnov test-statistic for normality. If the signal strenghts are roughtly normal, then we expect the p-values to have a uniform distribution. This leads to about 5% of the p-values for the 8000 tests to fall below 0.05.

```
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
```

```
        })

offlineSigSummary <- 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")
```




```
by_pos_angle_mac <- offlineSigSummary[, .(AvgSignal = mean(signal)), by = list(mac, angle, posX

with(by_pos_angle_mac, ks.test(AvgSignal, "pnorm")) # Kolmogorov-Smirnov test-statistic
```

```
Warning in ks.test(AvgSignal, "pnorm"): ties should not be present for the
Kolmogorov-Smirnov test


    One-sample Kolmogorov-Smirnov test

data:  AvgSignal
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
by_pos_angle <- offlineSigSummary[, .(AvgSignal = mean(signal)), by = list(angle, posXY)]

with(by_pos_angle, ks.test(AvgSignal, "pnorm")) # Kolmogorov-Smirnov test-statistic
```

```
Warning in ks.test(AvgSignal, "pnorm"): ties should not be present for the
Kolmogorov-Smirnov test


    One-sample Kolmogorov-Smirnov test

data:  AvgSignal
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
by_mac <- offline[, .(AvgSignal = mean(signal)), by = list(mac)]

with(by_mac, ks.test(AvgSignal, "pnorm")) # Kolmogorov-Smirnov test-statistic
```

```

    One-sample Kolmogorov-Smirnov test

data:  AvgSignal
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```

7.)

Write the surfaceSS() function that creates plots such as those in Figure 1.10. This function takes 3 arguments: data for the offline summary data frame, mafc and angle. The parameters mac and angle are used to specify which MAC address and angle are to be selected from the data for smoothing and plotting.

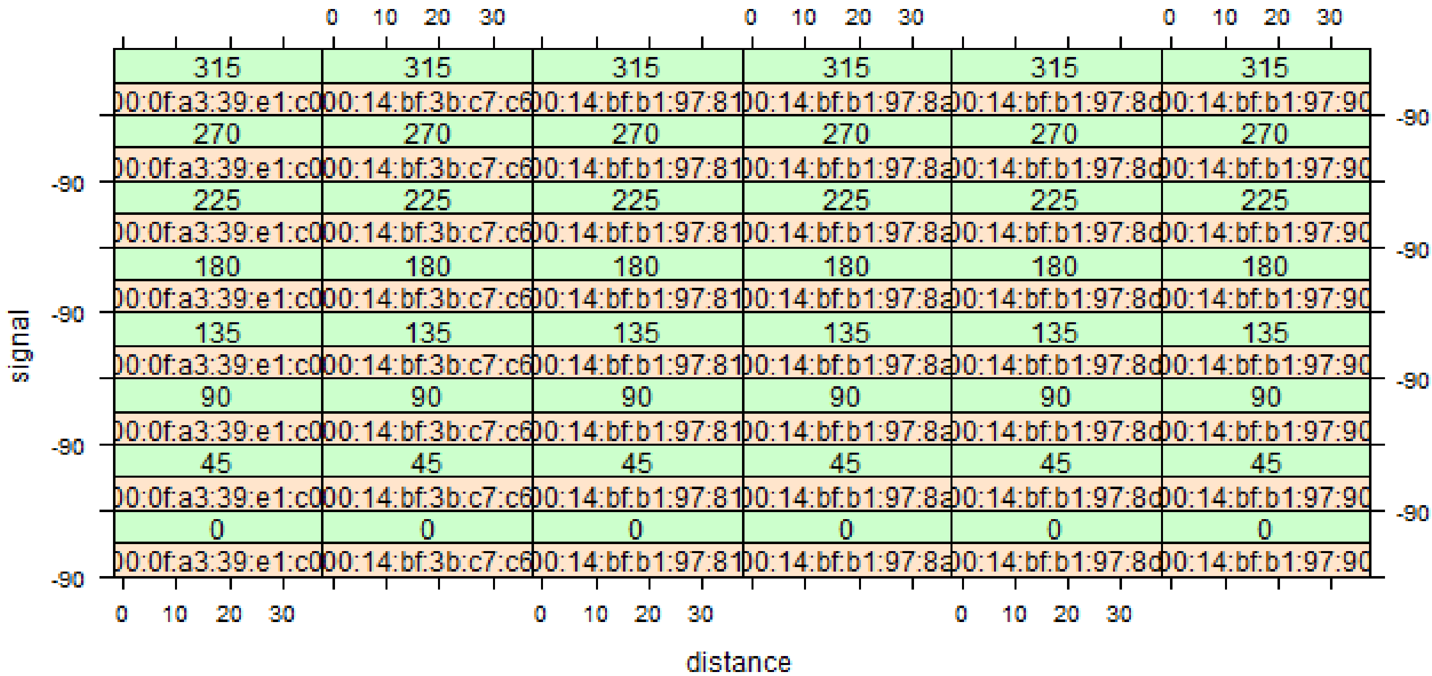
```
reshapeSS <- function(data,
                      varSignal = "signal",
                      keepVars = c("posXY", "posX", "posY"),
                      sampleAngle = F) {

   if(sampleAngle)
      data <- data[angle == sample(data$angle, size = 1), ]

   byLocation <- with(data,
                 by(data, list(posXY),
                    function(x) {
                       ans <- x[1, ..keepVars]
                       avgSS <- tapply(x$signal, x$mac, mean)
                       y = matrix(avgSS, nrow = 1, dimnames = list(ans$posXY, names(avgSS)))
                       cbind(ans, y)
```

```
                  }))

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

  col_names <- colnames(newDataSS)
  to_change <- !(col_names %in% keepVars)

  n_cols <- length(col_names)
  start <- length(keepVars)

  colnames(newDataSS)[to_change] <- sapply(col_names[to_change], function(col) {
    n <- nchar(col)
    substr(col, n - 2, n)
  })

  newDataSS[, start:ncol(newDataSS)] <- round(newDataSS[, start:ncol(newDataSS)])

  return(newDataSS)
}
```

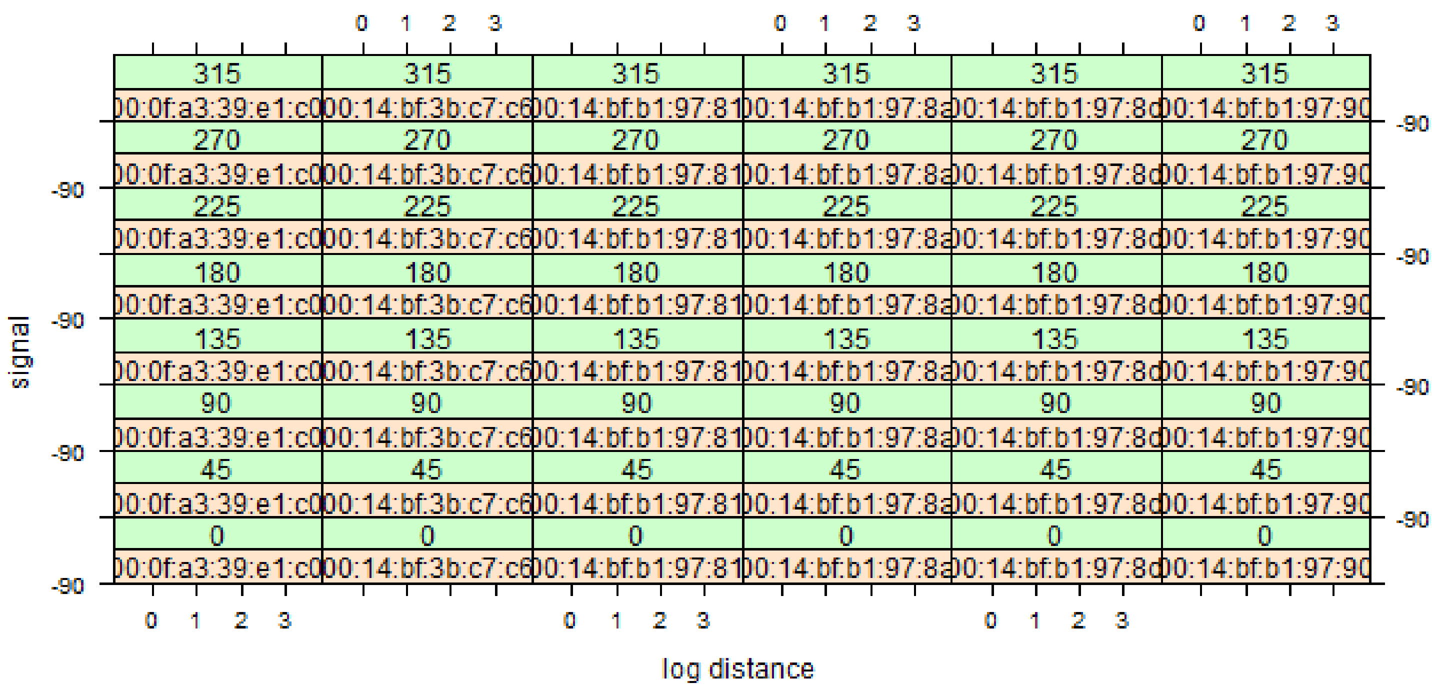
8.)

Consider the scatter plots in Figure 1.11. There appear to be curvature in the signal strength-distance relationship. Does a log transformation improve this relationship, i.e., make it linear?

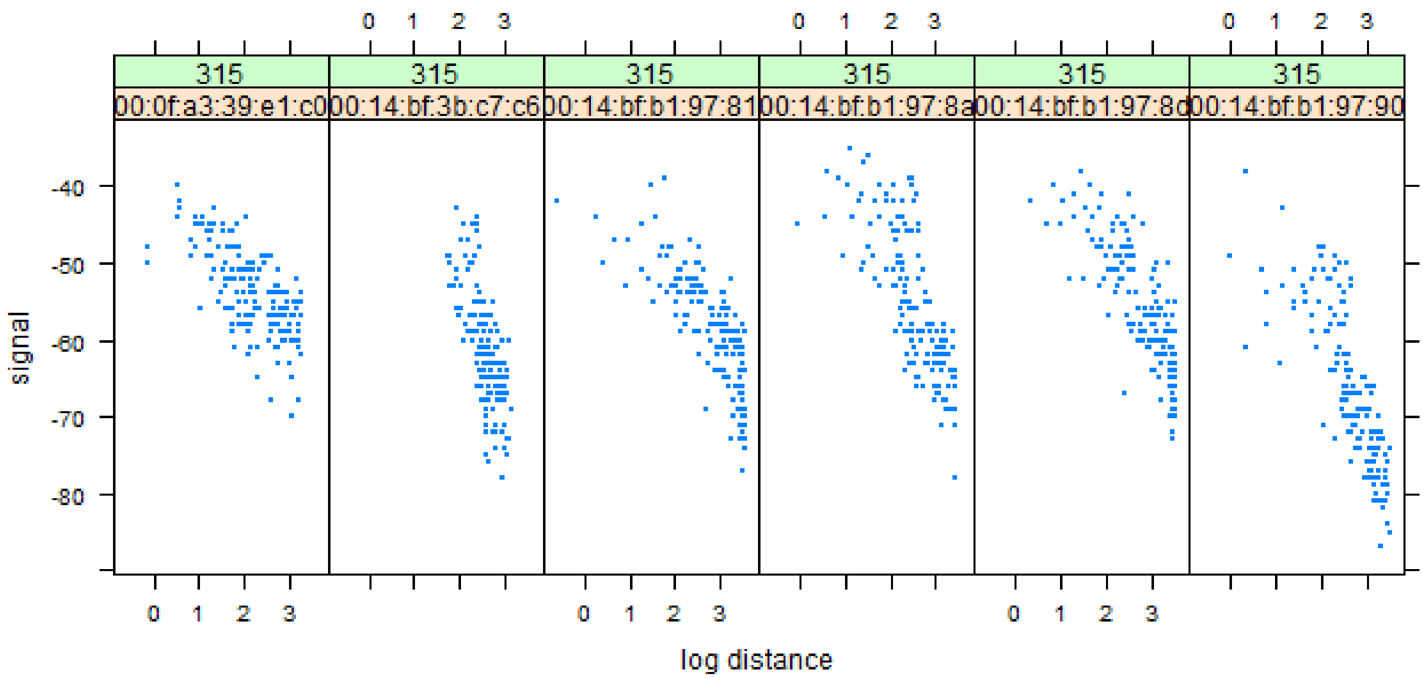
```
offlineSummary$dist <- sqrt(diffs[, 1]^2 + diffs[, 2]^2)
offlineSummary$log_dist <- log(offlineSummary$dist)

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



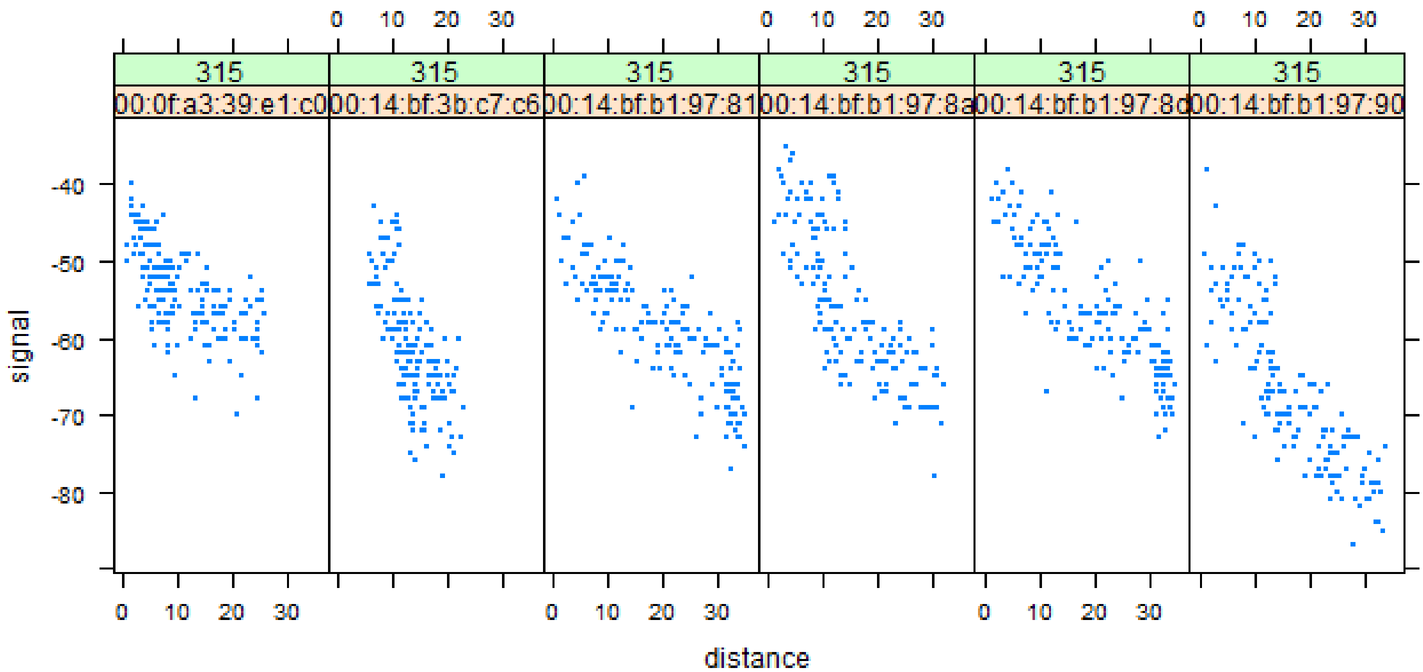
```r
xyplot(signal ~ log_dist | factor(mac) + factor(angle),
       data = offlineSummary, pch = 19, cex = 0.3,
       xlab = "log distance")
```



```r
xyplot(signal ~ log_dist | factor(mac) + factor(angle),
       data = offlineSummary[angle == 315], pch = 19, cex = 0.3,
       xlab = "log distance")
```




```r
xyplot(signal ~ dist | factor(mac) + factor(angle),
       data = offlineSummary[angle == 315], pch = 19, cex = 0.3,
       xlab = "distance")
```



The log transform removes a bit of the curvature, but not a great deal.

9.)

The floor plan for the building (see Figure 1.1) shows 65 access points. However, the data contain 7 access points with roughly the expected number of signals (166 locations x 8 orientations x 100 replications = 146,080 measurments).

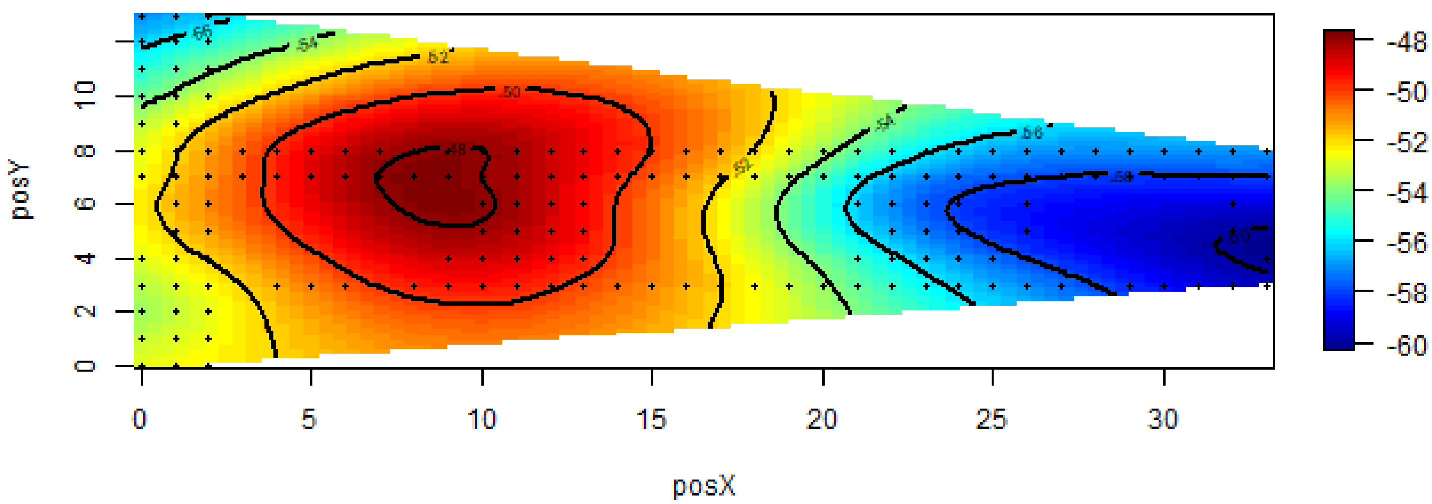
With the signal strength seen in the heat maps of Figure 1.10), we matched the access points to the corresponding MAC addresses. However, two of the MAC addresses seem to be for the same access point 00:0f:a3:39:e1:c0 and to eliminate the 00:0f:a3:39:dd:cd address.

Conduct a more thorough data analysis into these two MAC addresses. Did we make the correct decision? Does swapping out the one we kept for the one we discarded improve the prediction?

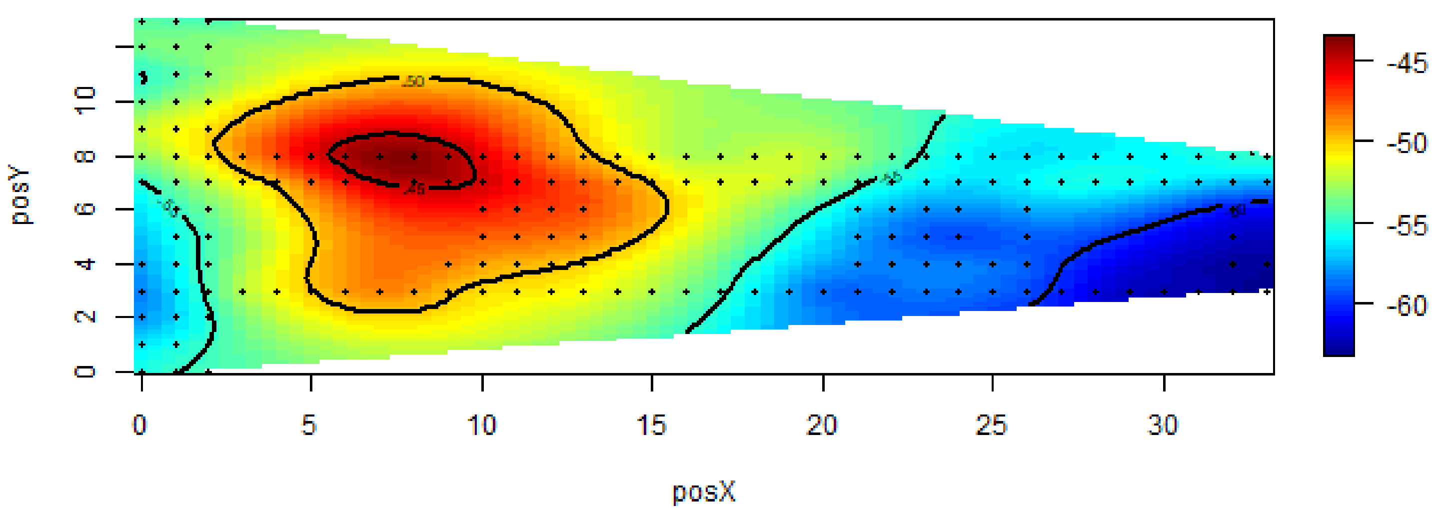
```
subMacs <- names(sort(table(offline$mac), decreasing = T)) # all sub macs

angles <- seq(0, by = 45, length = 8) # all angles

surfaceSS <- function(data, filter_mac, filter_angle) {
   oneAPAngle <- data[mac == filter_mac & angle == filter_angle]

   smothSS <- Tps(oneAPAngle[, c("posX", "posY")],
                  oneAPAngle$avgSignal)

   vizSmooth <- predictSurface(smothSS)
```

```
    plot.surface(vizSmooth, type = "C")
    points(oneAPAngle$posX, oneAPAngle$posY, pch=19, cex = 0.5)
}
```

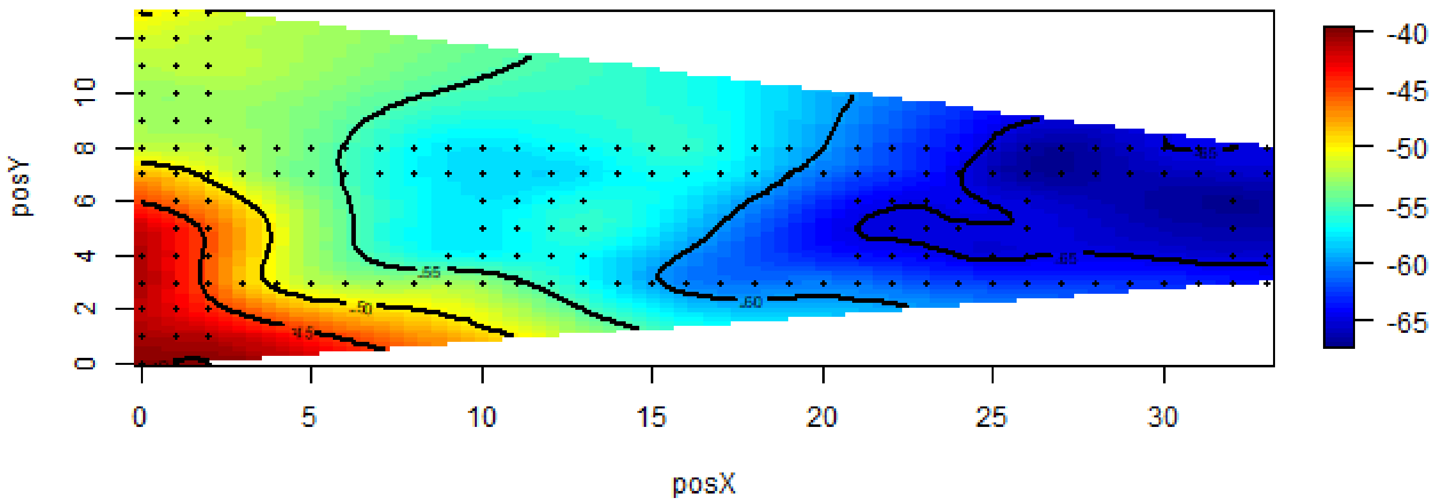
```
surfaceSS(offlineSummary, subMacs[1], angles[1])
```



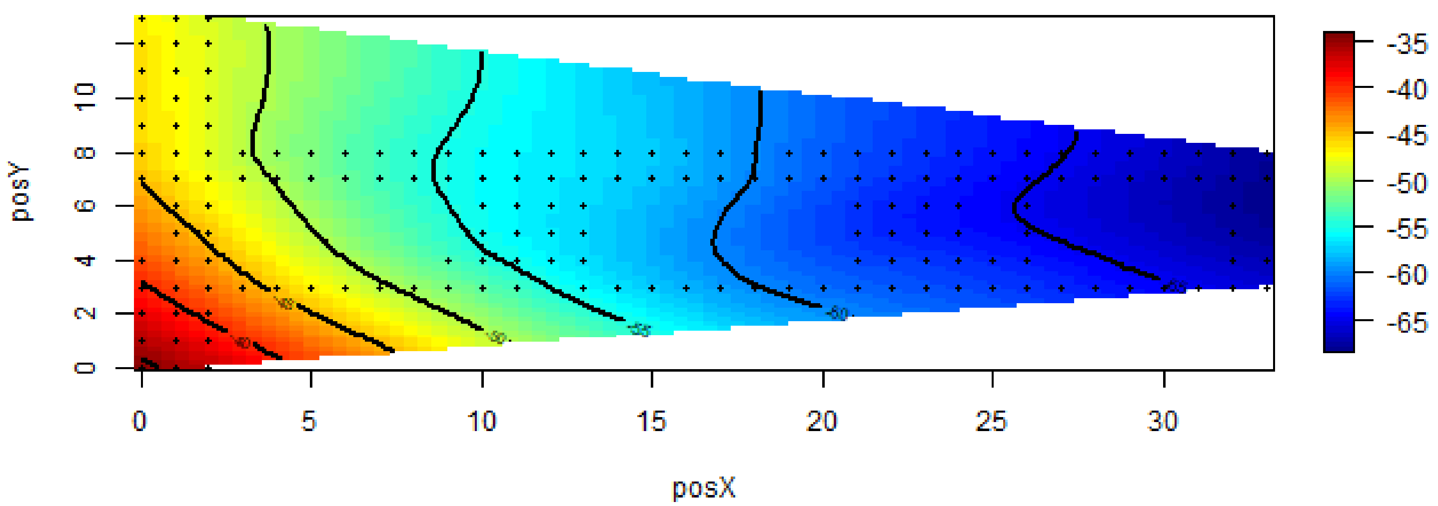

```
surfaceSS(offlineSummary, subMacs[1], angles[7])
```



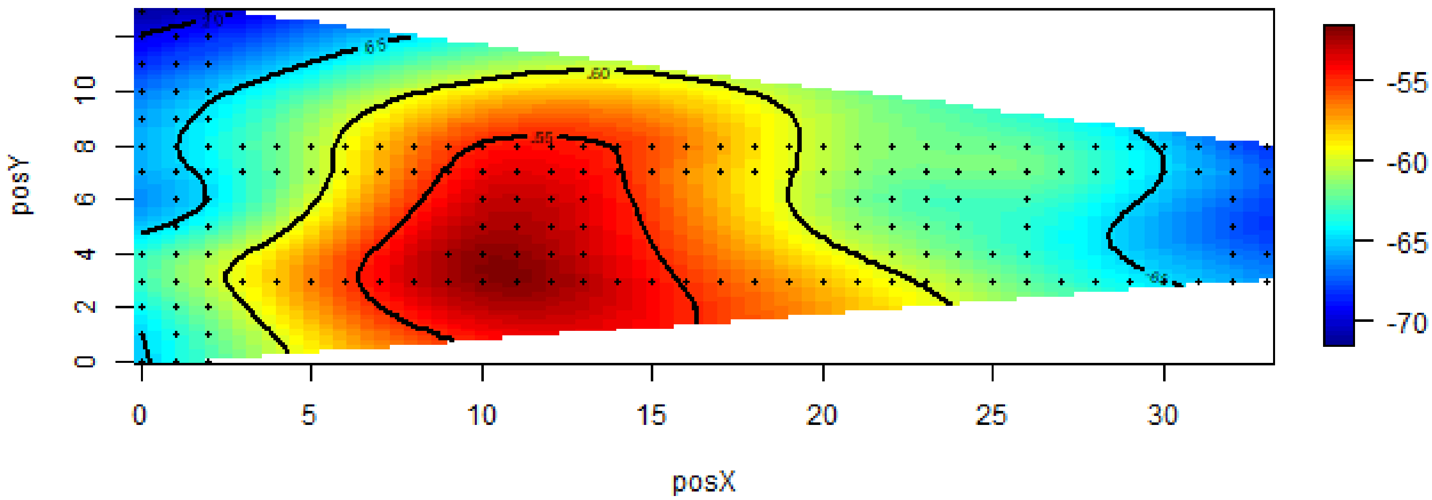
```r
surfaceSS(offlineSummary, subMacs[3], angles[2])
```



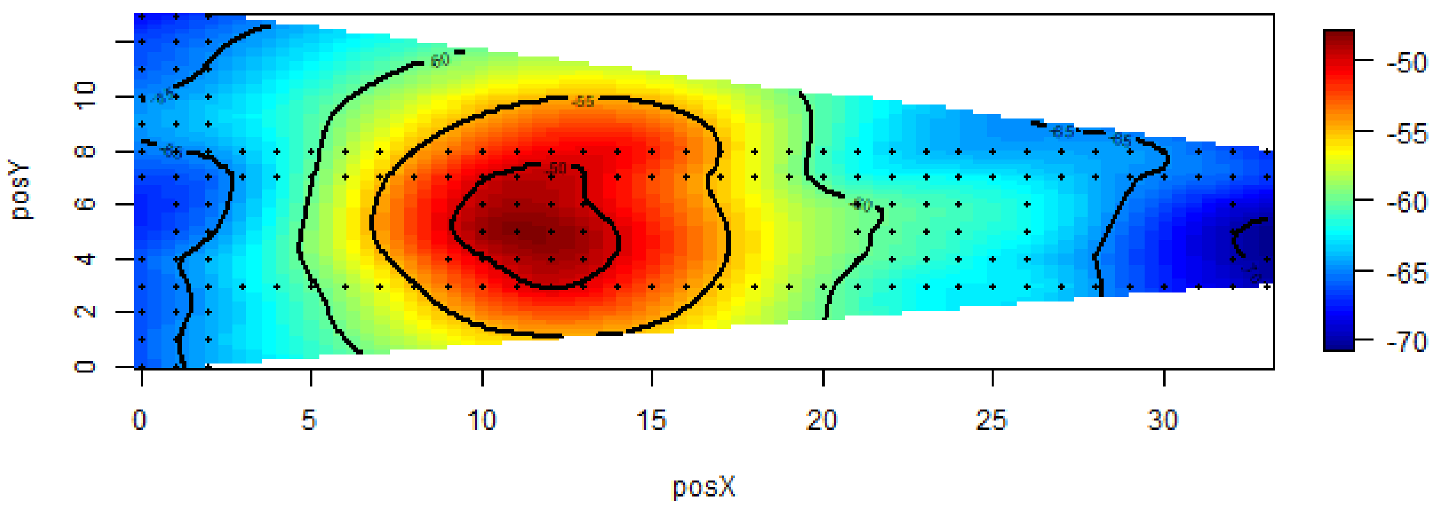

```r
surfaceSS(offlineSummary, subMacs[3], angles[7])
```



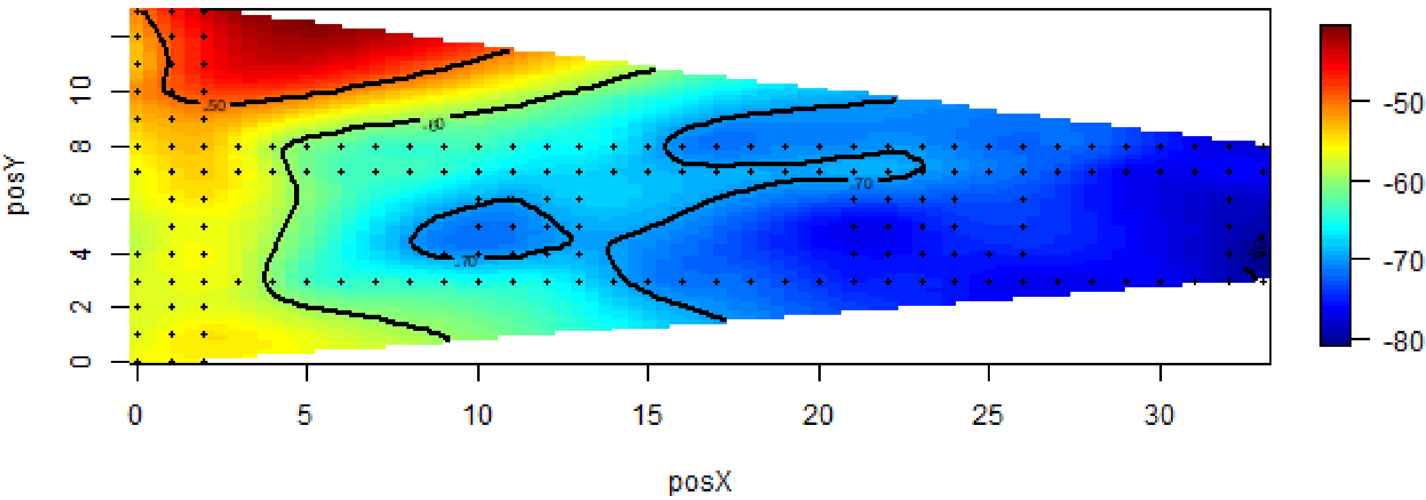

```r
surfaceSS(offlineSummary, subMacs[4], angles[2])
```



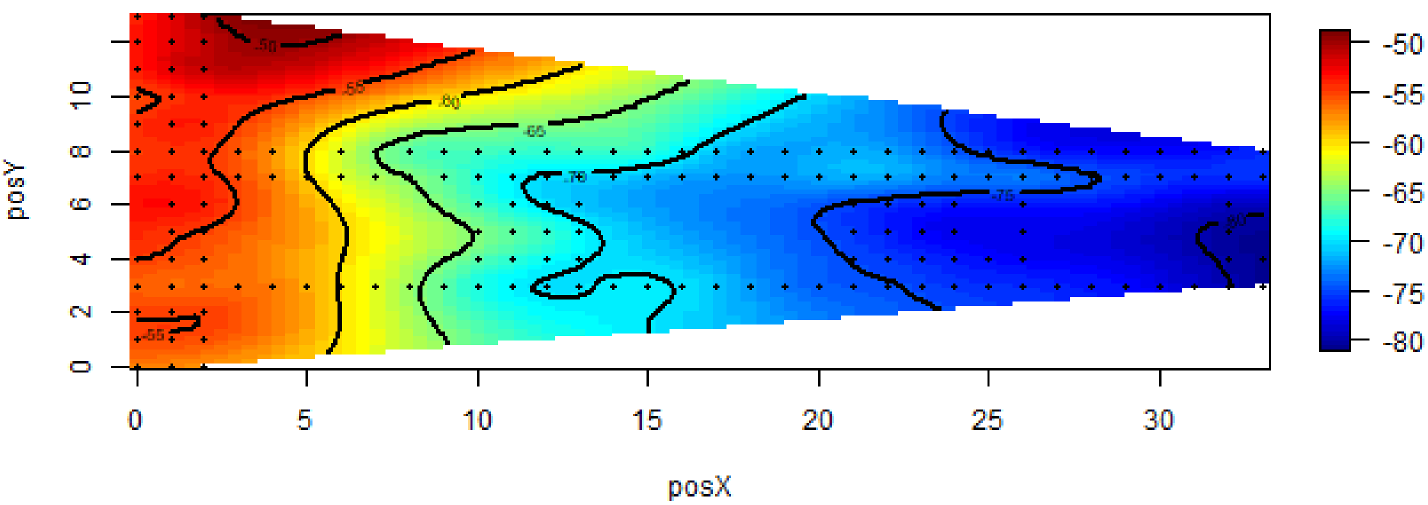

```r
surfaceSS(offlineSummary, subMacs[4], angles[7])
```



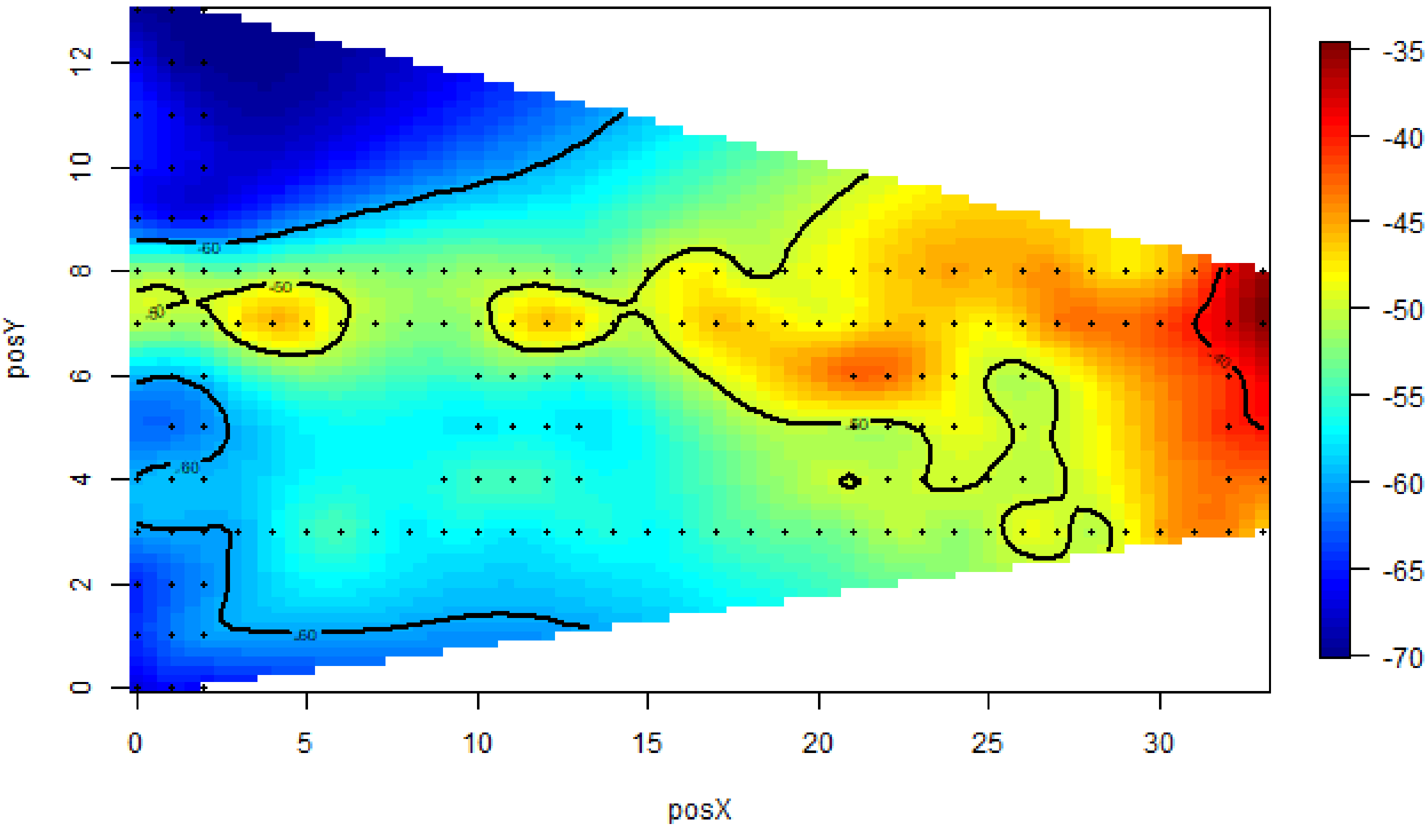

```r
surfaceSS(offlineSummary, subMacs[5], angles[2])
```



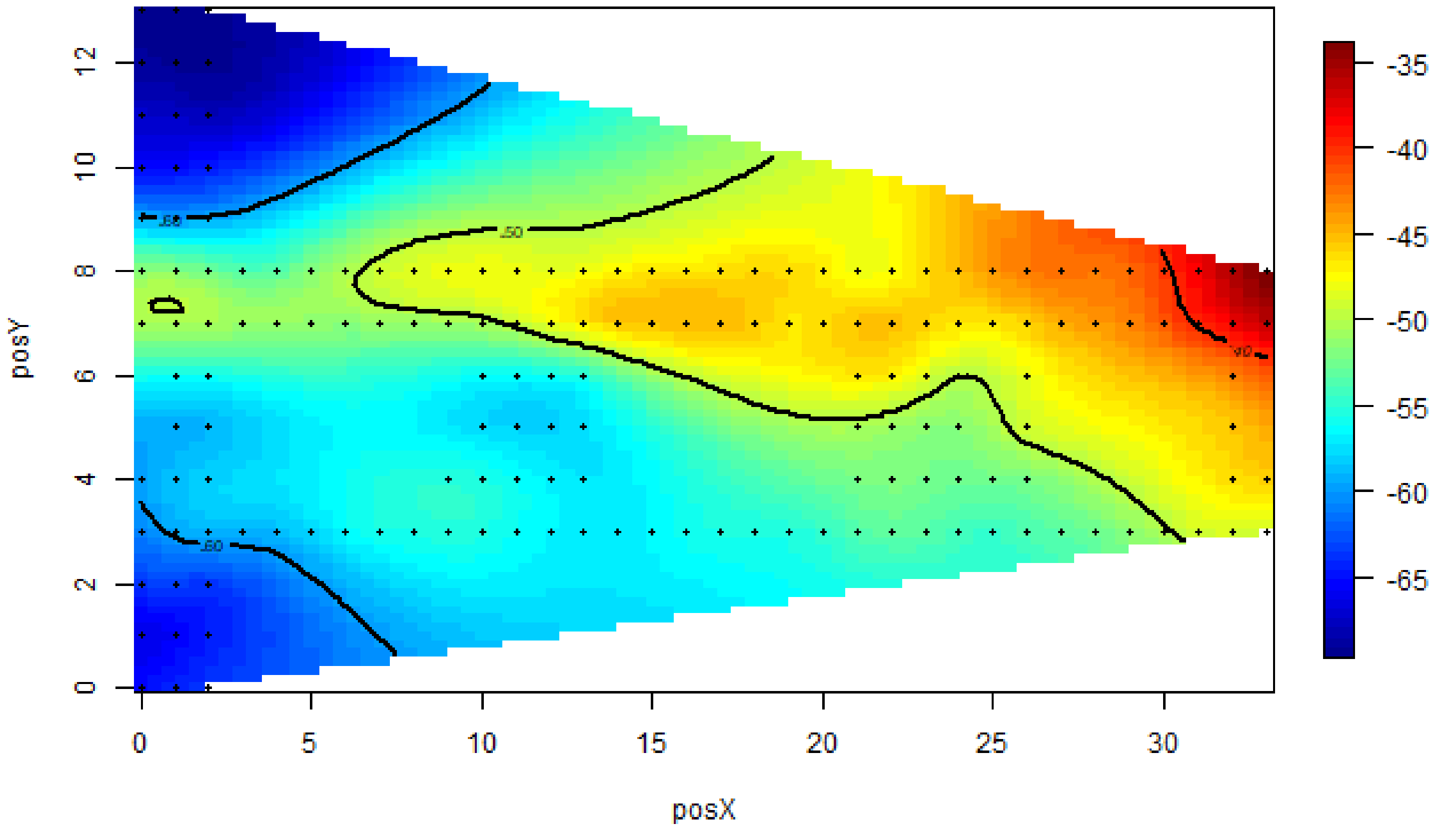

```r
surfaceSS(offlineSummary, subMacs[5], angles[7])
```



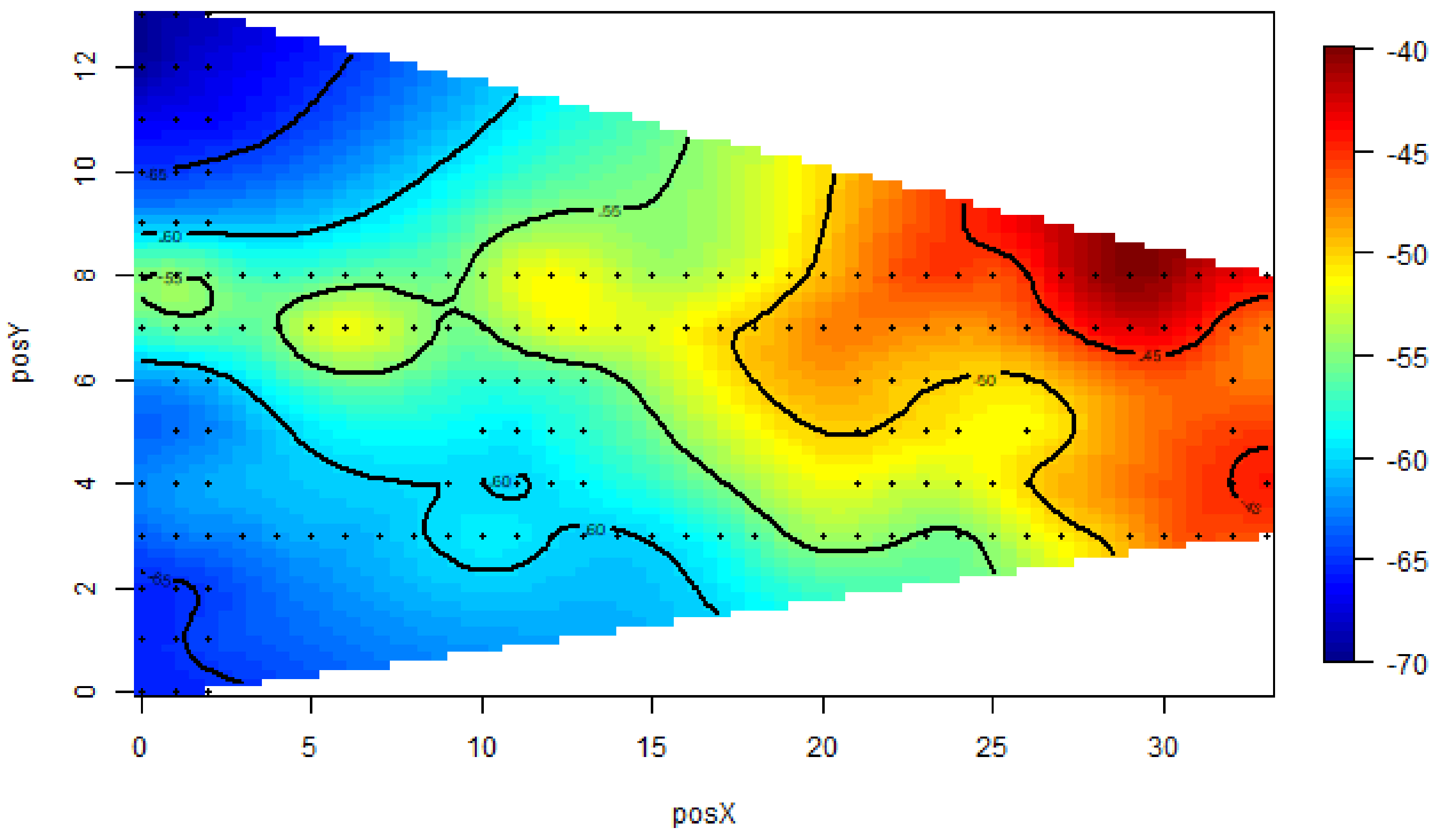

```r
surfaceSS(offlineSummary, subMacs[6], angles[2])
```




```r
surfaceSS(offlineSummary, subMacs[6], angles[5])
```




```r
surfaceSS(offlineSummary, subMacs[6], angles[7])
```



10.)

Write the selectTrain() function described in Section 1.5.2. This function has 3 parameters: angleNewObs, the angle of the new observation, signals, the training data, i.e., data in the format of offlineSummary; and m, the number of angles to include from signals.

The function returns a data frame that matches trainSS, i.e., selectTrain() calls reshapeSS().

```
reshapeSS <- function(data,
                      varSignal = "signal",
                      keepVars = c("posXY", "posX", "posY"),
                      sampleAngle = F) {

  if(sampleAngle)
    data <- data[angle == sample(data$angle, size = 1), ]

  byLocation <- with(data,
                by(data, list(posXY),
                   function(x) {
```

```
                        ans <- x[1, ..keepVars]
                        avgSS <- tapply(x$signal, x$mac, mean)
                        y = matrix(avgSS, nrow = 1, dimnames = list(ans$posXY, names(avgSS)))
                        cbind(ans, y)
                     }))

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

   col_names <- colnames(newDataSS)
   to_change <- !(col_names %in% keepVars)

   n_cols <- length(col_names)
   start <- length(keepVars)

   colnames(newDataSS)[to_change] <- sapply(col_names[to_change], function(col) {
      n <- nchar(col)
      substr(col, n - 2, n)
   })

   newDataSS[, start:ncol(newDataSS)] <- round(newDataSS[, start:ncol(newDataSS)])

   return(newDataSS)
}
```

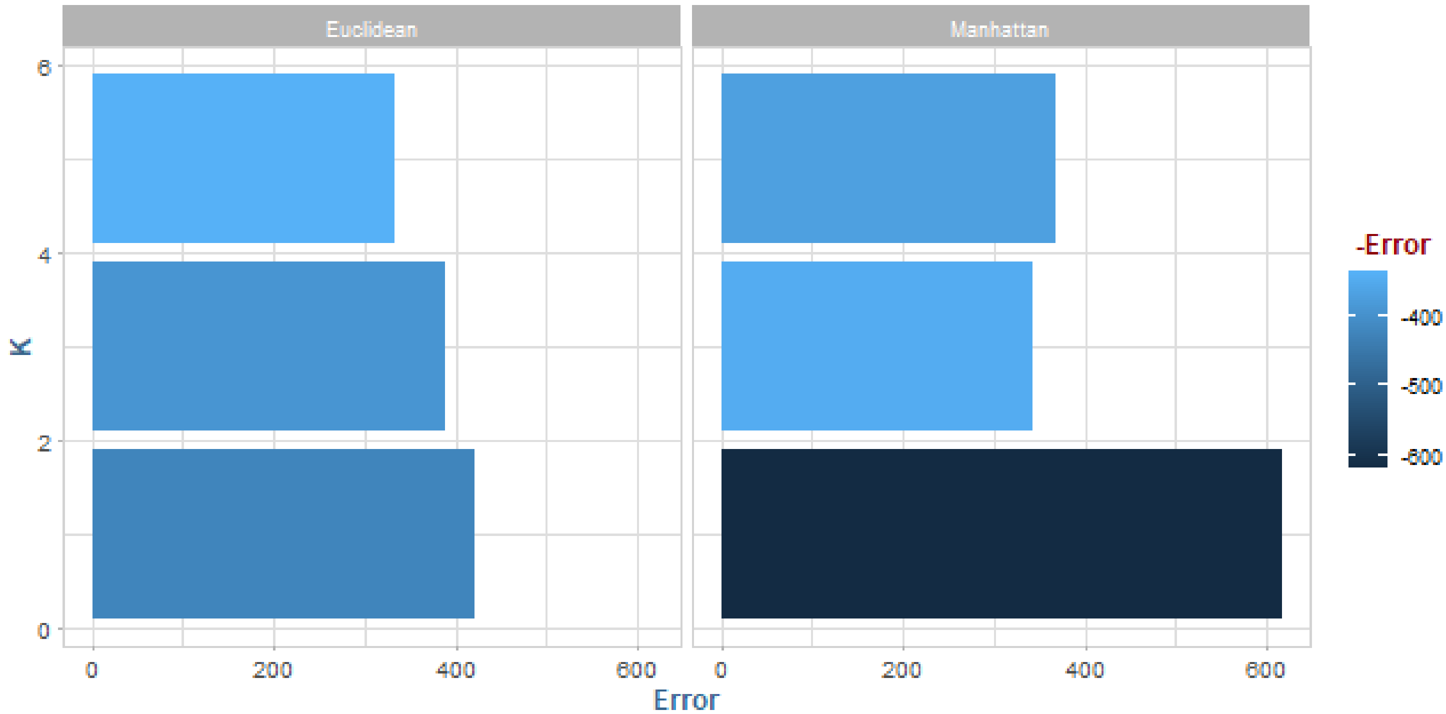
11.)

We use Euclidean distance to find the distance between the signal strength vectors. However, Euclidean distance is not robust in taht it is sensitive to outliers. Consider other metrics such as the L_1 distance, i.e., the absolute value of the difference. Modify the findNN() functions in sections 1.5.3 to use this alternative distance.

```
findNN_l1 <- function(newSignal, trainSubset) {
   diffs <- apply(trainSubset[ , 4:9], 1,
                  function(x) abs(x - newSignal))
   dists <- apply(diffs, 2, function(x) sum(x))
   closest <- order(dists)
   return(trainSubset[closest, 1:3 ])
}

predXY_l1 <- 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_l1(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)
}

estXYk1_l1 <- predXY_l1(newSignals = onlineSummary[, 6:11],
                  newAngles = onlineSummary[, 4],
                  offlineSummary, numAngles = 3, k = 1)

estXYk3_l1 <- predXY_l1(newSignals = onlineSummary[, 6:11],
                  newAngles = onlineSummary[, 4],
                  offlineSummary, numAngles = 3, k = 3)

estXYk5_l1 <- predXY_l1(newSignals = onlineSummary[, 6:11],
                  newAngles = onlineSummary[, 4],
                  offlineSummary, numAngles = 3, k = 5)

error_table <- data.table(K = c(1, 3, 5),
                          Error = c(calcError(estXYk1, actualXY),
                                    calcError(estXYk3, actualXY),
                                    calcError(estXYk5, actualXY)),
                          Method = "Euclidean")

error_table_l1 <- data.table(K = c(1, 3, 5),
                          Error = c(calcError(estXYk1_l1, actualXY),
                                    calcError(estXYk3_l1, actualXY),
                                    calcError(estXYk5_l1, actualXY)),
                          Method = "Manhattan")

errors_combined <- rbind(error_table, error_table_l1)

ggplot(errors_combined) +
  geom_bar(aes(K, Error, fill = -Error), stat = "identity") +
```

```
  facet_wrap(~Method) +
  coord_flip()
```



Does it improve predictions?

Euclidean seems to perform better.

12.)

To predict locaiton, we use the k nearest neighbors to a set of signal strenghts. We average the known (x, y) values for these neighbors. However, a better predictor might be a weighted average, wehre the weights are inversely proportional to the "distance" (in signal strength) from the test observation. This allows us to include the k points that are close, but to differentiate between them by how close they actually are.

The weights might be:

$$\frac{1/d_i}{\sum_{i=1}^{k} 1/d_i}$$

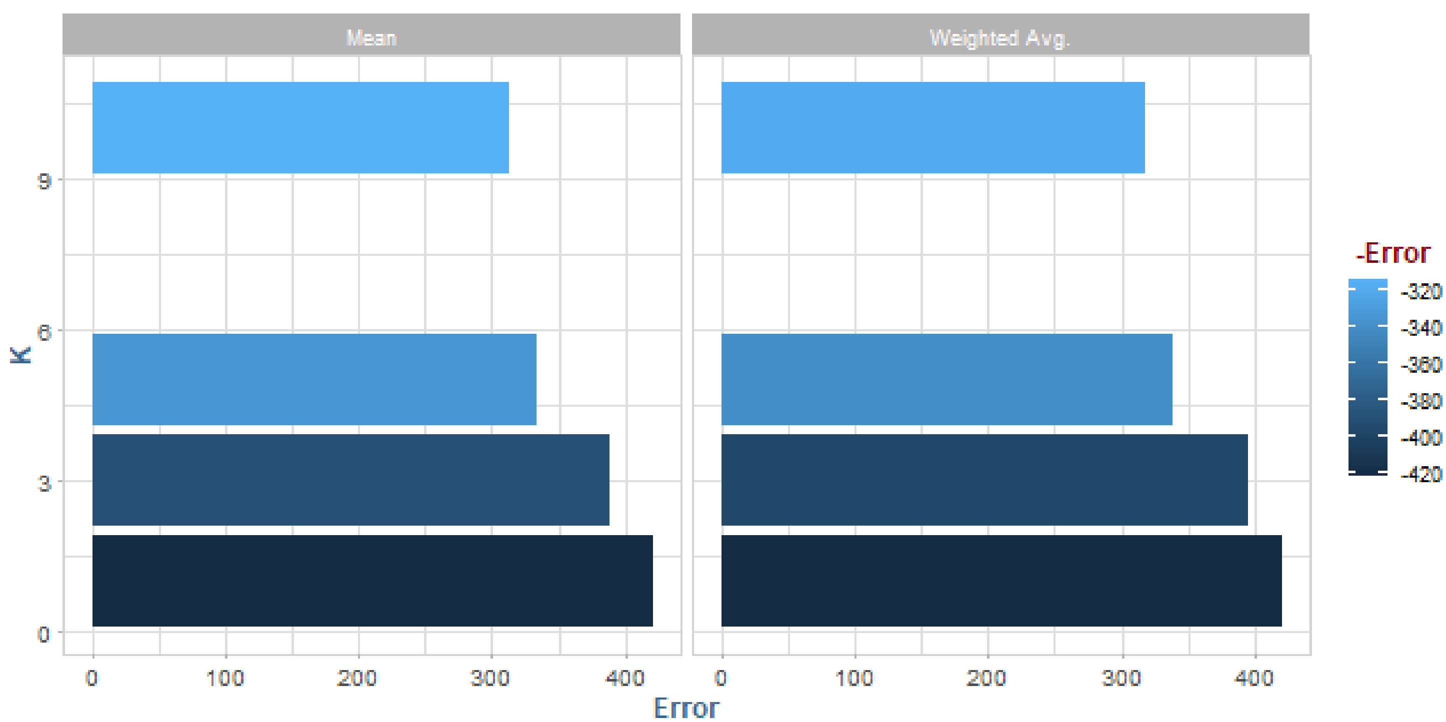
for the i-th closest neighboring observation where d_i is the distance from our new test point to this reference point (in signal strength space).

Implement this alternative prediciton method.

```
predXY_weighted <- 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) {
                                      points <- x[1:k]

                                      if(k > 1) {
                                        weights <- points/sum(points)

                                        return(sum(points * weights))
                                      } else {
                                        return(points)
                                      }
                                    }))

  estXY <- do.call("rbind", estXY)

  return(estXY)
}

estXYk10 <- predXY(newSignals = onlineSummary[, 6:11],
                   newAngles = onlineSummary[, 4],
                   offlineSummary, numAngles = 3, k = 10)

estXYk1_weighted <- predXY_weighted(newSignals = onlineSummary[, 6:11],
                   newAngles = onlineSummary[, 4],
                   offlineSummary, numAngles = 3, k = 1)

estXYk3_weighted <- predXY_weighted(newSignals = onlineSummary[, 6:11],
                   newAngles = onlineSummary[, 4],
                   offlineSummary, numAngles = 3, k = 3)

estXYk5_weighted <- predXY_weighted(newSignals = onlineSummary[, 6:11],
                   newAngles = onlineSummary[, 4],
                   offlineSummary, numAngles = 3, k = 5)

estXYk10_weighted <- predXY_weighted(newSignals = onlineSummary[, 6:11],
                   newAngles = onlineSummary[, 4],
```

```r
                  offlineSummary, numAngles = 3, k = 10)

error_table <- data.table(K = c(1, 3, 5, 10),
                          Error = c(calcError(estXYk1, actualXY),
                                    calcError(estXYk3, actualXY),
                                    calcError(estXYk5, actualXY),
                                    calcError(estXYk10, actualXY)),
                          Method = "Mean")

error_table_weighted <- data.table(K = c(1, 3, 5, 10),
                          Error = c(calcError(estXYk1_weighted, actualXY),
                                    calcError(estXYk3_weighted, actualXY),
                                    calcError(estXYk5_weighted, actualXY),
                                    calcError(estXYk10_weighted, actualXY)),
                          Method = "Weighted Avg.")

errors_combined <- rbind(error_table, error_table_weighted)

ggplot(errors_combined) +
   geom_bar(aes(K, Error, fill = -Error), stat = "identity") +
   facet_wrap(~Method) +
   coord_flip()
```



```
errors_combined
```

```
    K    Error        Method
1:  1 420.7603          Mean
2:  3 388.1070          Mean
3:  5 334.3683          Mean
4: 10 313.7643          Mean
5:  1 420.7603 Weighted Avg.
6:  3 395.4816 Weighted Avg.
7:  5 339.1897 Weighted Avg.
8: 10 317.5429 Weighted Avg.
```

Does this improve the predictions?

The weighted avg approach is of course identical in the base case (k=1), however, it performs slightly worse than the simple average in the k=3 and k=5 cases. I even upped k to 10 (the best CV k), and we still see simple average beats the weighted approach.

13.)

In Section 1.5.4 we used cross-validation to choose k, the number of neighbors. Another parameter to choose is the number of angles at which the signal strength was measured.

Use cross-validation to select this value.

Also, consider N + K cross-validation pairs.

```r
v <- 11 # n-folds
N <- 6 # num angles
K <- 10 # also choose k

nk_err <- matrix(rep(0, N * K), nrow = N, ncol = 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) {
      for(n in 1:N) {
         estFold <- predXY(newSignals = onlineFold[, 6:11],
                           newAngles = onlineFold[, 4],
                           offlineFold, numAngles = n, k = k)
```

```
        nk_err[n, k] <- calcError(estFold, actualFold)
      }
   }
}

nk_err
```

```
     [,1]   [,2]     [,3]     [,4]   [,5]     [,6]     [,7]     [,8]     [,9]
[1,]  353 280.25 235.0000 229.3125 222.40 229.2778 229.2449 224.0781 210.7901
[2,]  243 262.25 255.3333 198.2500 201.36 199.6667 188.3469 200.9062 207.3951
[3,]  252 253.75 218.7778 201.3750 180.40 178.1389 199.7347 204.3750 202.4321
[4,]  208 195.50 161.0000 174.4375 191.64 199.7500 200.2041 179.1562 168.2840
[5,]  208 195.50 161.0000 174.4375 191.64 199.7500 200.2041 179.1562 168.2840
[6,]  152 157.75 192.3333 186.0625 161.08 142.5833 144.5510 149.4844 151.2469
      [,10]
[1,] 206.95
[2,] 215.20
[3,] 197.61
[4,] 172.81
[5,] 172.81
[6,] 162.43
```

```
which(nk_err == min(nk_err), arr.ind = TRUE)
```

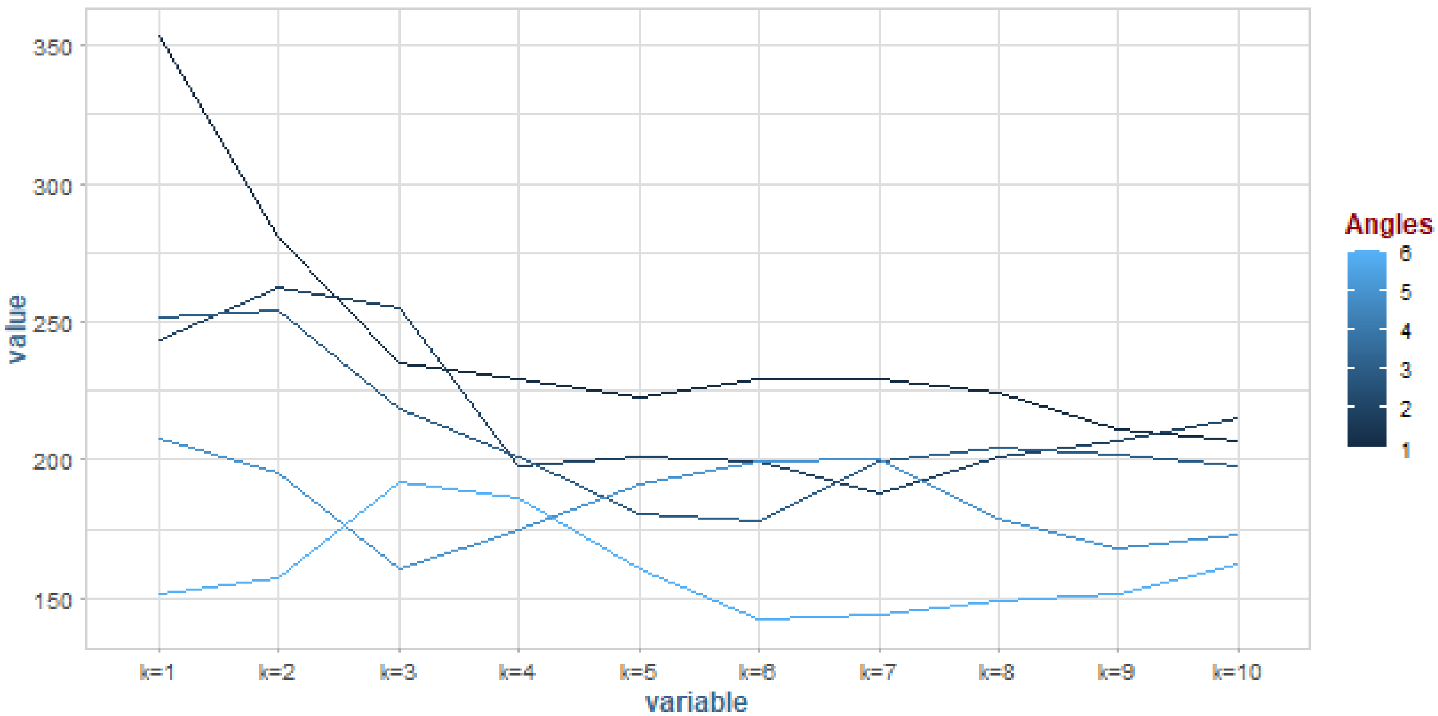
```
     row col
[1,]   6   6
```

```
head(order(nk_err))
```

```
[1] 36 42 48 54  6 12
```

```
# 1, 6
# 1, 5
# 1, 7
# 4, 8

n_take <- 10
cv_results <- data.table(nk_err)[, 1:n_take]
colnames(cv_results) <- sapply(1:n_take, function(x) {paste0("k=", x)})
cv_results[, Angles := .I]

min_cv <- which(nk_err == min(nk_err), arr.ind = TRUE)

cv_results_flat <- melt(cv_results, id.vars = "Angles")

ggplot(cv_results_flat, aes(variable, value, group = Angles)) +
```

```
geom_line(aes(col = Angles)) +
labs("K, N Cross-Validation Results")
```



The cross-validation of both N and K yield the optimal N at *1* and K at *6*. 4 of the top 5 CV results have *N=1*.