modeid usage example

Duncan Procter

8 September 2017

Requirements

To use this example you will need quite a few packages installed. Going though it you can see where each is loaded. The modeid package is required throughout, and can be installed from github via a devtools function, install.github:

```
install.packages("devtools","zoo","randomForest","sp","rgdal","maptools","spatstat")
library(devtools)
install.githhub("dprocter/modeid",quiet=TRUE)
```

Training data

How you extract your training data-set based on our rules will depend on how your demgraphic data is set up, providing an example of how we did it is not particularly useful. if you need help feel free to contact the author.

We have privided an anonymised version of our training dataset, so you can see how we use it:

```
library(modeid)
training.data<-Proc2017tdata
summary(training.data)
```

```
##      Axis1             Axis2             Axis3              ID
##  Min.   :   0.00   Min.   :   0.00   Min.   :   0.0   134    :  2588
##  1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.0   140    :  2471
##  Median :   0.00   Median :   0.00   Median :   0.0   18     :  2135
##  Mean   :  99.33   Mean   :  71.01   Mean   :  62.9   136    :  2086
##  3rd Qu.:  44.00   3rd Qu.:  69.00   3rd Qu.:  69.0   236    :  2001
##  Max.   :3858.00   Max.   :1959.00   Max.   :1556.0   107    :  1785
##                                                       (Other):118507
##         day            speed              pdop            hdop
##  Friday   :21339   Min.   :  0.000   Min.   : 0.920   Min.   :0.640
##  Monday   :18306   1st Qu.:  0.439   1st Qu.: 1.300   1st Qu.:0.930
##  Saturday :15539   Median :  1.785   Median : 1.530   Median :1.060
##  Sunday   :14212   Mean   : 12.326   Mean   : 1.714   Mean   :1.205
##  Thursday :20353   3rd Qu.: 16.672   3rd Qu.: 1.900   3rd Qu.:1.300
##  Tuesday  :19872   Max.   :159.999   Max.   :99.990   Max.   :5.000
##  Wednesday:21952
##       vdop            sumsnr        near.train       dist.next.min
##  Min.   : 0.650   Min.   : 30.0   Min.   :    0.00   Min.   :   0.001
##  1st Qu.: 0.840   1st Qu.:155.0   1st Qu.:   74.58   1st Qu.:   6.141
##  Median : 0.910   Median :213.0   Median :  269.11   Median :  52.999
##  Mean   : 1.164   Mean   :221.8   Mean   :  568.62   Mean   : 198.793
##  3rd Qu.: 1.370   3rd Qu.:281.0   3rd Qu.:  573.94   3rd Qu.: 267.682
##  Max.   :99.990   Max.   :564.0   Max.   :69652.48   Max.   :2909.710
##                                                      NA's   :3549
##  dist.last.min         ax1.mean           ax1.sd          ax1.cent.10
```

```
##  Min.   :   0.001   Min.   :   0.00   Min.   :   0.000   Min.   :   0.00
##  1st Qu.:   6.219   1st Qu.:   1.20   1st Qu.:   4.061   1st Qu.:   0.00
##  Median :  54.678   Median :  12.96   Median :  27.556   Median :   0.00
##  Mean   : 199.102   Mean   : 100.48   Mean   :  61.216   Mean   :  49.26
##  3rd Qu.: 267.694   3rd Qu.:  74.08   3rd Qu.:  86.822   3rd Qu.:   0.00
##  Max.   :2909.710   Max.   :1446.44   Max.   :1049.435   Max.   :1254.60
##  NA's   :3266
##   ax1.cent.90        spd.mean            spd.sd          spd.cent.10
##  Min.   :   0.0   Min.   :  0.01436   Min.   : 0.00791   Min.   :  0.004
##  1st Qu.:   0.0   1st Qu.:  0.80556   1st Qu.: 0.49780   1st Qu.:  0.135
##  Median :  31.6   Median :  4.00044   Median : 1.95215   Median :  0.450
##  Mean   : 165.9   Mean   : 11.99806   Mean   : 7.47013   Mean   :  3.245
##  3rd Qu.: 192.8   3rd Qu.: 17.37984   3rd Qu.:12.83787   3rd Qu.:  1.275
##  Max.   :2001.8   Max.   :159.09295   Max.   :69.43334   Max.   :157.867
##                                       NA's   :1
##   spd.cent.90        sumsnr.mean     near.train.4min
##  Min.   :  0.0252   Min.   : 35.44   Min.   :    0.49
##  1st Qu.:  1.4286   1st Qu.:159.24   1st Qu.:   94.96
##  Median :  5.9868   Median :214.16   Median :  271.10
##  Mean   : 21.2301   Mean   :222.26   Mean   :  568.58
##  3rd Qu.: 33.7308   3rd Qu.:277.71   3rd Qu.:  558.01
##  Max.   :159.8492   Max.   :502.28   Max.   :69631.47
##
##  dist.4min.nextmin   dist.4min.lastmin     cv.marker       true.mode
##  Min.   :   0.0615   Min.   :   0.0615   Min.   :1.000   bus    :12579
##  1st Qu.:   8.5951   1st Qu.:   8.6352   1st Qu.:2.000   cycle  :11607
##  Median :  75.0145   Median :  75.2581   Median :3.000   stat   :59499
##  Mean   : 195.7339   Mean   : 196.5133   Mean   :2.904   train  :18269
##  3rd Qu.: 281.2682   3rd Qu.: 281.6794   3rd Qu.:4.000   vehicle:16828
##  Max.   :2654.4335   Max.   :2656.6931   Max.   :5.000   walk   :12791
##  NA's   :74          NA's   :63
```

Replicating our cross.validation

The function *cross.validator* fits a number of random forest models to the subsets of the data you specify.

As you can see below you must first provide a dataset free from NAs, so we first kick out unused variables and then use na.omit to remove NAs

cross.validator then requires the NA free dataste, the formula of the model you wish to repeatedly fit, and a marker to denote the cross-validation subsets. We simply randomly assigned each participant to one of 5 subsets, see ?sample for random number selection

cross.validator currently has horrible looking output, which consists of 4 columns (the fitted models, the test data.frames, the confusion matrices and model accuracy scores). The rows correspond to the cross-validation subsets, so in our example there will be 5. This will be tidied in future versions.

We use set.seed to give the randomness a start point, but we cannot account for all of it in the cross-validation. Therefor, the output will slightly disagree with our reported scores. It also needs to be linked back to the original training data to include points we couldn't predict to (variables contained NAs), as errors, though these are tiny in number (under 100 points)

```
train.for.cv<-subset(training.data,select=-c(Axis1,Axis2,Axis3,speed,pdop,hdop,vdop,sumsnr,near.train
                                             ,dist.next.min,dist.last.min))
train.for.cv<-na.omit(train.for.cv)
```

```
set.seed(315)
cv.model<-cross.validator(training.dataset = train.for.cv
                ,formula = true.mode~ax1.mean+ax1.sd+ax1.cent.90+ax1.cent.10
                         +spd.mean+spd.sd+spd.cent.90+spd.cent.10
                         +sumsnr.mean+near.train.4min+dist.4min.nextmin+dist.4min.lastmin
                ,cv.marker = train.for.cv$cv.marker)

#This is ugly, but gives an overall confusion matrix
cv.overall<-cv.model[1,3][[1]]+cv.model[2,3][[1]]+cv.model[3,3][[1]]+cv.model[4,3][[1]]+cv.model[5,3][[
cv.overall
```

```
##          bus cycle  stat train vehicle  walk
## bus     8145   605   102   126    2598    13
## cycle    299 10338    24     1     307    18
## stat     186   112 58883    85      12   586
## train    109     9    34 17800      78    18
## vehicle 3813   516    42   201   13825     4
## walk      27    21   377    21       6 12144
## error      0     0     0     0       0     0
```

```
model.acc(cv.overall)
```

```
##     modes      ppv sensitivity      npv specificity accuracy f1.score
## 1     bus 70.28216    64.75078 96.30179    97.10359 94.00844 67.40318
## 2   cycle 94.09302    89.11301 98.95185    99.45864 98.54584 91.53533
## 3    stat 98.36129    99.02627 99.19158    98.63794 98.81355 98.69266
## 4   train 98.62589    97.61983 99.61741    99.78102 99.48131 98.12028
## 5 vehicle 75.13179    82.16451 97.34622    96.00904 94.23737 78.49093
## 6    walk 96.41156    95.00117 99.46252    99.61921 99.17025 95.70117
```

Data Processing

Let us assume:

1. You have Accelerometer files which have been exported to whatever epoch length you used e.g. 10 seconds.
2. You have corresponding GPS data

Merging accelerometer and GPS data

First we need to merge the data together. The following takes accfile, and gpsfile, which I have assigned to the appropriate file paths.

Cutoff.method = 3 means that we will cut the first day of data, then take 7 days of data, see ?gps.acc.merged for more details

British.time=1 means that the data is from the UK, so we need to convert UTC time from the GPS unit to British Summer time, during the appropriate months, so it matches the correct accelerometer data.

```
merged.data<-gps.acc.merge(accfile = accfile
                          ,gpsfile = gpsfile
                          ,participant.id = "bob"
                          ,cutoff.method = 3
                          ,epoch.length = 10
                          ,british.time = 1
```

```
                          ,acc.model = "Actigraph")

summary(merged.data)
```

```
##      Axis1              Axis2              Axis3              ID
##  Min.   :   0.00   Min.   :  0.00   Min.   :   0.00   Length:60480
##  1st Qu.:   0.00   1st Qu.:  0.00   1st Qu.:   0.00   Class :character
##  Median :   0.00   Median :  0.00   Median :   0.00   Mode  :character
##  Mean   :  43.58   Mean   : 32.98   Mean   :  18.79
##  3rd Qu.:   0.00   3rd Qu.:  0.00   3rd Qu.:   0.00
##  Max.   :1929.00   Max.   :988.00   Max.   :1292.00
##
##      index              day           latitude       longitude
##  Min.   :  991   Tuesday  : 5989   Min.   :51.51   Min.   :-0.13
##  1st Qu.: 7346   Thursday : 4803   1st Qu.:51.53   1st Qu.:-0.12
##  Median :13702   Monday   : 4764   Median :51.53   Median :-0.12
##  Mean   :13702   Wednesday: 3954   Mean   :51.53   Mean   :-0.07
##  3rd Qu.:20058   Friday   : 3707   3rd Qu.:51.53   3rd Qu.:-0.01
##  Max.   :26413   (Other)  : 2206   Max.   :51.59   Max.   : 0.00
##  NA's   :35057   NA's     :35057   NA's   :35057   NA's   :35057
##      speed             height             pdop             hdop
##  Min.   :  0.00   Min.   :-266.11   Min.   : 0.98   Min.   : 0.69
##  1st Qu.:  0.56   1st Qu.:  45.24   1st Qu.: 1.38   1st Qu.: 0.99
##  Median :  1.14   Median :  62.78   Median : 1.68   Median : 1.24
##  Mean   :  2.31   Mean   :  63.87   Mean   : 2.82   Mean   : 2.30
##  3rd Qu.:  2.33   3rd Qu.:  78.26   3rd Qu.: 2.35   3rd Qu.: 1.70
##  Max.   :194.36   Max.   :2140.84   Max.   :99.99   Max.   :99.99
##  NA's   :35057    NA's   :35057     NA's   :35057   NA's   :35057
##       vdop            sumsnr         date.time
##  Min.   : 0.68   Min.   :  0.0   Min.   :2013-07-10 00:00:00
##  1st Qu.: 0.86   1st Qu.:127.0   1st Qu.:2013-07-11 17:59:57
##  Median : 0.92   Median :160.0   Median :2013-07-13 11:59:55
##  Mean   : 1.31   Mean   :177.1   Mean   :2013-07-13 11:59:55
##  3rd Qu.: 1.09   3rd Qu.:198.0   3rd Qu.:2013-07-15 05:59:52
##  Max.   :99.99   Max.   :541.0   Max.   :2013-07-16 23:59:50
##  NA's   :35057   NA's   :35057
```

Removing accelerometer non-wear time

Accelerometer non-wear time will be identified as periods of 0 counts on all accelerometer axes.

window.length specifies the width of the window in minutes

interruption.length allows you to specify any allowed interruptions (epochs with over 0 counts), again in minutes

The following code therefore identifies non-wear time in the dataset merged.data, where you require 1 hour of 0 counts to denote non-wear, and we allow 2 minutes of interruptions.

All accelerometer axis data is set as NA where it seems to be non-wear

```
merged.data<-acc.nonwear(dataset= merged.data
                          ,epoch.length = 10
                          ,window.length = 60
```

```
                          ,interruption.length = 2)
summary(merged.data)
```

```
##      Axis1             Axis2             Axis3              ID
##  Min.   :   0.0   Min.   :  0.00   Min.   :   0.00   Length:60480
##  1st Qu.:   0.0   1st Qu.:  0.00   1st Qu.:   0.00   Class :character
##  Median :   0.0   Median :  0.00   Median :   0.00   Mode  :character
##  Mean   : 100.4   Mean   : 75.37   Mean   :  42.25
##  3rd Qu.:   9.0   3rd Qu.: 48.00   3rd Qu.:  37.00
##  Max.   :1929.0   Max.   :906.00   Max.   :1247.00
##  NA's   :34509    NA's   :34509    NA's   :34509
##      index              day           latitude       longitude
##  Min.   :  991   Tuesday  : 5989   Min.   :51.51   Min.   :-0.13
##  1st Qu.: 7346   Thursday : 4803   1st Qu.:51.53   1st Qu.:-0.12
##  Median :13702   Monday   : 4764   Median :51.53   Median :-0.12
##  Mean   :13702   Wednesday: 3954   Mean   :51.53   Mean   :-0.07
##  3rd Qu.:20058   Friday   : 3707   3rd Qu.:51.53   3rd Qu.:-0.01
##  Max.   :26413   (Other)  : 2206   Max.   :51.59   Max.   : 0.00
##  NA's   :35057   NA's     :35057   NA's   :35057   NA's   :35057
##      speed            height            pdop            hdop
##  Min.   :  0.00   Min.   :-266.11   Min.   : 0.98   Min.   : 0.69
##  1st Qu.:  0.56   1st Qu.:  45.24   1st Qu.: 1.38   1st Qu.: 0.99
##  Median :  1.14   Median :  62.78   Median : 1.68   Median : 1.24
##  Mean   :  2.31   Mean   :  63.87   Mean   : 2.82   Mean   : 2.30
##  3rd Qu.:  2.33   3rd Qu.:  78.26   3rd Qu.: 2.35   3rd Qu.: 1.70
##  Max.   :194.36   Max.   :2140.84   Max.   :99.99   Max.   :99.99
##  NA's   :35057    NA's   :35057     NA's   :35057   NA's   :35057
##       vdop           sumsnr        date.time
##  Min.   : 0.68   Min.   :  0.0   Min.   :2013-07-10 00:00:00
##  1st Qu.: 0.86   1st Qu.:127.0   1st Qu.:2013-07-11 17:59:57
##  Median : 0.92   Median :160.0   Median :2013-07-13 11:59:55
##  Mean   : 1.31   Mean   :177.1   Mean   :2013-07-13 11:59:55
##  3rd Qu.: 1.09   3rd Qu.:198.0   3rd Qu.:2013-07-15 05:59:52
##  Max.   :99.99   Max.   :541.0   Max.   :2013-07-16 23:59:50
##  NA's   :35057   NA's   :35057
```

Cleaning GPS data

There are sevral circumstances is which GPS data can be unreliable (usually caused by poor signal). Therefore we remove points we do not think we can trust.

We clean the data in 3 ways:

1. Using a speed cut-off, to remove implausibly high speed points
2. Using a Horisontal Dilution of Precision cut-off, to remove points where the satellites are aligned and so signal is poor
3. By removing points that are isolated, and thefore have no context

The following therefore marks all GPS data where speed is under 160kph, hdop is under 5, or there are less than 3 points within 5 minutes (2.5 minutes before and after the points, inluding the point itself, therefore 2 neighbours).

The data.loss.gps function tell you how many points are removed at each level of processing

```
data.loss.gps(speed.cutoff = 160
              ,hdop.cutoff = 5
              ,neighbour.number = 3
              ,neighbour.window = 300
              ,epoch.length = 10
              ,dataset = merged.data)
```

```
##               labels data.amounts data.removed
## 1 total.dataset.size        60480            0
## 2   invalid.gps.data        25423        35057
## 3       no.neigbours        25410           13
## 4       excess.speed        25409            1
## 5        poor.signal        24204         1205
```

```
merged.data<-gps.cleaner(speed.cutoff = 160
                         ,hdop.cutoff = 5
                         ,neighbour.number = 3
                         ,neighbour.window = 300
                         ,epoch.length = 10
                         ,dataset = merged.data)
```

Calculating distance to train lines

This doesn't cover getting the neccessary data on train lines. As a start point, if you're in a UK institution there is lots of data freely available on Digimap.

We use the sp and rgdal packages to import an ArcGIS shapefile into R of train lines, then convert the SpatialLinesDataFrame into a psp so we can use the spatstat function nncross to measure distance from each point to the nearest train line. For the participants data we make SpatialPointsDataFrame, then reproject it to OSGB1936, used by the train data, and then convert it into a ppp to allow us to use the spatstat function.

```
library(sp)
library(maptools)
```

```
## Checking rgeos availability: TRUE
```

```
library(rgdal)
```

```
## rgdal: version: 1.2-8, (SVN revision 663)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.0.1, released 2015/09/15
##  Path to GDAL shared files: C:/Duncan/R libraries/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.2, 08 September 2015, [PJ_VERSION: 492]
##  Path to PROJ.4 shared files: C:/Duncan/R libraries/rgdal/proj
##  Linking to sp version: 1.2-5
```

```
library(spatstat)
```

```
## Loading required package: nlme
```

```
## Loading required package: rpart
```

```
##
## spatstat 1.52-1       (nickname: 'Apophenia')
## For an introduction to spatstat, type 'beginner'
```

```
train.lines<-readOGR("C:/Duncan/Train lines","all_train_lines")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:/Duncan/Train lines", layer: "all_train_lines"
## with 38316 features
## It has 8 fields
## Integer64 fields read as strings:  OBJECTID CODE
```

```
train.coords<-coordinates(train.lines)
max.x<-max(unlist(lapply(train.coords,FUN=function(x){x[[1]][,1]})))
max.y<-max(unlist(lapply(train.coords,FUN=function(x){x[[1]][,2]})))
min.x<-min(unlist(lapply(train.coords,FUN=function(x){x[[1]][,1]})))
min.y<-min(unlist(lapply(train.coords,FUN=function(x){x[[1]][,2]})))

train.win<-owin(xrange=c(min.x,max.x),yrange=c(min.y,max.y))

train.psp<-as.psp(train.lines,W=train.win)
```

```
## Warning in as.psp.SpatialLinesDataFrame(train.lines, W = train.win): 7
## columns of data frame discarded
```

```
 # add the near train data to the merged dataset
  merged.data$near.train<-NA

  # take a subset of the data that has valid GPS data
  # and turn it into a SpatialPointsDataFrame, with projection information
  only.gps<-subset(merged.data,!is.na(speed))
  merged.data$easting<-NA
  merged.data$northing<-NA
  if (length(only.gps[,1])>0){
    gps.spatial<-SpatialPointsDataFrame(cbind(only.gps$longitude,only.gps$latitude)
                                        ,data=only.gps,proj4string = CRS("+proj=longlat +datum=WGS84"))

    # convert gps.spatial to have the same projection as the train.lines data
    gps.spatial<-spTransform(gps.spatial,CRS(proj4string(train.lines)))

    #creates a bounding box around the points (ppp's need these)
    gps.win<-owin(xrange=c(min(coordinates(gps.spatial)[,1]-1000),max(coordinates(gps.spatial)[,1])+100
                  ,yrange=c(min(coordinates(gps.spatial)[,2]-1000),max(coordinates(gps.spatial)[,2]+100
    # turns the gps data into a sptial point pattern
    gps.ppp<-as.ppp(coordinates(gps.spatial),W=gps.win)

    # the nncross function from spatstat gives you sitance from each point to the nearest line
    merged.data$near.train[!is.na(merged.data$speed)]<-nncross(gps.ppp,train.psp)[,1]

    merged.data$easting[!is.na(merged.data$longitude)]<-coordinates(gps.spatial)[,1]
    merged.data$northing[!is.na(merged.data$longitude)]<-coordinates(gps.spatial)[,2]
  }
```

Distance 1 minute away

When not travelling, people stay in one spot. To take this into consideration we include distance moved in the next minute and distance moved in the last minute. Both are included so that we can detect no movement both just as you stopped and just before you start moving too.

```
merged.data$dist.next.min<-distance.moved(dataset = merged.data,
                                          last=FALSE,
                                          time.window = 60,
                                          epoch.length = 10)

merged.data$dist.last.min<-distance.moved(dataset = merged.data,
                                          last=TRUE,
                                          time.window = 60,
                                          epoch.length = 10)
summary(merged.data)
```

```
##      Axis1            Axis2            Axis3              ID
##  Min.   :   0.0   Min.   :  0.00   Min.   :   0.00   Length:60480
##  1st Qu.:   0.0   1st Qu.:  0.00   1st Qu.:   0.00   Class :character
##  Median :   0.0   Median :  0.00   Median :   0.00   Mode  :character
##  Mean   : 100.4   Mean   : 75.37   Mean   :  42.25
##  3rd Qu.:   9.0   3rd Qu.: 48.00   3rd Qu.:  37.00
##  Max.   :1929.0   Max.   :906.00   Max.   :1247.00
##  NA's   :34509    NA's   :34509    NA's   :34509
##      index              day           latitude       longitude
##  Min.   :  991   Tuesday  : 5989   Min.   :51.51   Min.   :-0.13
##  1st Qu.: 7346   Thursday : 4803   1st Qu.:51.53   1st Qu.:-0.12
##  Median :13702   Monday   : 4764   Median :51.53   Median :-0.12
##  Mean   :13702   Wednesday: 3954   Mean   :51.53   Mean   :-0.08
##  3rd Qu.:20058   Friday   : 3707   3rd Qu.:51.53   3rd Qu.:-0.01
##  Max.   :26413   (Other)  : 2206   Max.   :51.55   Max.   : 0.00
##  NA's   :35057   NA's     :35057   NA's   :36274   NA's   :36274
##      speed            height             pdop            hdop
##  Min.   :  0.00   Min.   :-266.11   Min.   : 0.98   Min.   :0.69
##  1st Qu.:  0.55   1st Qu.:  45.24   1st Qu.: 1.36   1st Qu.:0.98
##  Median :  1.11   Median :  62.78   Median : 1.63   Median :1.21
##  Mean   :  2.23   Mean   :  63.87   Mean   : 1.97   Mean   :1.47
##  3rd Qu.:  2.23   3rd Qu.:  78.26   3rd Qu.: 2.21   3rd Qu.:1.57
##  Max.   :101.18   Max.   :2140.84   Max.   :25.57   Max.   :5.00
##  NA's   :36274    NA's   :35057     NA's   :36274   NA's   :36274
##       vdop            sumsnr         date.time
##  Min.   : 0.68   Min.   : 26.0   Min.   :2013-07-10 00:00:00
##  1st Qu.: 0.86   1st Qu.:131.0   1st Qu.:2013-07-11 17:59:57
##  Median : 0.91   Median :163.0   Median :2013-07-13 11:59:55
##  Mean   : 1.21   Mean   :181.3   Mean   :2013-07-13 11:59:55
##  3rd Qu.: 1.12   3rd Qu.:201.0   3rd Qu.:2013-07-15 05:59:52
##  Max.   :25.14   Max.   :541.0   Max.   :2013-07-16 23:59:50
##  NA's   :36274   NA's   :36274
##    near.train       easting          northing      dist.next.min
##  Min.   :  0.0   Min.   :529863   Min.   :180864   Min.   :   0.00
##  1st Qu.:209.6   1st Qu.:530654   1st Qu.:183135   1st Qu.:   8.15
##  Median :235.6   Median :530674   Median :183163   Median :  17.98
##  Mean   :252.9   Mean   :533493   Mean   :183263   Mean   :  36.75
##  3rd Qu.:325.9   3rd Qu.:538207   3rd Qu.:183441   3rd Qu.:  39.12
##  Max.   :917.6   Max.   :538633   Max.   :185302   Max.   :1931.55
##  NA's   :36274   NA's   :36274    NA's   :36274    NA's   :37436
##  dist.last.min
##  Min.   :   0.00
##  1st Qu.:   8.15
```

```
##  Median :  17.98
##  Mean   :  36.75
##  3rd Qu.:  39.12
##  Max.   :1931.55
##  NA's   :37436
```

Calculating moving windows

None of the previous functions remove invalid data from the dataset, they only set the relevant accelerometer or GPS variables as NA when we have reason to think they are untrustworthy. As a result the dataset is a continuous set of epochs from the start to end of the data. We can therefore treat a moving window across the cleaned merged dataset as a time window, as long as we allow for how long each epoch represents.

To calculate moving windows we use the zoo package, and particularly the *rollapply* function. Rollapply allows us to specify a function and then apply it across a window. We use a width of four minutes, centered on the point of interest, so here that is 25 epochs (12 before, 12 after, each 10 seconds plus the center point). All the functions are simple mean, sd or quantile, but with na.omit, so they ignore small numbers of NAs in the window. partial=TRUE means that if not all pointsjion ;/ in the 25 are there (e.g. it is the start of the dataset), the function is still applied to the remaining points.

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
  merged.data$ax1.mean<-rollapply(merged.data$Axis1,width=25,align="center",FUN=function(x){mean(na.omi
  merged.data$ax1.sd<-rollapply(merged.data$Axis1,width=25,align="center",FUN=function(x){sd(na.omit(x)
  merged.data$ax1.cent.10<-rollapply(merged.data$Axis1,width=25,align="center",FUN=function(x){quantile
  merged.data$ax1.cent.90<-rollapply(merged.data$Axis1,width=25,align="center",FUN=function(x){quantile

  merged.data$spd.mean<-rollapply(merged.data$speed,width=25,align="center",FUN=function(x){mean(na.omi
  merged.data$spd.sd<-rollapply(merged.data$speed,width=25,align="center",FUN=function(x){sd(na.omit(x)
  merged.data$spd.cent.10<-rollapply(merged.data$speed,width=25,align="center",FUN=function(x){quantile
  merged.data$spd.cent.90<-rollapply(merged.data$speed,width=25,align="center",FUN=function(x){quantile

  merged.data$sumsnr.mean<-rollapply(merged.data$sumsnr,width=25,align="center",FUN=function(x){mean(na
  merged.data$near.train.4min<-rollapply(merged.data$near.train,width=25,align="center",FUN=function(x)

    merged.data$dist.4min.lastmin<-rollapply(merged.data$dist.last.min,width=25,align="center",FUN=funct
      merged.data$dist.4min.nextmin<-rollapply(merged.data$dist.next.min,width=25,align="center",FUN=fu
```

Predicting to the example data

Prediction to participants is simple once you have calculated the rolling means:

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(315)
base.model<-randomForest(true.mode~ax1.mean+ax1.sd+ax1.cent.90+ax1.cent.10
                         +spd.mean+spd.sd+spd.cent.90+spd.cent.10
                         +sumsnr.mean+near.train.4min+dist.4min.lastmin+dist.4min.nextmin
                         ,data=train.for.cv,importance=TRUE)

merged.data$pred.mode<-predict(base.model,newdata = merged.data,type="response")
summary(merged.data$pred.mode)
```

```
##     bus   cycle    stat   train vehicle    walk    NA's
##     155      87   17243     630       4    3673   38688
```

```
plot(merged.data$longitude,merged.data$latitude,axes=FALSE,xlab="",ylab="",col=merged.data$pred.mode,pch
```