GPX Cleaning

wolf zinke

April 13, 2020

Try to remove outliers from a gpx track and smooth the data.

some inspirations came from these resources: * https://github.com/Robinlovelace/Creating-maps-in-R/blob/master/vignettes/gpx-files.Rmd * https://bookdown.org/robinlovelace/geocompr * https://rspatial.org/raster/spatial/index.html * https://www.r-bloggers.com/stay-on-track-plotting-gps-tracks-with-r * https://digital-geography.com/gpx-overview-r-function-create-overview-gpx-files-using-leaflet-rgooglemaps

load required packages

Load all packages required for the analysis. If these packages are not available, install them on the fly with a `install.packages(<packagesname>)` call.

read in gpx file

```
# from https://stackoverflow.com/questions/36377252/import-gpx-track-using-the-xml-library
filename = "EM_Lunch_Walk.gpx"

gpx_dt <- filename %>%
  xmlTreeParse(useInternalNodes = TRUE) %>%
  xmlRoot %>%
  xmlToList %>%
  (function(x) x$trk) %>%
  (function(x) unlist(x[names(x) == "trkseg"], recursive = FALSE)) %>%
  map_df(function(x) as.data.frame(t(unlist(x)), stringsAsFactors=FALSE))

# names(gpx_dt) # verify attributes labels
names(gpx_dt) = c("Elevation", "Time", "Latitude", "Longitude")

gpx_dt$Longitude = as.numeric(gpx_dt$Longitude)
gpx_dt$Latitude  = as.numeric(gpx_dt$Latitude)
gpx_dt$Elevation = as.numeric(gpx_dt$Elevation)
gpx_dt$Time      = as.POSIXct(gpx_dt$Time, tz="GMT", format="%Y-%m-%dT%H:%M:%OS")
gpx_dt$Duration  = as.numeric(difftime(gpx_dt$Time, gpx_dt$Time[1]), units="hours")

Npts = length(gpx_dt$Longitude)
sTm = gpx_dt$Time[1]
```

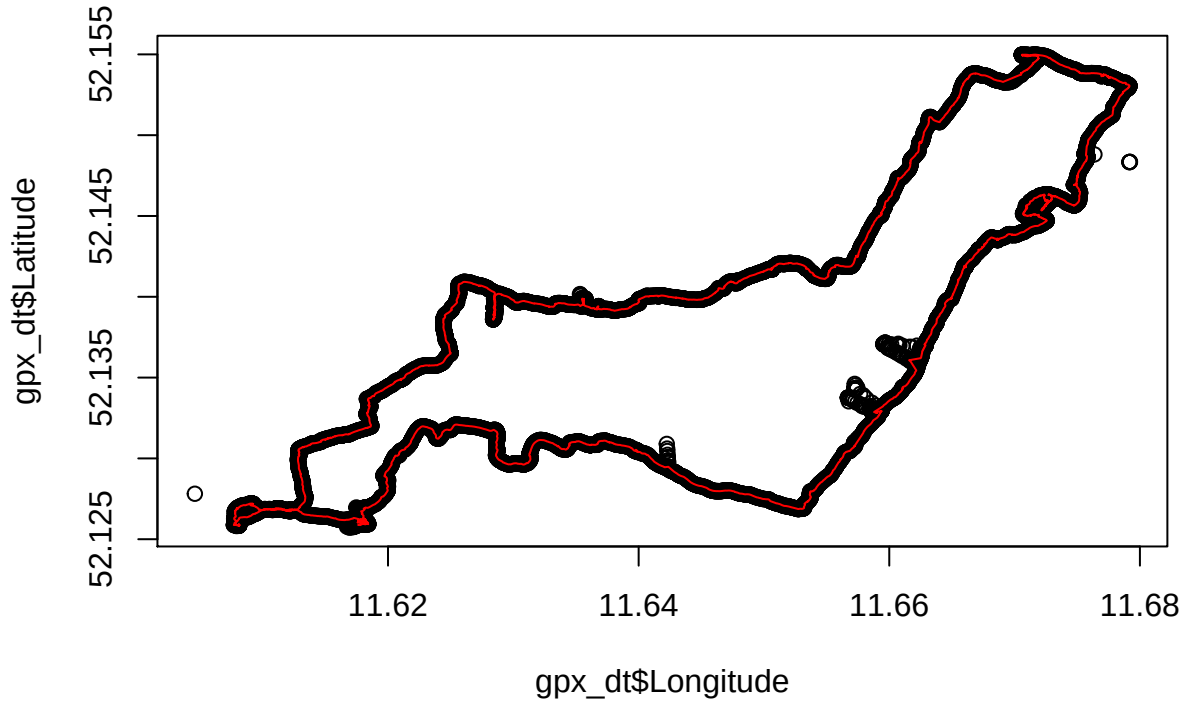
Determine Distance and Speed

```
Dist     = spDists(x=cbind(gpx_dt$Longitude, gpx_dt$Latitude), longlat=TRUE, segments = TRUE)
interval = diff(gpx_dt$Duration)
smpldur  = round(median(interval)/2, 10)

Speed = c(0, Dist / interval)   # km/h
```

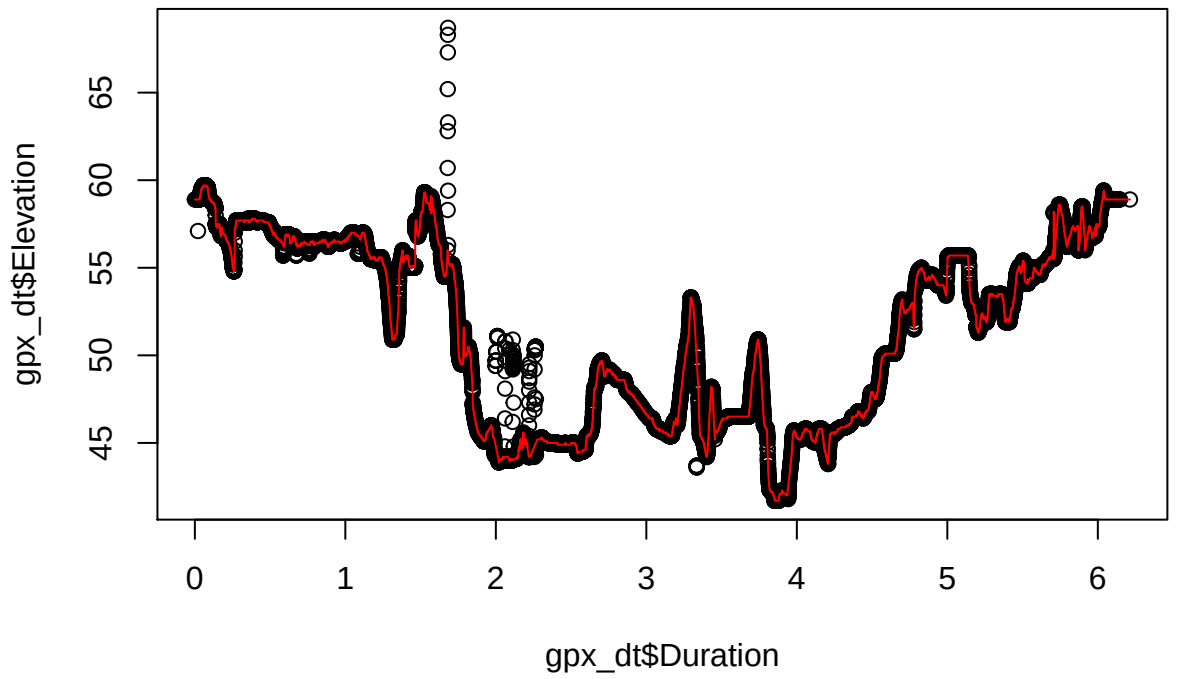
identify outliers and create new gpx data

```
# running median residuals
# rMed_win = 151 # window size of data points used for the running median
#
# rLat = gpx_dt$Latitude  - runmed(gpx_dt$Latitude,  rMed_win)
# rLon = gpx_dt$Longitude - runmed(gpx_dt$Longitude, rMed_win)

# loess residuals
ResThr = 0.0001
WinDur = 1 / 60 # loess Window duration in hours

NWinPts = WinDur/smpldur
Lspan   = NWinPts / Npts

rLon = loess(gpx_dt$Latitude  ~ gpx_dt$Duration, span=Lspan)$residuals
rLat = loess(gpx_dt$Longitude ~ gpx_dt$Duration, span=Lspan)$residuals

rDist = sqrt(rLat^2 + rLon^2)

# remove outliers

# ol = which(rDist > 10*mad(rDist))
ol = which(rDist > ResThr)

gpx_ol_rm = gpx_dt
gpx_ol_rm[ol, ] = NA

gpx_ol_rm = droplevels(subset(gpx_ol_rm, is.finite(gpx_ol_rm$Longitude)))

# define interpolation functions
nDuration = seq(from=0, to=max(gpx_dt$Duration), by=smpldur)

Lon_ip = approxfun(gpx_ol_rm$Duration, gpx_ol_rm$Longitude, rule=2)
Lat_ip = approxfun(gpx_ol_rm$Duration, gpx_ol_rm$Latitude,  rule=2)
Ele_ip = approxfun(gpx_ol_rm$Duration, gpx_ol_rm$Elevation, rule=2)

nLon = Lon_ip(nDuration)
nLat = Lat_ip(nDuration)
nEle = Ele_ip(nDuration)

# visualize
plot(gpx_dt$Longitude, gpx_dt$Latitude)
```

```
lines(nLon, nLat, col='red')
```

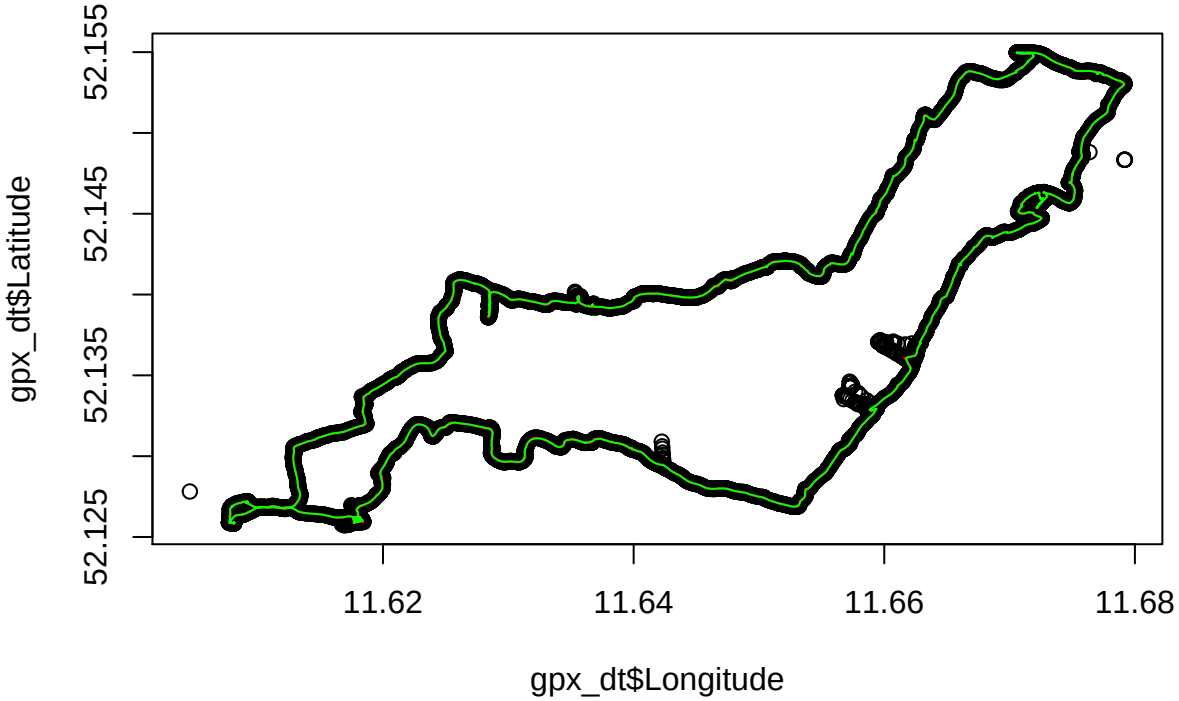



```
plot(gpx_dt$Duration,gpx_dt$Elevation)
lines(nDuration,nEle, col='red')
```

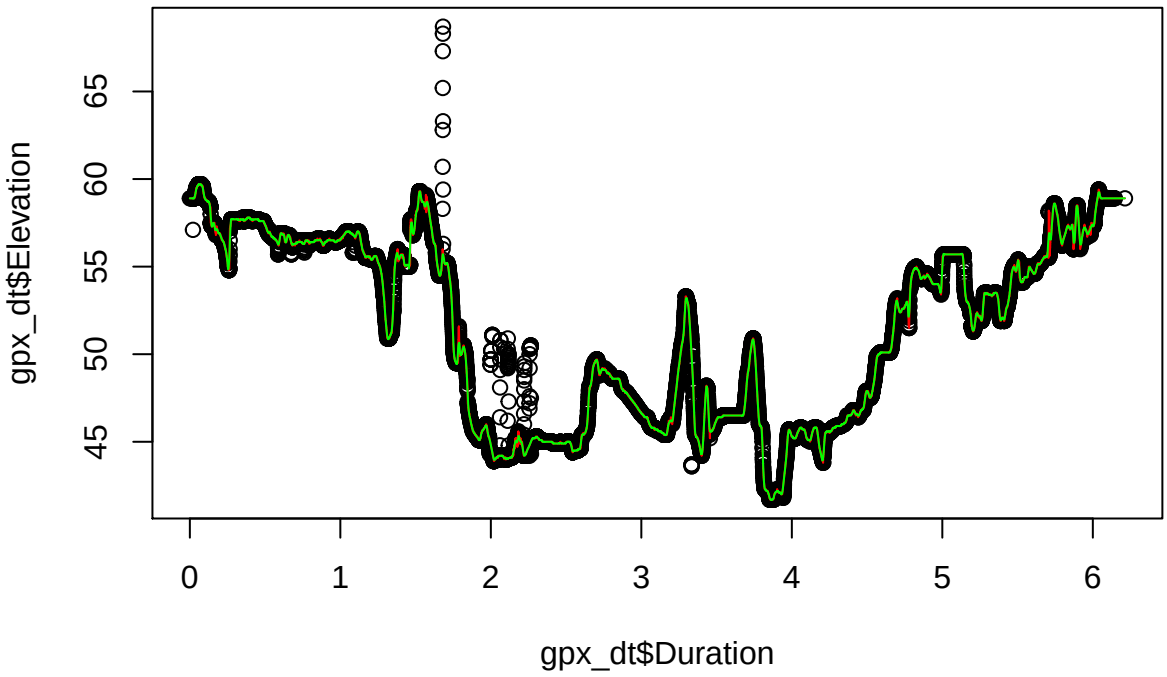


smooth the data

```
WinDur = 2 / 60 # loess Window duration in hours

NWinPts = WinDur/smpldur
Lspan = NWinPts / length(nDuration)

lLon = loess(nLon ~ nDuration, span=Lspan)$fitted
lLat = loess(nLat ~ nDuration, span=Lspan)$fitted
lEle = loess(nEle ~ nDuration, span=Lspan)$fitted

plot(gpx_dt$Longitude, gpx_dt$Latitude)
lines(nLon, nLat, col='red')
lines(lLon, lLat, col='green')
```




```
plot(gpx_dt$Duration,gpx_dt$Elevation)
lines(nDuration,nEle, col='red')
lines(nDuration,lEle, col='green')
```



write processed GPX file

define Write GPX

from: https://stackoverflow.com/questions/54726758/merging-multiple-gpx-files-into-a-single-gpx-file-with-multiple-tracks

```
writeGPX = function(Time, Longitude, Latitude, Elevation, file="file.gpx"){
  o = c('<gpx version="1.1" creator="R">','<trk>','<trkseg>')
  o = c(o, paste('<trkpt lat="', Latitude,'" lon="', Longitude,'"><time>',
           paste("<ele>", Elevation,"</ele>",sep=""),
           paste(gsub(' ','T', as.character(Time)), 'Z', sep=''),
           '</time></trkpt>', sep=''))

  o = c(o, '</trkseg>', '</trk>', '</gpx>')
  cat(o, file=file, sep='\n')
}
```

Write data

```
lTm = as.difftime(nDuration, units="hours") + sTm

writeGPX(lTm, lLon, lLat, lEle, file="gpx_cleaned.gpx")
```

just some plots to play with

```
#gpx_dt = readGPX(filename)
#gpx_raw <- readOGR(dsn = filename)
#pfile <- htmlTreeParse(filename, useInternalNodes = T)

gpx.raw = readGPS(i = "gpx", f = filename, type="w")
(layers = ogrListLayers(filename))
```

```
## [1] "waypoints"    "routes"       "tracks"       "route_points" "track_points"
## attr(,"driver")
## [1] "GPX"
## attr(,"nlayers")
## [1] 5
```

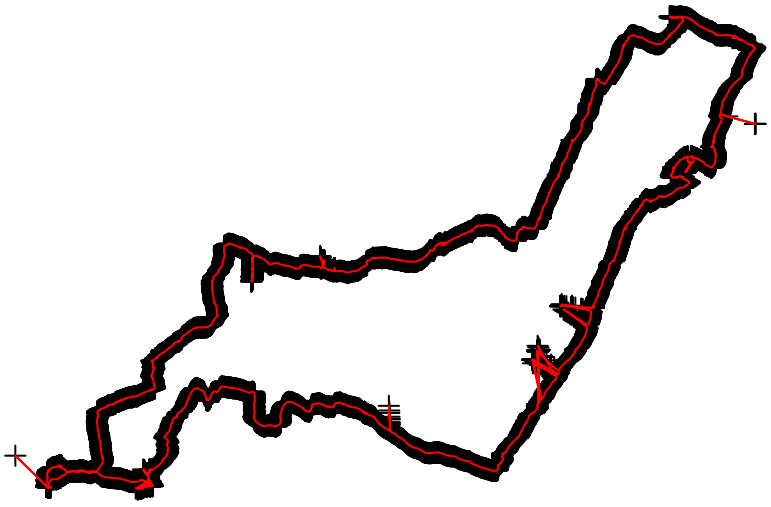
```
gp1 = readOGR(filename, layer=layers[5])
```

```
## OGR data source with driver: GPX
## Source: "/DATA/git_repos/wzinke/gpx_playground/gpx_clean/EM_Lunch_Walk.gpx", layer: "track_points"
## with 15114 features
## It has 26 fields
```

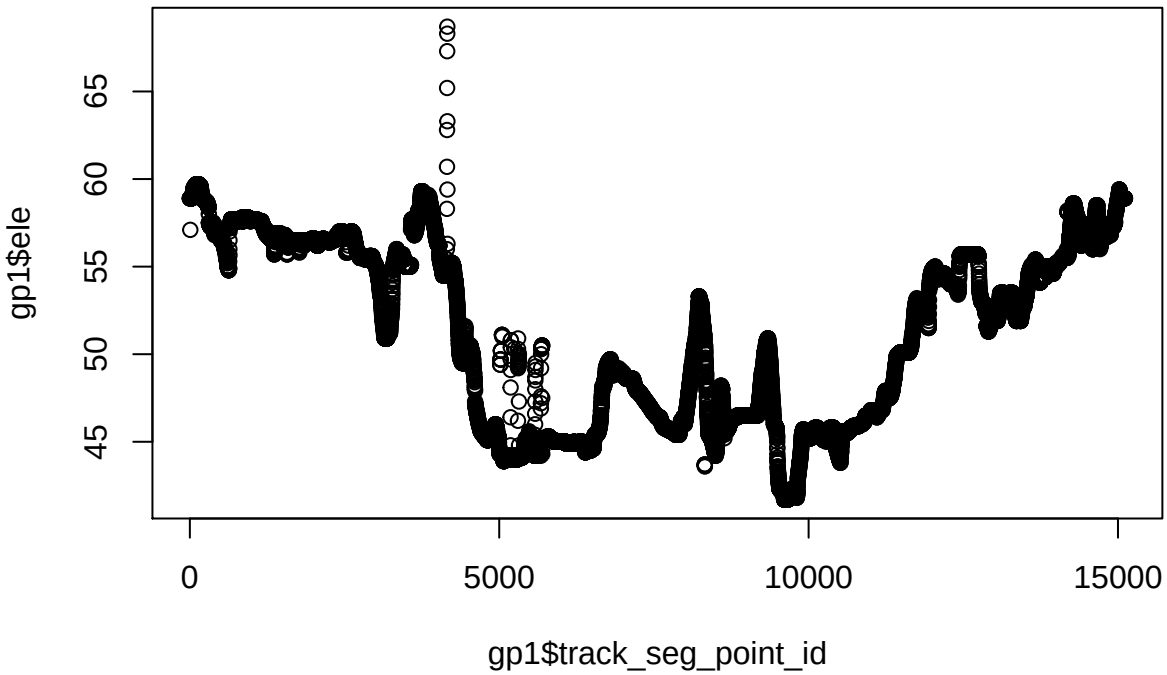
```
gt1 = readOGR(filename, layer=layers[3])
```

```
## OGR data source with driver: GPX
## Source: "/DATA/git_repos/wzinke/gpx_playground/gpx_clean/EM_Lunch_Walk.gpx", layer: "tracks"
## with 1 features
## It has 12 fields
```

```
plot(gp1)
plot(gt1, add = T, col = "red")
```



```
plot(gp1$track_seg_point_id, gp1$ele )
```




```
gp1.ppp = as(SpatialPoints(gp1), "ppp")
plot(density(gp1.ppp, adjust=0.5))
plot(gt1, add = T, col = "green")
```

density(gp1.ppp, adjust = 0.5)