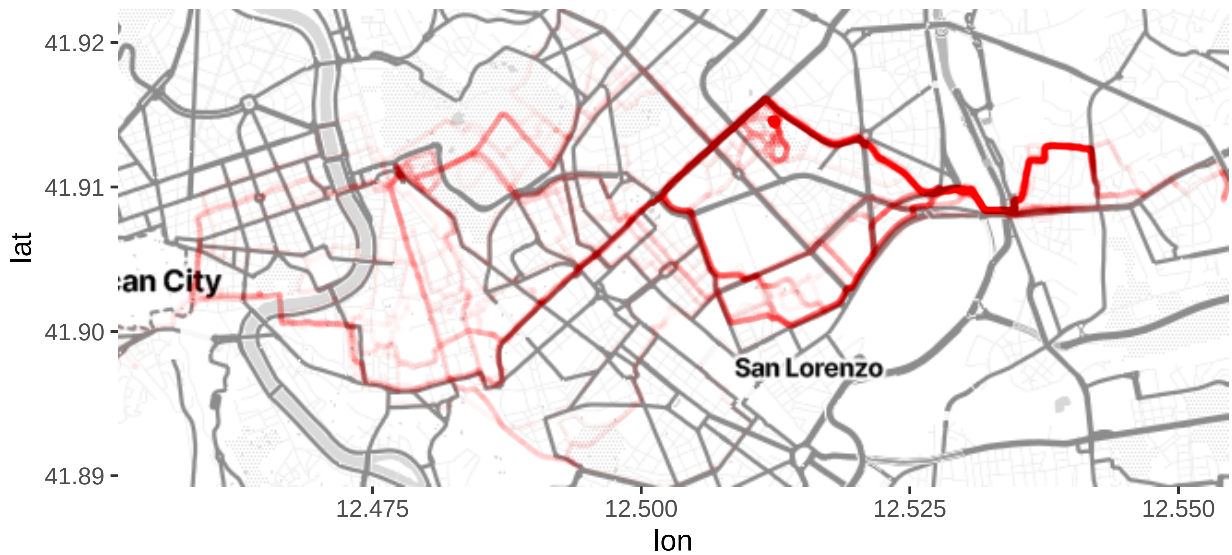
Statistical Learning

Due Sunday, May 12 on Moodle

Homework-01 | Exercise 02

Track me if you can...

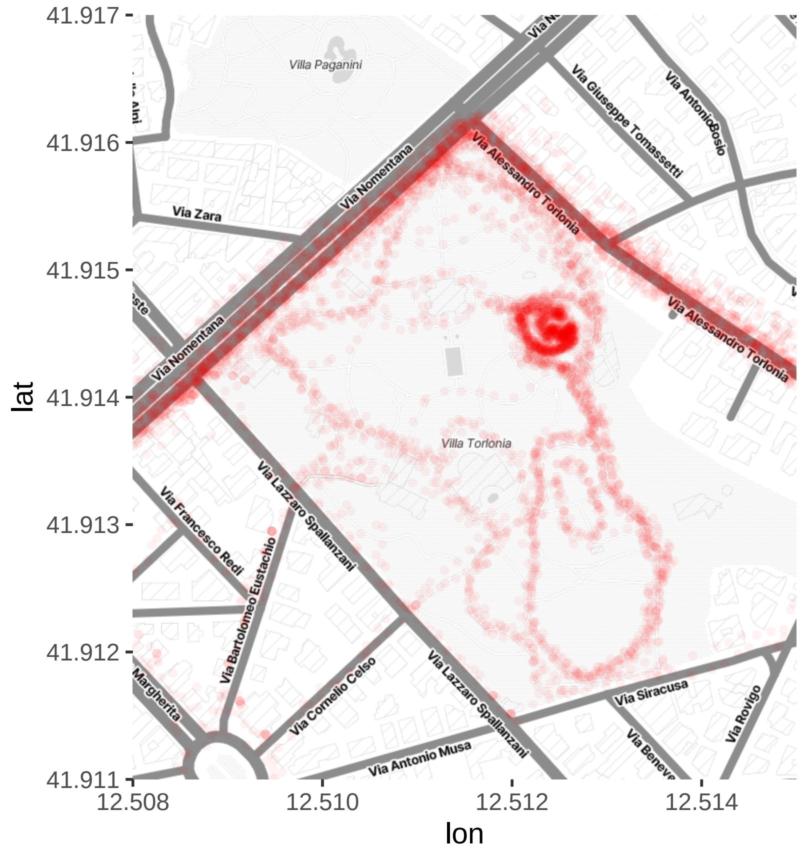
Personal data coming from (personal) tracking devices are massive these days. So, let's take a very... *personal*... point of view on the issue. The dataset `trackme.RData` collects the gps info of 60 of my (old) running sessions. To take a look we can use the `ggmap` package as follow:




```
# The package
# url: https://github.com/dkahle/ggmap
require(ggmap)

# Get the map from Stadia
# To retrieve the API (no credit card required), go here:
# url for API: https://client.stadiamaps.com/signup
# register_stadiamaps(key = "PLACE YOUR STADIA API HERE")

# The data
load("trackme.RData")

# Map boundaries
myLocation <- c(min(runtrack$lon, na.rm = T),
                min(runtrack$lat, na.rm = T),
                max(runtrack$lon, na.rm = T),
                max(runtrack$lat, na.rm = T))

# Get the map from Stadia
myMapInD <- get_map(location = myLocation, source = "stadia",
                    maptype = "stamen_toner_lite", zoom = 13)

# Plot gps coordinates (without elevation data)
gp <- ggmap(myMapInD) + geom_point(data = runtrack,
                                   aes(x = lon, y = lat),
```

```r
                                           size = .5, colour = I("red"), alpha = .01)
# Take a look
print(gp)
```

```r
# Zoom in / Restricted map boundaries
reLocation <- c(12.508, 41.911, 12.515, 41.917)
# Get the map from Google (default) and plot
reMapInD <- get_map(location = reLocation, source = "stadia",
                    maptype = "stamen_toner_lite")
ggmap(reMapInD) + geom_point(data = runtrack, aes(x = lon, y = lat),
                             size = 1, colour = I("red"), alpha = .05)
```



⇝ Your job ⇜

1. First of all, considering only the lon–lat information, treat `runtrack` simply as a 2D point cloud in the Euclidean space and use **any** method, `R` package and function you like to **estimate** and **visualize** properly and meaningfully the density of this data (…**boldface** is there for a reason). Please notice: this dataset is medium sized but you can still have problems of speed, be smart and do your best to get around them.

2. Can you figure out a way to single out the places where I run the most? One option is the **mean-shift**, can you do better? Conclude **commenting** your results. For the sake of completness, let us briefly review the *mean-shift* here. So, given data $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\} \overset{\text{IID}}{\sim} p$, we construct an estimate $\widehat{p}(\cdot)$ of the density. Let $\{\widehat{\boldsymbol{m}}_1, \ldots, \widehat{\boldsymbol{m}}_K\}$ be the estimated modes, and let $\{\widehat{\mathcal{A}}_k\}_{k=1}^{K}$ be the corresponding ascending manifolds implied by $\widehat{p}(\cdot)$. The <u>sample</u> or <u>empirical</u> clusters $\{C_1, \ldots, C_K\}$ are then defined as

$$C_k = \{\boldsymbol{X}_i \,:\, \boldsymbol{X}_i \in \widehat{\mathcal{A}}_k\}, \quad \forall\, k \in \{1, \ldots, K\}.$$

To be more specific, consider a (isotropic) kernel density estimator

$$\widehat{p}_h(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d}\mathsf{K}\left(\frac{\|\boldsymbol{x} - \boldsymbol{X}_i\|}{h}\right).$$

To locate the mode of $\widehat{p}_h(\boldsymbol{x})$ we use the *mean-shift algorithm* which finds modes by approximating the steepest ascent paths. The algorithm consists of the following steps:

Input: $\widehat{p}_h(\boldsymbol{x})$ and a mesh of points $A = \{\boldsymbol{a}_1, \dots, \boldsymbol{a}_N\}$ often taken to be the data points.

Inits: For each point $\boldsymbol{a}_j$, set $\boldsymbol{a}_j^{(0)} = \boldsymbol{a}_j$.

Iterate: Until convergence the following equation

$$\boldsymbol{a}_j^{(s+1)} \leftarrow \sum_{i=1}^{n} \left[\frac{\mathsf{K}\left(\frac{\|\boldsymbol{a}_j^{(s)} - \boldsymbol{X}_i\|}{h}\right)}{\sum_{r=1}^{n} \mathsf{K}\left(\frac{\|\boldsymbol{a}_j^{(s)} - \boldsymbol{X}_r\|}{h}\right)} \right] \boldsymbol{X}_i.$$

Output: $\widehat{\mathcal{M}} = \texttt{unique}\{\boldsymbol{a}_1^{(\infty)}, \dots, \boldsymbol{a}_N^{(\infty)}\}$.

The result of this algorithm is then the set of estimated modes $\widehat{\mathcal{M}}$ together with the actual clustering: the mean-shift algorithm, in fact, shows us exactly what mode each point is attracted to![1]

3. Let's be honest here: those red dots are not really dots, they are part of a single track, a single curve in the plane. Let's look at two of them:

```
# Plot gps coordinates (without elevation data)
runsmall <- subset(runtrack, id %in% c("run_1", "run_36"))
gp2 <- ggmap(myMapInD) + geom_path(data = runsmall,
                                   aes(x = lon, y = lat, col = id),
                                   size = 1.5, lineend = "round",
                                   alpha = .6)
# Take a look
print(gp2)
```

- We are in the realm of what is now called **functional data analysis**, a broad name to denote a branch of statistics that starts from the very idea that a single datapoint, a single observation is **not** just a number or a label as we use to think, but a more complex beast, *usually* a whole function that we observe sampled at a given frequency.
  - A single observation X then, is *not* just high dimensional, it is **infinite** dimensional: in our case each X_i is a *curve*. An immediate problem is that the very concept of a density $p(\cdot)$ over such a space is no longer well defined…sooooo, can you figure out a way to find the **top-5 paths** with the highest chance to be run?

In the end, whatever you do: comment, comment, comment, comment, comment!

---

[1] More formally, the mean-shift is simply tracing the so called *gradient flow*. The flow lines lead to the modes and define the clusters. In general, a *flow* is a map $\phi : \mathbb{R}^d \times \mathbb{R} \mapsto \mathbb{R}^d$, where the second argument can be interpreted as time, such that: 1. $\phi(\boldsymbol{x}, 0) = \boldsymbol{x}$; 2. $\phi\big(\phi(\mathbf{x}, t), s\big) = \phi(\mathbf{x}, s + t)$. The latter is called the *semi–group property*.