Sometimes, geographic information is given as street intersections rather than GPS coordinates or street addresses with house numbers. Intersections can be geocoded with Google Maps (Limited free usage with API), ArcGIS (paid software) or other web services which parse text and return coordinates. If you are lucky, there may be a government geocoding service for free.

On the other hand, you can perform your own geocoding of intersections by taking advantage of the detailed Open Street Maps data accessed with `osmdata`.

With `osmdata`, it is possible to download specific features of an area (buildings, shopping locations, streets etc) separately rather than the entire map. Here, all highways for Vancouver are downloaded and the resulting data is transformed into an `osmdata` object containing `simple features` data frames for manipulation and plotting with the `sf` package.

```
library(osmdata)
```

```
## Warning: package 'osmdata' was built under R version 3.6.3
```

```
## Data (c) OpenStreetMap contributors, ODbL 1.0. https://www.openstreetmap.org/copyright
```

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 3.6.3
```

```
## Linking to GEOS 3.8.0, GDAL 3.0.4, PROJ 6.3.1
```

```
vancouver_highways <- opq(bbox = 'Vancouver Canada') %>%
  add_osm_feature(key = 'highway') %>%
  osmdata_sf()
```

```
vancouver_highways
```

```
## Object of class 'osmdata' with:
##                  $bbox : 49.1984452,-123.2249611,49.3161714,-123.0232419
##         $overpass_call : The call submitted to the overpass API
##                 $meta : metadata including timestamp and version numbers
##            $osm_points : 'sf' Simple Features Collection with 88427 points
##             $osm_lines : 'sf' Simple Features Collection with 21747
linestrings
##          $osm_polygons : 'sf' Simple Features Collection with 456 polygons
##        $osm_multilines : NULL
##     $osm_multipolygons : 'sf' Simple Features Collection with 18
multipolygons
```

In OSM, highways are usually ordered as a list of nodes (points) or a “way”, equivalent to lines in the `osmdata` object. All highways in Vancouver can then be plotted with a call to `osm_lines` of the `osmdata` object.

```
plot(vancouver_highways$osm_lines$geometry)
```

Through fuzzy matching and use of `sf` functions, it then becomes possible to geocode an intersection. In this case, the intersection to be geocoded is Alma St at West 11 Ave (see on Google Maps. I found the intersection and used "What's Here" to get the coordinates).

I chose this intersection specifically because OSM cannot locate the intersection easily with a text search. OSM is very good at finding intersections if a bus stop/transit stop is named after the intersection.

The first step is to find one of the streets from the `osmdata` object. Street names are located in the `name` column of `osm_lines`.

```
head(vancouver_highways$osm_lines$name)
```

```
## [1] "West King Edward Avenue" "Granville Street"
## [3] "East 12th Avenue"        "South Grandview Highway"
## [5] "East Broadway"           "Kingsway"
```

Matching the name directly can fail if the formatting isn't correct. For example:

```
which(vancouver_highways$osm_lines$name == "Alma St")
```

```
## integer(0)
```

OSM appears to use no abbreviations at all.

```
which(vancouver_highways$osm_lines$name == "Alma Street")
```

```
##  [1]   109   154   193  1596  1748  1749  4257  6671  6672  6906  7180 12029
## [13] 15445
```

Using "street" rather than "st" works perfectly. However, abbreviations are likely to be frequent in data, and to avoid replacing every possible abbreviation (NSEW, Rd, Blvd etc), a fuzzy match is a good solution, especially with possible misspellings.

```
row_index <- agrep("Alma St", vancouver_highways$osm_lines$name)
```

We can plot these matching lines and see that Alma Street is composed of several disconnected lines which reflect the real Alma Street.

```
plot(vancouver_highways$osm_lines$geometry[row_index])
```

We first use `osm_lines` to find all the other lines (streets) that intersect with Alma Street to narrow down our candidates for the second street in the intersection. We pass in the OSM id of the Alma Street lines.

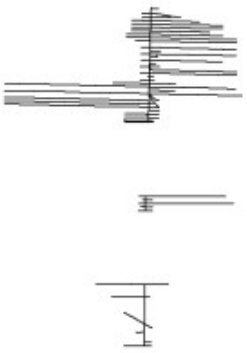
```
potential_intersections <- osm_lines(vancouver_highways,
rownames(vancouver_highways$osm_lines[row_index,]))
```

```
dim(potential_intersections)
```

```
## [1]  82 224
```

There are 82 streets which intersect with Alma Street.

```
plot(potential_intersections$geometry)
```



We can then use fuzzy matching again for “West 11th Avenue”.

```
candidate_streets <- agrep("West 11th Avenue", potential_intersections$name)
```

```
candidate_streets
```

```
##  [1]  1  4  5  7  8 10 12 13 15 24 35 37 38 47 48 56 57 58 63 64 65 67 70 71
72
## [26] 74 75 78 82
```

There are a lot of potential matches. `agrep` returns so many because of the similarity of names.

```
candidate_streets <- potential_intersections[candidate_streets,]
candidate_streets$name
```

```
##  [1] "West 12th Avenue" "West 15th Avenue" "West 28th Avenue" "West 14th
Avenue"
##  [5] "West 27th Avenue" "West 48th Avenue" "West 6th Avenue"  "West 5th
Avenue"
##  [9] "West 13th Avenue" "West 7th Avenue"  "West 4th Avenue"  "West 1st
Avenue"
## [13] "West 1st Avenue"  "West 4th Avenue"  "West 16th Avenue" "West 7th
Avenue"
## [17] "West 6th Avenue"  "West 5th Avenue"  "West 11th Avenue" "West 12th
Avenue"
## [21] "West 11th Avenue" "West 16th Avenue" "West 13th Avenue" "West 10th
Avenue"
## [25] "West 10th Avenue" "West 8th Avenue"  "West 8th Avenue"  "West 29th
Avenue"
## [29] "West 39th Avenue"
```

Since the street names are so similar, we can compute the Levenshtein distance with `adist`.

```
distances <- adist("West 11th Avenue", candidate_streets$name)

matches <- which(distances == min(distances))
matches
```

```
## [1] 19 21
```

Using the row indexes, the matches can be extracted. The two matches appear to create a connecting line.

```
intersections <- candidate_streets[matches,]
plot(intersections$geometry)
```

We can then find the intersection of West 11th Avenue and Alma with the `st_intersection` function from the `sf` package.

```
coordinates <- st_intersection(vancouver_highways$osm_lines[row_index,],
intersections)
```

```
## although coordinates are longitude/latitude, st_intersection assumes that
they are planar
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
```

```
all
## geometries

coordinates$geometry

## Geometry set for 2 features
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: -123.1859 ymin: 49.26257 xmax: -123.1859 ymax: 49.26257
## geographic CRS: WGS 84

## POINT (-123.1859 49.26257)

## POINT (-123.1859 49.26257)

plot(vancouver_highways$osm_lines$geometry[row_index])
plot(intersections$geometry, add = TRUE)
plot(coordinates$geometry, add = TRUE)
```

There are two points of intersection which overlap, possibly because both lines are West 11 Avenue line segments which intersect with Alma from opposite directions at the same point.

The predicted intersection is (-123.1859 49.26257) which is very close to the manual marker at (-123.185954 49.262582).

Here is the intersection plotted against all the streets with text labels.

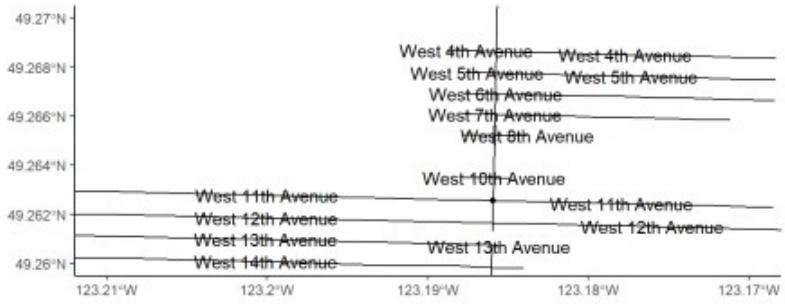
```
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.6.3

ggplot() +
  lims(y = c(49.26, 49.27), x = c(-123.21, -123.17)) +
  geom_sf(data = vancouver_highways$osm_lines[row_index,]) +
  geom_sf(data = candidate_streets) +
  geom_sf_text(data = candidate_streets, aes(label = name), check_overlap =
TRUE) +
  geom_sf(data = coordinates) +
  theme_classic() + labs(x = "", y = "")

## Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
## give correct results for longitude/latitude data

## Warning: Removed 10 rows containing missing values (geom_text).
```



This example can be easily modified for batch geocoding. For example, `agrep` can be replaced with `adist`. `adist` can return a matrix if two vectors are provided. Then it would be possible to find the first street of multiple intersections in one function. Using optimized versions of functions such as `min_max` and `rowMins` from the `Rfast` package, and `stringdist` can also speed up the code.

One caveat about this method is having to know the general area for the geocoding. If the area is large, such as a country, the chances of getting a wrong match is higher. The OSM data can also get big in memory depending on the area. For example, `vancouver_highways` is approximately 220 Mb. Fortunately, it is possible to save the `osmdata` object to a file for future uses.
