Exploratory Data Analysis of Airbnb Listings in Berlin

Silvia Ventoruzzo

Ladislaus von Bortkiewicz Chair of Statistics
Humboldt–Universität zu Berlin
http://lvb.wiwi.hu-berlin.de

Outline

Objectives

- Airbnb is a website where private people, and commercial entities, can rent apartments, or part of them.
- Aim of the project:
  - Answer the following questions:
    - How are the Airbnb properties distributed in Berlin?
    - Which variables drive the property price?
  - Display the information through a ShinyApp

Creating Polygons for the districts and neighbourhoods

- Loading the shapefile of Berlin neighbourhoods: (downloaded from www.statistik-berlin-brandenburg.de)
- For the neighbourhoods we just need to transform the SpatialPolygonsDataFrame into an sf object.

```
berlin_neighbourhood_sf <- berlin %>%
  sf::st_as_sf() %>%
  sf::st_set_crs(proj4string(berlin)) %>%
  dplyr::rename(id    = Name,
                group = BEZNAME) %>%
  dplyr::select(id, group, geometry) %>%
  dplyr::arrange(group) %>%
  dplyr::mutate(view = "Neighbourhoods")
```

- The neighbourhood Buckow is composed of two separate parts, which we unite by grouping the neighbourhoods by their id.

```
berlin_neighbourhood_singlebuckow_sf <-
  berlin_neighbourhood_sf %>%
  dplyr::group_by(id, group, view) %>%
  dplyr::summarize(do_union = TRUE) %>%
  dplyr::arrange(group)
```

- To get the districts we unite the neighbourhoods by their district (group).

```
berlin_district_sf <- berlin_neighbourhood_sf %>%
  dplyr::group_by(group, view) %>%
  dplyr::summarize(do_union = TRUE) %>%
  dplyr::mutate(id    = group,
                view  = "Districts")
```

Mapping the polygons

- Districts:

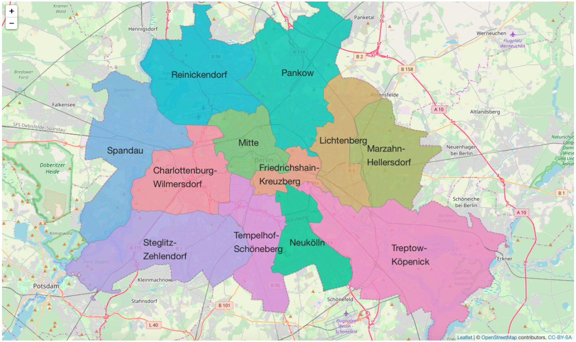

Figure 1: Leaflet map of Berlin Districts

- Neighbourhoods:

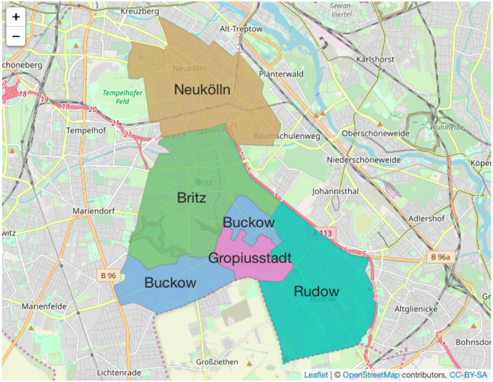

Figure 2: Leaflet map of Neukölln's Neighbourhoods

Creating Polygons of the VBB Areas

- Loading the shapefile of the stations and stops in Berlin: (downloaded from www.geofabrik.de)
- Create dataframe with the information for train/subway stations.

```
bahn_stations_df <- stations %>%
  base::as.data.frame() %>%
  dplyr::filter(fclass %like% "railway") %>%
  dplyr::rename(id    = name,
                long  = coords.x1,
                lat   = coords.x2) %>%
```

```r
dplyr::mutate(id      = gsub("Berlin ", "", id),
              id      = gsub("Berlin-", "", id),
              id      = gsub(" *\\(.*?\\) *", "",
                             id),
              s_bahn  = ifelse(startsWith(id,
                               "S "), TRUE, FALSE),
              u_bahn  = ifelse(startsWith(id,
                               "U "), TRUE, FALSE),
              id      = gsub("S ", "", id),
              id      = gsub("U ", "", id)) %>%
dplyr::group_by(id) %>%
dplyr::summarize(long   = mean(long),
                 lat    = mean(lat),
                 s_bahn = any(s_bahn),
                 u_bahn = any(u_bahn))
```

- Create dataframe with names and order position of stations that form the Ringbahn (VBB Area A).

```
ringbahn_names_df <- base::data.frame(
    id = c("Südkreuz", "Schöneberg",
        "Innsbrucker Platz", "Bundesplatz",
        "Heidelberger Platz", "Hohenzollerndamm",
        "Halensee", "Westkreuz", "Messe Nord/ICC",
        "Westend", "Jungfernheide",
        "Beusselstraße", "Westhafen", "Wedding",
        "Gesundbrunnen", "Schönhauser Allee",
        "Prenzlauer Allee", "Greifswalder Straße",
        "Landsberger Allee", "Storkower Straße",
        "Frankfurter Allee", "Ostkreuz",
        "Treptower Park", "Sonnenallee",
        "Neukölln", "Hermannstraße",
        "Tempelhof", "Südkreuz"),
        stringsAsFactors = FALSE) %>%
```

```
    tibble::rownames_to_column(var = "order") %>%
    dplyr::mutate(order = as.numeric(order))
```

- Create the sf object of the VBB Areas by binding the Area A with the complete-Berlin polygon and finding their interaction.

```
vbb_AB_sf <- bahn_stations_df %>%
  dplyr::right_join(ringbahn_names_df,
                    by = "id") %>%
  dplyr::arrange(order) %>%
  dplyr::select(long, lat) %>%
  base::as.matrix() %>%
  base::list() %>%
  sf::st_polygon() %>%
  sf::st_sfc() %>%
  sf::st_sf(crs = proj4string(berlin)) %>%
```

```
base::rbind(berlin_all_sf) %>%
sf::st_intersection() %>%
```

- Area A: interaction (n.overlaps $>$ 1)
- Area B: no interaction (n.overlaps $=$ 1)

```
dplyr::mutate(id    = ifelse(n.overlaps > 1,
                             "A", "B"),
              view  = "VBB Areas",
              group = id) %>%
dplyr::select(-n.overlaps, -origins) %>%
dplyr::arrange(desc(id))
```

Mapping the VBB Areas

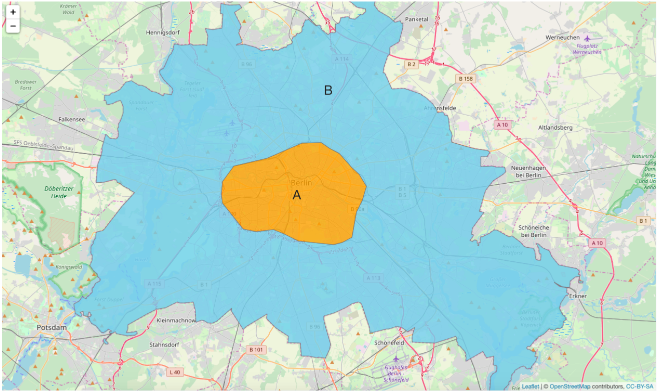

Figure 3: Leaflet map of VBB Areas

Initial variables

- Loading the following files (dowloaded from www.insideairbnb.com):
  - *listings.csv.gz:* Detailed Listings data for Berlin
  - *listings.csv:* Summary information and metrics for listings in Berlin
  - *calendar.csv.gz:* Detailed Calendar Data for listings in Berlin
- Last update: 7th November 2018
- Keeping the following variables:
  - *id:* Reference to the listing
  - *long, lat:* Coordinates of the listing
  - *price:* Price per night in dollars
  - *security_deposit_yn, cleaning_fee_yn:* If extra money will be asked as deposit and/or for cleaning
  - *property_type, room_type:* Type of accommodation

- *accommodates, bedrooms, beds:* Important for the number of people who could stay in the property
- *minimum_nights:* How many night at least a person has to book
- *reviewed_yn, number_of_reviews, review_scores_rating:* If a listing has been reviewed, by how many peope and what its average scoring is
- *availability_30, availability_60, availability_90, availability_365:* Availability in days in the next 30, 60, 90 and 365 days
- *listing_url:* Website of the specific listing

New variables

- district, neighbourhood, vbb_area:

```
listings <- point_in_polygons(
                points_df = listings,
                polys_sf  = berlin_neighbourhood_sf,
                var_name  = "neighbourhood")
```

- station_count, station_dist, attraction_count, attraction_dist:

```
listings <- distance_count(
                main      = listings,
                reference = bahn_stations_df,
                var_name  = "station",
                distance  = 1000)
```

Functions

- point_in_polygons:

```r
point_in_polygons <-
    function(points_df, polys_sf, var_name) {
        is_in <- data.frame(
                        matrix(ncol = nrow(polys_sf),
                               nrow = nrow(points_df)))
        is_in[,var_name] <- NA
        coordinates <- as.data.frame(st_coordinates(
          polys_sf))
        name <- polys_sf$id
```

```
    for (k in 1:nrow(polys_sf)) {
        is_in[,k] <- sp::point.in.polygon(
                    point.x = points_df$long,
                    point.y = points_df$lat,
                    pol.x   = coordinates$X
                              [coordinates$L2 == k],
                    pol.y   = coordinates$Y
                              [coordinates$L2 == k])
        is_in[,var_name][is_in[,k] == 1] <- name[k]
    }
    is_in <- is_in %>%
    dplyr::select(var_name) %>%
    dplyr::mutate(id = points_df$id)
    points_df <- dplyr::full_join(points_df, is_in,
                                  by = "id")
    return(points_df)
  }
```

- distance_count:

```
distance_count <- function(main, reference,
                           var_name, distance) {
  var_name_count <- paste(var_name, "count",
                          sep = "_")
  var_name_dist <- paste(var_name, "dist",
                         sep = "_")
  point_distance <- geosphere::distm(
                        x   = main %>%
                              dplyr::select(long,
                                            lat),
                        y   = reference %>%
                              dplyr::select(long,
                                            lat),
                        fun = distHaversine) %>%
    as.data.frame() %>%
    data.table::setnames(as.character(reference$id))
```

```r
  point_distance[,var_name_count] <- rowSums(
    point_distance <= distance)
  point_distance[,var_name_dist] <- apply(
        point_distance[,-ncol(point_distance)],
        MARGIN = 1, FUN = min) %>% round(0)
  main <- point_distance %>%
    dplyr::mutate(id = main$id) %>%
    dplyr::select(id,
                  var_name_count, var_name_dist) %>%
    dplyr::right_join(main, by = "id")
  return(main)
}
```

Correlation with price

- Correlation of each variable with price.

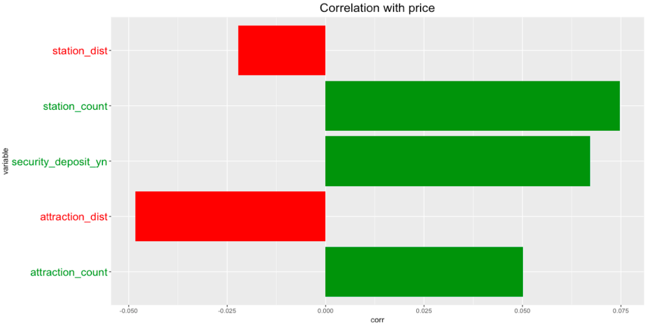

Figure 4: Correlation plot

Tab Berlin map

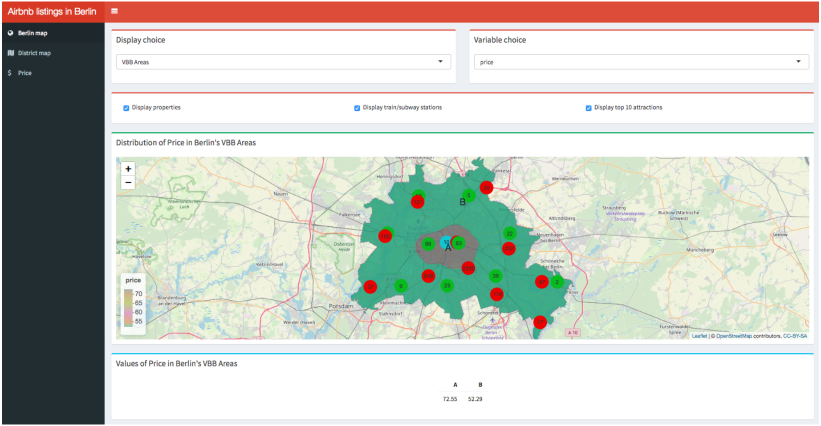

Figure 5: Tab Berlin map

Tab District map

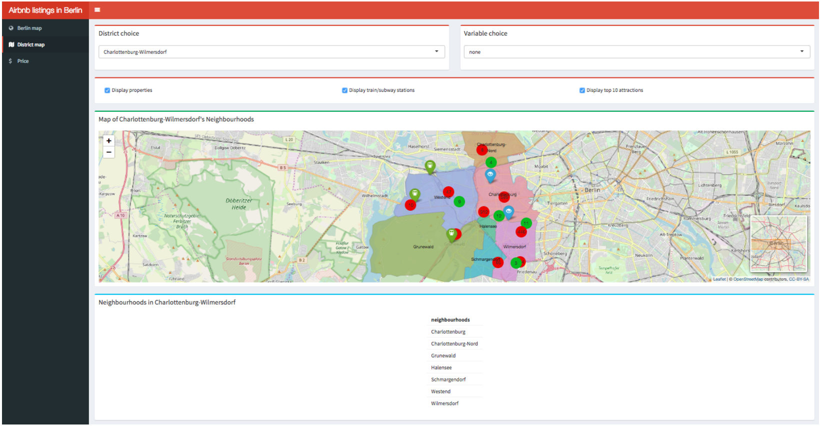

Figure 6: Tab District map

Tab Price

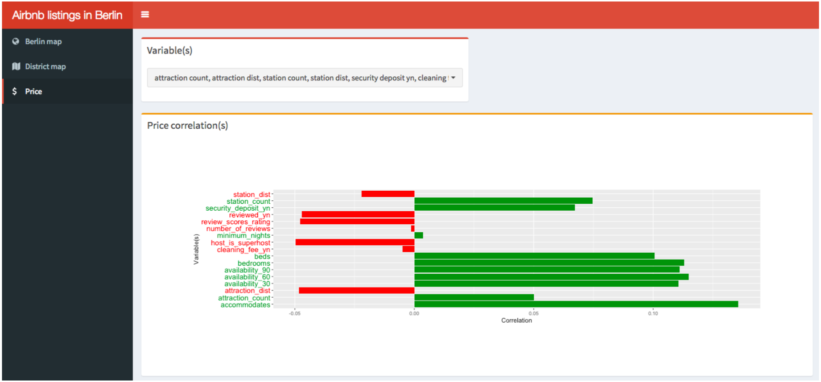

Figure 7: Tab Price