An Analysis of Customer Reviews on Airbnb

Marcell P. Granát & Zója M. Szabó

March 9, 2021

Contents

Abstract

Public perception of shared goods and/or services has changed significantly in the last few years. Shared accommodations have gained so great popularity, that house and flat sharing platforms like Airbnb now rival some of the world's largest businesses in hospitality. Sharing of personal properties provides an opportunity for owners to lower the transaction costs of operating short-term rentals and online rental marketplaces connect people who want to rent out their dwellings with the ones who are looking for accommodations. This study is aimed at determining the perceived behavior of individuals choosing Airbnb and introducing what factors influence user ratings and consumer adoption of Airbnb while assuming that customer feedbacks significantly influence consumer choice. We also analyze the market trends of the Hungarian Airbnb accommodations as primary examples of sharing or collaborative economy. Weekly data was collected for the Hungarian accommodation establishments all over the country. We aimed to build a complete dataset of the active suppliers by using automated "web scraping" techniques during a certain window of time. Our database contained customer ratings, reviews and pieces of public information concerning the rooms. We performed an advanced text analysis of the variables mentioned previously.

List of Tables

List of Figures

Introduction

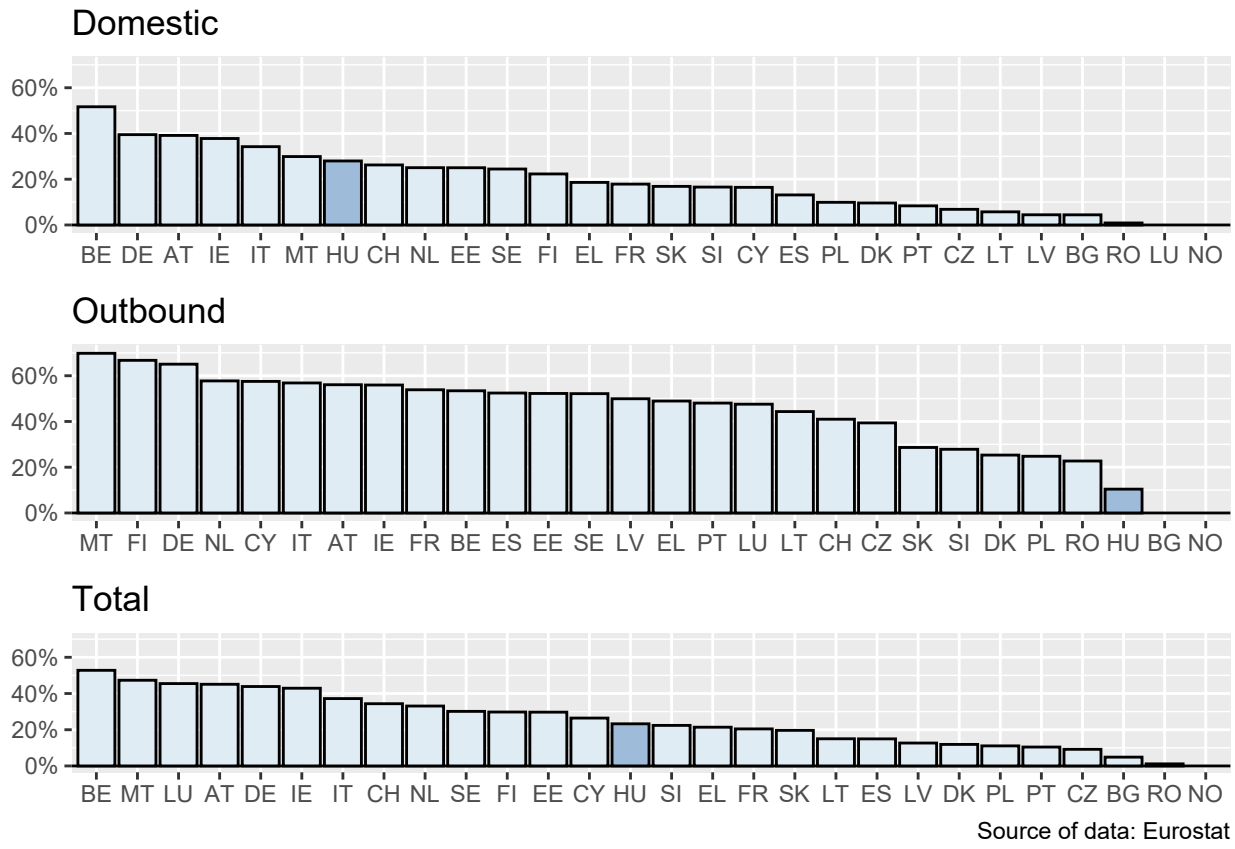

Figure 1: Proportion of internet bookings of the main means of accommodation by countries and the partner type

[leíró statok: eu bizottság felhívása]

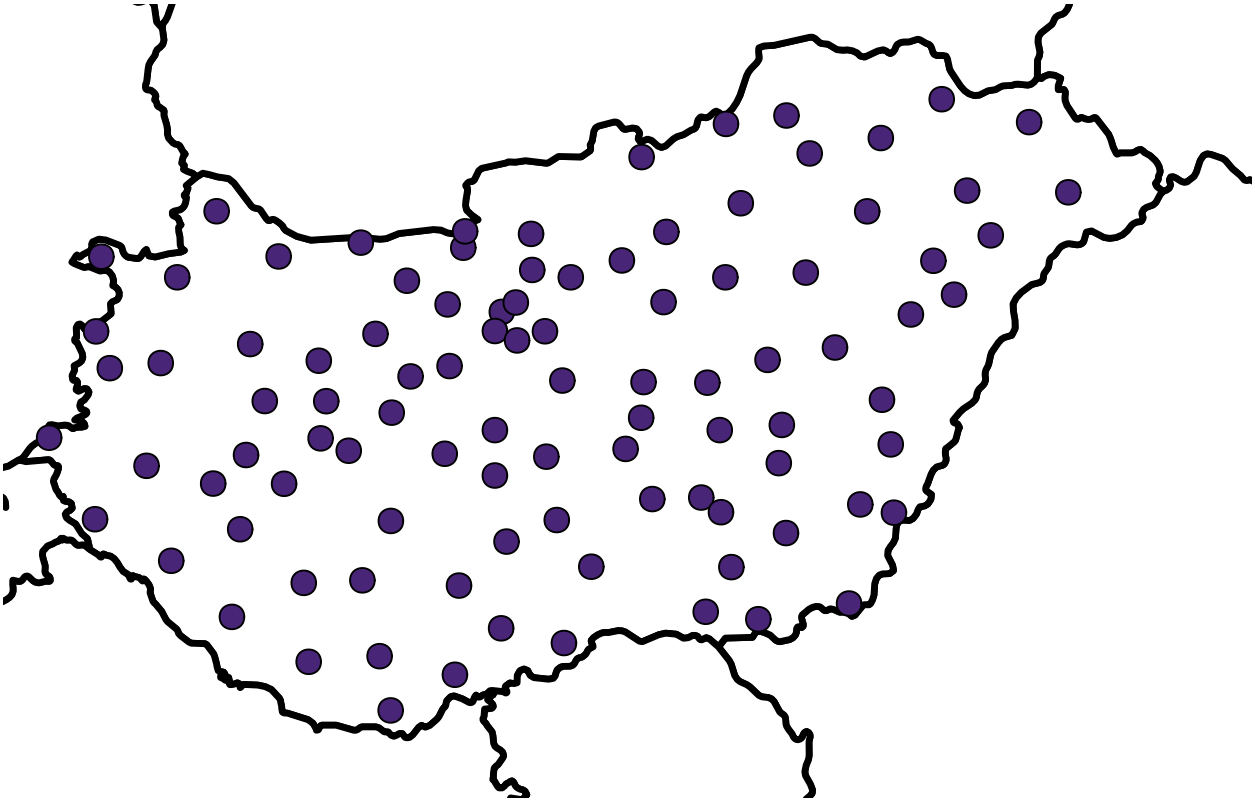

Figure 2: Starting points to our scrapping algorithm

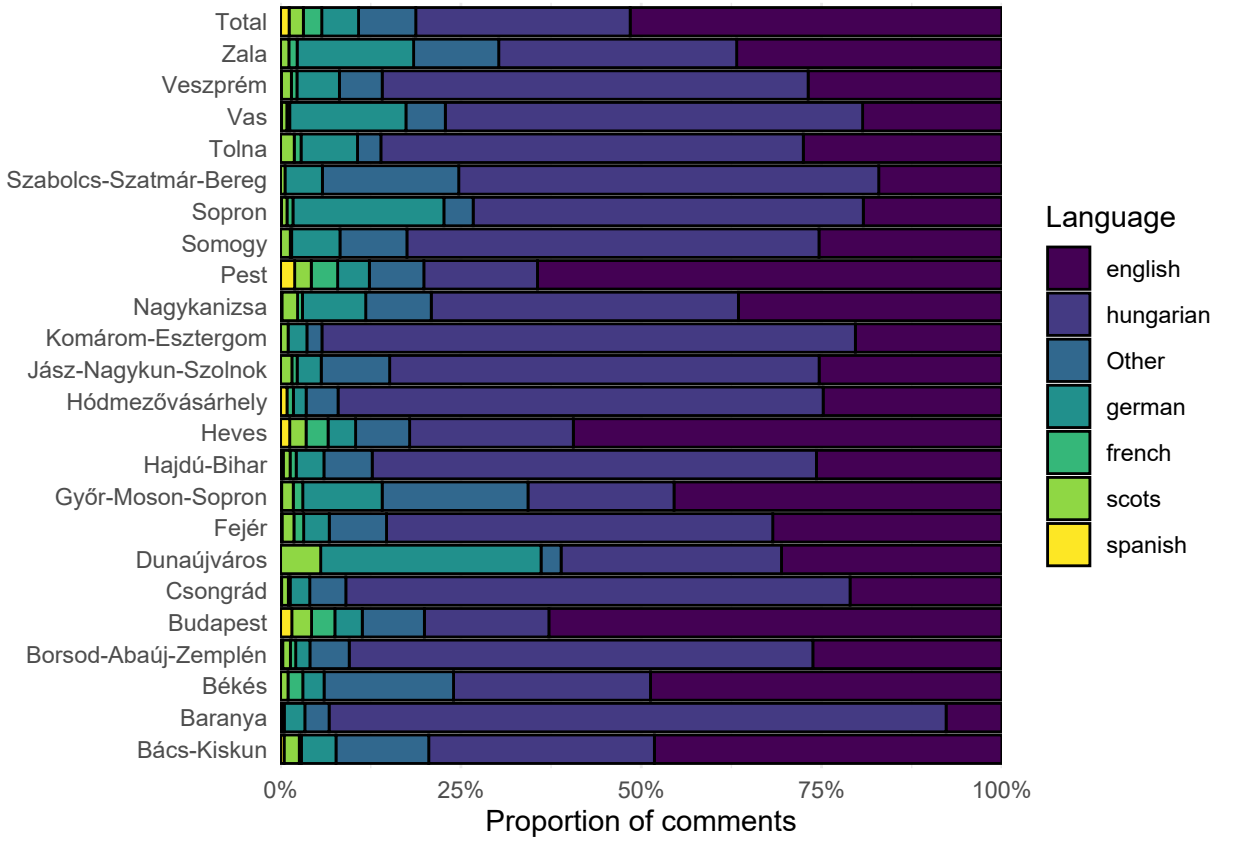

Figure 3: Most common languages found in the comments by counties

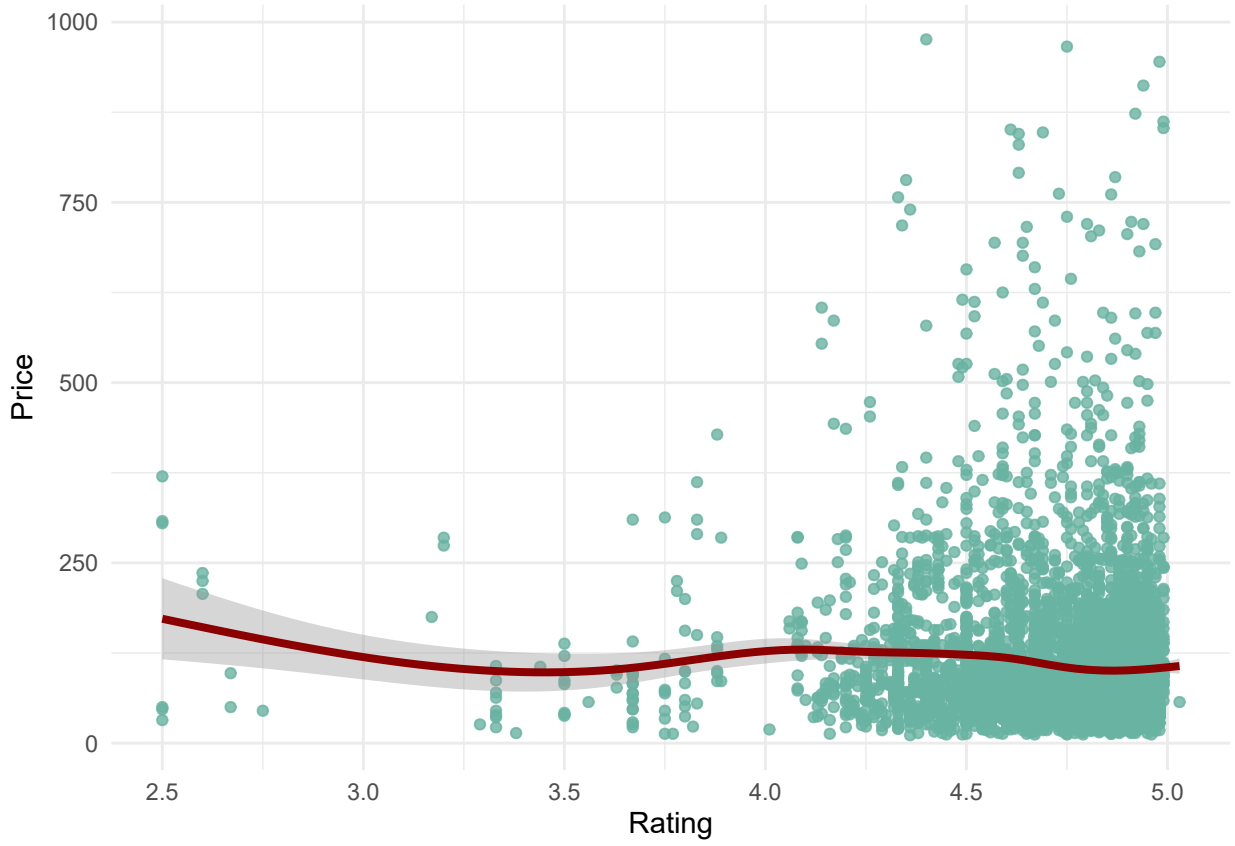

Figure 4: Scatter plot of overall ratings and prices

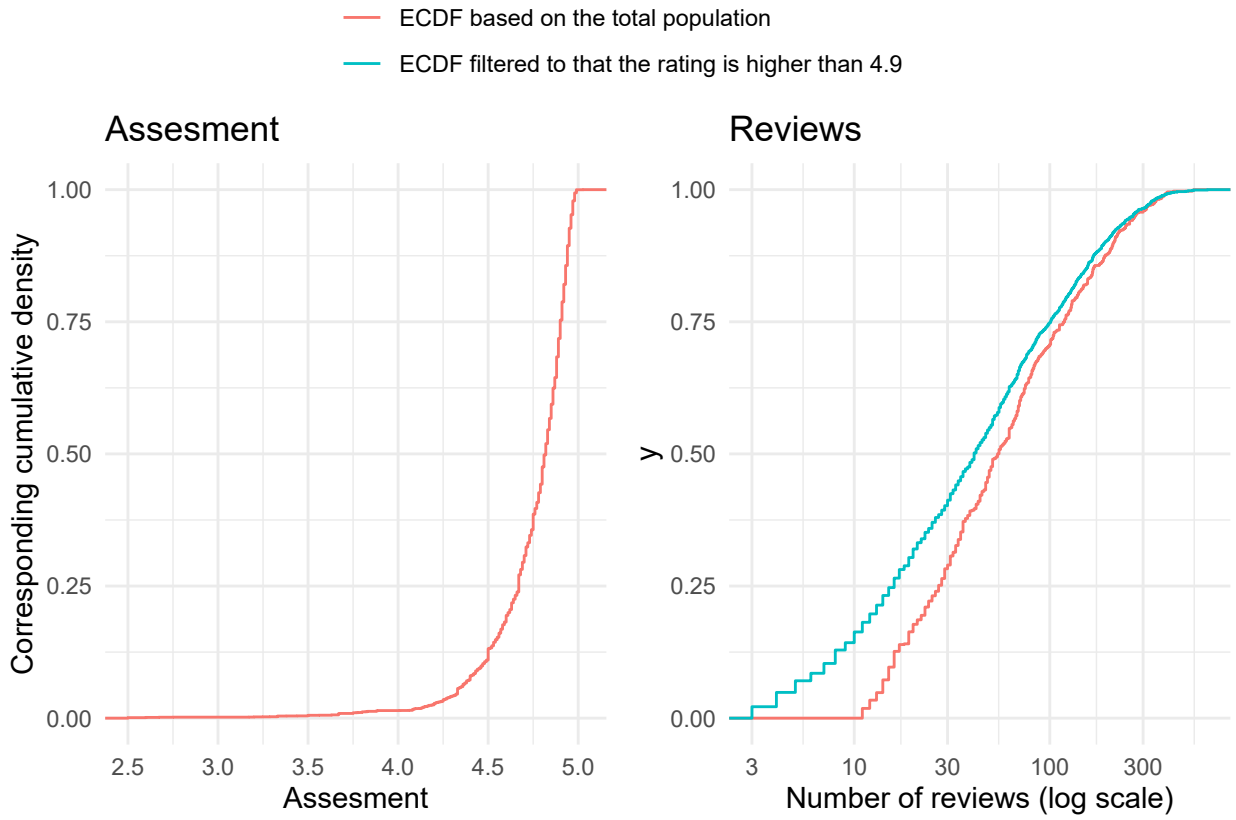

Figure 5: Empirical cumulative distribution functions of overall rating scores and the number of reviews

Theoretical consideration

@hornik2013

Data

Explore data

Modell building

Consequences

Summary

References

Hornik, K., Mair, P., Rauch, J., Geiger, W., Buchta, C., and Feinerer, I. (2013). The textcat package for n-gram based text categorization in r. *Journal of Statistical Software*, 52(6).

Silge, J. and Robinson, D. (2017). *Text mining with R: A tidy approach*. O'Reilly Media, Sebastopol, CA.

Zervas, G., Proserpio, D., and Byers, J. W. (2017). The rise of the sharing economy: Estimating the impact of airbnb on the hotel industry. *Journal of Marketing Research*, 54(5):687–705.

Appendix: R codes

```r
# Packages ---------------------------------------------------------------------

library(tidyverse)
library(knitr)
library(tidytext)
library(rvest)
library(parallel)
library(RSelenium)
library(rnaturalearth)
library(rnaturalearthdata)

# Introduction -----------------------------------------------------------------

f.plot_eurostat <- function(x){
  eurostat::get_eurostat('tour_dem_ttorg', time_format = 'num') %>%
    filter(trip_arr %in% c('ACC_WEB', 'TOTAL') & duration == 'N1-3' & time == 2017 &
             purpose == 'TOTAL') %>%
    pivot_wider(names_from = trip_arr, values_from = values) %>%
    mutate(
      value = ACC_WEB/TOTAL
    ) %>%
    filter(partner == x) %>%
    mutate(geo = fct_reorder(geo, -value)) %>%
    ggplot() +
    aes(geo, value, fill = geo == 'HU') +
    geom_hline(yintercept = 0) +
    geom_col(color = 'black') +
    scale_fill_brewer(palette = 3, guide = F) +
    scale_y_continuous(labels = scales::percent, limits = c(0, .7)) +
    labs(
      x = NULL, y = NULL, title = case_when(
        x == 'DOM' ~ 'Domestic',
        x == 'OUT' ~ 'Outbound',
        T ~ 'Total'
      )
    )
}

ggpubr::ggarrange(
  f.plot_eurostat('DOM'),
  f.plot_eurostat('OUT'),
  f.plot_eurostat('WORLD') +
    labs(caption = 'Source of data: Eurostat'),
  ncol = 1
)

# Data -------------------------------------------------------------------------

# Webscraping ==================================================================

cities <- readxl::read_excel("cities.xlsx") %>%
  # Please find this attached file at the corresponding GitHub repository
```

```r
  # This contains the cities we searched on Airbnb.com and the URL to the searching
  mutate(
    URL = str_replace_all(URL, 'airbnb.hu', 'airbnb.com')
  )

cities <- cities %>%
  apply(1, function(x) {
    n_rooms <- read_html(x[2]) %>%
      html_nodes("._1snxcqc") %>%
      html_text() %>%
      {gsub(' .*', '', .)}
    data.frame(city = x[1], url = x[2], n_rooms = n_rooms)
  }
  ) %>%
  reduce(rbind)

### Recursive price filter ########################################################################

RecPri_df <- cities %>%
  filter(n_rooms == 'Több' | n_rooms == '300+' | n_rooms == '300' )

for (i in 1:nrow(RecPri_df)) {
  run <-  T
  df <- seq(from = 10, to = 1500, length.out = 5) %>%
    floor() %>%
    {na.omit(data.frame(p1 = ., p2 = lead(.)))} %>%
    mutate(
      url = paste0(RecPri_df[i, 2], '&price_max=', p2, '&price_min=', p1)
    )

  df$n_rooms <-  sapply(df$url, function(url) {
    read_html(url) %>%
      html_nodes("._1snxcqc") %>%
      html_text()
  }
  )

  while (run) {
    df_still <- df %>%
      filter(str_detect(n_rooms, "300"))

    df <- df %>%
      filter(!str_detect(n_rooms, "300"))

    if (nrow(df_still) > 0) {


      df_still <- apply(df_still, 1, function(x) {
        seq(from = x[1], to = x[2], length.out = 3) %>%
          floor() %>%
          {na.omit(data.frame(p1 = ., p2 = lead(.)))} %>%
          mutate(
            url = paste0(RecPri_df[i, 2], '&price_max=', p2, '&price_min=', p1)
```

```
          )
      }
      ) %>%
        reduce(rbind)

      df_still$n_rooms <-  sapply(df_still$url, function(url) {
        tryCatch(
        read_html(url) %>%
          html_nodes("._1snxcqc") %>%
          html_text() %>%
          as.character(),
          error = function(e) NA)
      }
      )

      df <- rbind(df, na.omit(df_still))
      print(df)
    } else {
      run <- F
      cities <- df %>%
        mutate(city = RecPri_df[i, 1]) %>%
        select(city, url, n_rooms) %>%
        rbind(cities)
    }
  }
}

### Repeat the search to different dates and merge the result ############################

cities <- cities %>%
  filter(!str_detect(n_rooms, 'Több') & !str_detect(n_rooms, '300'))

start_dates <- c(seq(from = 1, to = 89, by = 7), seq(from = 5, to = 92, by = 7)) %>%
  sort()

room_list_total <- data.frame()

for (i in start_dates) {
    cities_current <- cities %>%
      mutate(
        url = str_replace_all(url, '2021-07-01',
                              as.character(as.Date(i, origin = '2021-05-30'))),
        url = str_replace_all(url, '2021-07-04',
                              as.character(as.Date(ifelse(i %% 7 == 1, i + 3, i + 2),
                                                   origin = '2021-05-30')))
      )

cl <- makeCluster(7)
clusterExport(cl, list("cities"), envir = environment())
clusterEvalQ(cl, library(rvest))
clusterEvalQ(cl, library(tidyverse))

all_source <- parApply(cl = cl, cities_current, 1, function(x) {
```

```r
  page <- read_html(x[2])
  n_rooms <- page %>%
    html_nodes("._1snxcqc") %>%
    html_text() %>%
    {gsub(" s.*", "", .)} %>%
    as.numeric()

  df <- data.frame(city = x[1], url = x[2])

  if (!is.na(n_rooms) && n_rooms > 20) {

    df <- data.frame(
      city = x[1],
      url = html_nodes(page, xpath =
                         paste0('/html/body/div[4]/div/div/div/div[1]/main/div',
                                '/div/div[1]/div[2]/div/div/div[1]/nav/div/a[1]')) %>%
        html_attr('href') %>%
        {paste0('https://www.airbnb.com', .)},
      v = seq(from = 20, to = (n_rooms %/% 20)*20, by = 20)
    ) %>%
      mutate(
        url = str_replace(url, 'items_offset=20', paste0('items_offset=', v))
      ) %>%
      select(-v) %>%
      rbind(df)
  }
  df
})

all_source <- reduce(all_source, rbind)

room_list <- parApply(cl = cl, all_source, 1, function(x) {
  tryCatch({
    page <- read_html(x[2])
    URL <- page %>%
      html_nodes('._gjfol0') %>%
      html_attr('href') %>%
      {paste0('https://www.airbnb.com', .)}
    price <- page %>%
      html_nodes('span._olc9rf0') %>%
      html_text() %>%
      .[1:length(URL)]
    place <- page %>%
      html_nodes('._b14dlit') %>%
      html_text() %>%
      .[1:length(URL)]
    assesment <- page %>%
      html_nodes('._18khxk1') %>%
      html_text() %>%
      .[1:length(URL)]
    data.frame(
      city = x[1],
      title = page %>%
```

```
        html_nodes('._gjfol0') %>%
        html_attr('aria-label'),
      URL, price, place, assesment
    )
  }, error = function(e) NULL)
})

stopCluster(cl)

room_list <- reduce(Filter(f = Negate(is.null), room_list), rbind)
room_list_total <- rbind(room_list_total, room_list)

}

### Scrape all the rooms ##############################################################

room_list_total <- tibble(room_list_total) %>%
  filter(!duplicated(id)) %>%
  filter(!is.na(assesment)) %>%
  filter(str_detect(assesment, '\\.')) %>%
  mutate(
    price = as.numeric(str_remove_all(price, '\\$')),
    n_reviews = gsub(pattern = '.*[(]', replacement = '', x = assesment) %>%
      gsub(pattern = ' .*', replacement = '') %>%
      as.numeric(),
    assesment = gsub('\\s.*', '', assesment) %>%
      as.numeric()
  )

room_interval <- 1:nrow(room_list_total)

raw_dat <- list()

rD <- rsDriver(verbose = TRUE,
               port=48458L,
               chromever = '88.0.4324.27',
               check = TRUE)

remDr <- rD$client

for (i in room_interval) {
remDr$navigate(pull(room_list_total[i, ], URL))

    url_descript <- character()
    url_amenities <- character()
    url_reviews <- character()
    host <- character()
    rules <- character()
    assesment <- character()
    bed <- character()
    comments <- character()
    amenities <- character()
    stars <- character()
```

```
265
266     Sys.sleep(4)
267     page_room <- remDr$getPageSource()[[1]] %>%
268       read_html()
269
270     page_room %>%
271       html_nodes("._13e0raay") %>%
272       html_attr("href") %>%
273       {paste0("https://www.airbnb.com", .)} %>%
274       {
275         url_amenities <<- str_subset(., "amenities")
276         url_reviews <<- str_subset(., "reviews")
277       }
278
279     host <- page_room %>%
280       html_nodes("._14i3z6h") %>% # host
281       html_text()
282
283     rules <- page_room %>%
284       html_nodes("._u827kd") %>% # rules
285       html_text()
286
287     bed <- page_room %>%
288       html_nodes("._1a5glfg") %>% # bed
289       html_text()
290
291     stars <- page_room %>%
292       html_nodes("._1s11ltsf") %>%
293       html_text()
294
295 if (length(url_reviews) != 0 && url_reviews != 'https://www.airbnb.com') {
296
297 remDr$navigate(url_reviews)
298     Sys.sleep(6)
299     comments <- remDr$getPageSource()[[1]] %>%
300       read_html() %>%
301       html_nodes('._1xib9m0') %>%
302       html_text()
303
304 remDr$navigate(url_amenities)
305     Sys.sleep(3)
306     amenities <- remDr$getPageSource()[[1]] %>%
307       read_html() %>%
308       html_nodes("._vzrbjl") %>%
309       html_text()
310 }
311
312   raw_dat[[length(raw_dat) + 1]] <- list(
313     source = room_list_total[i, ],
314     url_amenities = url_amenities,
315     url_reviews = url_reviews,
316     host = host,
317     rules = rules,
318     bed = bed,
```

```
    comments = comments,
    amenities = amenities,
    stars = stars
  )
}

hun_cities <- read_csv("worldcities.csv") %>%
  filter(country == "Hungary") %>%
  select(city, lat, lng, admin_name, population)

cities <- readxl::read_excel("cities.xlsx")

world <- ne_countries(scale = "large", returnclass = "sf")

merge(cities, hun_cities, by = 'city') %>%
  tibble() %>%
  ggplot() +
  geom_sf(data = world, size = 1.2, fill = 'white', color = 'black') +
  coord_sf(xlim = c(16, 23.4), ylim = c(45.5, 48.7), expand = FALSE) +
  geom_point(aes(x = lng, y = lat), size = 4, alpha = 1, color = 'black',
             shape = 21, fill = viridis::viridis(1, begin = .1)) +
  theme_void()

lapply(dat, function(x) {
  tibble(city = x[["source"]][["city"]], comments = x$comments) %>%
    mutate(language = textcat::textcat(comments))
}) %>%
  reduce(rbind) %>%
  mutate(
    language = fct_lump(language, n = 6) %>%
      fct_infreq()
  ) %>%
  merge(hun_cities) %>%
  {rbind(., mutate(., admin_name = 'Total'))} %>%
  mutate(admin_name = fct_reorder(admin_name, admin_name == 'Total')) %>%
  select(admin_name, language) %>%
  na.omit() %>%
  ggplot() +
  aes(y = admin_name, fill = language) +
  scale_x_continuous(labels = scales::percent, limits = c(0,1), expand = c(0,0)) +
  geom_bar(color = "black", position = position_fill()) +
  scale_fill_viridis_d() +
  labs(x = 'Proportion of comments', y = NULL, fill = "Language")

room_list <- room_list_total %>%
  filter(!duplicated(id)) %>%
  filter(!is.na(assesment)) %>%
  filter(str_detect(assesment, '\\.'))

room_list %>%
  ggplot() +
  aes(assesment, price) +
  geom_point(color="#69b3a2", alpha=0.8) +
```

```r
  geom_smooth() +
  labs(x = 'Rating', y = 'Price')

ggpubr::ggarrange(
room_list %>%
  ggplot() +
  stat_ecdf(aes(x = assesment, color = 'ECDF based on the total population')) +
  geom_blank(aes(color = 'ECDF filtered to that the rating is higher than 4.9')) +
  labs(x = 'Assesment', y = 'Corresponding cumulative density', title = 'Assesment',
       color = NULL),
room_list %>%
  ggplot() +
  stat_ecdf(data = filter(room_list, assesment > 4.9), mapping = aes(n_reviews,
                          color = 'ECDF filtered to that the rating is higher than 4.9')) +
  stat_ecdf(aes(x = n_reviews, color = 'ECDF in the total population')) +
  scale_x_log10() +
  labs(x = 'Number of reviews (log scale)', title = 'Reviews') +
  theme(
    legend.position = 'bottom'
  ), common.legend = T
)

dat_words_eng <- lapply(dat, function(x) {
  tryCatch({
    tibble(comments = x$comments) %>%
      mutate(language = textcat::textcat(comments)) %>%
      filter(language == "english") %>%
      tail(-2) %>%
      tidytext::unnest_tokens(words, comments, to_lower = T) %>%
      mutate(assesment = x[["source"]][["assesment"]],
             n_reviews = x[["source"]][["n_reviews"]])
  }, error = function(e) NULL)
}
) %>%
  {reduce(Filter(f = Negate(is.null), .), rbind)}

freq_by_rank_eng <- dat_words_eng %>%
  filter(n_reviews >= 15) %>%
  mutate(
    type = as.numeric(Hmisc::cut2(assesment, g = 10, levels.mean = T)),
    type = case_when(
    type == min(type) ~ 'Low',
    type == max(type) ~ 'High',
    T ~ 'Middle'
    ),
  ) %>%
{
df <- .
      df <- df %>%
        group_by(type, words) %>%
        summarise(n = n()) %>%
        ungroup()
      merge(
```

```
      df,
      df %>%
        group_by(type) %>%
        summarise(total = sum(n))
    )
  } %>%
  arrange(desc(n)) %>%
  group_by(type) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total) %>%
  ungroup()

freq_by_rank_eng %>%
  select(words, type, n) %>%
  bind_tf_idf(term = words, document = type, n = n) %>%
  anti_join(data.frame(words = c(stopwords::stopwords(), "also", "can"))) %>%
  filter(type != 'Middle' & n > 20) %>%
  arrange(desc(tf_idf)) %>%
  group_by(type) %>%
  group_modify(~ head(mutate(.x, rank = row_number()), 50)) %>%
  reshape2::acast(words ~ type, value.var = "rank", fill = 0) %>%
  wordcloud::comparison.cloud(colors = viridis::viridis(2, direction = -1 , end = .7),
                              max.words = 50)
```