wrangling

Clare Tang

2021/11/9

Data wrangling

```
listing <- listings %>% dplyr::select(id, host_id,
                              host_response_time,
                              host_response_rate,
                              host_is_superhost,
                              # host_total_listings_count,
                              host_has_profile_pic,
                              host_identity_verified,
                              neighbourhood_cleansed,
                              room_type,
                              price,
                              number_of_reviews,
                              review_scores_value,
                              license,
                              longitude,
                              latitude
                              )


listing <- as.data.frame(listing)
# dim(listing) # 3123   15

# summary(is.na(listing)) # 861 na values in 'review_scores_value'
listing <- listing %>% filter(!is.na(review_scores_value))
# dim(listing) # 2262   15

# originally, the host_total_listings_count does not match the unique number of listings each host has
# maybe the reason that some hosts have listings not in boston
# create new host_total_listings_count
listing$host_total_listings_count <- rep(NA, dim(listing)[1])
for(i in 1:dim(listing)[1]){
  listing$host_total_listings_count[i] <- sum(listing$host_id == listing$host_id[i])
}

# number of hosts
length(unique(listing$host_id)) # 1183
```

```
## [1] 1016
```

```
sum(listing$host_response_time == "N/A") # 682
```

```
## [1] 528
```

```r
sum(listing$host_response_rate == "N/A") # 682
```

```
## [1] 528
```

```r
# filter na value
unique <- lapply(listing, unique)
unique$host_response_time # N/A + 3
```

```
## [1] "N/A"                "within an hour"     "within a few hours"
## [4] "within a day"       "a few days or more"
```

```r
unique$host_response_rate # N/A
```

```
##  [1] "N/A"  "100%" "67%"  "90%"  "60%"  "62%"  "0%"   "94%"  "86%"  "25%"
## [11] "50%"  "83%"  "80%"  "96%"  "20%"  "97%"  "38%"  "99%"  "91%"  "89%"
## [21] "69%"  "46%"  "33%"  "75%"  "88%"  "93%"  "98%"  "71%"  "92%"  "81%"
## [31] "70%"  "10%"  "84%"  "78%"  "43%"  "82%"  "29%"  "79%"  "63%"  "14%"
## [41] "40%"  "95%"  "42%"
```

```r
unique$host_is_superhost # t/f
```

```
## [1] "f" "t"
```

```r
unique$host_has_profile_pic # t/f
```

```
## [1] "t" "f"
```

```r
unique$host_identity_verified # t/f
```

```
## [1] "f" "t"
```

```r
length(unique$neighbourhood_cleansed) # 25
```

```
## [1] 25
```

```r
unique$room_type # 4
```

```
## [1] "Entire home/apt" "Private room"    "Shared room"     "Hotel room"
```

```r
# unique$license
unique$host_total_listings_count
```

```
##  [1]  1 10  5  7 24  4  6  2  3 34 22 27 55 20  8 11 15 17 12 13 23 14  9 36 19
## [26] 21 29 26
```

```r
listin <- listing %>% filter(host_response_time != "N/A" &  host_response_rate != "N/A" )
# summary(is.na(listin))

# crerate new variables
listin$host_response_time <- factor(listin$host_response_time,
                                    levels = c("within an hour",
                                               "within a few hours",
                                               "within a day",
                                               "a few days or more"
                                               ))
listin$host_is_superhost <- factor(listin$host_is_superhost,
                                   levels = c("f", "t"))
listin$host_has_profile_pic <- factor(listin$host_has_profile_pic,
                                   levels = c("f", "t"))
listin$host_identity_verified <- factor(listin$host_identity_verified,
```

```
                                    levels = c("f", "t"))
listin$room_type <- factor(listin$room_type,
                                    levels = c("Entire home/apt", "Private room", "Hotel room",  "Shared

listin$license_ornot <- ifelse(listin$license == "", 0, 1)
listin$license_ornot <- factor(listin$license_ornot)

listin$host_response_rate <- as.numeric(gsub("[\\%, ]", "", listin$host_response_rate))
listin$host_response_rate <- listin$host_response_rate/100

listin$price <- as.numeric(gsub("\\$", "", listin$price))
sum(is.na(listin$price))
```

```
## [1] 9
```

```
listin <- listin %>% filter(price != 0, !is.na(price))

dim(listin)
```

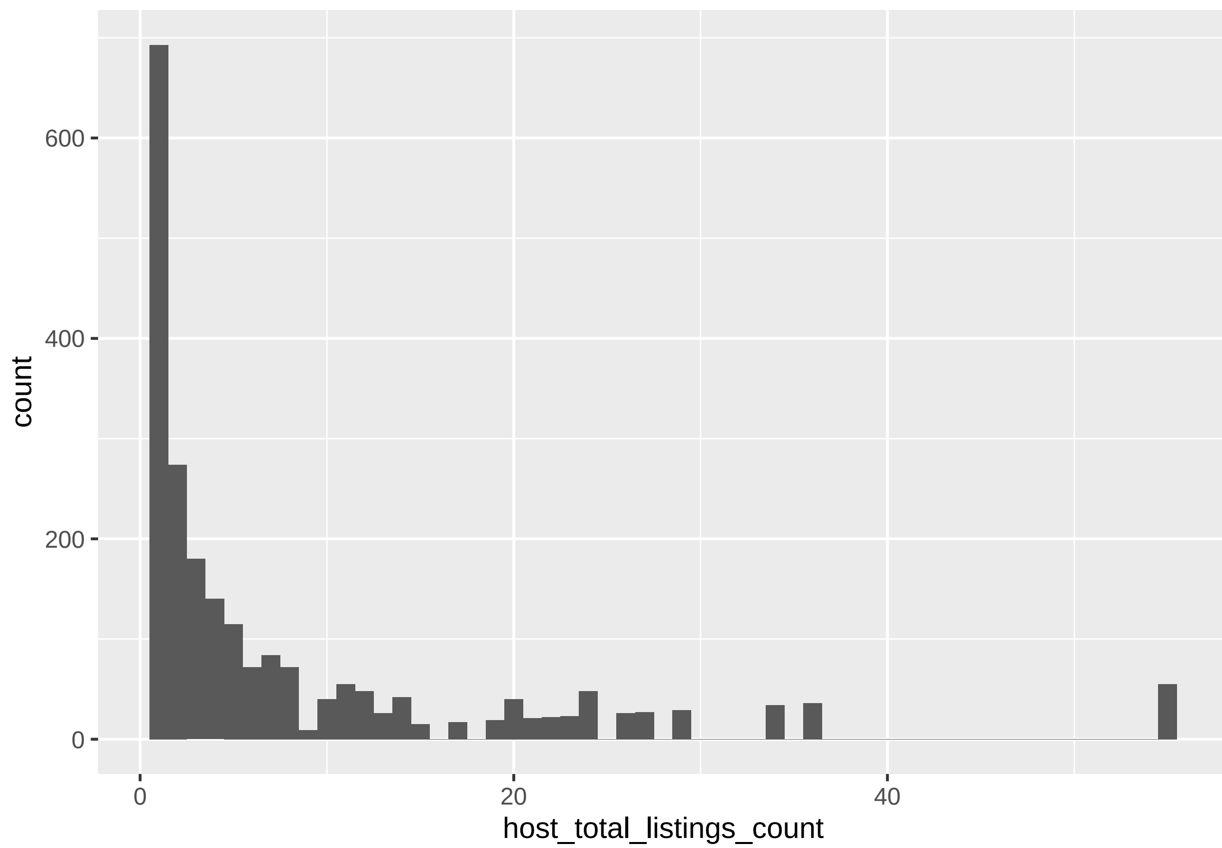
```
## [1] 1721   17
```

```
count_hid <- count(listin, host_id)
```

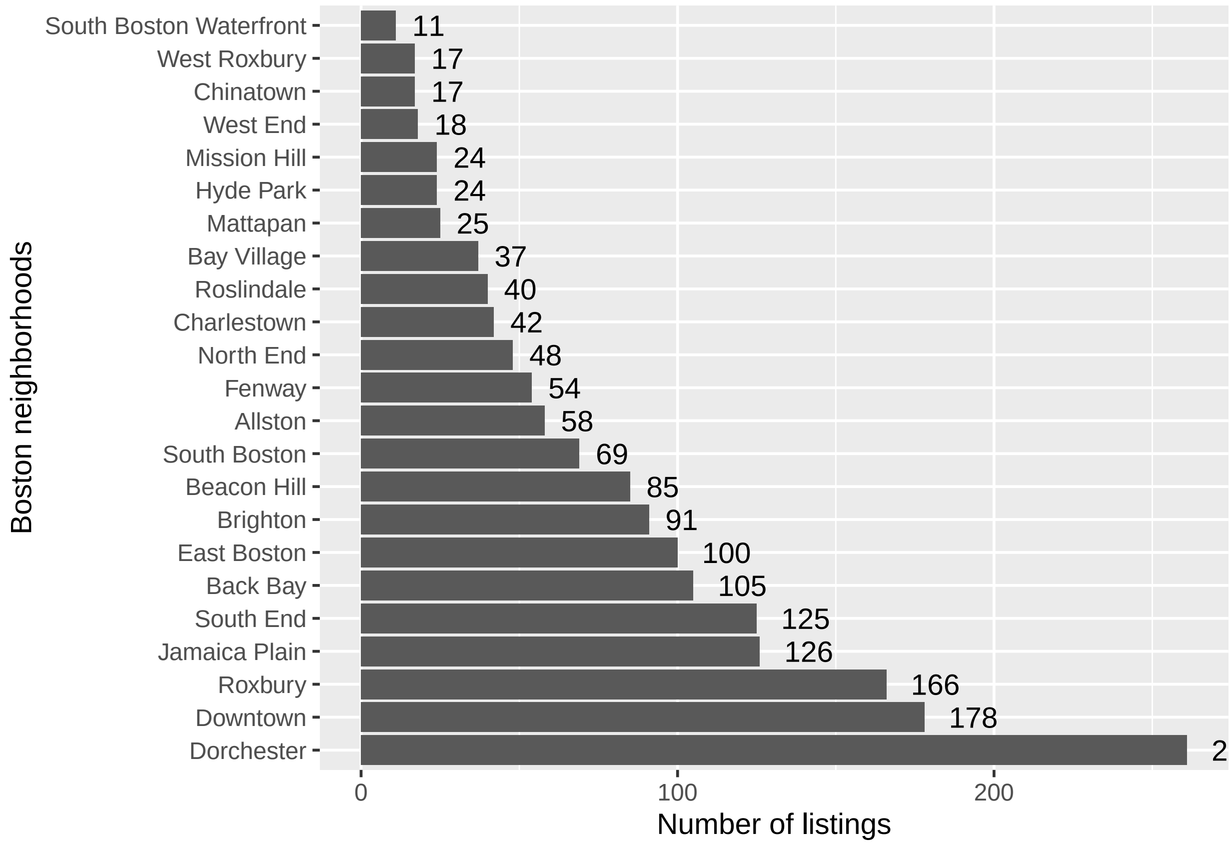
Basic plot

1. Distribution of the number of listings own by different hosts Most hosts own less than 5 listings
2. Number of listings in different Boston neighborhoods
3. Density of reviews of listings in different Boston neighborhoods
4. Number of different type of listings in different Boston neighborhoods

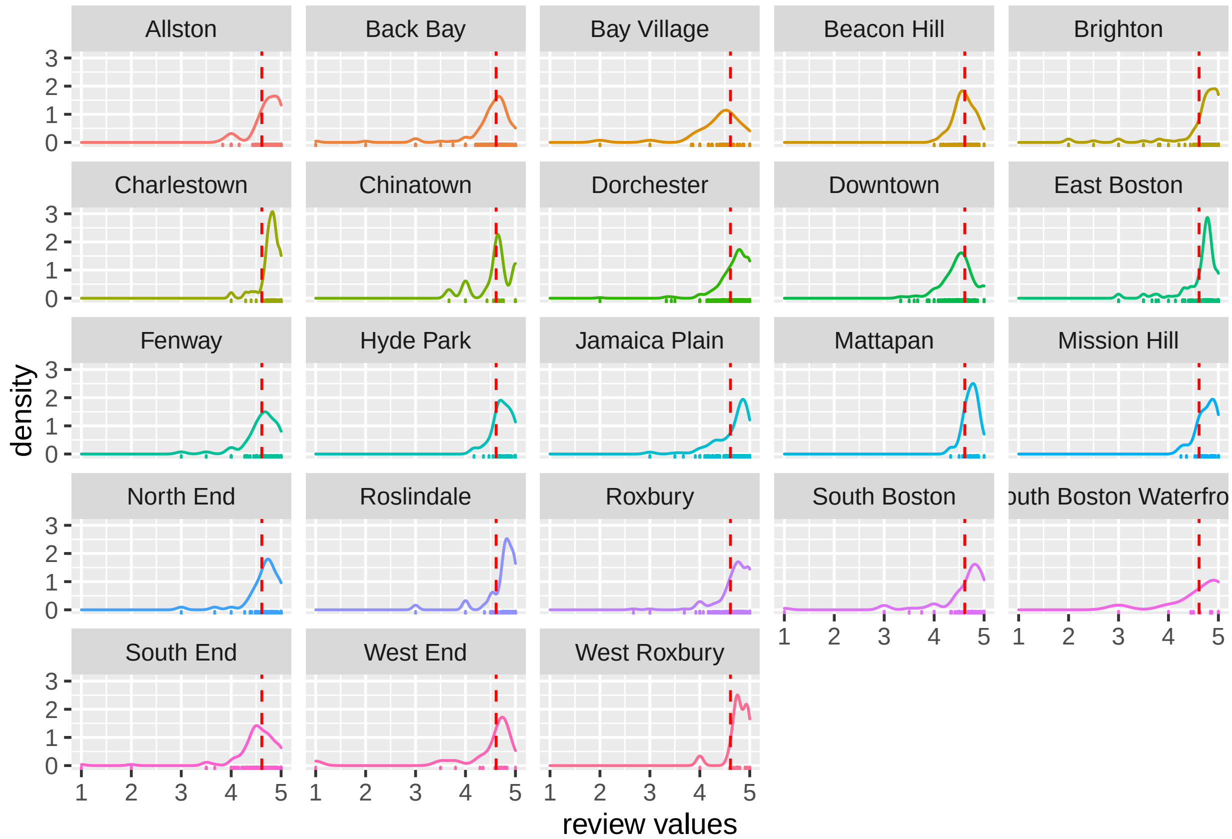
```
# distribuiton of host_total_listings_count
ggplot(data = listing, aes(x = host_total_listings_count))+
  geom_histogram(binwidth = 1)
```




```
count <- count(listin, neighbourhood_cleansed)
ggplot(count, aes(x = reorder(neighbourhood_cleansed, -n), y = n))+
  geom_bar(stat = "identity")+
  coord_flip()+
  ylab("Number of listings")+
  xlab("Boston neighborhoods")+
  geom_text(aes(label = n), hjust=-0.5, position = "dodge")
```



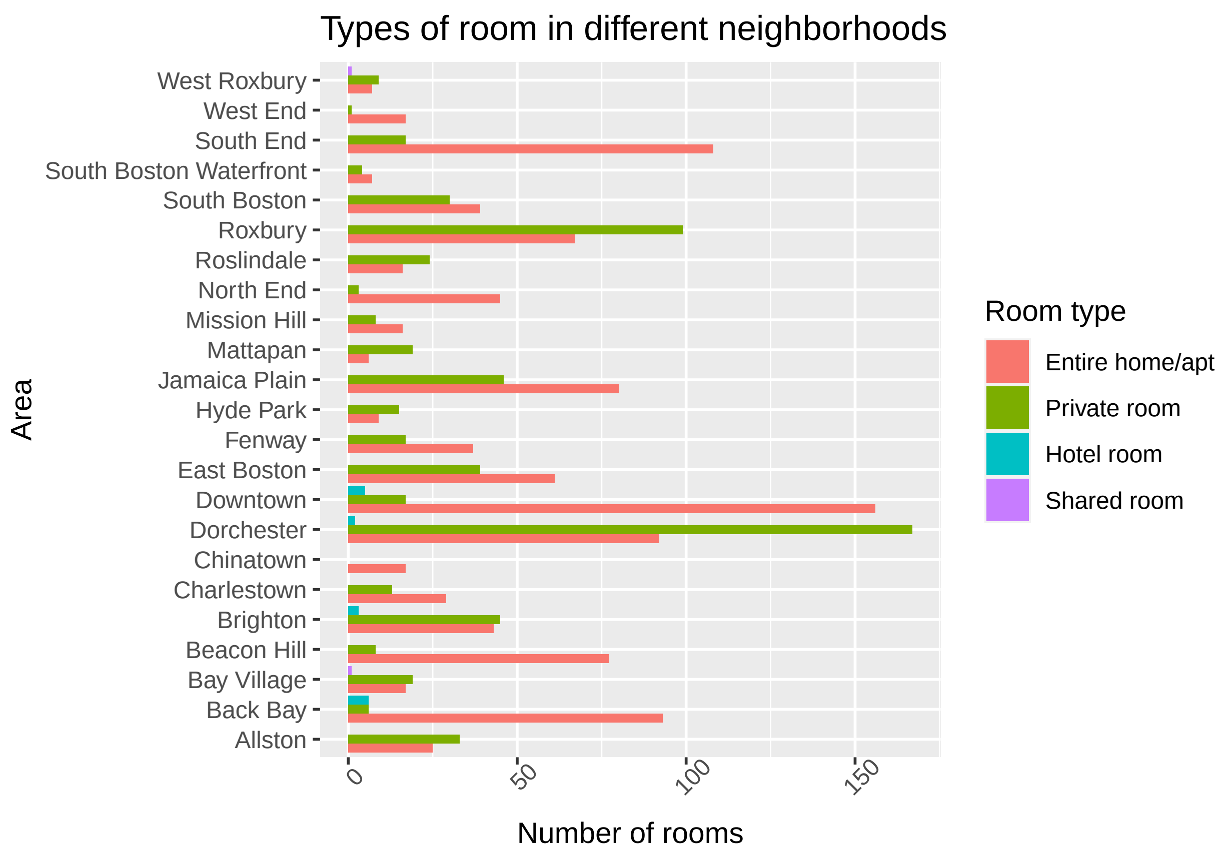

```r
# draw the distribution of review scores of 20 hosts
# set.seed(1)
# oh_hid <- sample(unique(listin$host_id), 20, replace = FALSE)
# oh_listin <- listin %>% filter(host_id %in% oh_hid)
# ggplot(oh_listin)+
#   geom_density(alpha = .3)+
#   aes(x = review_scores_value, color = host_id)+
#   facet_wrap(~ host_id)+
#   theme(legend.position = "none")+
#   geom_rug()+
#   xlab("review values")+
#   geom_vline(xintercept = mean(oh_listin$review_scores_value), color = "red", lty = 2)


ggplot(listin)+
  geom_density(alpha = .3)+
  aes(x = review_scores_value, color = neighbourhood_cleansed)+
  facet_wrap(~ neighbourhood_cleansed)+
  theme(legend.position = "none")+
  geom_rug()+
  xlab("review values")+
  geom_vline(xintercept = mean(listin$review_scores_value), color = "red", lty = 2)
```




```
# bar plot: type of listings
roomdf <- listin %>% group_by(neighbourhood_cleansed, room_type) %>% summarize(Freq = n())
```

```
## `summarise()` has grouped output by 'neighbourhood_cleansed'. You can override using the `.groups` a
```

```
total_room <- listin %>% group_by(neighbourhood_cleansed) %>% summarize(sum = n())
ratio_room <- merge(roomdf, total_room, by = "neighbourhood_cleansed")
ratio_room <- ratio_room %>% mutate(ratio = Freq/sum)

ggplot(ratio_room, aes(x = Freq, y = neighbourhood_cleansed, fill = room_type))+
  geom_bar(position = position_dodge(preserve = 'single'), stat = "identity")+
  xlab("Number of rooms")+ ylab("Area")+
  scale_fill_discrete(name = "Room type")+
  ggtitle("Types of room in different neighborhoods")+
  theme(axis.text.x = element_text(angle = 45))
```



Boston neighborhoods

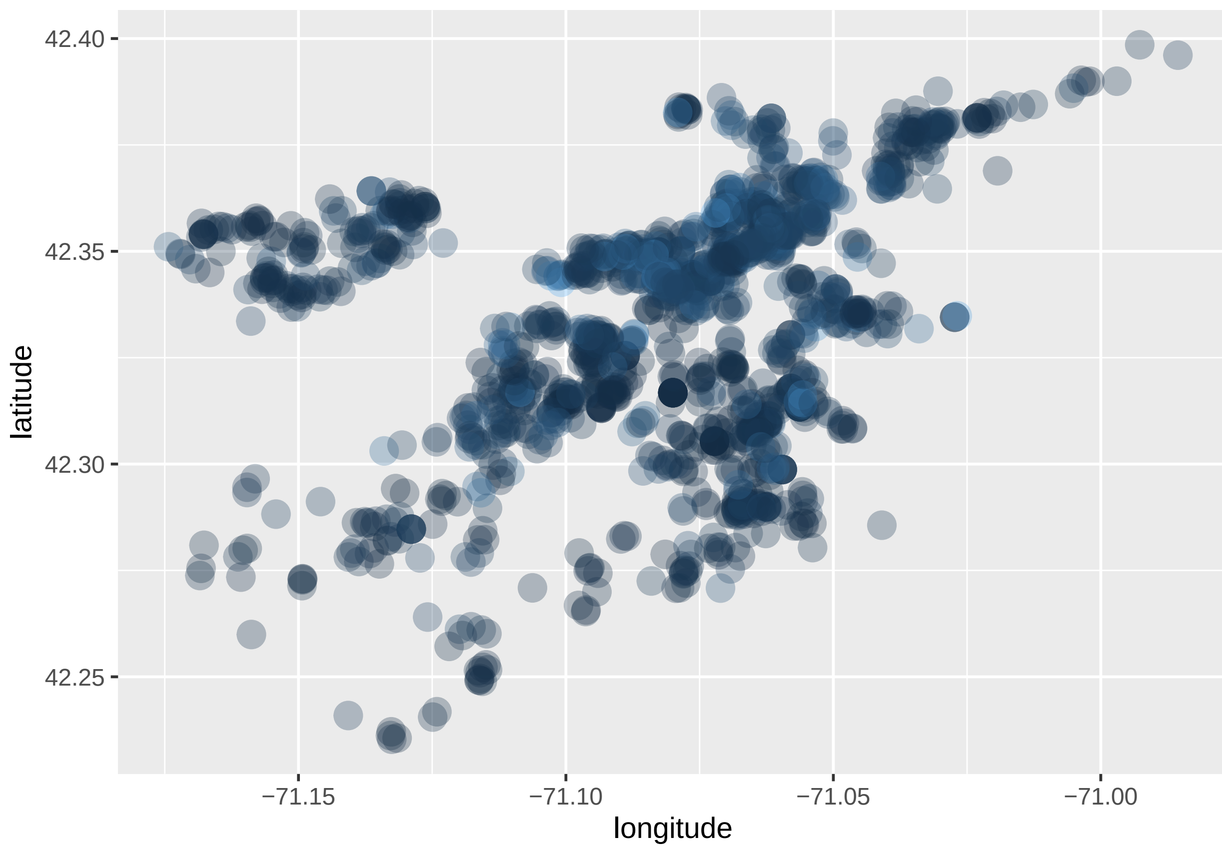
1. read boston neighborhoods shape file
2. transfer lisitng dataframe into shape file by using the location of listings (longitude and latitude)
3. maps dot plot of price of listings for different Boston neighborhoods

```
boston <- st_read("Boston_Neighborhoods/Boston_Neighborhoods.shp", quiet = TRUE)
epsg_wgs84 <- 4326
# boston %>% st_transform(epsg_wgs84)

sf_listin <- listin %>% st_as_sf(coords = c("longitude", "latitude")) %>% st_set_crs(epsg_wgs84)
print(sf_listin, n = 5)
```

```
## Simple feature collection with 1721 features and 15 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -71.17429 ymin: 42.23533 xmax: -70.98558 ymax: 42.39853
## Geodetic CRS:  WGS 84
## First 5 features:
##      id host_id host_response_time host_response_rate host_is_superhost
## 1  5506    8229     within an hour                  1                 t
## 2  6695    8229     within an hour                  1                 t
## 3  8789   26988     within an hour                  1                 t
## 4 10730   26988     within an hour                  1                 t
## 5 10813   38997 within a few hours                  1                 t
##   host_has_profile_pic host_identity_verified neighbourhood_cleansed
## 1                    t                      t                Roxbury
```

```
## 2                    t                      t                Roxbury
## 3                    t                      t               Downtown
## 4                    t                      t               Downtown
## 5                    t                      t               Back Bay
##         room_type price number_of_reviews review_scores_value
## 1 Entire home/apt   124               108                4.77
## 2 Entire home/apt   169               115                4.70
## 3 Entire home/apt   110                25                4.56
## 4 Entire home/apt   100                32                4.43
## 5 Entire home/apt   116                 5                4.75
##                       license host_total_listings_count license_ornot
## 1 Approved by the government                        10             1
## 2                   STR446650                        10             1
## 3                                                     5             0
## 4                                                     5             0
## 5                                                     7             0
##                      geometry
## 1 POINT (-71.09559 42.32981)
## 2 POINT (-71.09351 42.32994)
## 3 POINT (-71.06265 42.35919)
## 4  POINT (-71.06185 42.3584)
## 5 POINT (-71.08787 42.35061)
```

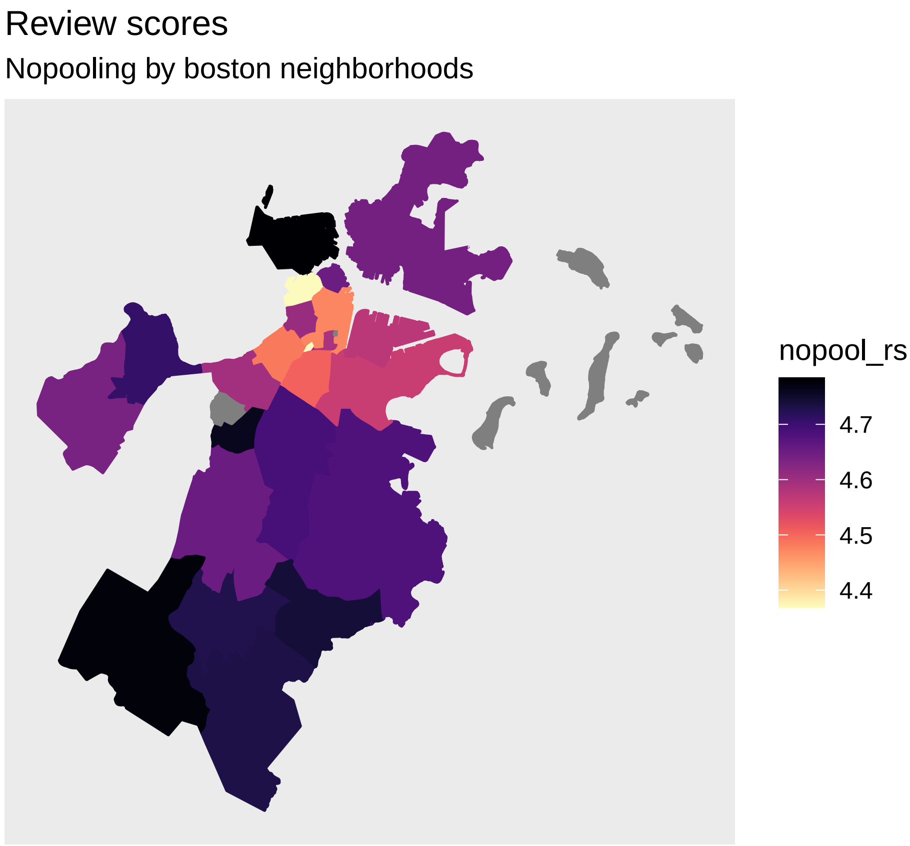
```
ggplot()+
  geom_point(data = listin,
             aes(longitude,
                 latitude,
                 color = price, size = .8), alpha = .3)+
  theme(legend.position = "none")
```



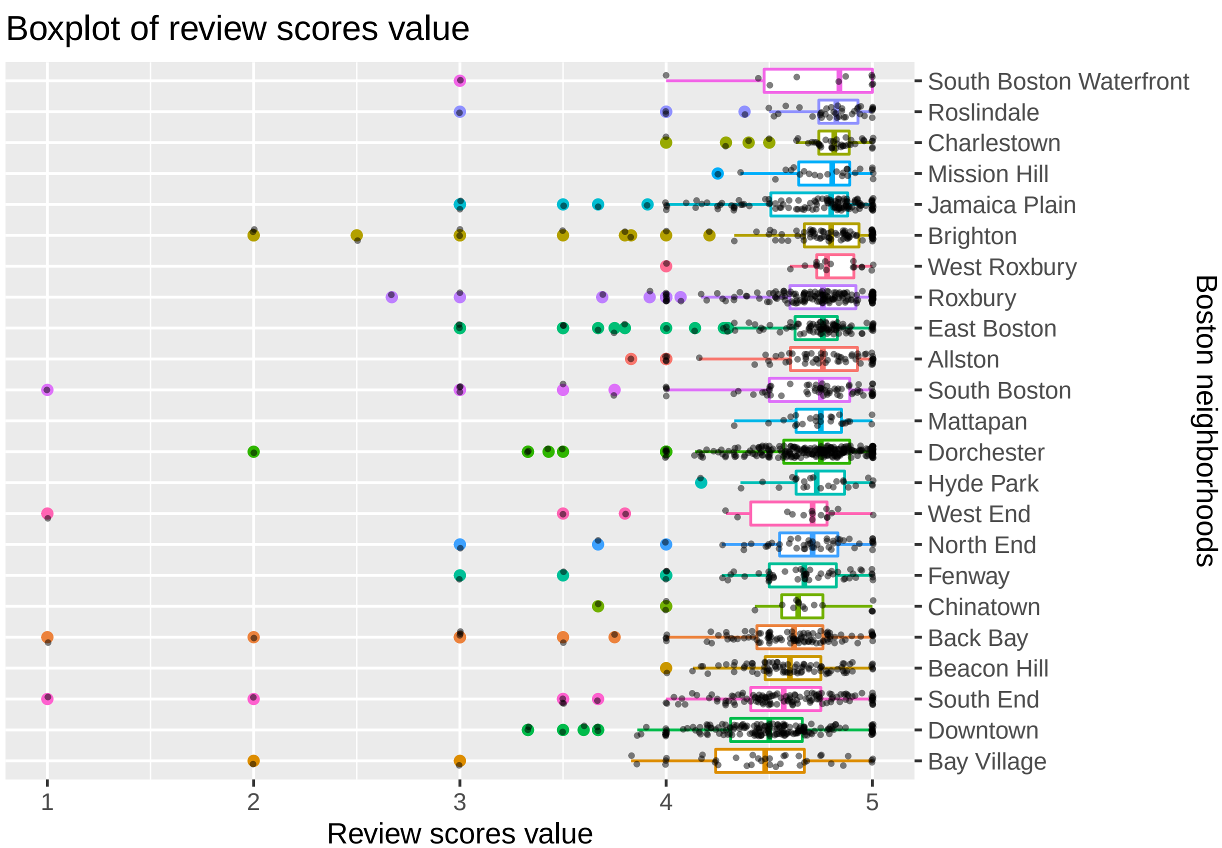
Map of review_scores_value

1. calculate the mean value of reviews score of different Boston neighborhoods and plot
2. boxplot of reviews scores of different Boston neighborhoods

```
nopooling_rs <- listin %>%
  group_by(neighbourhood_cleansed) %>%
  do(tidy(lm(review_scores_value ~ 1, .)))

rs <- data.frame(Name = count$neighbourhood_cleansed, nopool_rs = nopooling_rs$estimate, stringsAsFacto

np_rs <- left_join(boston, rs, by = "Name")

plot_nopool_rs <- np_rs %>%
  ggplot(aes(fill = nopool_rs, color = nopool_rs))+
  geom_sf()+
  coord_sf(crs = 5070, datum = NA)+
  scale_fill_viridis(direction = -1, option = "A")+
  scale_color_viridis(direction = -1, option = "A")+
  labs(title = "Review scores", subtitle = "Nopooling by boston neighborhoods")
plot_nopool_rs
```




```r
ggplot(listin, aes(x = fct_reorder(neighbourhood_cleansed, review_scores_value),
                   y = review_scores_value,
                   color = neighbourhood_cleansed))+
  geom_boxplot()+
  geom_jitter(color = "black", width = .2, size = .5, alpha = .5)+
  coord_flip()+
  theme(legend.position = "none")+
  labs(y = "Review scores value", x = "Boston neighborhoods")+
  scale_x_discrete(position = "top")+
  ggtitle("Boxplot of review scores value")
```



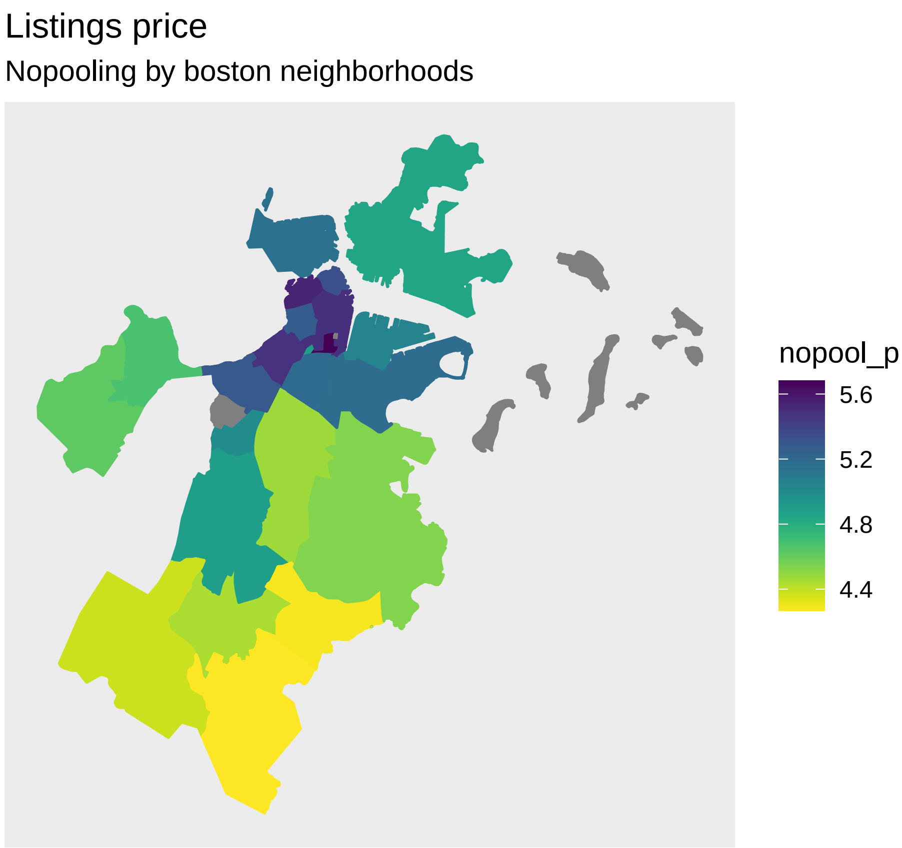
https://map-rfun.library.duke.edu/032_thematic_mapping_geom_sf.html

Mean of price by neighborhood

1. nopooling map of price of listings
2. boxplot of price of lisitngs of different Boston neighborhoods

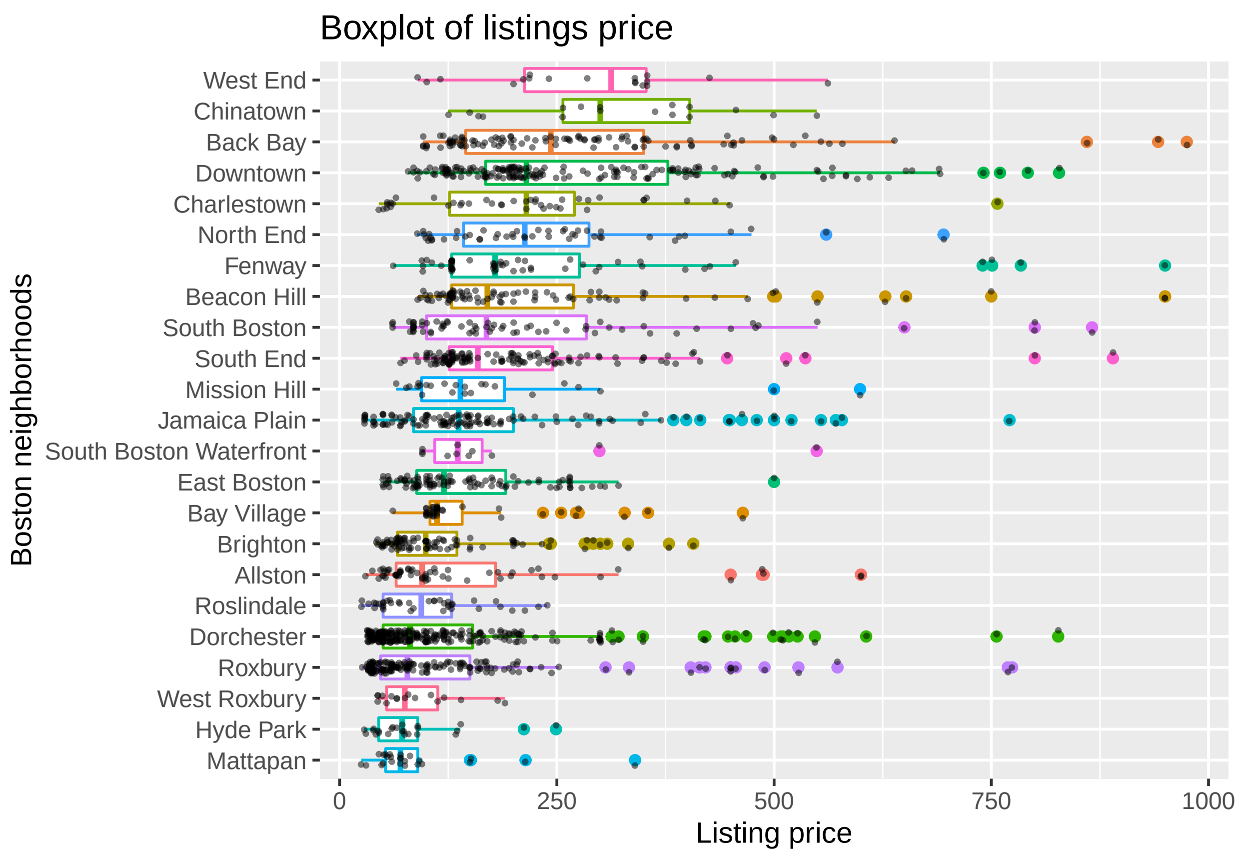
```
mp <- listin %>% group_by(neighbourhood_cleansed) %>%
  summarise_at(vars(price), list(mean_p = mean)) %>%
  mutate(log_p = log(mean_p))
names(mp)[1] <- "Name"
#
join_p <- boston %>% left_join(mp, by = "Name")
#
# join_p %>%
#   ggplot(aes(fill = log_p, color = log_p))+
#   geom_sf()+
#   coord_sf(crs = 5070, datum = NA)+
#   scale_color_viridis(direction = -1, option = "A")+
#   scale_fill_viridis(direction = -1, option = "A")+
#   labs(title = "Average price of listings by boston neighborhoods")
#
# plot_nopool_p <- join_p %>%
#   ggplot(aes(fill = log_p, color = log_p))+
#   geom_sf()+
#   coord_sf(crs = 5070, datum = NA)+
#   scale_fill_viridis(direction = -1)+
```

```r
#   scale_color_viridis(direction = -1)+
#   labs(title = "Average listing price", subtitle = "Nopooling by boston neighborhoods")



nopooling_p <- listin %>%
  group_by(neighbourhood_cleansed) %>%
  do(tidy(lm(log(price) ~ 1, .)))

p <- data.frame(Name = count$neighbourhood_cleansed, nopool_p = nopooling_p$estimate, stringsAsFactors =

np_p <- left_join(boston, p, by = "Name")

plot_nopool_p <- np_p %>%
  ggplot(aes(fill = nopool_p, color = nopool_p))+
  geom_sf()+
  coord_sf(crs = 5070, datum = NA)+
  scale_fill_viridis(direction = -1)+
  scale_color_viridis(direction = -1)+
  labs(title = "Listings price", subtitle = "Nopooling by boston neighborhoods")
plot_nopool_p
```




```r
ggplot(listin, aes(x = fct_reorder(neighbourhood_cleansed, price), y = price, color = neighbourhood_clea
  geom_boxplot()+
  geom_jitter(color = "black", width = .2, size = .5, alpha = .5)+
  coord_flip()+
```

```
  theme(legend.position = "none")+
  labs(x = "Boston neighborhoods", y = "Listing price")+
  ggtitle("Boxplot of listings price")
```

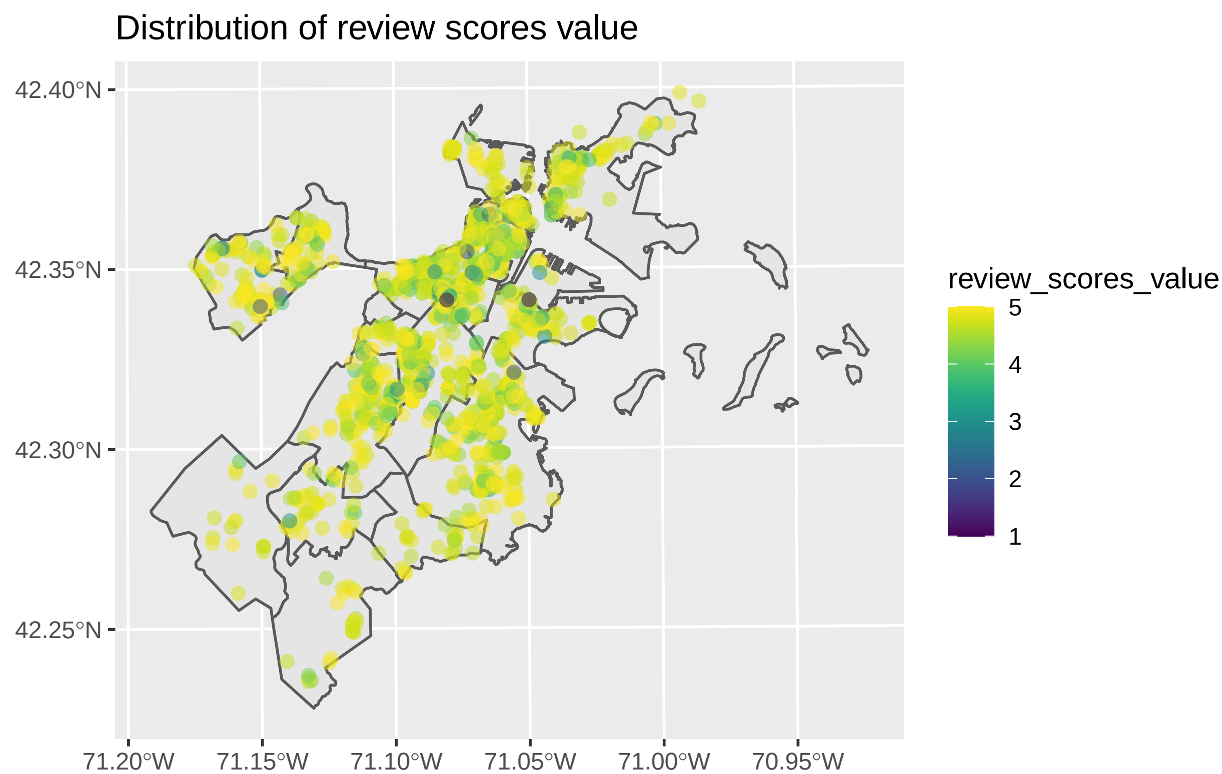


Other predictors map

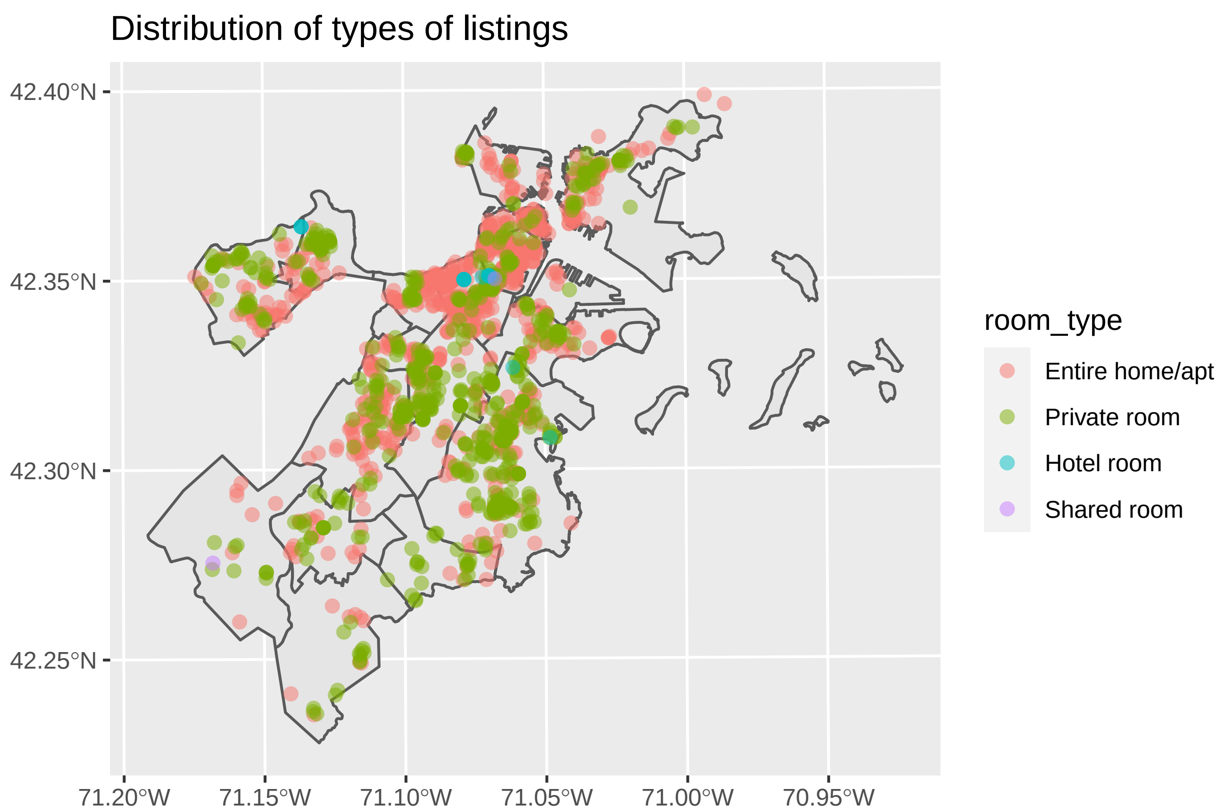
1. points map of review scores value
2. points map of types of listings
3. points map of number of reviews for each listing
4. points map of number of listings each host own

```
# shapefile
mrv_listin <- sf_listin %>%
 group_by(neighbourhood_cleansed) %>%
 summarise_at(vars(review_scores_value), list(mean_rs = mean)) %>%
 dplyr::select(neighbourhood_cleansed, mean_rs)
names(mrv_listin)[1] <- "Name"

# ggplot()+geom_sf(data = boston)+ geom_sf(data = mrv_listin, aes(color = Name))+
#  theme(legend.position = "none")

# tm_shape(sf_listin) +
#  tm_bubbles(col = "room_type", palette = "YlOrBr", size = .2)+
#  tm_legend(outside = TRUE)

ggplot()+
```

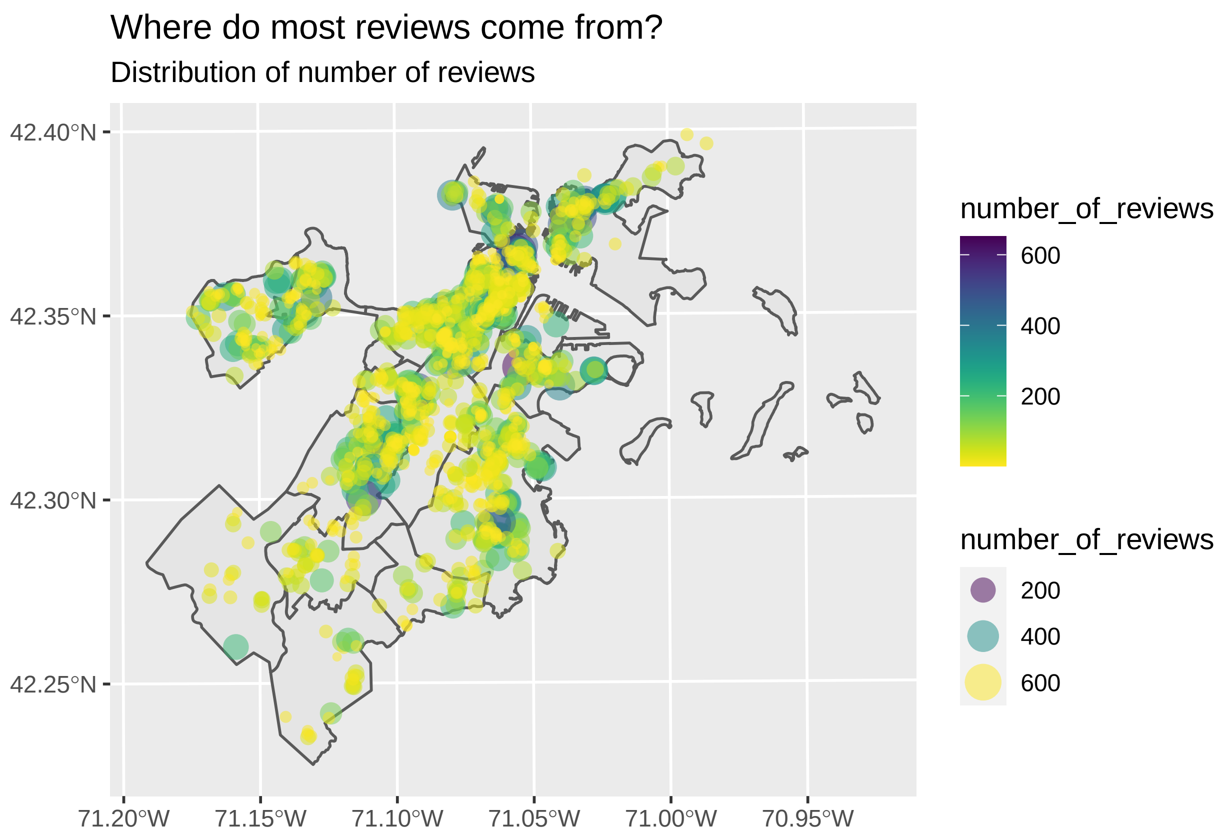
```
  geom_sf(data = boston)+
  geom_sf(data = sf_listin, aes(color = review_scores_value), size = 2, alpha = .5)+
  scale_color_viridis() +
  guides(size=guide_legend(override.aes = list(color = viridis(1))))+
  ggtitle("Distribution of review scores value")
```



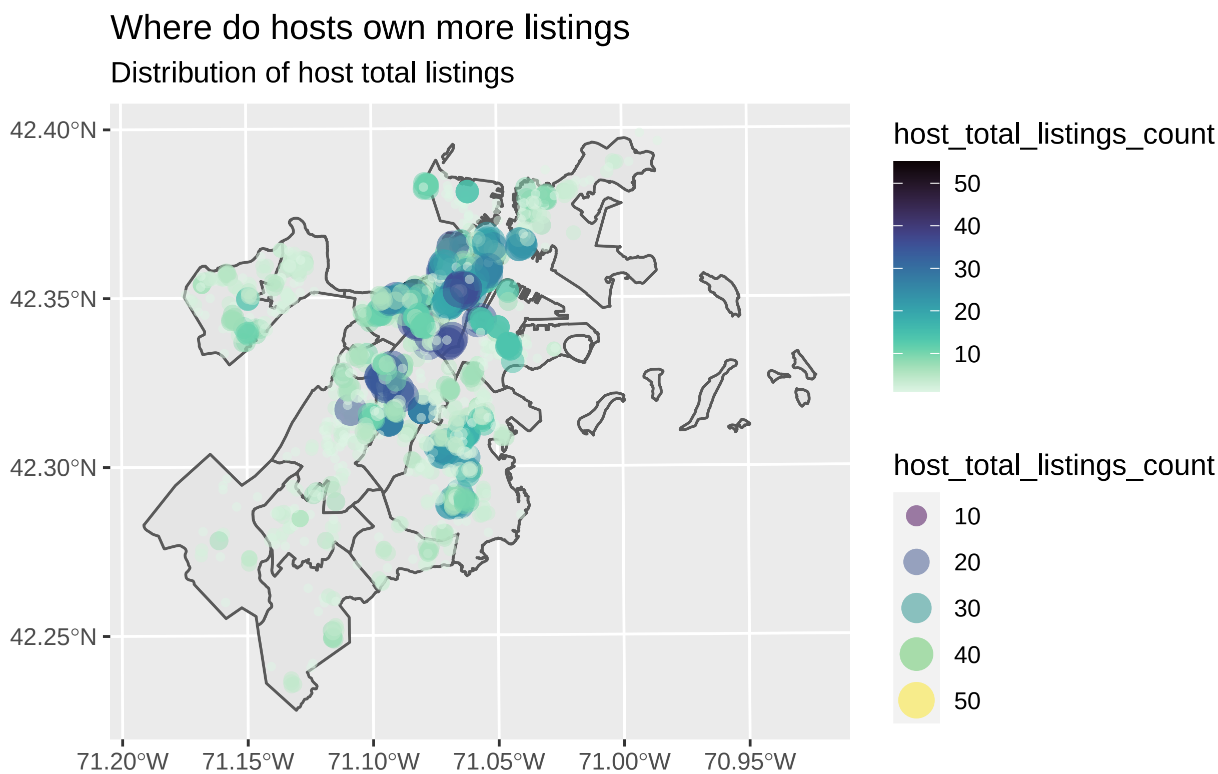

```
ggplot()+
  geom_sf(data = boston)+
  geom_sf(data = sf_listin, aes(color = room_type), size = 2, alpha = .5)+
  ggtitle("Distribution of types of listings")
```




```
ggplot()+
  geom_sf(data = boston)+
  geom_sf(data = sf_listin,
          aes(color = number_of_reviews, size = number_of_reviews), alpha = .5)+
  scale_color_viridis(direction = -1) +
  guides(size=guide_legend(override.aes = list(color = viridis(3))))+
  ggtitle("Where do most reviews come from?", subtitle = "Distribution of number of reviews")
```




```
  # theme(legend.position = "none")

ggplot()+
  geom_sf(data = boston)+
  geom_sf(data = sf_listin,
          aes(color = host_total_listings_count, size = host_total_listings_count), alpha = .5)+
  scale_color_viridis(direction = -1, option = "G")+
  guides(size=guide_legend(override.aes = list(color = viridis(5))))+
  ggtitle("Where do hosts own more listings", subtitle = "Distribution of host total listings")
```



Kriging

1. test the variogram assumptions of the price of listings
2. smooth the data of price of listings
3. present in maps

Use census tracts to get more block units

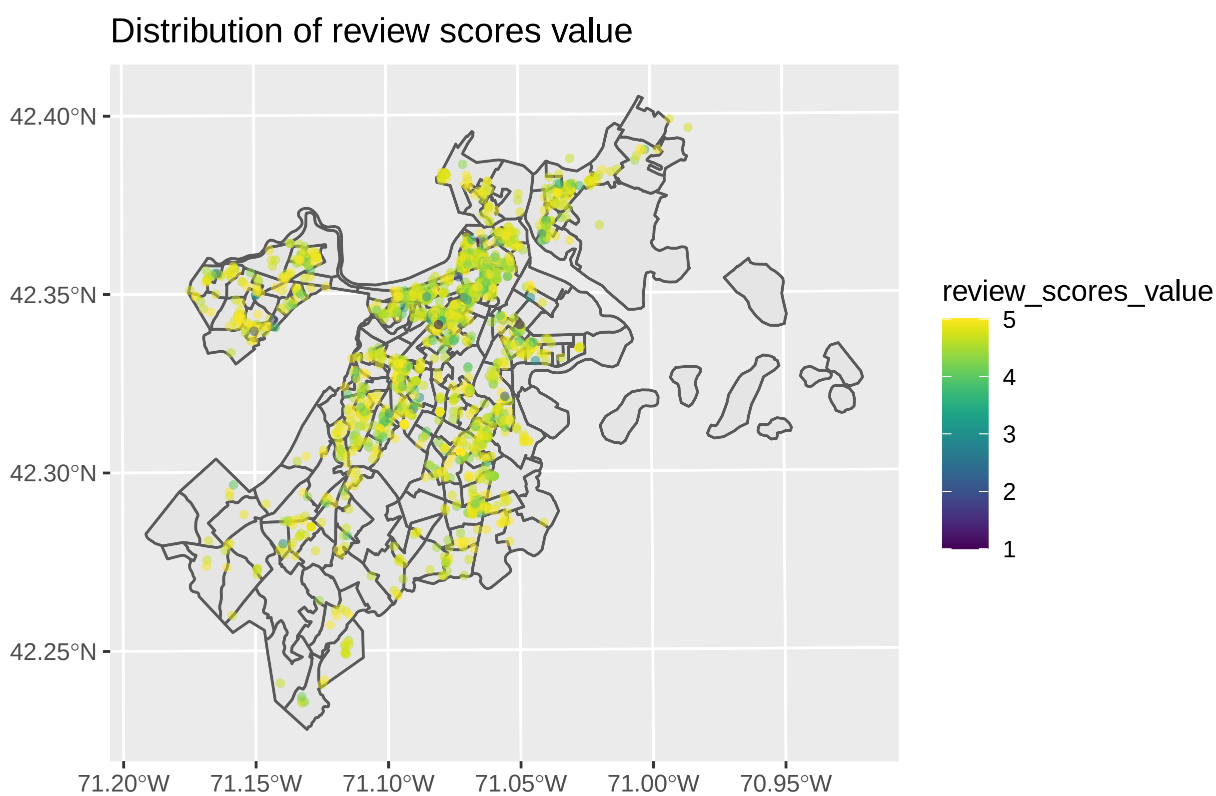
this part I am trying to use Boston tracts instead of neighborhoods to divide Boston into more areas, to get more data on variogram

```
tracts <- st_read("Census2020_Tracts/Census2020_tracts.shp")
```

```
## Reading layer `Census2020_Tracts' from data source
##   `D:\BU STUDY\MA 678\HW\midterm\MA678_midterm_project\Census2020_Tracts\Census2020_Tracts.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 207 features and 15 fields
## Geometry type: MULTIPOLYGON
```

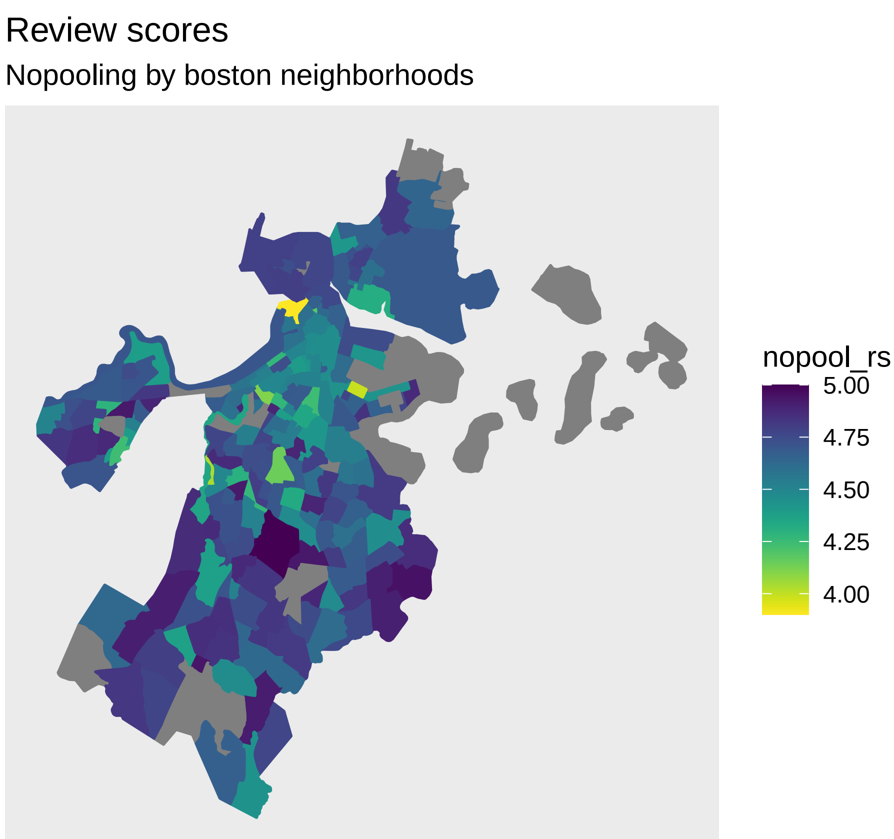
```
## Dimension:     XY
## Bounding box:  xmin: 739715.8 ymin: 2908294 xmax: 812981.4 ymax: 2972975
## Projected CRS: NAD83 / Massachusetts Mainland (ftUS)
```

```
ggplot()+
  geom_sf(data = tracts)+
  geom_sf(data = sf_listin, aes(color = review_scores_value), size = 1, alpha = .5)+
  scale_color_viridis() +
  guides(size=guide_legend(override.aes = list(color = viridis(1))))+
  ggtitle("Distribution of review scores value")
```



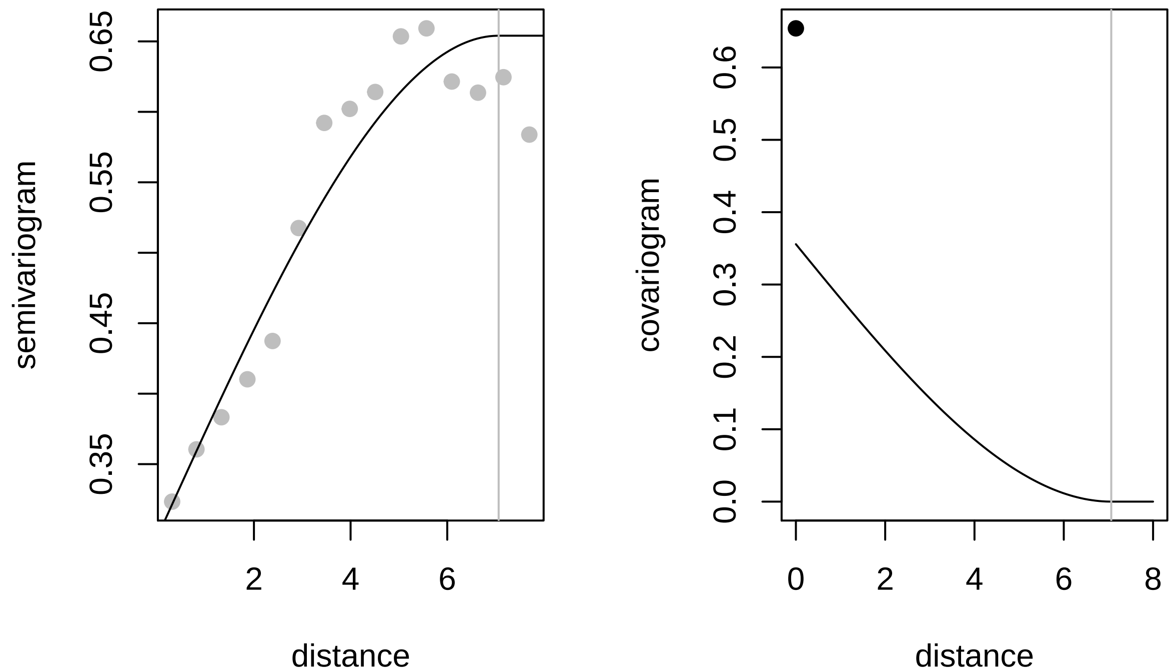

```
# coordinates(listin) <- ~longitude+ latitude


# st_crs(sf_listin) <- st_crs(tracts)
# st_join(tracts, sf_listin)

library(tigris)
# coord <- data.frame(lat = listin$latitude, long = listin$longitude)
# coord$census_code <- apply(coord, 1,
#                             function(row) call_geolocator_latlon(row['lat'], row['long']))
# coord$GEOID20 <- substr(coord$census_code, start = 1, stop = 11)
# coord$id <- listin$id
# coord$price <- listin$price
#
# write.csv(coord, 'coord.csv')
coord <- read.csv("coord.csv")
```

```r
with_code <- left_join(sf_listin, coord, by = "id")

try <- with_code %>%
  group_by(GEOID20) %>%
  do(tidy(lm(review_scores_value ~ 1, .)))

try3 <- coord %>% count(GEOID20)

try1 <- data.frame(GEOID20 = try3$GEOID20,
                   nopool_rs = try$estimate, stringsAsFactors = FALSE)
try1$GEOID20 <- as.character(try1$GEOID20)

try2 <- left_join(tracts, try1, by = "GEOID20")

try2 %>%
  ggplot(aes(fill = nopool_rs, color = nopool_rs))+
  geom_sf()+
  coord_sf(crs = 5070, datum = NA)+
  scale_fill_viridis(direction = -1)+
  scale_color_viridis(direction = -1)+
  labs(title = "Review scores", subtitle = "Nopooling by boston neighborhoods")
```




```r
spherical_variogram <- function (n, ps, r) function (h) {
  h <- h / r
  n + ps * ifelse(h < 1, 1.5 * h - .5 * h ^ 3, 1)
```

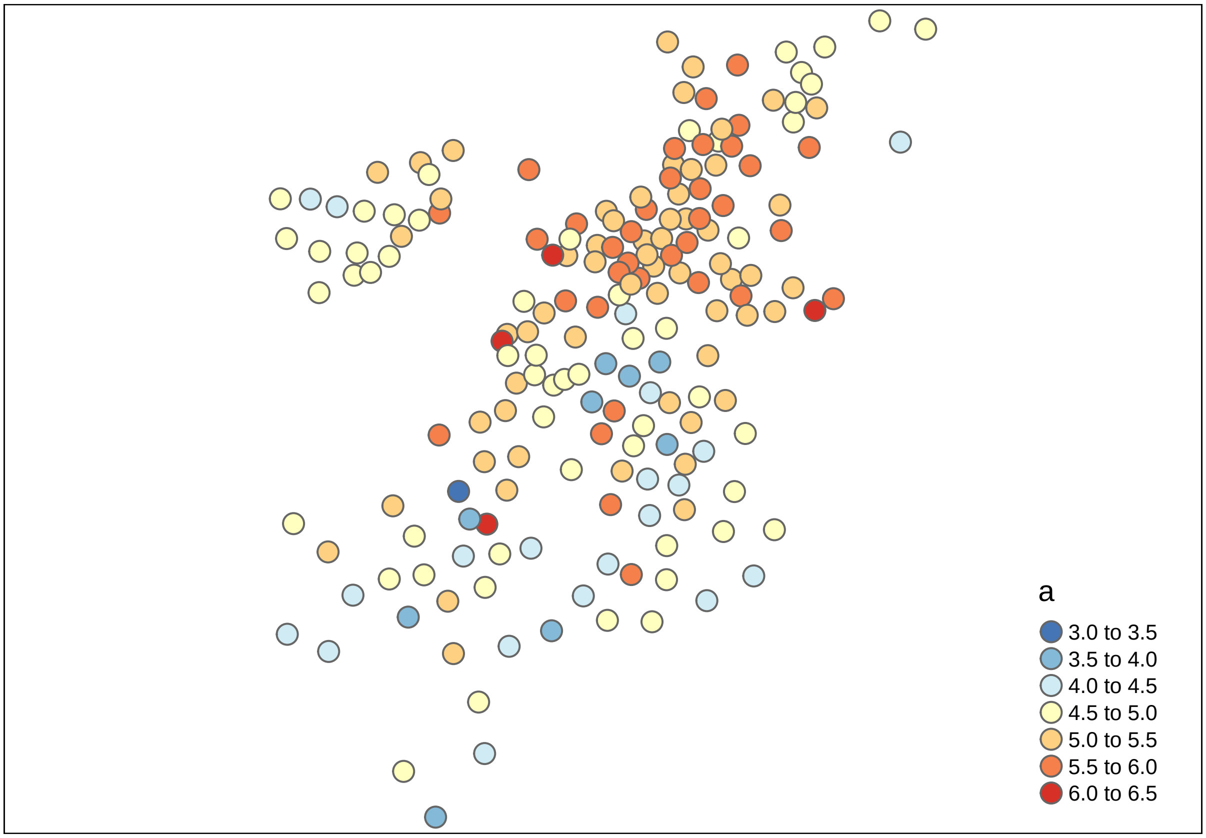
```r
}

chol_solve <- function (C, v) backsolve(C, backsolve(C, v, transpose = TRUE))

kriging_smooth_spherical <- function (formula, data, ...) {
  v <- variogram(formula, data)
  v_fit <- fit.variogram(v, vgm("Sph", ...))
  v_f <- spherical_variogram(v_fit$psill[1], v_fit$psill[2], v_fit$range[2])
  Sigma <- v_f(as.matrix(dist(coordinates(data)))) # semivariogram
  Sigma <- sum(v_fit$psill) - Sigma # prior variance
  tau2 <- v_fit$psill[1] # residual variance
  C <- chol(tau2 * diag(nrow(data)) + Sigma)
  y <- model.frame(formula, data)[, 1] # response
  x <- model.matrix(formula, data)
  # generalized least squares:
  beta <- coef(lm.fit(backsolve(C, x, transpose = TRUE),
                      backsolve(C, y, transpose = TRUE))) # prior mean
  Sigma_inv <- chol2inv(chol(Sigma))
  C <- chol(Sigma_inv + diag(nrow(data)) / tau2)
  # posterior mean (smoother):
  mu <- drop(chol_solve(C, y / tau2 + Sigma_inv %*% x %*% beta))
  list(smooth = mu, prior_mean = beta)
}

v <- variogram(log(price.x) ~ 1, with_code)
v_fit <- fit.variogram(v, vgm("Sph"))
v_f <- spherical_variogram(v_fit$psill[1], v_fit$psill[2], v_fit$range[2])
#
# # check variogram and covariance
op <- par(mfrow = c(1, 2))
h <- seq(0, 8, length = 100)
plot(v$dist, v$gamma,  pch = 19, col = "gray",
     xlab = "distance", ylab = "semivariogram")
lines(h, v_f(h))
abline(v = v_fit$range[2], col = "gray")
plot(h, sum(v_fit$psill) - v_f(h), type = "l",
     xlab = "distance", ylab = "covariogram",
     ylim = c(0, sum(v_fit$psill)))
points(0, sum(v_fit$psill), pch = 19)
abline(v = v_fit$range[2], col = "gray")
```




```
par(op)

mean_p_tracts <- coord %>%
 group_by(GEOID20) %>%
 summarise_at(vars(price), list(mean_p = mean)) %>%
 dplyr::select(GEOID20, mean_p)
mean_p_tracts$GEOID20 <- as.character(mean_p_tracts$GEOID20)

join_tracts <- left_join(tracts, mean_p_tracts, by = "GEOID20")

tract_2<-st_centroid(join_tracts) #Center the polygon
```

```
## Warning in st_centroid.sf(join_tracts): st_centroid assumes attributes are
## constant over geometries of x
```

```
tract_2 = na.omit(tract_2)
tract_2$a = log(tract_2$mean_p) #The distribution is un-normal, so we use the log transformation here.
# breaks <- seq(4.4, 6, by = .1)
tmap_arrange(
  tm_shape(tract_2) +
  tm_bubbles(col = "a", palette = "-RdYlBu", size = .3))
```

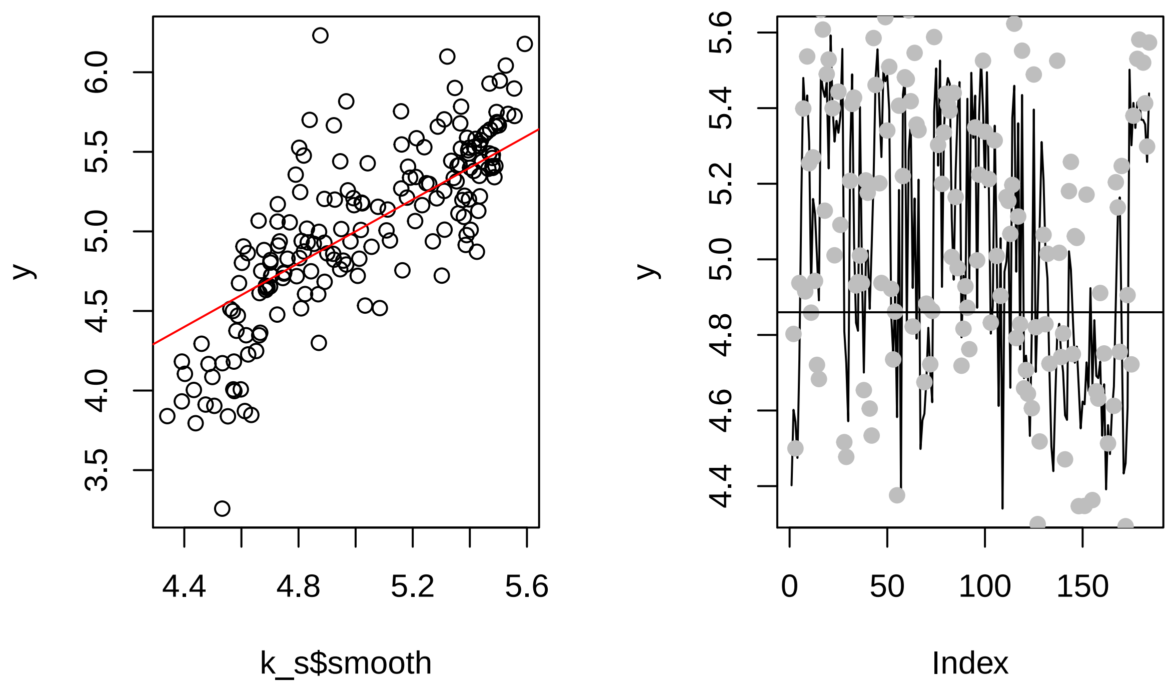



```
library(purrr)
```

```
## Warning: package 'purrr' was built under R version 4.0.5

##
## Attaching package: 'purrr'

## The following object is masked from 'package:magrittr':
##
##     set_names

## The following object is masked from 'package:data.table':
##
##     transpose
```

```
tract_3 <- tract_2 %>%
    mutate(x = unlist(map(tract_2$geometry,1)),
           y = unlist(map(tract_2$geometry,2)))
# tract_3
tract_4 <- tract_3 %>% st_sf() %>% as_Spatial()


k_s <- kriging_smooth_spherical(a ~ 1, tract_4)
y <- tract_4$a
op <- par(mfrow = c(1, 2))
plot(k_s$smooth, y); abline(0, 1, col = "red")
plot(k_s$smooth, type = "l", ylab = "y")
points(y, pch = 19, col = "gray")
```

```r
abline(h = k_s$prior_mean)
```

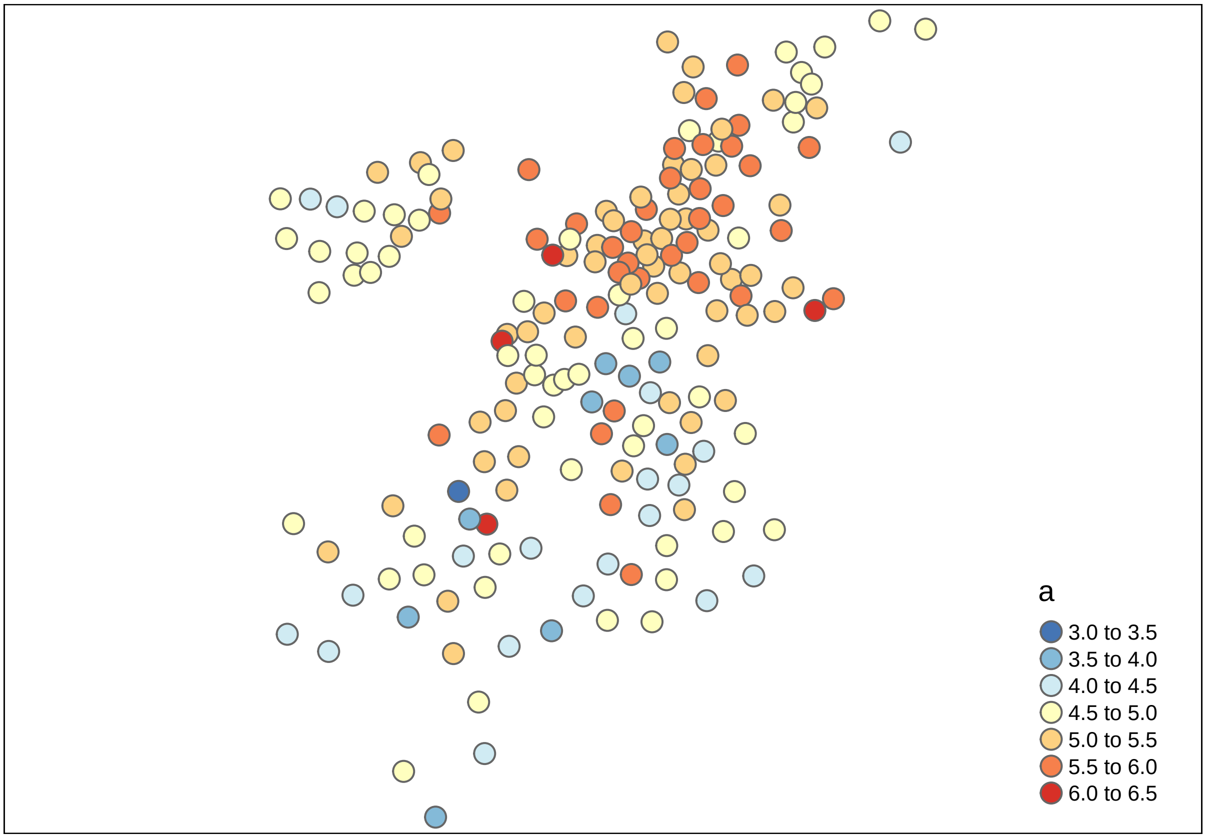



```r
par(op)
tract_2$smooth <- k_s$smooth

tmap_mode("plot")
```

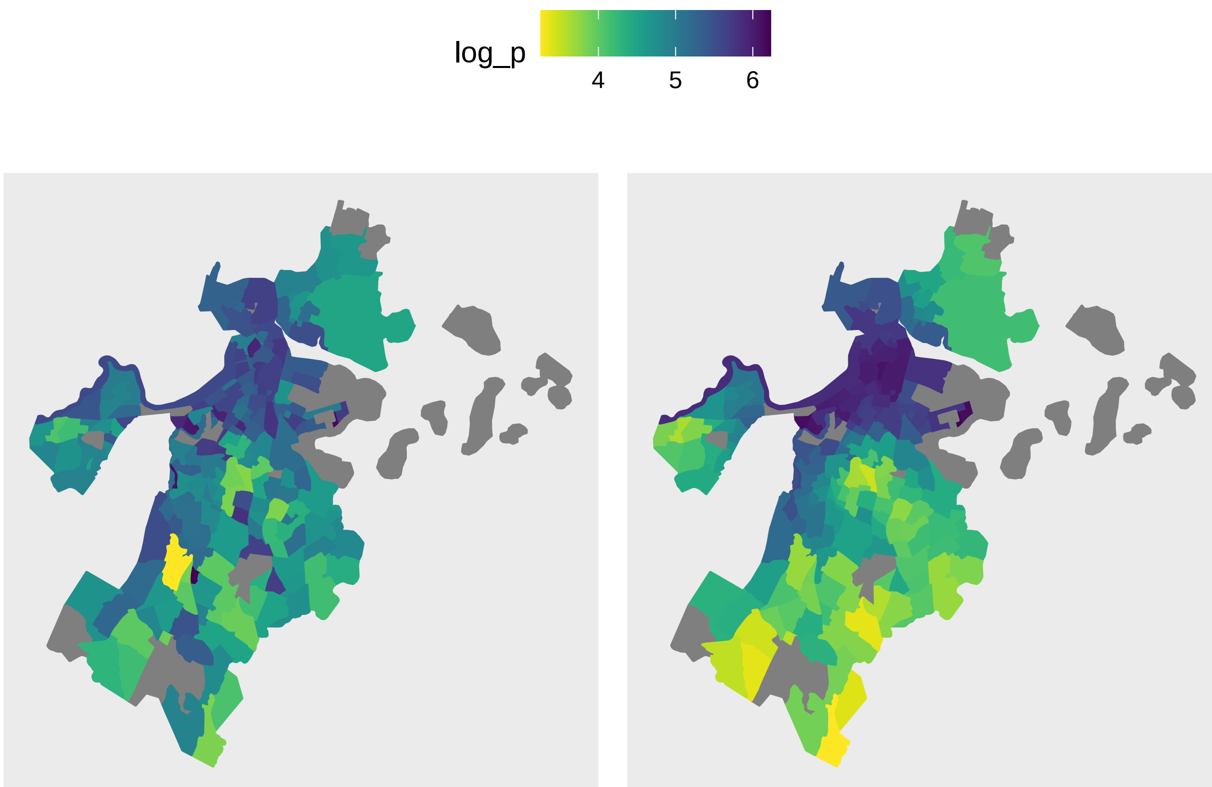
```
## tmap mode set to plotting
```

```r
tmap_arrange(
  tm_shape(tract_4) +
    tm_bubbles(col = "a", palette = "-RdYlBu", size = .3))
```

```
## Warning in sp::proj4string(obj): CRS object has comment, which is lost in output

## Warning in sp::proj4string(obj): CRS object has comment, which is lost in output
```




```r
# smoothed map comparison
smooth_p_t <- data.frame(GEOID20 = tract_2$GEOID20,
                    log_p = tract_2$a,
                    smooth = tract_2$smooth,
                    stringsAsFactors = FALSE)

smooth_p_t <- left_join(tracts, smooth_p_t, by = "GEOID20")

plot_original_t <-
  smooth_p_t %>%
  ggplot(aes(fill = log_p, color = log_p))+
  geom_sf()+
  coord_sf(crs = 5070, datum = NA)+
  scale_fill_viridis(direction = -1)+
  scale_color_viridis(direction = -1)

plot_smooth_t <-
  smooth_p_t %>%
  ggplot(aes(fill = smooth, color = smooth))+
  geom_sf()+
  coord_sf(crs = 5070, datum = NA)+
  scale_fill_viridis(direction = -1)+
  scale_color_viridis(direction = -1)

ggarrange(plot_original_t, plot_smooth_t, common.legend = TRUE)
```



Text mining (review's sentiment analysis)

1. load the dataframe containing the reviews text of listings
2. tidy the content of text: remove numbers, stop words (remove words without true meanings), convert each reviews text to one sentence so that sentiment analysis of each review can be analyzed
3. ten most positive and negative words used by customers in reviews
4. wordcloud of positive and negative words in reviews

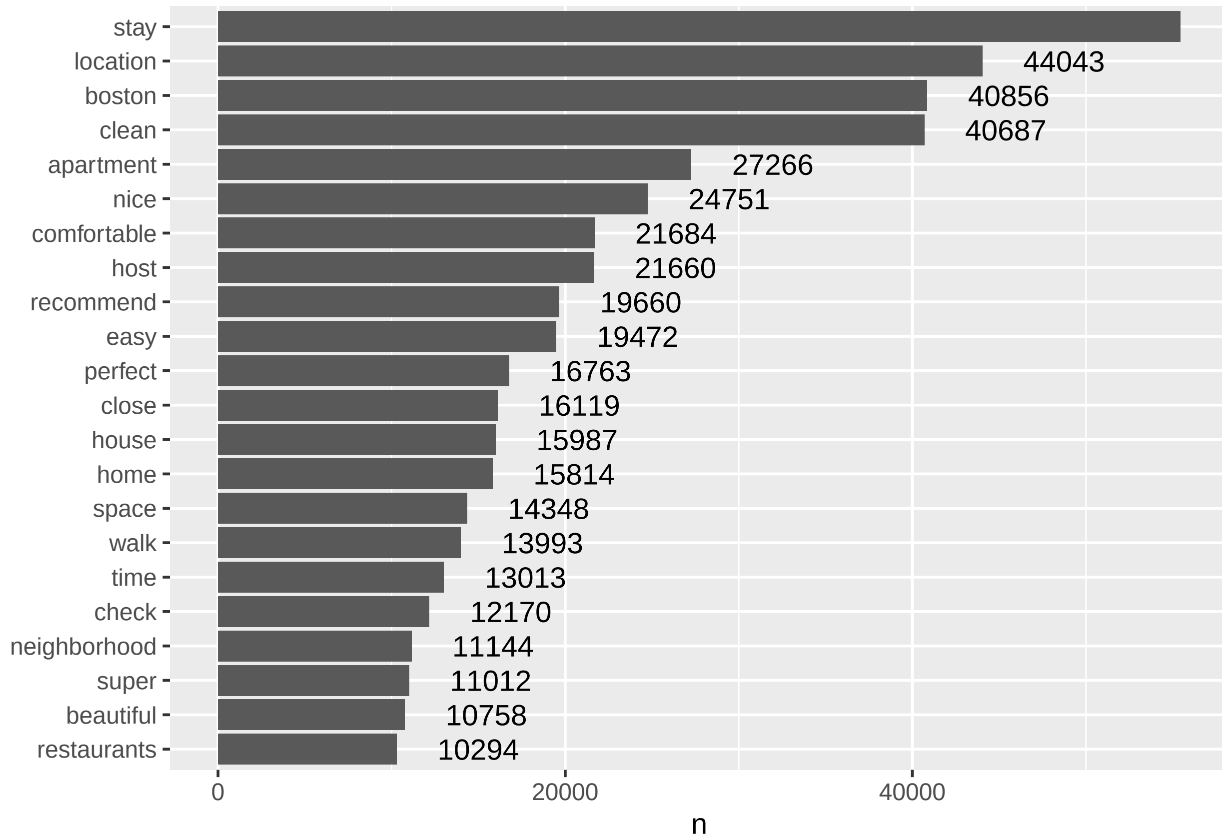
```
# word sentiment analysis
re$comments <- removeNumbers(re$comments)
tidy <- re %>%
  unnest_tokens(word, comments)

tidy <- tidy %>%
  anti_join(stop_words) %>%
  dplyr::select(listing_id, word)
```

```
## Joining, by = "word"
```

```
# table of word count
tidy %<>%
  filter(word != "br")

tidy %>%
  count(word, sort = TRUE) %>%
  filter(n > 10000) %>%
  mutate(word = reorder(word, n)) %>%
```

```
  ggplot(aes(n, word))+
  geom_col()+
  labs(y = NULL)+
  geom_text(aes(label = n), hjust=-0.5, position = "dodge")
```

```
## Warning: Width not defined. Set with `position_dodge(width = ?)`
```




```
get_sentiments("afinn")
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # ... with 2,467 more rows
```

```
get_sentiments("bing")
```

```
## # A tibble: 6,786 x 2
```

```
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # ... with 6,776 more rows
```

```
get_sentiments("nrc")
```

```
## # A tibble: 13,875 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abacus      trust
##  2 abandon     fear
##  3 abandon     negative
##  4 abandon     sadness
##  5 abandoned   anger
##  6 abandoned   fear
##  7 abandoned   negative
##  8 abandoned   sadness
##  9 abandonment anger
## 10 abandonment fear
## # ... with 13,865 more rows
```

```
id_nb <- listin %>% group_by(id, neighbourhood_cleansed) %>% dplyr::select(id, neighbourhood_cleansed)
names(id_nb)[1] <- "listing_id"
length(unique(tidy$listing_id)) # 2269
```

```
## [1] 2269
```

```
tidy_nb <- merge(tidy, id_nb, by = "listing_id")
dim(tidy_nb)
```

```
## [1] 1869502       3
```

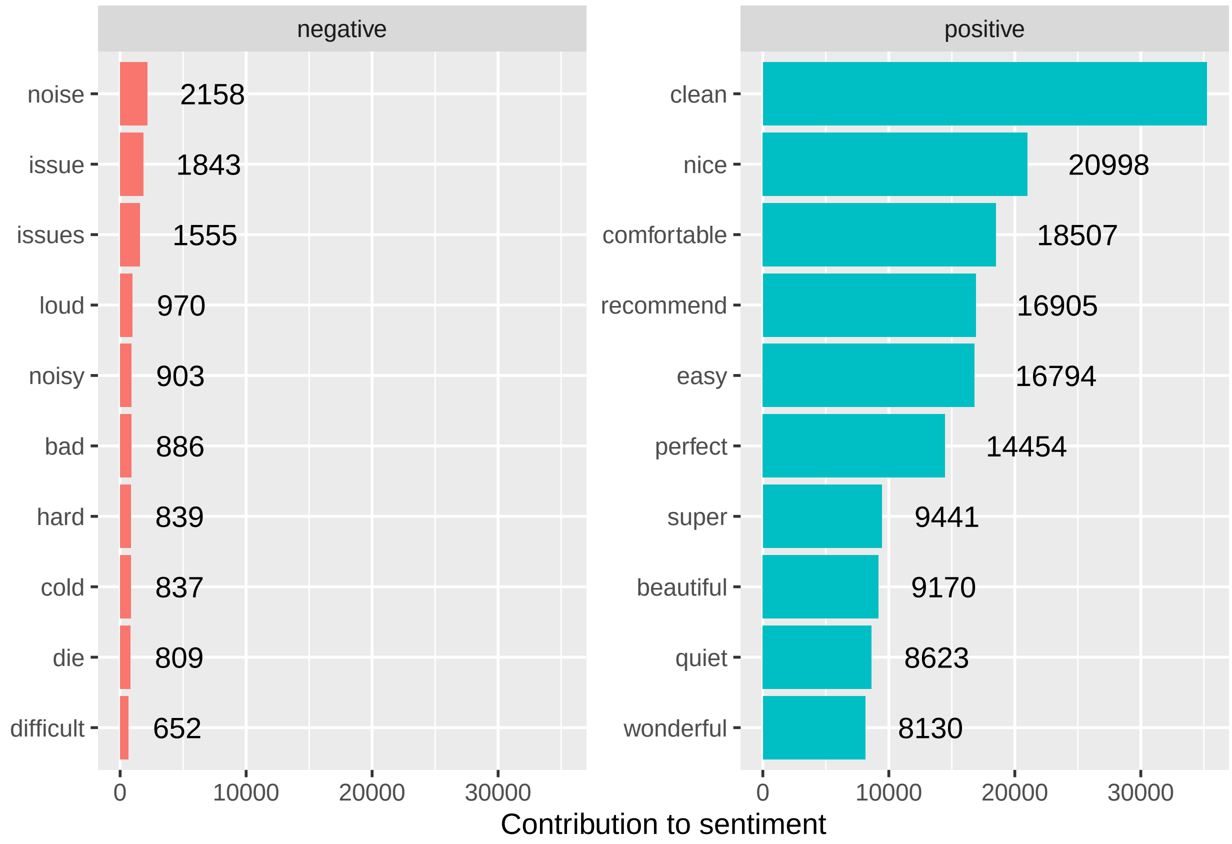
```
bing_word_counts <- tidy_nb %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

```
## Joining, by = "word"
```

```
bing_word_counts %>%
  group_by(sentiment) %>%
  slice_max(n, n = 10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  geom_text(aes(label = n), hjust=-0.5, position = "dodge")+
```

```
  facet_wrap(~sentiment, scales = "free_y") +
  labs(x = "Contribution to sentiment",
       y = NULL)
```

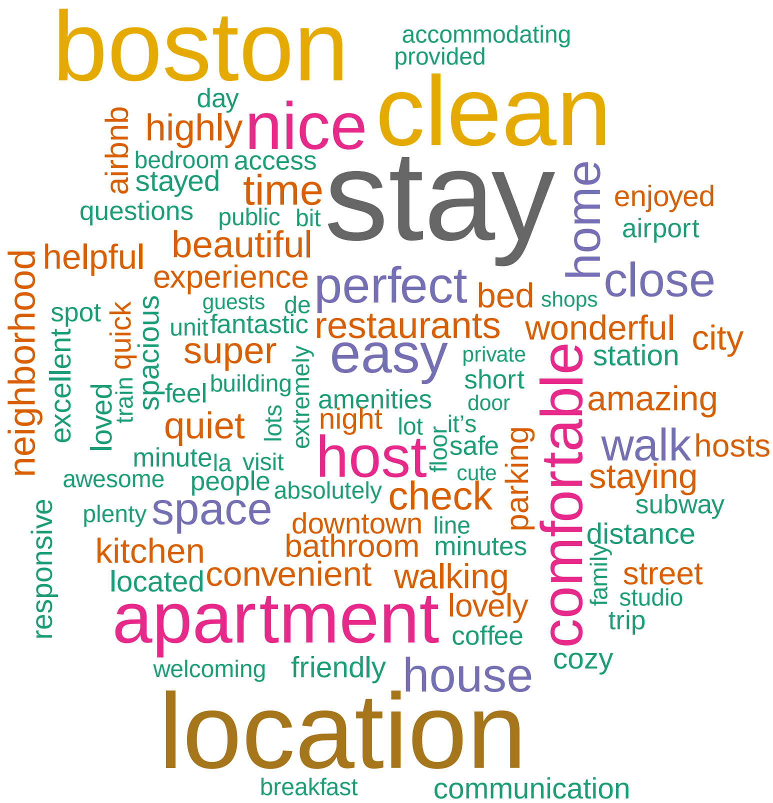
```
## Warning: Width not defined. Set with `position_dodge(width = ?)`
```




```
tidy_nb %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100, colors=brewer.pal(8, "Dark2")))
```

```
## Joining, by = "word"
```

```
## Warning in wordcloud(word, n, max.words = 100, colors = brewer.pal(8, "Dark2")):
## recommend could not be fit on page. It will not be plotted.
```

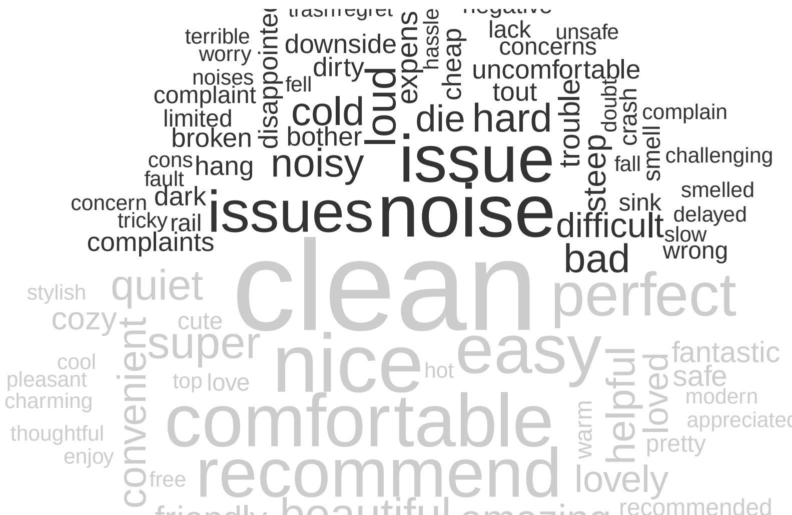



```
# positive and negative wordcloud
po_word_counts <- bing_word_counts %>% filter(sentiment == "positive") %>% dplyr::select(word, n)
ne_word_counts <- bing_word_counts %>% filter(sentiment == "negative") %>% dplyr::select(word, n)

wordcloud2(po_word_counts, size=1.6, color='random-dark')
```

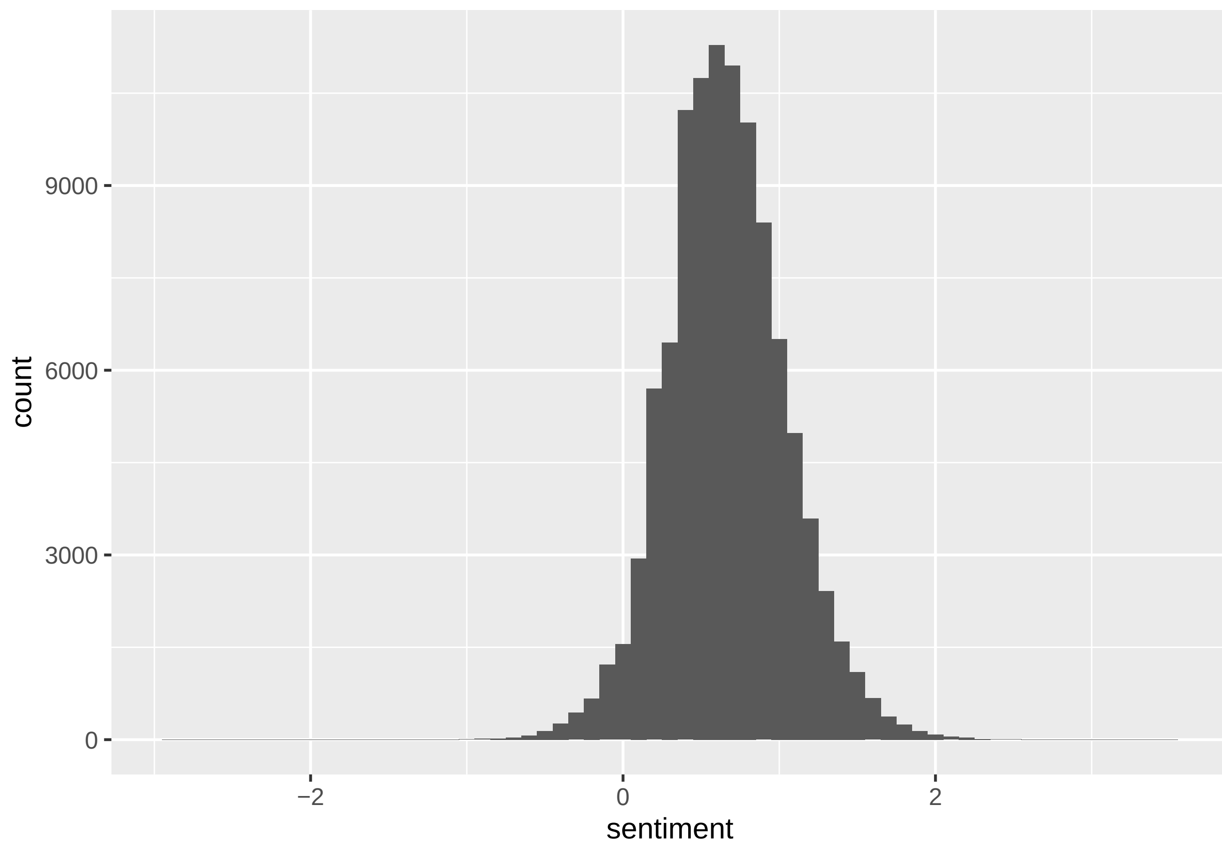

```
po_cloud <- wordcloud2(po_word_counts, size = 1, minRotation = -0.52, maxRotation = -0.52, rotateRatio =

wordcloud2(ne_word_counts, size=1.6, color='random-dark')
```


```r
ne_cloud <- wordcloud2(ne_word_counts, size = 1, minRotation = -0.52, maxRotation = -0.52, rotateRatio

# save image
# webshot::install_phantomjs()
# library("htmlwidgets")
# saveWidget(po_cloud,"po_cloud.html",selfcontained = F)
# webshot("po_cloud.html", "po_cloud.png", delay =5, vwidth = 650, vheight=650)

tidy_nb %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray80"),
                   max.words = 100)
```

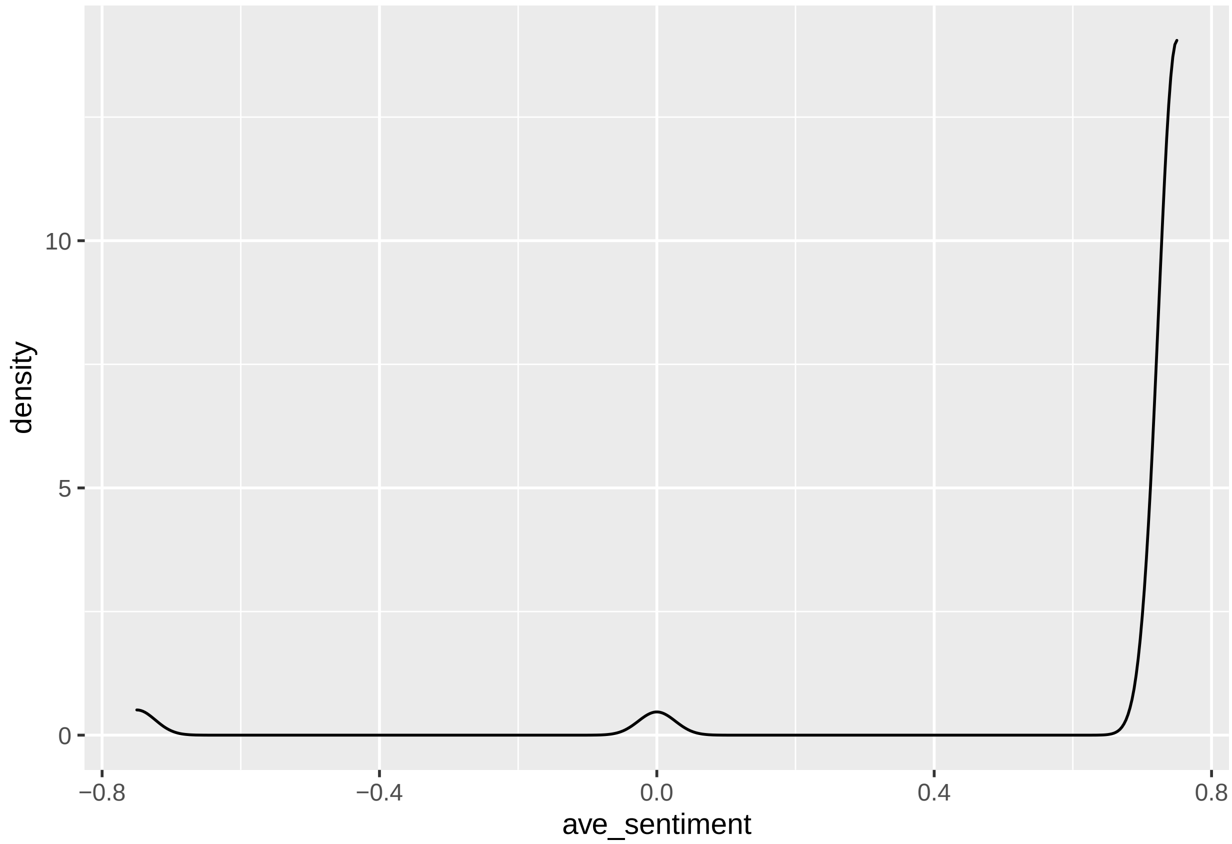
```
## Joining, by = "word"
```




```
# sentence sentiment analysis

review$raword <- removePunctuation(review$comments)
review$raword <- paste(review$raword, ". ")

# dat4$raword <- gsub('\\.', '', dat4$comments)
# dat4$raword <- tolower(dat4$raword)
# install.packages("splus2R")
# library(splus2R)
# lowerCase(REVIEWS$raword)

# comments <- review$raword %>%
#   get_sentences() %>%
#    sentiment() %>%
#    mutate(polarity_level = ifelse(sentiment < 0, "Negative",
#                                ifelse(sentiment > 0,
#                                       "Positive","Neutral")))

# write.csv(comments, 'comments.csv')
comments <- read.csv("comments.csv")

comments %>% filter(polarity_level != "Neutral") %>%
  ggplot() + geom_histogram(aes(x = sentiment), binwidth = .1, bins = 30)
```




```
# density plot
comments %>%
  get_sentences() %>%
  sentiment_by(by = NULL) %>% #View()
  ggplot() + geom_density(aes(ave_sentiment))
```



Model fitting

log(price) and neighborhoods

1. complete pooling model
2. unpooled model
3. partial pooling model
4. plot comparison of unpooled and partial pooling

```
# listin group by neighborhoods
pooled <- lm(log(price) ~ 1, data = listin)
display(pooled)
```

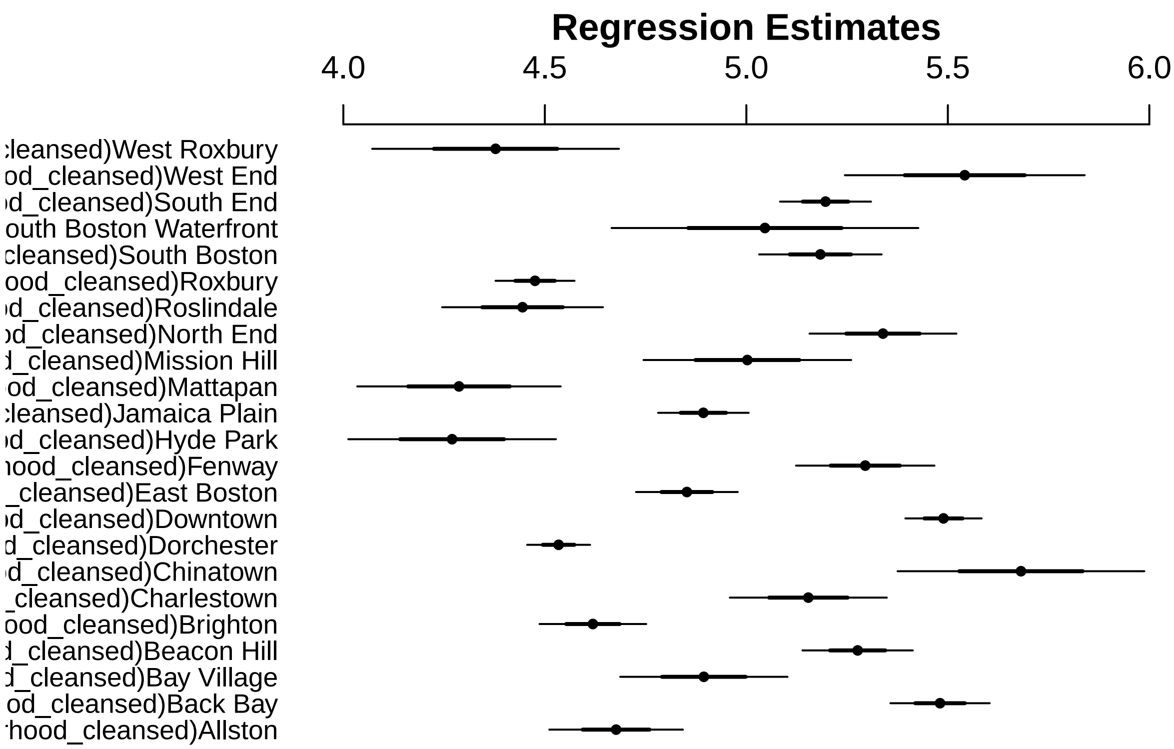
```
## lm(formula = log(price) ~ 1, data = listin)
##             coef.est coef.se
## (Intercept) 4.94     0.02
## ---
## n = 1721, k = 1
## residual sd = 0.74, R-Squared = 0.00
```

```
# mrv
unpooled <- lm(log(price) ~ factor(neighbourhood_cleansed) -1, data = listin)
display(unpooled)
```

```
## lm(formula = log(price) ~ factor(neighbourhood_cleansed) - 1,
##     data = listin)
##                                                         coef.est coef.se
## factor(neighbourhood_cleansed)Allston                   4.68     0.08
```

```
## factor(neighbourhood_cleansed)Back Bay                 5.48     0.06
## factor(neighbourhood_cleansed)Bay Village              4.89     0.10
## factor(neighbourhood_cleansed)Beacon Hill              5.28     0.07
## factor(neighbourhood_cleansed)Brighton                 4.62     0.07
## factor(neighbourhood_cleansed)Charlestown              5.15     0.10
## factor(neighbourhood_cleansed)Chinatown                5.68     0.15
## factor(neighbourhood_cleansed)Dorchester               4.53     0.04
## factor(neighbourhood_cleansed)Downtown                 5.49     0.05
## factor(neighbourhood_cleansed)East Boston              4.85     0.06
## factor(neighbourhood_cleansed)Fenway                   5.30     0.09
## factor(neighbourhood_cleansed)Hyde Park                4.27     0.13
## factor(neighbourhood_cleansed)Jamaica Plain            4.89     0.06
## factor(neighbourhood_cleansed)Mattapan                 4.29     0.13
## factor(neighbourhood_cleansed)Mission Hill             5.00     0.13
## factor(neighbourhood_cleansed)North End                5.34     0.09
## factor(neighbourhood_cleansed)Roslindale               4.44     0.10
## factor(neighbourhood_cleansed)Roxbury                  4.48     0.05
## factor(neighbourhood_cleansed)South Boston             5.18     0.08
## factor(neighbourhood_cleansed)South Boston Waterfront 5.05     0.19
## factor(neighbourhood_cleansed)South End                5.20     0.06
## factor(neighbourhood_cleansed)West End                 5.54     0.15
## factor(neighbourhood_cleansed)West Roxbury             4.38     0.15
## ---
## n = 1721, k = 23
## residual sd = 0.63, R-Squared = 0.98
```

```
coefplot(unpooled)
```




```
partial <- lmer(log(price) ~ 1+ (1 | neighbourhood_cleansed),
                data = listin)
display(partial)
```

```
## lmer(formula = log(price) ~ 1 + (1 | neighbourhood_cleansed),
##     data = listin)
## coef.est  coef.se
##     4.96     0.09
##
## Error terms:
##  Groups                 Name        Std.Dev.
##  neighbourhood_cleansed (Intercept) 0.41
##  Residual                           0.63
## ---
## number of obs: 1721, groups: neighbourhood_cleansed, 23
## AIC = 3381.8, DIC = 3369.7
## deviance = 3372.8
```

```
head(coef(partial)$neighbourhood_cleansed)
```

```
##              (Intercept)
## Allston         4.687673
## Back Bay        5.469297
## Bay Village     4.898238
## Beacon Hill     5.267651
## Brighton        4.627626
## Charlestown     5.143435
```

```r
head(ranef(partial)$neighbourhood_cleansed)
```

```
##             (Intercept)
## Allston     -0.26926049
## Back Bay     0.51236273
## Bay Village -0.05869636
## Beacon Hill  0.31071700
## Brighton    -0.32930824
## Charlestown  0.18650149
```

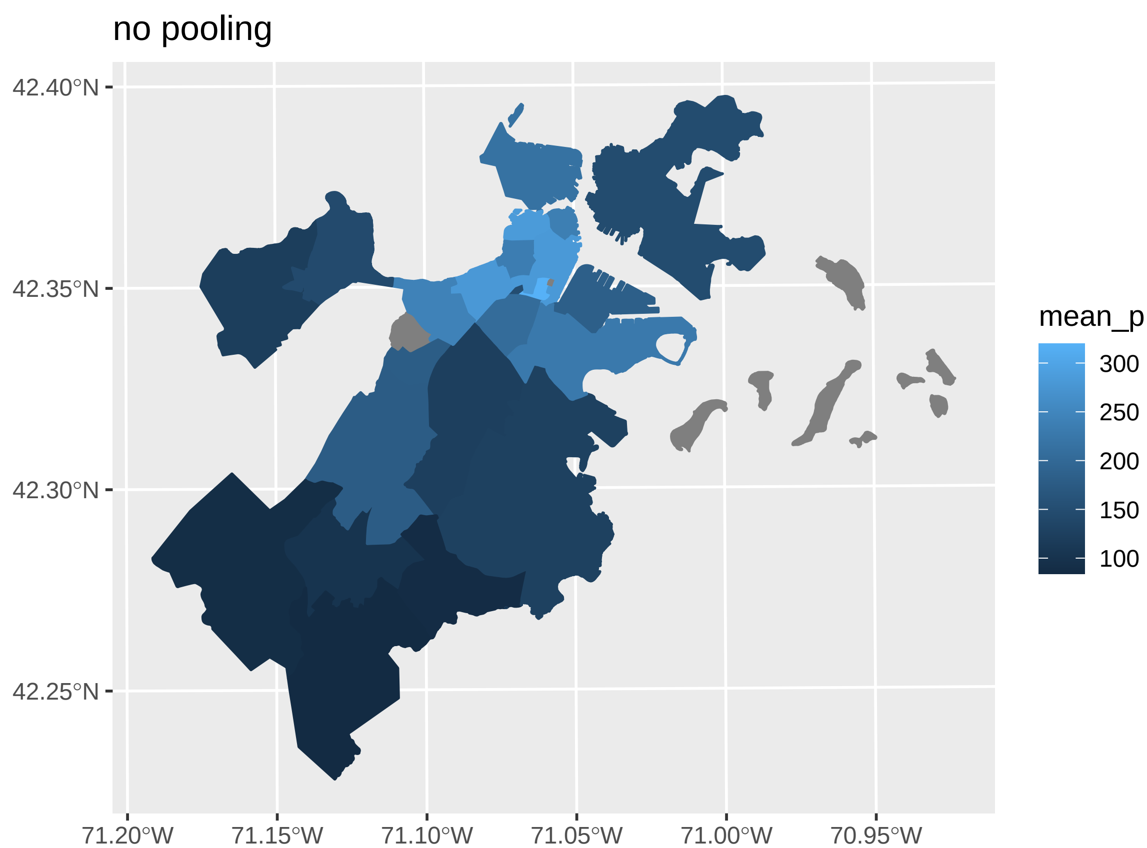
```r
# prediction for partial pooling
p_pred <- predict(partial,
                  newdata = data.frame(neighbourhood_cleansed = mp$Name))
length(p_pred) # 23
```

```
## [1] 23
```

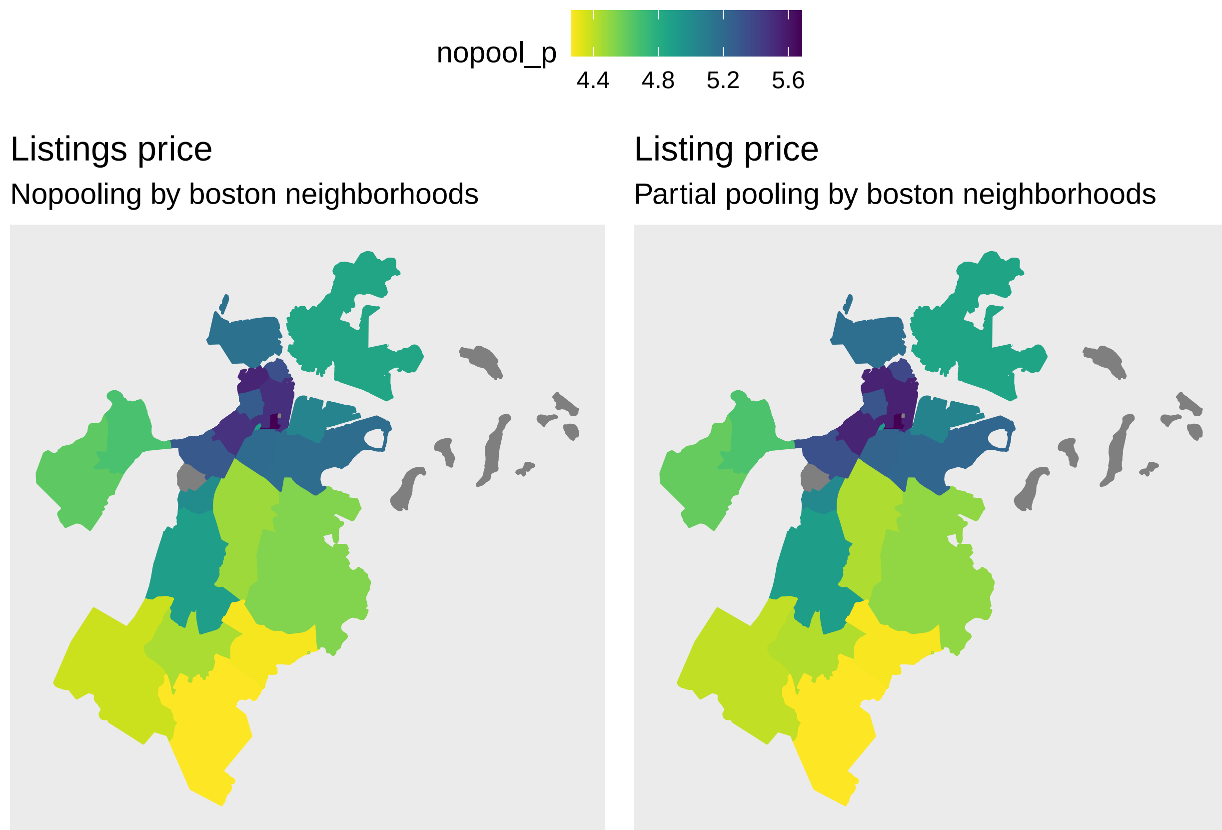
```r
tdp2 <- data.frame(Name = mp$Name,
                   pre_p = p_pred,
                   stringsAsFactors = FALSE)
# head(tdp2)
# dim(tdp2) # 23 2
all_join <- left_join(join_p, tdp2, by = "Name")
dim(all_join)
```

```
## [1] 26 11
```

```r
# prediction for complete pooling
cpred <- predict(pooled, newdata = data.frame(1))


# unpooled map
ggplot(data = all_join) +
  geom_sf(aes(fill =mean_p, color = mean_p)) +
  # scale_fill_gradient(limits = c(min,max), na.value = NA)+
  ggtitle("no pooling")
```




```
plot_partial <-
  all_join %>%
  ggplot(aes(fill = pre_p, color = pre_p))+
  geom_sf()+
  coord_sf(crs = 5070, datum = NA)+
  scale_fill_viridis(direction = -1)+
  scale_color_viridis(direction = -1)+
  # scale_fill_gradientn(colors = viridis_pal()(9), limits=c(min, max),
                       # na.value = "grey50")+
  labs(title = "Listing price", subtitle = "Partial pooling by boston neighborhoods")

# grid.arrange(plot_nopool_p, plot_partial, ncol = 2)
ggarrange(plot_nopool_p, plot_partial, common.legend = TRUE)
```



Other models

1. model filtering out insignificant predictors
2. include different slope
3. residual plot

```
# density of superhost&review score value
gg_color_hue <- function(n) {
  hues = seq(15, 375, length = n + 1)
  hcl(h = hues, l = 65, c = 100)[1:n]
}
n = 2
cols = gg_color_hue(n)

# ggplot(listin)+
#   geom_density(aes(x = review_scores_value, color = factor(host_is_superhost)))+
#   xlab("Review scores value")+
#   ggtitle("Density of review scores value")+
#   annotate("text", x = 3.9, y = 1, label = "not super hosts",color=cols[1])+
#   annotate("text", x = 4.3, y = 2.5, label = "super hosts",color=cols[2])+
#   theme(legend.position = "none")


partial_2 <- lmer(log(price) ~ 1+
                    host_response_time+
                    host_response_rate+
```

```r
                     host_is_superhost+
                     host_has_profile_pic+
                     host_identity_verified+
                     review_scores_value+
                     room_type+
                     host_total_listings_count+
                     license_ornot+
                     (1 | neighbourhood_cleansed),
                 data = listin)
display(partial_2)
```

```
## lmer(formula = log(price) ~ 1 + host_response_time + host_response_rate +
##     host_is_superhost + host_has_profile_pic + host_identity_verified +
##     review_scores_value + room_type + host_total_listings_count +
##     license_ornot + (1 | neighbourhood_cleansed), data = listin)
##                                       coef.est coef.se
## (Intercept)                            6.26     0.51
## host_response_timewithin a few hours -0.15     0.04
## host_response_timewithin a day       -0.14     0.04
## host_response_timea few days or more -0.53     0.15
## host_response_rate                   -0.60     0.16
## host_is_superhostt                   -0.13     0.03
## host_has_profile_pict                -0.51     0.47
## host_identity_verifiedt              -0.02     0.03
## review_scores_value                  -0.02     0.03
## room_typePrivate room                -0.81     0.03
## room_typeHotel room                   0.14     0.12
## room_typeShared room                 -1.31     0.33
## host_total_listings_count            -0.01     0.00
## license_ornot1                        0.45     0.03
##
## Error terms:
##  Groups                 Name        Std.Dev.
##  neighbourhood_cleansed (Intercept) 0.25
##  Residual                           0.46
## ---
## number of obs: 1721, groups: neighbourhood_cleansed, 23
## AIC = 2379.7, DIC = 2227.4
## deviance = 2287.5
```

```r
head(coef(partial_2)$neighbourhood_cleansed)
```

```
##             (Intercept) host_response_timewithin a few hours
## Allston        6.139338                           -0.1493199
## Back Bay       6.588553                           -0.1493199
## Bay Village    6.300797                           -0.1493199
## Beacon Hill    6.457424                           -0.1493199
## Brighton       6.045576                           -0.1493199
## Charlestown    6.411881                           -0.1493199
##             host_response_timewithin a day host_response_timea few days or more
## Allston                          -0.1432756                           -0.5341036
## Back Bay                         -0.1432756                           -0.5341036
## Bay Village                      -0.1432756                           -0.5341036
## Beacon Hill                      -0.1432756                           -0.5341036
```

```
## Brighton                        -0.1432756                              -0.5341036
## Charlestown                     -0.1432756                              -0.5341036
##             host_response_rate host_is_superhostt host_has_profile_pict
## Allston             -0.5992116         -0.1291349            -0.5064195
## Back Bay            -0.5992116         -0.1291349            -0.5064195
## Bay Village         -0.5992116         -0.1291349            -0.5064195
## Beacon Hill         -0.5992116         -0.1291349            -0.5064195
## Brighton            -0.5992116         -0.1291349            -0.5064195
## Charlestown         -0.5992116         -0.1291349            -0.5064195
##             host_identity_verifiedt review_scores_value room_typePrivate room
## Allston                 -0.02131382         -0.01761556           -0.8093631
## Back Bay                -0.02131382         -0.01761556           -0.8093631
## Bay Village             -0.02131382         -0.01761556           -0.8093631
## Beacon Hill             -0.02131382         -0.01761556           -0.8093631
## Brighton                -0.02131382         -0.01761556           -0.8093631
## Charlestown             -0.02131382         -0.01761556           -0.8093631
##             room_typeHotel room room_typeShared room host_total_listings_count
## Allston               0.1368516           -1.307374              -0.009512831
## Back Bay              0.1368516           -1.307374              -0.009512831
## Bay Village           0.1368516           -1.307374              -0.009512831
## Beacon Hill           0.1368516           -1.307374              -0.009512831
## Brighton              0.1368516           -1.307374              -0.009512831
## Charlestown           0.1368516           -1.307374              -0.009512831
##             license_ornot1
## Allston          0.4480474
## Back Bay         0.4480474
## Bay Village      0.4480474
## Beacon Hill      0.4480474
## Brighton         0.4480474
## Charlestown      0.4480474
```

```
fixef(partial_2)
```

```
##                          (Intercept) host_response_timewithin a few hours
##                          6.259800041                         -0.149319865
##       host_response_timewithin a day host_response_timea few days or more
##                         -0.143275593                         -0.534103594
##                   host_response_rate                   host_is_superhostt
##                         -0.599211640                         -0.129134906
##                host_has_profile_pict              host_identity_verifiedt
##                         -0.506419544                         -0.021313824
##                  review_scores_value                 room_typePrivate room
##                         -0.017615560                         -0.809363119
##                  room_typeHotel room                  room_typeShared room
##                          0.136851584                         -1.307374329
##            host_total_listings_count                       license_ornot1
##                         -0.009512831                          0.448047396
```

```
head(ranef(partial_2)$neighbourhood_cleansed)
```

```
##             (Intercept)
## Allston     -0.12046232
## Back Bay     0.32875296
## Bay Village  0.04099666
## Beacon Hill  0.19762413
```

```
## Brighton    -0.21422377
## Charlestown  0.15208087
```

```r
# filter not significant predictors
partial_3 <- lmer(log(price) ~
                    host_response_time+
                    host_response_rate+
                    host_is_superhost+
                    review_scores_value+
                    room_type+
                    host_total_listings_count+
                    license_ornot+
                    (1 + host_total_listings_count+ host_is_superhost| neighbourhood_cleansed),
                data = listin)
display(partial_3)
```

```
## lmer(formula = log(price) ~ host_response_time + host_response_rate +
##     host_is_superhost + review_scores_value + room_type + host_total_listings_count +
##     license_ornot + (1 + host_total_listings_count + host_is_superhost |
##     neighbourhood_cleansed), data = listin)
##                                      coef.est coef.se
## (Intercept)                           5.67     0.21
## host_response_timewithin a few hours -0.13     0.04
## host_response_timewithin a day       -0.13     0.04
## host_response_timea few days or more -0.53     0.15
## host_response_rate                   -0.61     0.16
## host_is_superhostt                   -0.12     0.04
## review_scores_value                   0.00     0.03
## room_typePrivate room                -0.81     0.03
## room_typeHotel room                   0.17     0.12
## room_typeShared room                 -1.42     0.34
## host_total_listings_count             0.00     0.00
## license_ornot1                        0.43     0.03
##
## Error terms:
##  Groups                 Name                      Std.Dev. Corr
##  neighbourhood_cleansed (Intercept)               0.33
##                         host_total_listings_count 0.02     -0.75
##                         host_is_superhostt        0.14     -0.88  0.96
##  Residual                                         0.45
## ---
## number of obs: 1721, groups: neighbourhood_cleansed, 23
## AIC = 2338.1, DIC = 2194.6
## deviance = 2247.4
```

```r
head(coef(partial_3)$neighbourhood_cleansed)
```

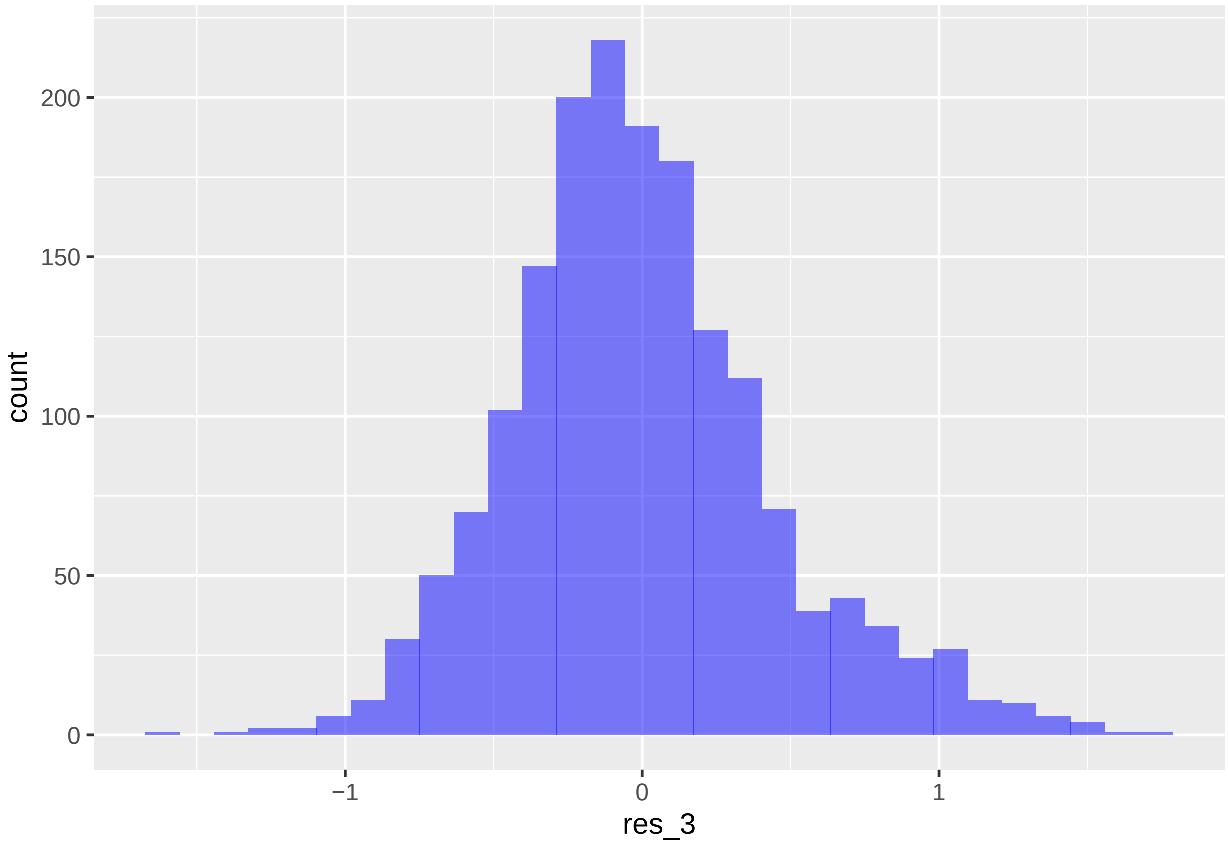
```
##             (Intercept) host_response_timewithin a few hours
## Allston        5.631716                           -0.1315033
## Back Bay       6.144909                           -0.1315033
## Bay Village    6.183978                           -0.1315033
## Beacon Hill    6.042324                           -0.1315033
## Brighton       5.422802                           -0.1315033
## Charlestown    5.973230                           -0.1315033
##             host_response_timewithin a day host_response_timea few days or more
```

```
## Allston                        -0.1295262                          -0.5276708
## Back Bay                       -0.1295262                          -0.5276708
## Bay Village                    -0.1295262                          -0.5276708
## Beacon Hill                    -0.1295262                          -0.5276708
## Brighton                       -0.1295262                          -0.5276708
## Charlestown                    -0.1295262                          -0.5276708
##             host_response_rate host_is_superhostt review_scores_value
## Allston             -0.6146354        -0.18589378         -0.00400253
## Back Bay            -0.6146354        -0.25685192         -0.00400253
## Bay Village         -0.6146354        -0.36770799         -0.00400253
## Beacon Hill         -0.6146354        -0.24624423         -0.00400253
## Brighton            -0.6146354        -0.07422797         -0.00400253
## Charlestown         -0.6146354        -0.25221477         -0.00400253
##             room_typePrivate room room_typeHotel room room_typeShared room
## Allston                 -0.811931           0.1679006            -1.418426
## Back Bay                -0.811931           0.1679006            -1.418426
## Bay Village             -0.811931           0.1679006            -1.418426
## Beacon Hill             -0.811931           0.1679006            -1.418426
## Brighton                -0.811931           0.1679006            -1.418426
## Charlestown             -0.811931           0.1679006            -1.418426
##             host_total_listings_count license_ornot1
## Allston                 -0.0146312565      0.4311717
## Back Bay                -0.0145754654      0.4311717
## Bay Village             -0.0343581024      0.4311717
## Beacon Hill             -0.0159926170      0.4311717
## Brighton                 0.0004071639      0.4311717
## Charlestown             -0.0185073030      0.4311717
```

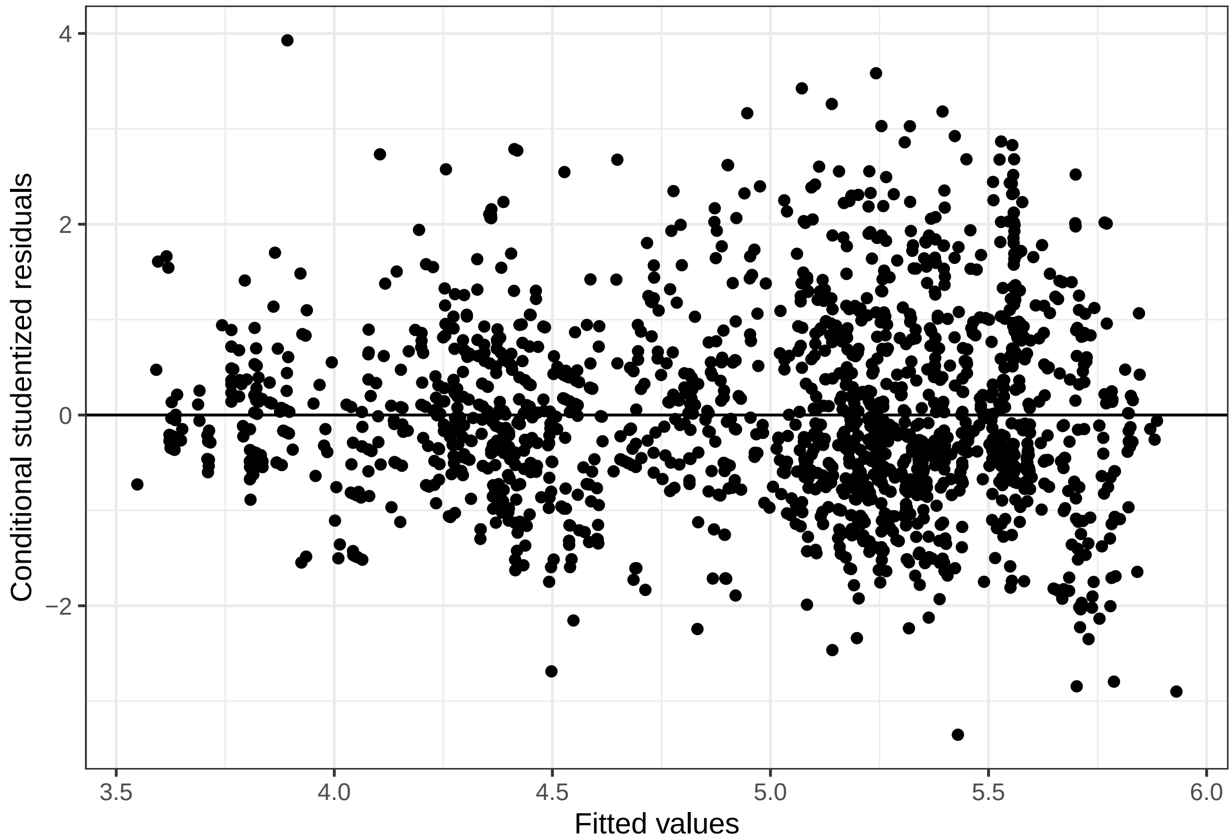
```r
head(ranef(partial_3)$neighbourhood_cleansed)
```

```
##             (Intercept) host_total_listings_count host_is_superhostt
## Allston     -0.03547663              -0.012476999        -0.06562851
## Back Bay     0.47771701              -0.012421207        -0.13658666
## Bay Village  0.51678526              -0.032203845        -0.24744272
## Beacon Hill  0.37513196              -0.013838359        -0.12597897
## Brighton    -0.24438982               0.002561422         0.04603729
## Charlestown  0.30603792              -0.016353045        -0.13194951
```

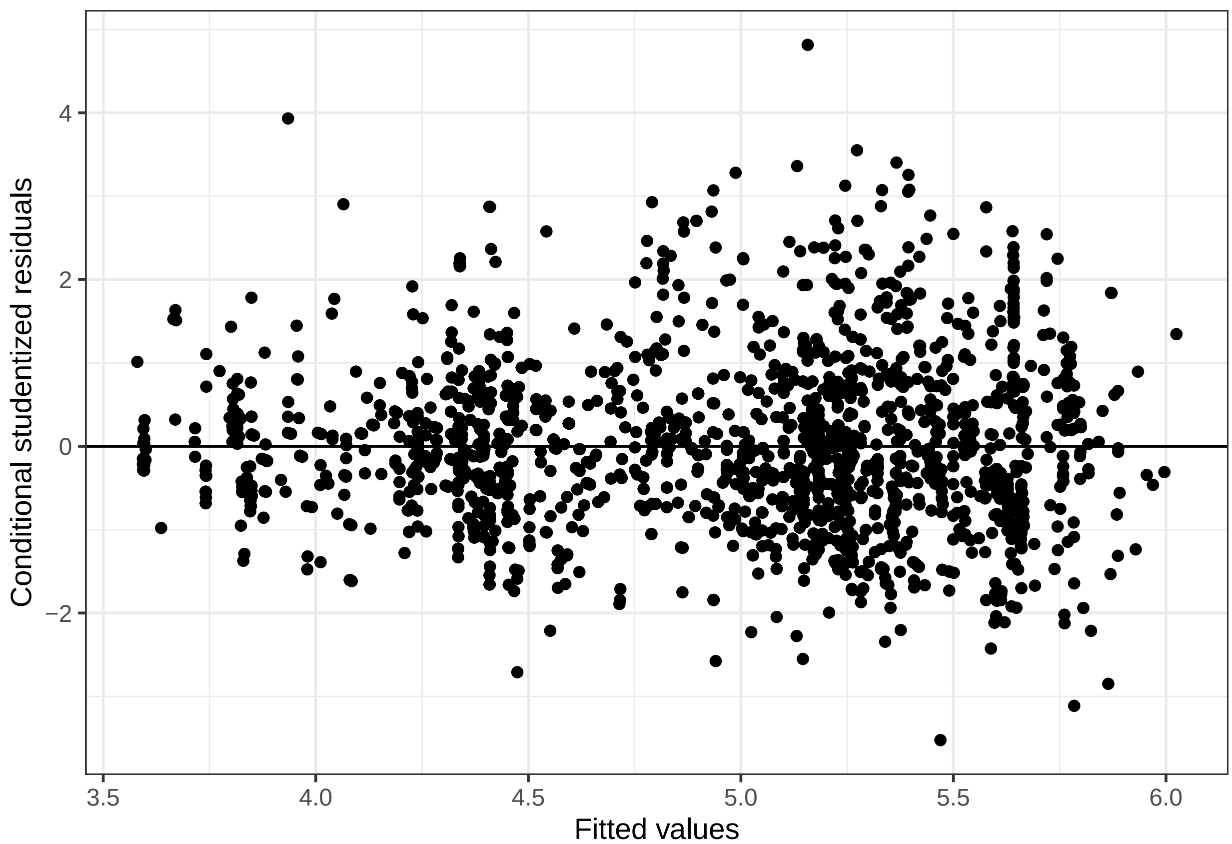
```r
res_3 <- residuals(partial_3)
res_3 <- as.data.frame(res_3)
ggplot(res_3, aes(res_3))+  geom_histogram(fill = 'blue', alpha = 0.5)
```



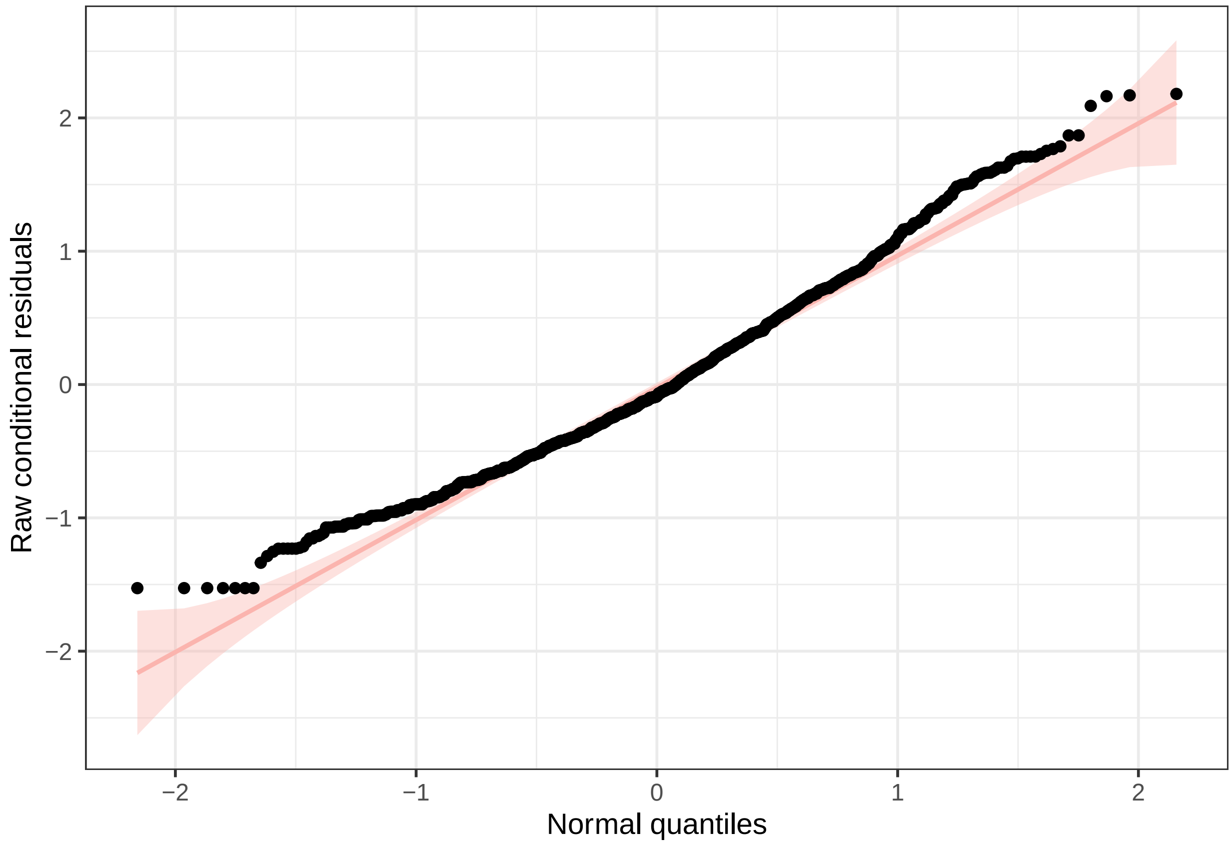

```
# plot(partial_3)

# redyes redres
# devtools::install_github("goodekat/redres")

# all kinds of residual of model3
rc_resids <- compute_redres(partial_3)
pm_resids <- compute_redres(partial_3, type = "pearson_mar")
sc_resids <- compute_redres(partial_3, type = "std_cond")


plot_redres(partial_2, type = "std_cond")
```



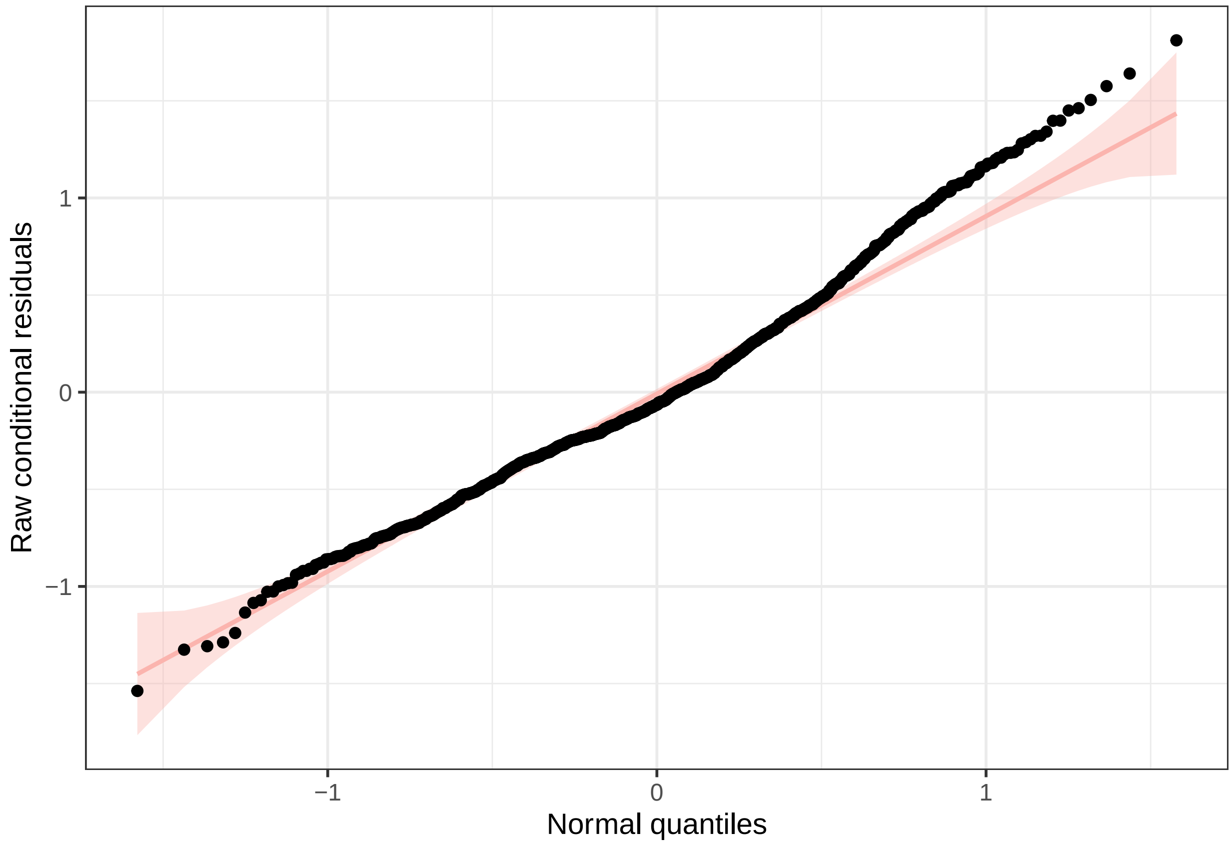

```
plot_redres(partial_3, type = "std_cond")
```



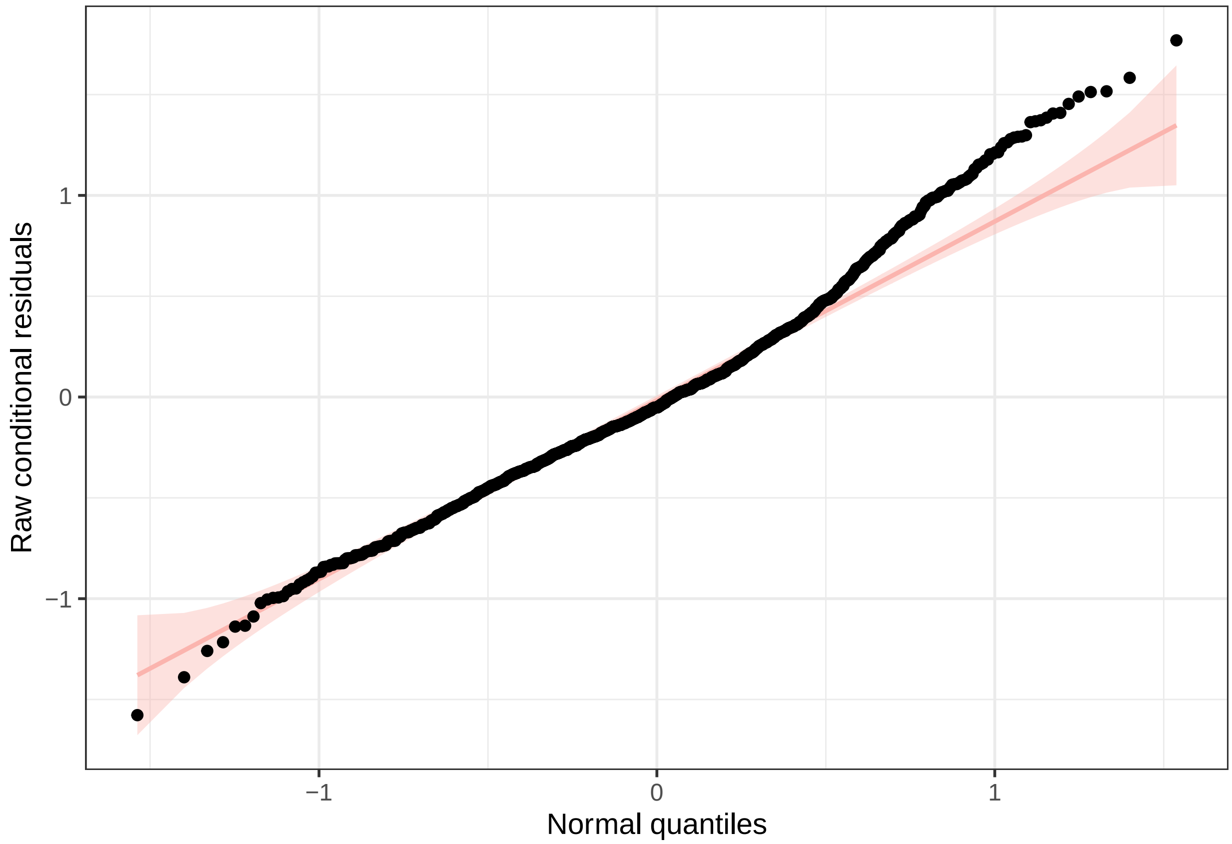

```
plot_resqq(partial)
```



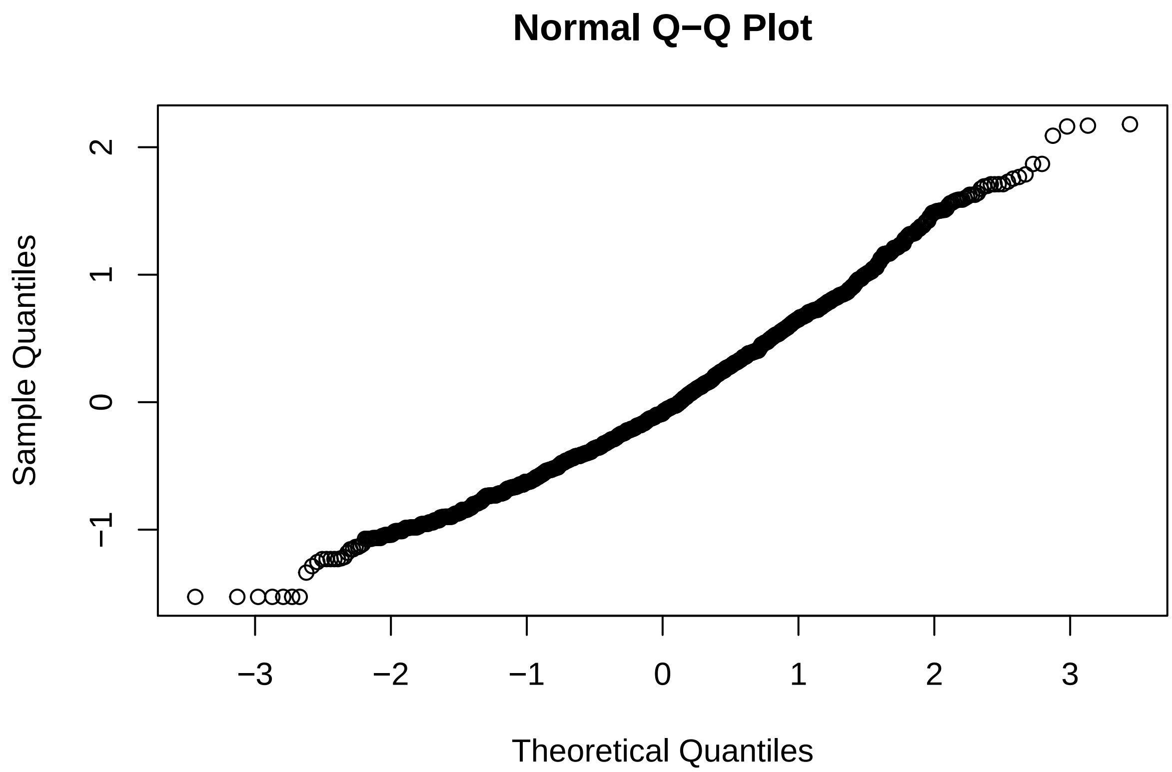

```
plot_resqq(partial_2)
```



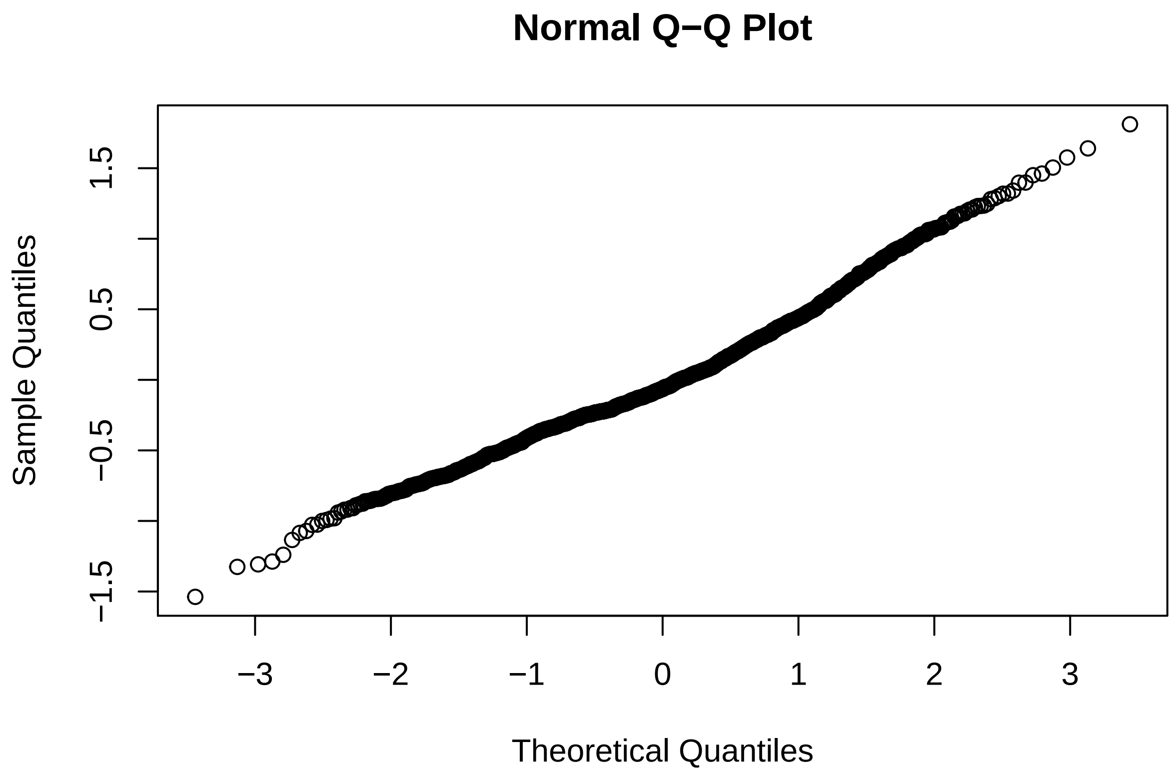

```
plot_resqq(partial_3)
```



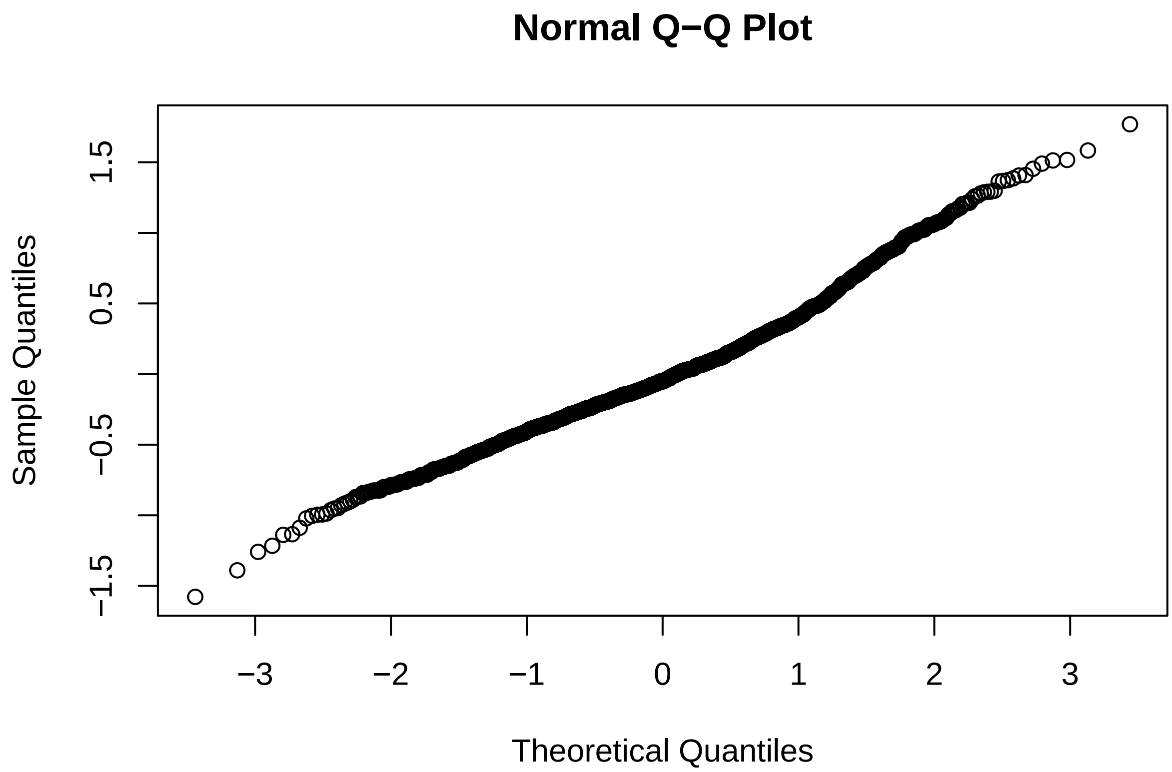

```
qqnorm(residuals(partial))
```




```
qqnorm(residuals(partial_2))
```




```
qqnorm(residuals(partial_3))
```




```r
anova(partial, partial_2, partial_3, test = "Chisq")
```

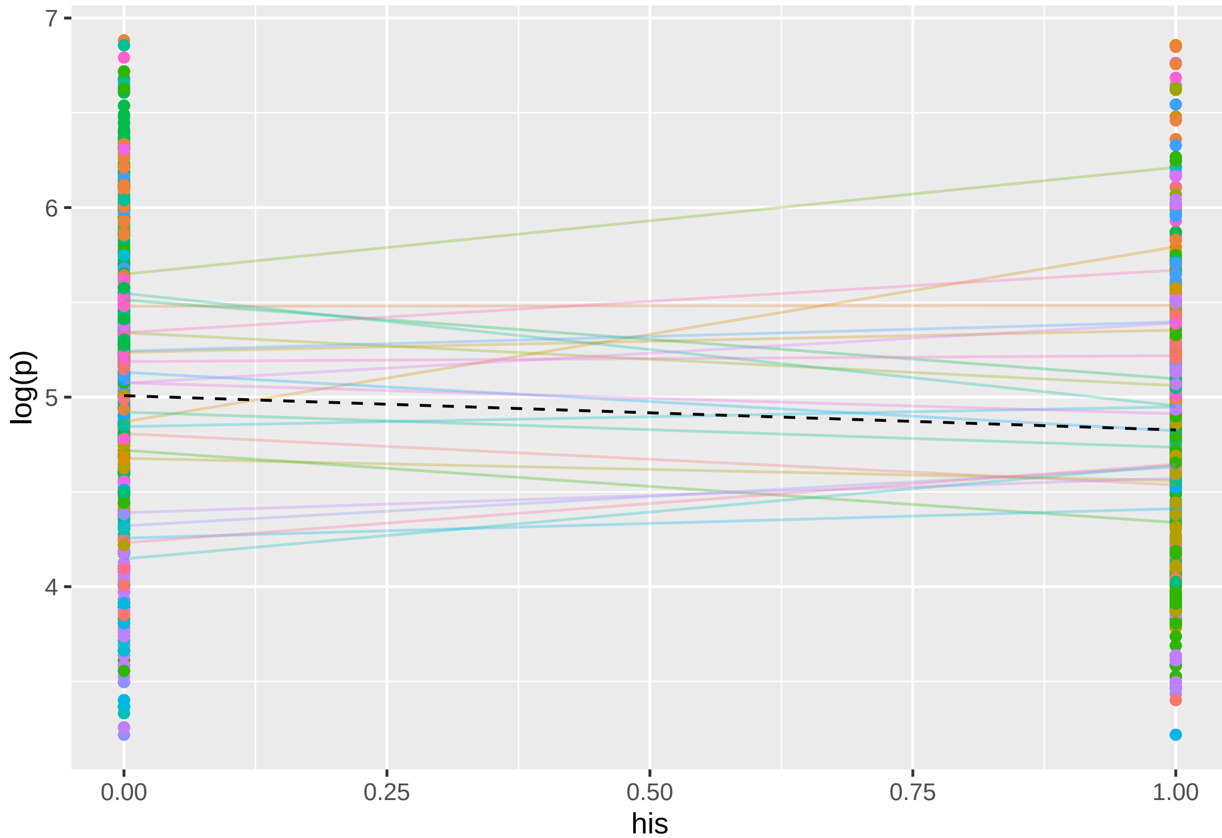
```
## Data: listin
## Models:
## partial: log(price) ~ 1 + (1 | neighbourhood_cleansed)
## partial_2: log(price) ~ 1 + host_response_time + host_response_rate + host_is_superhost + host_has_p
## partial_3: log(price) ~ host_response_time + host_response_rate + host_is_superhost + review_scores_
##           npar    AIC    BIC  logLik deviance    Chisq Df Pr(>Chisq)
## partial      3 3378.8 3395.1 -1686.4   3372.8
## partial_2   16 2319.5 2406.8 -1143.8   2287.5 1085.216 13  < 2.2e-16 ***
## partial_3   19 2285.4 2388.9 -1123.7   2247.4   40.194  3  9.692e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# consider interaction
# library(corrplot)
# library(GGally)
# cor.test(listin$number_of_reviews, listin$review_scores_value)
# ggpairs(listin[, c("price", "review_scores_value")])

# why choose multi-level model
try <- data.frame(his = ifelse(listin$host_is_superhost == "f", 0, 1),
                  p = listin$price,
                  nc = factor(listin$neighbourhood_cleansed))
count(try, nc)
```

```
##                           nc   n
```

```
## 1                    Allston  58
## 2                   Back Bay 105
## 3                Bay Village  37
## 4                Beacon Hill  85
## 5                   Brighton  91
## 6                Charlestown  42
## 7                  Chinatown  17
## 8                 Dorchester 261
## 9                   Downtown 178
## 10               East Boston 100
## 11                    Fenway  54
## 12                 Hyde Park  24
## 13             Jamaica Plain 126
## 14                  Mattapan  25
## 15              Mission Hill  24
## 16                 North End  48
## 17                Roslindale  40
## 18                   Roxbury 166
## 19              South Boston  69
## 20 South Boston Waterfront  11
## 21                 South End 125
## 22                  West End  18
## 23              West Roxbury  17
```

```
ggplot(try, aes(x = his, y = log(p), color = nc))+
  geom_point()+
  stat_summary(fun = "mean", geom = "line", alpha = .3)+
  stat_summary(fun = "mean", geom = "line", lty = 2, aes(group = 1), color = "black")+
  theme(legend.position="none")
```




```
# lis <- listings[, c("host_is_superhost", "neighbourhood_cleansed", "review_scores_value")]
# lis <- lis %>% filter(review_scores_value!= "NA" & neighbourhood_cleansed == "Allston")
# lis$host_is_superhost <- as.factor(lis$host_is_superhost)
# plot(lis$host_is_superhost, lis$review_scores_value)
```