Zillow Project Feature Engineering Part 1

Yue Wen

This RMD file is about performing feature engineering to the zillow data set.

Prepare data

```
setwd("D:/data_camp/zillow_project")

train <- read.csv("train_property.csv",stringsAsFactors = FALSE)

#dealing with missing data first
train$trans_year <- sapply(strsplit(train$transactiondate, '-'), '[[', 1)
train$trans_month <- sapply(strsplit(train$transactiondate, '-'), '[[', 2)
train$trans_day <- sapply(strsplit(train$transactiondate, '-'), '[[', 3)
train$trans_weekday <- weekdays(as.Date(train$transactiondate))
train$trans_DATE <- as.Date(train$transactiondate)


# change feautre name so  that it is easier to match the faeture name to real feature
train <- plyr::rename(train,
                      c("parcelid"="id_parcel",
                        "transactiondate" = "trans_date",
                        "yearbuilt" = "build_year",
                        "basementsqft"="area_base_living",
                        "yardbuildingsqft17"="area_patio",
                        "yardbuildingsqft26"="area_shed",
                        "poolsizesum"="area_pool",
                        "lotsizesquarefeet"="area_lot",
                        "garagetotalsqft"="area_garage",
                        "finishedfloor1squarefeet" = "area_firstfloor_finished",
                        "calculatedfinishedsquarefeet" = "area_total_calc",
                        "finishedsquarefeet6" = "area_base",
                        "finishedsquarefeet12" = "area_live_finished",
                        "finishedsquarefeet13" = "area_liveperi_finished",
                        "finishedsquarefeet15" = "area_total_finished",
                        "finishedsquarefeet50" = "area_unknown",
                        "unitcnt" = "num_unit",
                        "numberofstories" = "num_story",
                        "roomcnt" = "num_room",
                        "bathroomcnt" = "num_bathroom",
                        "bedroomcnt" = "num_bedroom",
                        "calculatedbathnbr" = "num_bathroom_calc",
                        "fullbathcnt" = "num_bath",
                        "threequarterbathnbr" = "num_75_bath",
                        "fireplacecnt" = "num_fireplace",
                        "poolcnt" = "num_pool",
                        "garagecarcnt" = "num_garage",
                        "regionidcounty" = "region_county",
                        "regionidcity" = "region_city",
```

```
                        "regionidzip" = "region_zip",
                        "regionidneighborhood" = "region_neighbor",
                        "taxvaluedollarcnt" = "tax_total",
                        "structuretaxvaluedollarcnt" = "tax_building",
                        "landtaxvaluedollarcnt" = "tax_land",
                        "taxamount" = "tax_property",
                        "assessmentyear" = "tax_year",
                        "taxdelinquencyflag" = "tax_delinquency",
                        "taxdelinquencyyear" = "tax_delinquency_year",
                        "propertyzoningdesc" = "zoning_property",
                        "propertylandusetypeid" = "zoning_landuse",
                        "propertycountylandusecode" = "zoning_landuse_county",
                        "fireplaceflag" = "flag_fireplace",
                        "hashottuborspa" = "flag_tub",
                        "buildingqualitytypeid" = "quality",
                        "buildingclasstypeid" = "framing",
                        "typeconstructiontypeid" = "material",
                        "decktypeid" = "deck",
                        "storytypeid" = "story",
                        "heatingorsystemtypeid" = "heating",
                        "airconditioningtypeid" = "aircon",
                        "architecturalstyletypeid" = "architectural_style",
                        "pooltypeid10" = "flag_spa",
                        "pooltypeid2" = "flag_pool_spa",
                        "pooltypeid7" = "flag_pool_tub",
                        "fips"="county"))


# correct variable type


# numerical variable
variable_numeric = c("area_firstfloor_finished",
                     "area_base", "area_base_living",
                     "area_garage",
                     "area_live_finished",
                     "area_liveperi_finished",
                     "area_lot",
                     "area_patio",
                     "area_pool",
                     "area_shed",
                     "area_total_calc",
                     "area_total_finished",
                     "area_unknown",
                     "tax_building",
                     "tax_land",
                     "tax_property",
                     "tax_total",
                     "latitude",
                     "longitude")
# discrete
variable_discrete = c("num_75_bath",
                      "num_bath",
```

```
                      "num_bathroom",
                      "num_bathroom_calc",
                      "num_bedroom",
                      "num_fireplace",
                      "num_garage",
                      "num_pool",
                      "num_room",
                      "num_story",
                      "num_unit")

variable_binary = c("flag_fireplace",
                    "flag_tub",
                    "flag_spa",
                    "flag_pool_spa",
                    "flag_pool_tub",
                    "tax_delinquency")

# categorical variable
variable_nominal = c("aircon",
                     "architectural_style",
                     "county",
                     "deck",
                     "framing",
                     "heating",
                     "id_parcel",
                     "material",
                     "region_city",
                     "region_county",
                     "region_neighbor",
                     "region_zip",
                     "story",
                     "zoning_landuse",
                     "zoning_landuse_county")
variable_ordinal = c("quality")

# date
variable_date = c("tax_year",
                  "build_year",
                  "tax_delinquency_year",
                  "trans_year",
                  "trans_month",
                  "trans_day",
                  "trans_date",
                  "trans_weekday")

# others
variable_unstruct = c("zoning_property")

# don't understand
variable_unknown = c('censustractandblock',
                     'rawcensustractandblock')
```

```
# Conversion
# - convert some binary to 0, 1
# - convert to date to int
# - convert to numeric to double
# - convert to discrete to int
# - convert to categorical to character

train[train$flag_fireplace == "", "flag_fireplace"] = 0
train[train$flag_fireplace == "true", "flag_fireplace"] = 1
train[train$flag_tub == "", "flag_tub"] = 0
train[train$flag_tub == "true", "flag_tub"] = 1
train[train$tax_delinquency == "", "tax_delinquency"] = 0
train[train$tax_delinquency == "Y", "tax_delinquency"] = 1

# convert to date to int
train[,variable_date] = sapply(train[,variable_date], as.character)

# convert to numeric to double
train[,variable_numeric] = sapply(train[,variable_numeric], as.numeric)

# convert to discrete to int
train[,c(variable_discrete, variable_binary)] = sapply(train[,c(variable_discrete, variable_binary)], as

# convert to categorical to character
train[,c(variable_nominal, variable_ordinal)] = sapply(train[,c(variable_nominal, variable_ordinal)], as

#drop  the redundant value
train <- train[, !(names(train) %in% c("num_bath","num_bathroom_calc","region_county","tax_year"))]
```

Feature Engineering

There are several ways to generate new features.

1:creating new level based on missing value

In particular, I care about the pool/spa equipment because it seems like a sign of whether the apartment is luxury building or not, which might influence the prediction result in future work. Here is a quicke example:

```
train$pool_missing <- ifelse(is.na(train$num_pool),1,0)
with(train, t.test(logerror~pool_missing ==0))
```

```
##
##  Welch Two Sample t-test
##
## data:  logerror by pool_missing == 0
## t = 2.7007, df = 27455, p-value = 0.006924
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.0009951843 0.0062627022
## sample estimates:
## mean in group FALSE  mean in group TRUE
##         0.012176818         0.008547874
```

We can see the missing value does not just randomly occured given the fact that the p-value is so small. We might want to see how the missing infomation is going to be utilized in the future to help improve the performance of the model

Of course, we can take a look at other features and then create new level for those of which missing value does not occur randomly.

Here, we just present the idea and do not repeat the above procedure for every single feature.

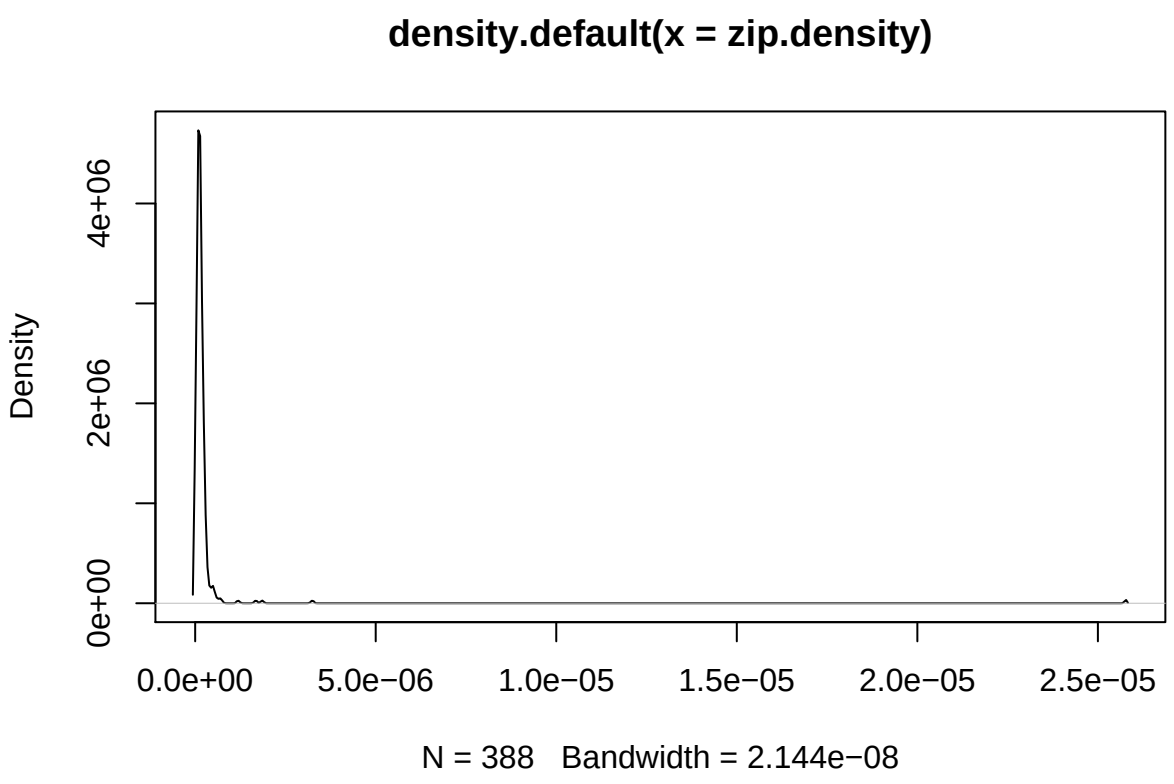
2. craeting new feature based on the current level.

We have already created two new feature(seasonality; bedrooom to bathroom ratio) in the EDA part, which is seasonaily and room/bath ratio. And Mrs. Here, I will present two idea I came up with:

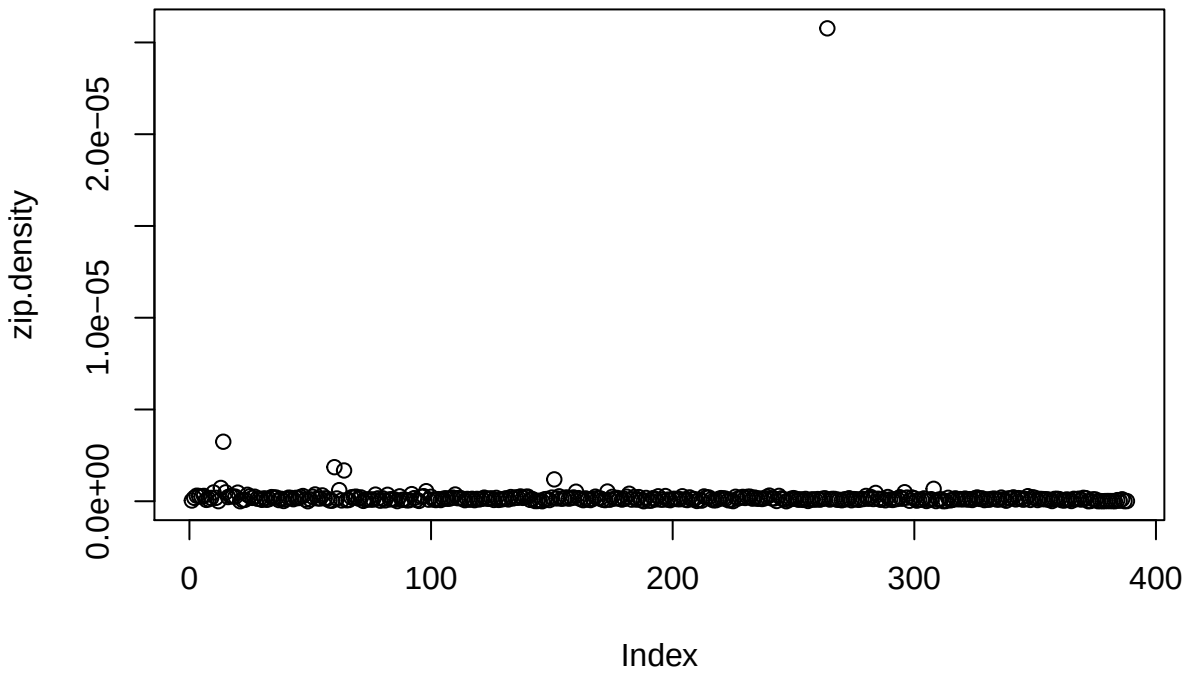
A. Zip density

Here, zip density is identified as the number of houses in each zip per unit area. We might want to care about this feature because the density of building probabily might influence the prediction accuracy. One possible explanatoinfor that can be whether the house is in a popular area or not-very-welcome area.

```
zip.num <- by(train, train$region_zip, function(x) nrow(x))

train$zip.num <- zip.num[train$region_zip]

zip.area  <- by(train, train$region_zip, function(x)
                    {abs((max(x$longitude) - min(x$longitude))*(max(x$latitude) - min(x$latitude)))})

zip.density <- ifelse(zip.area  == 0 , 0 ,zip.num/zip.area)

plot(density(zip.density))
```




```
plot(zip.density)
```



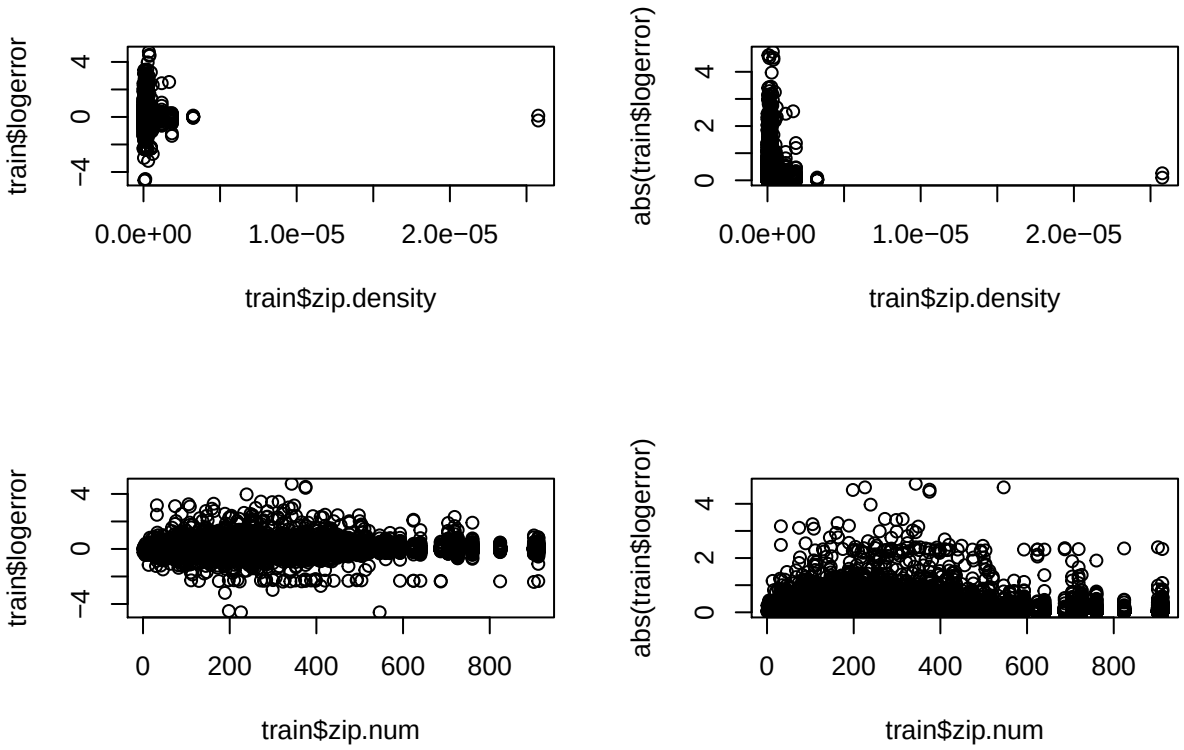

```
train$zip.density <- zip.density[train$region_zip]

with(train,cor(zip.density,logerror,use = "pairwise.complete.obs"))
```

```
## [1] -0.01232909
```

```
with(train,cor(zip.density,abs(logerror),use = "pairwise.complete.obs"))
```

```
## [1] 0.01616237
```

```
par(mfrow = c(2,2))
plot(train$zip.density, train$logerror)
plot(train$zip.density, abs(train$logerror))
plot(train$zip.num, train$logerror)
plot(train$zip.num, abs(train$logerror))
```



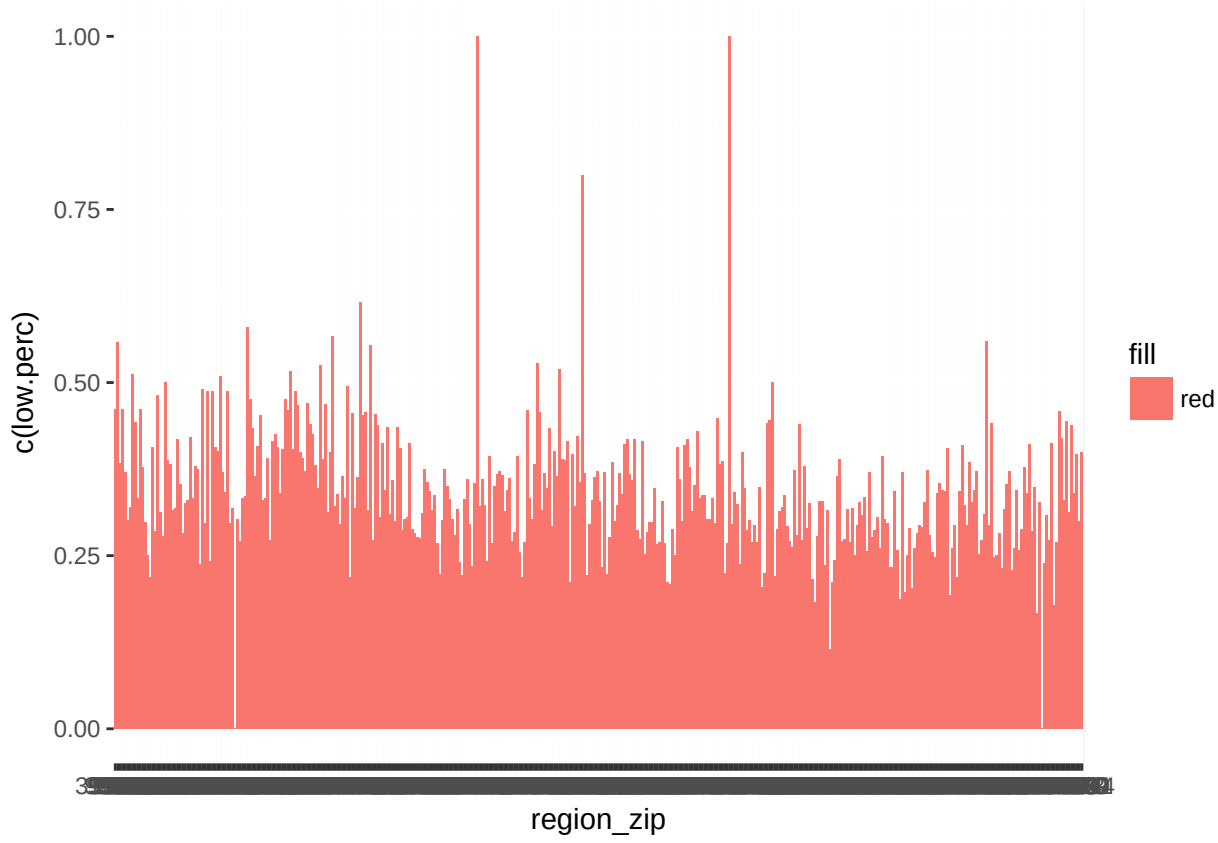
We can see there is a more clear trend between logerror and zip density.

B. Percentage of over-estimate, under-estimate, well-estimcate

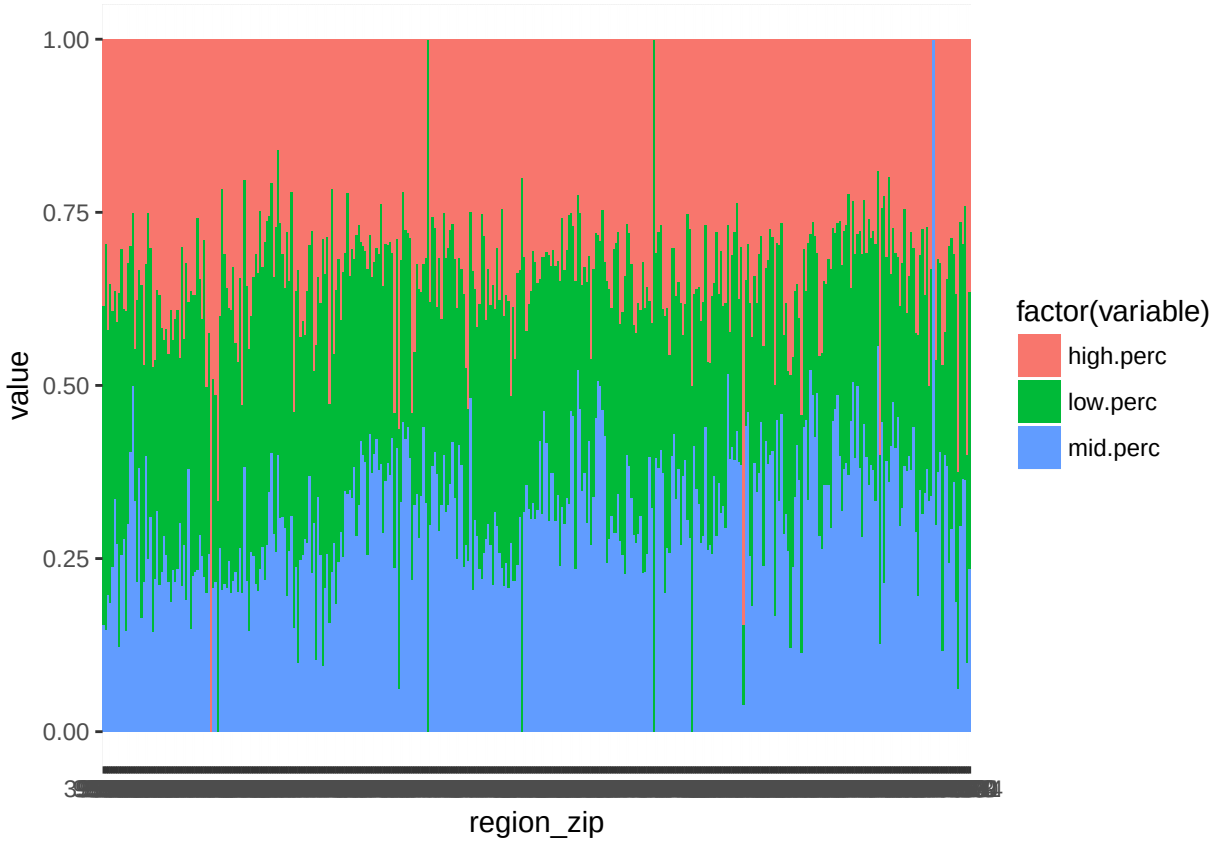
We want to take a look at what is the percentage of over-estimate, under-estimate, well-estimcate in each zip code.

```
library(ggplot2)
# up bound for houses that are over-estimated
up.bound <- quantile(train$logerror, 2/3)
# low bound for houses that are under-estimated
low.bound <- quantile(train$logerror, 1/3)

train$pred.level <- ifelse(train$logerror<low.bound,"low",
                           ifelse(train$logerror <= up.bound,"mid","high"))

#calculate percentage of three  types in each zip code
zip.high.per <-  by(train,train$region_zip, function(x)
                                    {sum(x$pred.level == "high")/nrow(x)})

zip.mid.per <-  by(train,train$region_zip, function(x)
{sum(x$pred.level == "mid")/nrow(x)})

zip.low.per <-  by(train,train$region_zip, function(x)
{sum(x$pred.level == "low")/nrow(x)})

library (plyr)
```

```
df <- ldply(zip.high.per, data.frame)

names(df) <- c("region_zip","high.perc")
df$low.perc <- unlist(zip.low.per)
df$mid.perc <- unlist(zip.mid.per)

ggplot(data=df, aes(x=region_zip, y=c(low.perc), fill = "red")) +
  geom_bar(stat="identity")
```




```
library(reshape2)
agg_df <- melt(df,id.vars="region_zip")

ggplot(data=agg_df, aes(x=region_zip, y=value,fill=factor(variable))) +
  geom_bar(stat="identity")
```



we can see there are actually differences in differnt zips, for some zip code area, it is more likely to be over-estimated or low-estimated, we can use this information to give each zip some features.