Addressing missing CoreLogic bedroom & bathroom data: Scraping method using Google and Bing

Neil Kattampallil, Steve Zhou, Leonel SIWE,

Social and Decision Analytics Division, Biocomplexity Institute,

Xin Wang, San Diego State University

June 25, 2024

Data needed for Broadband Infrastructure Program (BIP) Analysis

For our analysis, we consider the following property information: location, sale information, lot size, building size, age, number of bedrooms, and number of bathrooms.

- **Property location** is described by parcel-level centroid geographic coordinates
- **Sale information** includes the latest sale amount in dollars and the date when the sale contract was signed, dropping any sales prior to 2005
- The **size of a residence** is represented by the size of the building in sq. ft. and the size of the living area in sq. ft. The lot size was also measured in sq. ft.
- **Construction year** of the original building or the adjusted year when the building was last improved or remodeled was used to calculate the age of the property
- Used Fannie Mae and Freddie Mac Uniform Appraisal Dataset Specification to calculate the number of baths

Filling CoreLogic Missing Information

- In some counties, the number of bedrooms and bathrooms was completely or partially missing from the CoreLogic data.
- To address this problem, we obtained this information from Zillow real estate records when it was available.
- To address potential mismatches between these data sources, we kept Zillow values for the number of bedrooms and bathrooms only when the Zillow and CoreLogic living area in sq. ft. matched, within a 5% tolerance.

Amount of Missing Data

- Number of RUS_IDs we are analyzing in this subset: 96
- Number of CoreLogic House Observations in the RUS_ID areas: 3,122,391
- Number of CoreLogic observations after deduplication: 2,942,797
  - Number of Observations with missing bedrooms: 1,129,527 (38.4%)
  - Number of Observations with missing bathrooms: 299,292 (10.2%)
  - Number of Observations with missing bedrooms OR bathrooms: 1,176,494 (40.0%)
- This means nearly half the house observations in these rural project areas have significant missing data in the bedroom and bathroom columns.

Initial Thought Process to Scrape Bedroom and Bathrooom Data to Fill in CoreLogic

Proposed process

1) Iterate through the list of addresses for which there are missing bathrooms/bedrooms

2) Put that into Google and generate a set of search results. (Later, changed to Bing)

3) Get the first link in that page that mentions Zillow, which should be the Zillow link for the house

4) Visit the Zillow page for that house and use rvest to get the number of bedrooms/bathrooms

Switching from Google to Bing

Steve Zhou, one of the DSPG students on this project, realized that scraping data using Bing instead of Google has a much better hit rate and doesn't get blocked or rate-limited.

Scraping Timeline and Process

- Google Scraping occurred during April and May 2023.
- Bing Scraping occurred during July 2023.
- The scraped data are reflective of data available in Summer 2023 and not necessarily of the house characteristics during the time of the house sale.
- We scraped the home square footage data to compare Zillow results and Corelogic data. If these values were not within 5% tolerance, we would not accept the scraped data values.
    - The reasoning was that any change in bedrooms or bathrooms would be a significant enough renovation where the square footage would have been changed significantly.

Validation

- We manually checked addresses in our dataset for missing bedrooms and bathrooms.
- Consistently, we found that Zillow.com was a reliable source. It provided comprehensive data on square footage, number of bathrooms, number of bedrooms, and additional information such as the year built.
- Zillow is also incentivized to provide accurate information to potential home buyers
- We wrote code to automate this process of searching

Missing Data in BIP Regions filled by Scraping

- Number of Observations inside BIP areas: 139,196
  - 72,871 (52.4%) with missing #bedrooms or bathrooms
- Number of Observations within 20 miles of BIP: 2,803,601
  - 1,103,623 (39.4%) with missing #bedrooms or bathrooms
- In the Cleaned Dataset for Analysis, there were:
  - 61 RUSIDs
  - 1,172,287 total observations,
    - Of these, 170,673 (14.6%) had missing bedrooms/bathrooms filled in by scraping
  - 49,405 inside BIP
    - of these, 4,565 (9.2%) had missing data filled in by scraping
  - 1,122,882 within 20 miles of BIP
    - of these, 166,108 (14.8%) had missing data filled in by scraping

Matched Dataset used for Regression

Dataset contained:

- 61 RUSIDs
- 98,740 total observations
  - of these, 9,632 (9.8%) had missing data filled in by scraping
- 49,370 inside BIP
  - of these, 4,558 (9.2%) had missing data filled in by scraping
- 49,370 matched within 10 miles of BIP
  - of these, 5,074 (10.3%) had missing data filled in by scraping

Appendix: R code Description

Code in Notes Section of this slide

-The R code takes the pieces of the house address from each row of the dataset, combines it into an address string, and searches for this string on google.com

The HTML text that Google sends back is then analyzed for keywords such as 'square feet', 'sq ft', 'square ft', 'bed','beds', 'baths', etc., and then a regex is used to extract numerical values before those keywords.

- This is to capture patterns in house descriptions (2 bed, 3 bath)
- The results of the regex, if they exist, are saved as scraped_beds, scraped_baths, scraped_sqft and 'actual_text'. The last column is used to hold the raw text data t o which we applied the regex.

Appendix: R Code

--- title: "R Notebook"

output: html_notebook ---

ver 01: use Zillow to find missing info of properties in BK dataset.

Ver 02 update: add realtor and fastpeoplesearch as additional sources for properties missing info on Zillow.

Ver 03 update: use Bing instead of Google for Bing has much less restriction on number of queries. While google gives a HTTP 429 error for too frequent requests after approximately 50 queries, we have not hit such limitation on Bing in our sample.

```{r}

library(readr)

library(dplyr)

library(data.table)

library(stringr)

library(tidyr) library(scales)

library(rvest) ```

```{r} # load the data ----------

housings <- read.csv('housing_BIP_022227_deduplicated_missing_bedbath.csv.xz') ```

```{r}
```

Appendix: R Code

final_id_list_1<-c('OK1111-A40', ‘GA1106-A40', ‘IA1111-A40’, 'CO1107-A40’, 'AR1102-AG39’, 'TN1105-A40') f

final_id_list_2<-c('AZ1111-B39’, ‘DC1101-A40’, VA1112-A40’, 'WI1139-A39’, 'ID1107-A40’, 'KY1107-B40’)

final_id_list_3<-c('TN1104-B40', 'AR1102-C39', 'TN1106-A40', 'TN1102-A40', 'VA1109-A39', 'WI1126-A39', 'AL1105-B39’)

active_set <- final_id_list_1

...

...{r} s_not_valid = function(str){

helper function to deal with character(0) or logical(0) issue if(identical(str, character(0))){ return(TRUE) } if(is.na(str)){ return(TRUE)

}

return(FALSE) }

}

Appendix: R Code

```
search_result_select = function(html_page, domain_keyword, address, search_engine){
if (search_engine == "google"){
divs = rvest::html_nodes(html_page, "div") result_text = NA for (i in divs){
subdiv <- rvest::html_nodes(i, 'div’)
for (j in subdiv){
if (grepl(domain_keyword, html_text(j), fixed = TRUE)){ result_text <- html_text(j) } if (!is.na(result_text)){ break
}
 } if(!is.na(result_text)){ break
}
 }
 }
```

Appendix: R Code

```
if (search_engine == "bing"){
lis = rvest::html_nodes(page, "li") result_text = NA
for (i in lis){
 if (grepl(domain_keyword, html_text(i), fixed = TRUE)){ result_text <- html_text(i)
}
if (!is.na(result_text)){ break
 } } }
```

Appendix: R Code

```
if(!is.na(result_text)){
street_number_original = regmatches(address, regexpr("\\b\\d+\\b", address))
street_number_matched = street_number_original
 if(domain_keyword == "realtor" | domain_keyword == "zillow"){ street_number_matched =
regmatches(result_text, regexpr("\\b\\d+\\b", result_text))
}
if(domain_keyword == "fastpeoplesearch"){ street_number_matched = gsub( "\\D", "",
regmatches(result_text, regexpr("\\.\\.\\.\\D*\\d+\\b", result_text)) )
 }
 if (is_not_valid(street_number_matched)){ street_number_matched = "" } if (street_number_original
!= street_number_matched){ result_text = NA
}
result_text
}
 } ``
```

Appendix: R Code

```r
result_text_analyze = function(text){ print(text) if (identical(text, character(0))) {
return(data.frame("bedrooms" = NA, "bathrooms" = NA, "sqft" = NA, "text" = NA)) }

result_text = str_to_lower(text)
sq_feet <- str_extract(result_text, "(\\d+,)?\\d+\\.?\\d*?(?=\\s(square feet))")

if(is_not_valid(sq_feet)){ sq_feet <- str_extract(result_text, "(\\d+,)?\\d+\\.?\\d*?(?=\\s(sqft))") }
if(is_not_valid(sq_feet)){ sq_feet <- str_extract(result_text, "(\\d+,)?\\d+\\.?\\d*?(?=\\s(square foot))") }

 if(is_not_valid(sq_feet)){ sq_feet <- str_extract(result_text, "(\\d+,)?\\d+\\.?\\d*?(?=\\s(sq ft))")

 }

 if(is_not_valid(sq_feet)){ sq_feet <- str_extract(result_text, "(\\d+,)?\\d+\\.?\\d*?(?=\\s(sq\\. ft\\.))") }

bed_count <- str_extract(result_text, "\\d+\\.?\\d*?(?=\\s(bed))") if(is_not_valid(bed_count)){
bed_count <- str_extract(result_text, "\\d+\\.?\\d*?(?=\\s(-bed))")

}

bath_count<- str_extract(result_text, "\\d+\\.?\\d*?(?=\\s(bath))") if(is_not_valid(bath_count)){
bath_count<- str_extract(result_text, "\\d+\\.?\\d*?(?=\\s(-bath))")

 }

data.frame("bedrooms" = bed_count, "bathrooms" = bath_count, "sqft" = sq_feet, "text" = text)

} ```
```

Appendix: R Code

```r
count_row_valid_entries = function(row) {
 # helper function to deal with fake zeroes in zillow result
non_na = row[!is.na(row)]
zeroes = sum(non_na == "0")
zerodot = sum(non_na == "0.0")
 return(sum(!is.na(row)) - zeroes - zerodot)
}
```

Appendix: R Code

```r
source_keywords = c("zillow","realtor", "fastpeoplesearch")
search_engines = c("google", "bing")
start = i
engine = "bing"
#subscr_addr[,c('bedrooms', 'bathrooms', 'sqft', 'text', 'source')] for (x in active_set){
temporary<-housings[housings$rusid == x,] for(i in start : nrow(temporary)){
index = row.names(temporary)[i]
print(index)
print(temporary[i,'address’])
flag429 = FALSE # HTTP error 429 is for too frequent requests
tryCatch({
#Sys.sleep(1) # if using Google, change sleep to be at least 5 seconds
address <- temporary[index, 'address’]
url <- URLencode(paste0("https://www.", engine, ".com/search?q=",address)) page <-
xml2::read_html(url) #
```

Appendix: R Code

```
We will loop through source keywords and choose the one with the most info
best_row = NA
best_source = NA
for (source in source_keywords){ result_text = search_result_select(page, source, address, engine) result_row =
result_text_analyze(result_text) print(result_row) if (count_row_valid_entries(result_row) >
count_row_valid_entries(best_row)){ best_row = result_row best_source = source } if (sum(is.na(result_row)) ==
0){ break
 }
}
If (!is.null(nrow(best_row))){ # filter out empty matches that are just NA
best_row['source'] = best_source
row.names(best_row) = c(index)
print("Best Row Is")
print(best_row)
 #active_set[index, c('bedrooms', 'bathrooms', 'sqft', 'text', 'source')] = best_row
temporary[index, 'scraped_beds']<-best_row['bedrooms’]
temporary[index, 'scraped_baths']<-best_row['bathrooms’]
temporary[index, 'scraped_sq_ft']<-best_row['sqft’]
 temporary[index, 'actual_text']<-best_row['text’]
}
}
```

Appendix: R Code

```
error = function(e){ print(e)
 if(grepl("HTTP error 429", e, fixed = TRUE)
){ flag429 <<- TRUE # global assignment print("Too frequent requests, change IP or retry later")
 }
}) if (flag429) { break }
}
write.csv(temporary, paste0(x,"_1",".csv"), row.names=FALSE)
}
```

```{r}
# save backup copies write.csv(active_set, "subscriber_scrapping_working.csv")
# write.csv(active_set, "buffer_scrapped_working3.csv") ```
```

Appendix: R Code

```r
# keep row names after subsetting, replace any row in original data that appears in
# the active set with the processed row in active set row.names(active_set) = row.names(buffer_addr)[is.na(buffer_addr$source)]
buffer_addr[match(row.names(active_set), row.names(buffer_addr)), ] = active_set
write.csv(buffer_addr, "buffer_scrapped.csv")
```

```r
print(sum(is.na(buffer_addr$sqft)))
print((sum(is.na(buffer_addr$sqft))) / nrow(buffer_addr))
print(sum(is.na(buffer_addr$bedrooms)))
print(sum(is.na(buffer_addr$bathrooms)))
```