Collecting, Storing and Retrieving Data

DA:5020 -> Spring 2017

Faculty Advisor: Prof. Kathleen Durant

TA: (1) Japan Mehta

(2) Jianchao Yang

Name: Sagar Ghiya

M.S in Operations Research

Index

Accknowledgement

I would like to thank **Prof. Kathleen Durant** for teaching us this wonderful course. At the end of this course, I have become quite confident and proficient with R Programming. This course has also helped me earn calls for coop interview. I would also like to thank TA’s of this course **Jianchao Yang** and **Japan Mehta** for being there and solving doubts whenever needed.

Problem Description

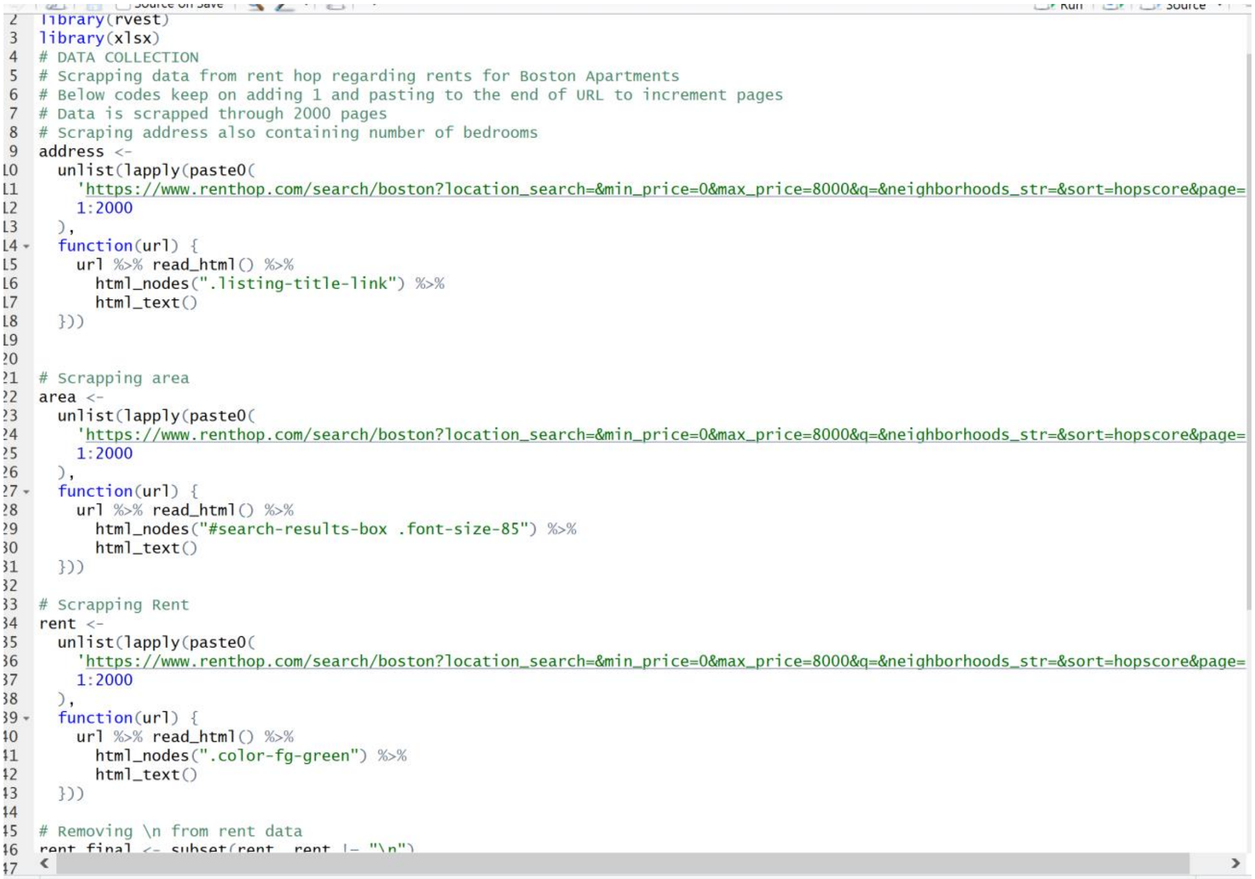
- Finding apartments for rent and getting a good deal is a must for a person looking for a rental apartment.
- Especially for university students coming from other countries as they must optimize their search.
- So my goal in this project is to analyze Boston Apartments rental data and finding neighbourhoods with low rent as per bedroom requirements.
- Data is scrapped from RentHop, one of the most common websites for searching rental apartments in Boston.
- Data scrapped includes address, area, rent and number of bedrooms and bathrooms.
- These are the main factors taken into consideration while looking for apartments.

Project Proposal

- Scrapping data from RentHop.com
- Cleaning it and getting it into desired format.
- Storing data into SQL database by converting it into 3NF form.
- Retrieving the required data.
- Analyzing it by creating some basic visualizations.

Data Collection

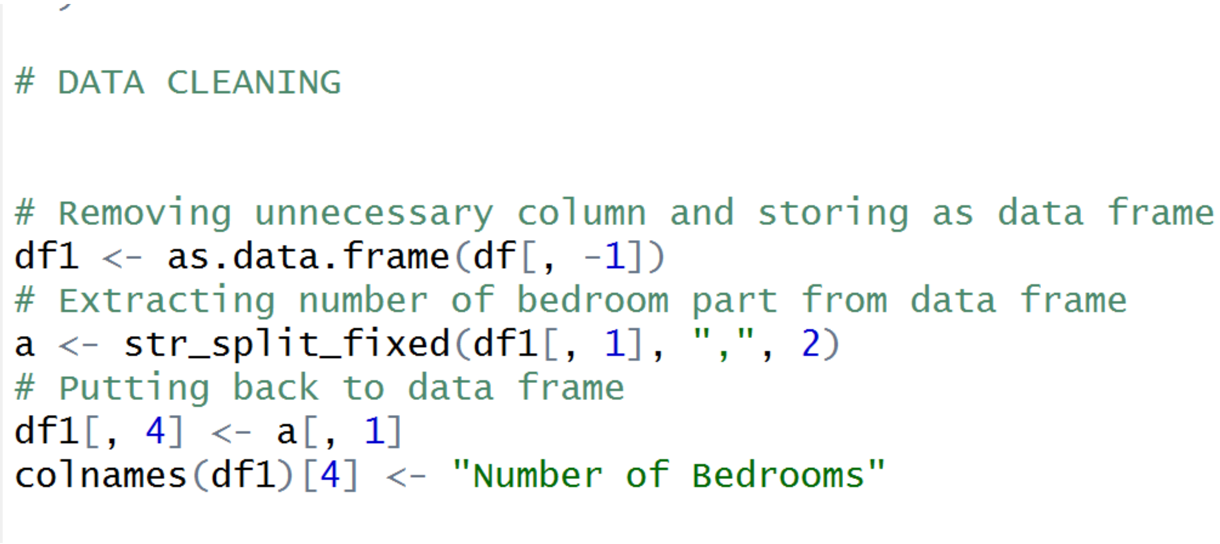
- 2000 pages of data is scrapped from RentHop.com.
- Package used is Rvest and xlsx.
- Data source i.e RentHop website keeps on changing.
- This can create problem and refute analysis such as calculating average.
- So as a better option, data needs to be fixed.
- This is done using **write.xlsx** command which store data into excel file in working directory.
- Data can then be read in another R Script to do further operations.



```r
library(rvest)
library(xlsx)
# DATA COLLECTION
# Scrapping data from rent hop regarding rents for Boston Apartments
# Below codes keep on adding 1 and pasting to the end of URL to increment pages
# Data is scrapped through 2000 pages
# Scraping address also containing number of bedrooms
address <-
  unlist(lapply(paste0(
    'https://www.renthop.com/search/boston?location_search=&min_price=0&max_price=8000&q=&neighborhoods_str=&sort=hopscore&page=
    1:2000
  ),
  function(url) {
    url %>% read_html() %>%
      html_nodes(".listing-title-link") %>%
      html_text()
  }))


# Scrapping area
area <-
  unlist(lapply(paste0(
    'https://www.renthop.com/search/boston?location_search=&min_price=0&max_price=8000&q=&neighborhoods_str=&sort=hopscore&page=
    1:2000
  ),
  function(url) {
    url %>% read_html() %>%
      html_nodes("#search-results-box .font-size-85") %>%
      html_text()
  }))

# Scrapping Rent
rent <-
  unlist(lapply(paste0(
    'https://www.renthop.com/search/boston?location_search=&min_price=0&max_price=8000&q=&neighborhoods_str=&sort=hopscore&page=
    1:2000
  ),
  function(url) {
    url %>% read_html() %>%
      html_nodes(".color-fg-green") %>%
      html_text()
  }))

# Removing \n from rent data
rent_final <- subset(rent, rent != "\n")
```

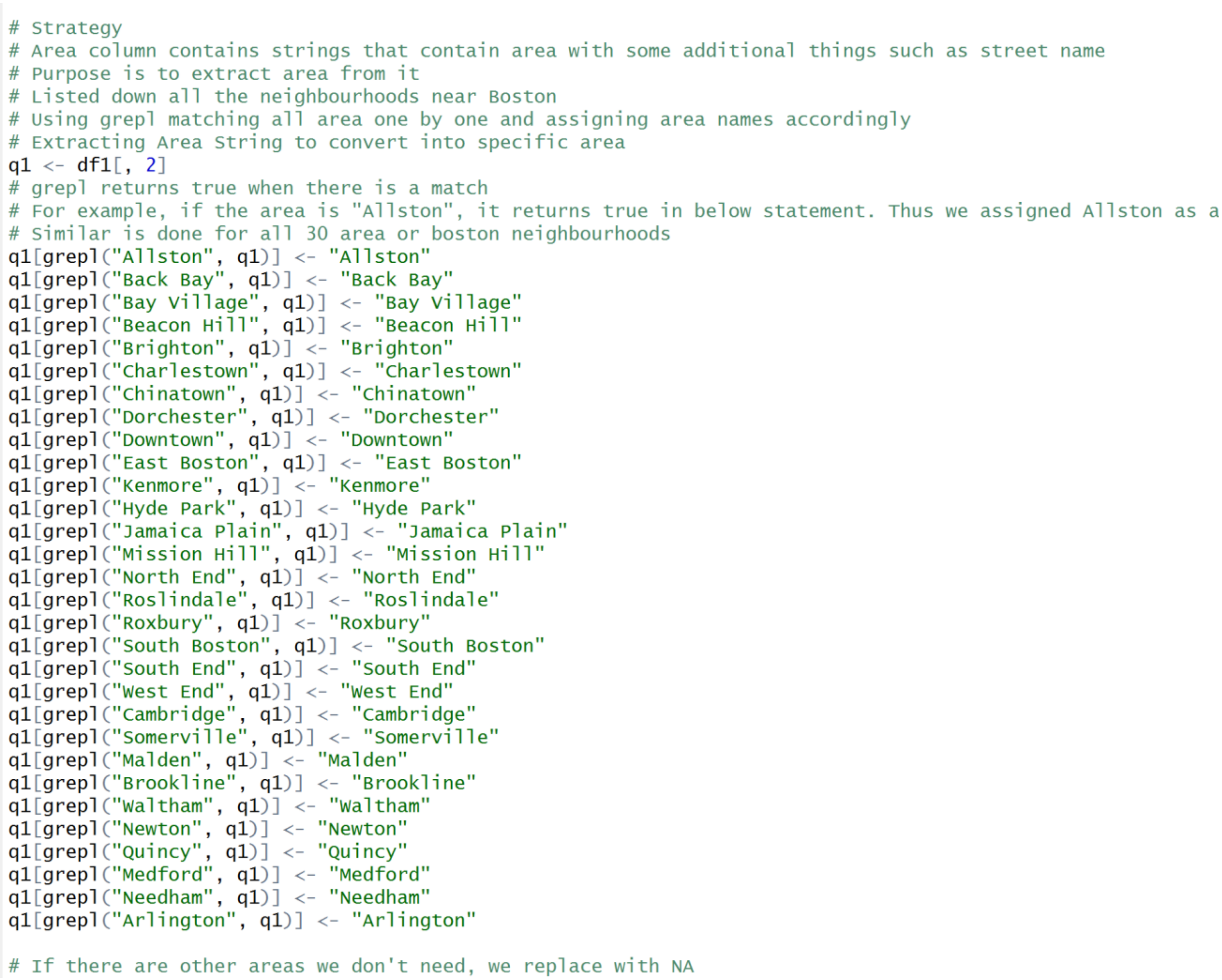
Data Cleaning

- Number of bedrooms were connected with address.
- String splitting done to extract number of bedrooms and store them as another column in data frame.



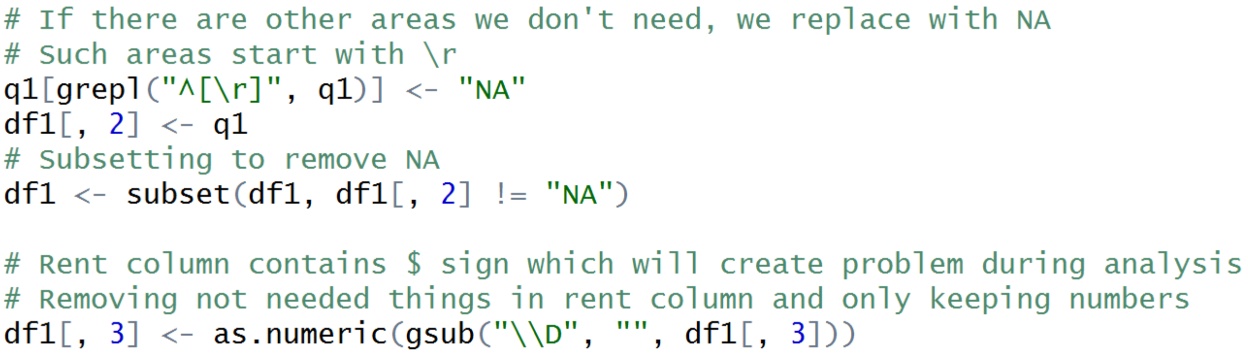
```
# DATA CLEANING


# Removing unnecessary column and storing as data frame
df1 <- as.data.frame(df[, -1])
# Extracting number of bedroom part from data frame
a <- str_split_fixed(df1[, 1], ",", 2)
# Putting back to data frame
df1[, 4] <- a[, 1]
colnames(df1)[4] <- "Number of Bedrooms"
```

- Area column contained extra things such as street name.
- Our purpose is just to have area names of Boston and neighbourhoods.
- So listed down all the area names.
- Used grepl function to match area column with particular area name.
- If true, area name is assigned thus removing unwanted things from the string.



```
# Strategy
# Area column contains strings that contain area with some additional things such as street name
# Purpose is to extract area from it
# Listed down all the neighbourhoods near Boston
# Using grepl matching all area one by one and assigning area names accordingly
# Extracting Area String to convert into specific area
q1 <- df1[, 2]
# grepl returns true when there is a match
# For example, if the area is "Allston", it returns true in below statement. Thus we assigned Allston as a
# Similar is done for all 30 area or boston neighbourhoods
q1[grepl("Allston", q1)] <- "Allston"
q1[grepl("Back Bay", q1)] <- "Back Bay"
q1[grepl("Bay Village", q1)] <- "Bay Village"
q1[grepl("Beacon Hill", q1)] <- "Beacon Hill"
q1[grepl("Brighton", q1)] <- "Brighton"
q1[grepl("Charlestown", q1)] <- "Charlestown"
q1[grepl("Chinatown", q1)] <- "Chinatown"
q1[grepl("Dorchester", q1)] <- "Dorchester"
q1[grepl("Downtown", q1)] <- "Downtown"
q1[grepl("East Boston", q1)] <- "East Boston"
q1[grepl("Kenmore", q1)] <- "Kenmore"
q1[grepl("Hyde Park", q1)] <- "Hyde Park"
q1[grepl("Jamaica Plain", q1)] <- "Jamaica Plain"
q1[grepl("Mission Hill", q1)] <- "Mission Hill"
q1[grepl("North End", q1)] <- "North End"
q1[grepl("Roslindale", q1)] <- "Roslindale"
q1[grepl("Roxbury", q1)] <- "Roxbury"
q1[grepl("South Boston", q1)] <- "South Boston"
q1[grepl("South End", q1)] <- "South End"
q1[grepl("West End", q1)] <- "West End"
q1[grepl("Cambridge", q1)] <- "Cambridge"
q1[grepl("Somerville", q1)] <- "Somerville"
q1[grepl("Malden", q1)] <- "Malden"
q1[grepl("Brookline", q1)] <- "Brookline"
q1[grepl("Waltham", q1)] <- "Waltham"
q1[grepl("Newton", q1)] <- "Newton"
q1[grepl("Quincy", q1)] <- "Quincy"
q1[grepl("Medford", q1)] <- "Medford"
q1[grepl("Needham", q1)] <- "Needham"
q1[grepl("Arlington", q1)] <- "Arlington"

# If there are other areas we don't need, we replace with NA
```

- Few basic cleaning work such as removing \r and removing $ sign from rent.

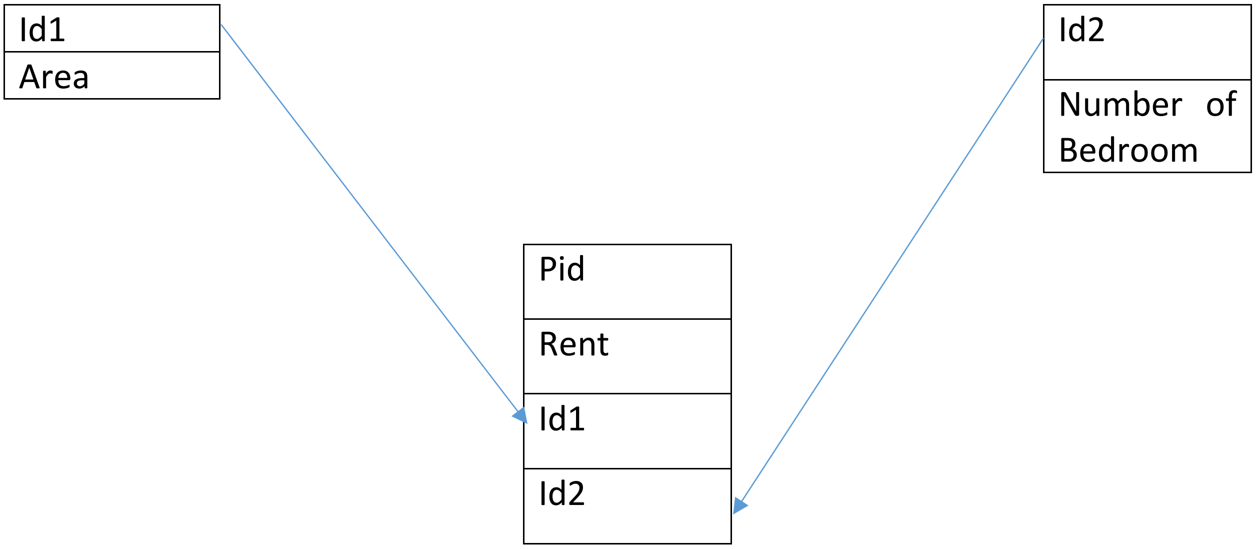


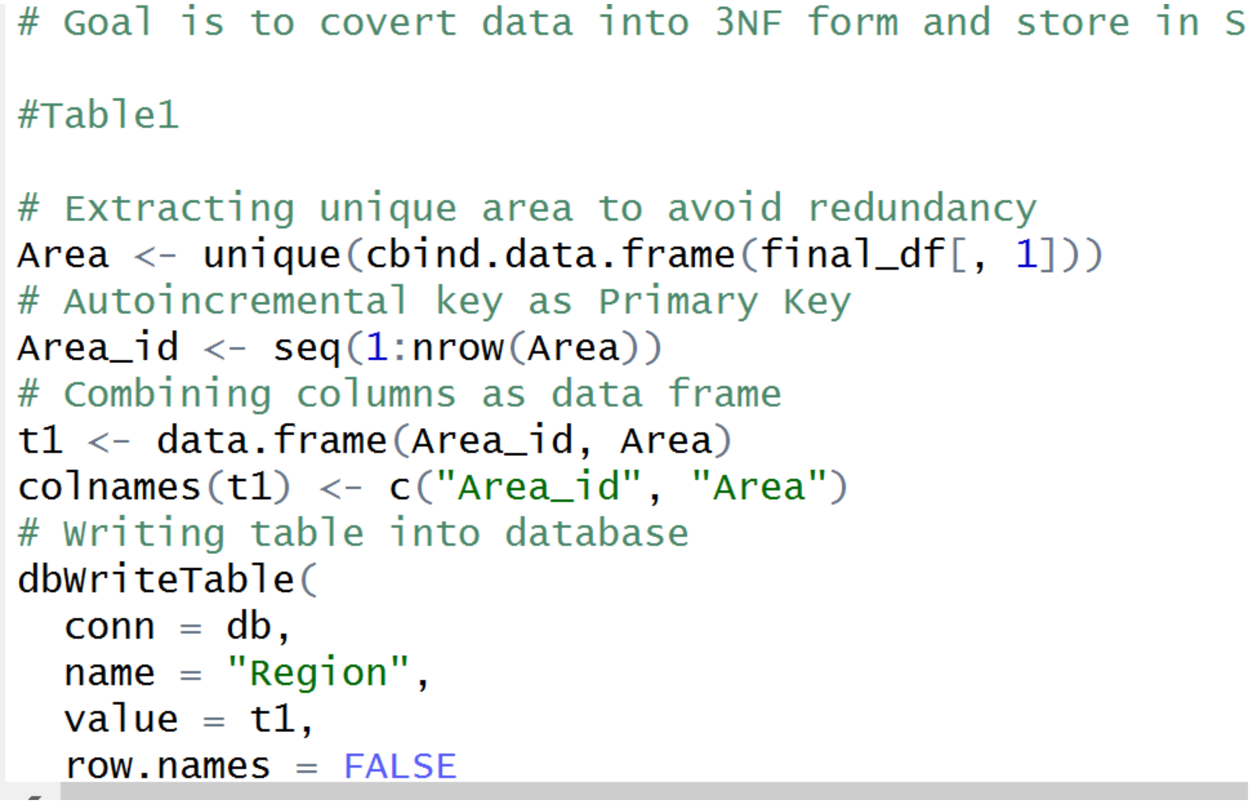
```
# If there are other areas we don't need, we replace with NA
# Such areas start with \r
q1[grepl("^[\r]", q1)] <- "NA"
df1[, 2] <- q1
# Subsetting to remove NA
df1 <- subset(df1, df1[, 2] != "NA")

# Rent column contains $ sign which will create problem during analysis
# Removing not needed things in rent column and only keeping numbers
df1[, 3] <- as.numeric(gsub("\\D", "", df1[, 3]))
```

Data Storage

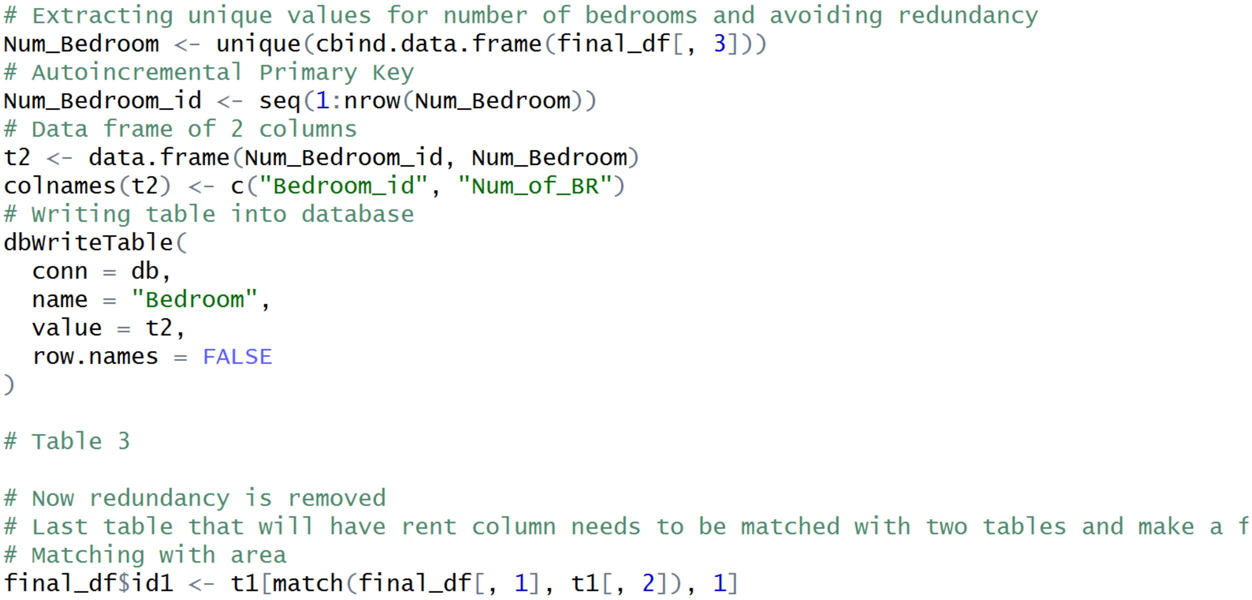
- Data contains lots of redundancy with areas and number of bedrooms repeated for postings.
- So best option is to store in SQL database by converting into 3NF form.
- First table contains unique area names and auto incremental id1 as primary key.
- Second table contains unique number of bedrooms and auto incremental id2 as primary key.
- Third and final table stores rent and id as primary key. It also contains id1 and id2 as foreign keys of other tables.

Database Schema





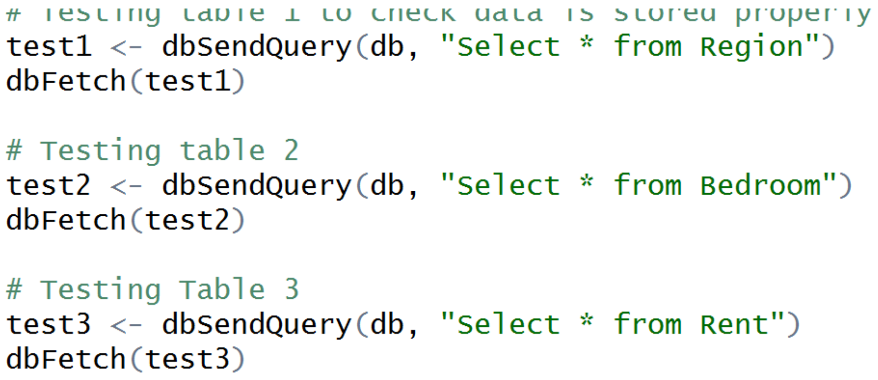
```
# Goal is to covert data into 3NF form and store in S

#Table1

# Extracting unique area to avoid redundancy
Area <- unique(cbind.data.frame(final_df[, 1]))
# Autoincremental key as Primary Key
Area_id <- seq(1:nrow(Area))
# Combining columns as data frame
t1 <- data.frame(Area_id, Area)
colnames(t1) <- c("Area_id", "Area")
# Writing table into database
dbWriteTable(
  conn = db,
  name = "Region",
  value = t1,
  row.names = FALSE
```



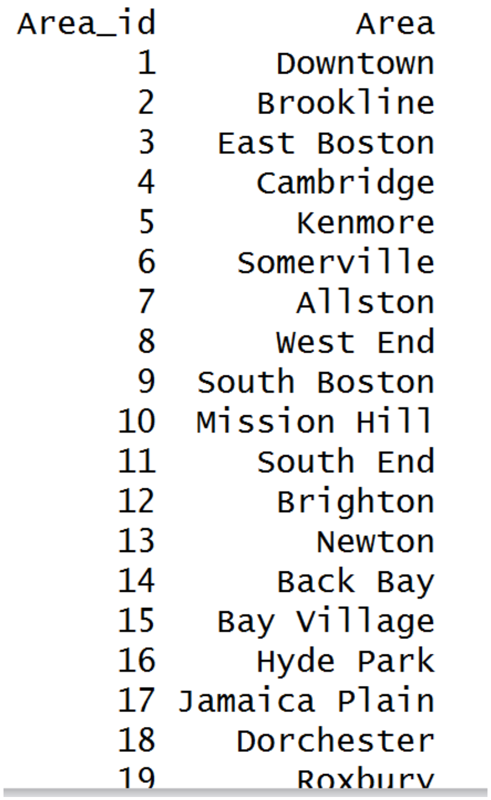
```
# Extracting unique values for number of bedrooms and avoiding redundancy
Num_Bedroom <- unique(cbind.data.frame(final_df[, 3]))
# Autoincremental Primary Key
Num_Bedroom_id <- seq(1:nrow(Num_Bedroom))
# Data frame of 2 columns
t2 <- data.frame(Num_Bedroom_id, Num_Bedroom)
colnames(t2) <- c("Bedroom_id", "Num_of_BR")
# Writing table into database
dbWriteTable(
  conn = db,
  name = "Bedroom",
  value = t2,
  row.names = FALSE
)

# Table 3

# Now redundancy is removed
# Last table that will have rent column needs to be matched with two tables and make a f
# Matching with area
final_df$id1 <- t1[match(final_df[, 1], t1[, 2]), 1]
```

Data Validation

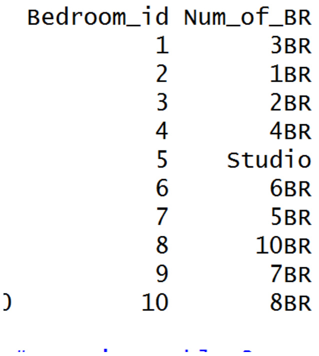
- Select * query to check whether data is stored properly in database.



```
# Testing table 1 to check data is stored properly
test1 <- dbSendQuery(db, "Select * from Region")
dbFetch(test1)

# Testing table 2
test2 <- dbSendQuery(db, "Select * from Bedroom")
dbFetch(test2)

# Testing Table 3
test3 <- dbSendQuery(db, "Select * from Rent")
dbFetch(test3)
```



```
Area_id           Area
      1       Downtown
      2      Brookline
      3    East Boston
      4      Cambridge
      5        Kenmore
      6     Somerville
      7        Allston
      8       West End
      9   South Boston
     10   Mission Hill
     11      South End
     12       Brighton
     13         Newton
     14       Back Bay
     15    Bay Village
     16      Hyde Park
     17  Jamaica Plain
     18     Dorchester
     19        Roxbury
```

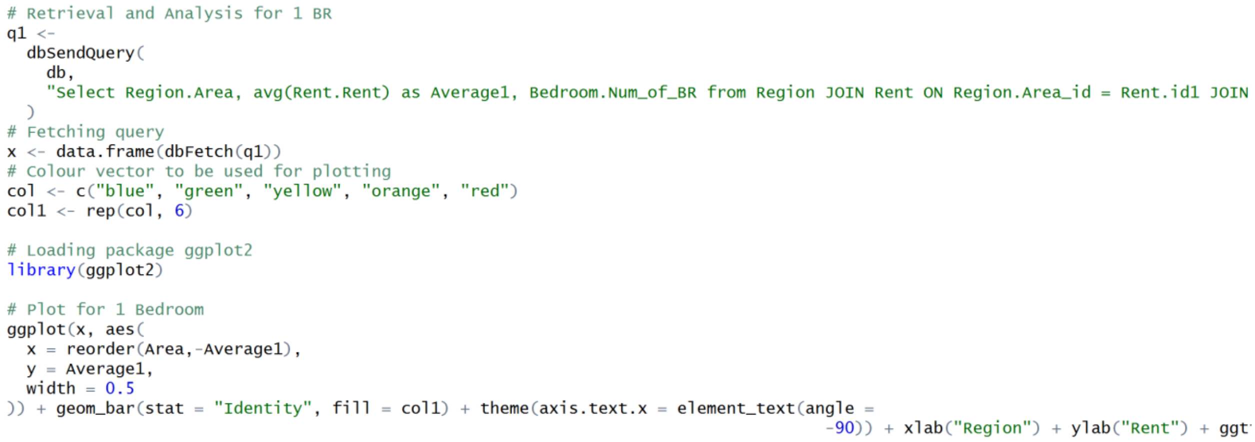


```
Bedroom_id Num_of_BR
         1       3BR
         2       1BR
         3       2BR
         4       4BR
         5    Studio
         6       6BR
         7       5BR
         8      10BR
         9       7BR
        10       8BR
```

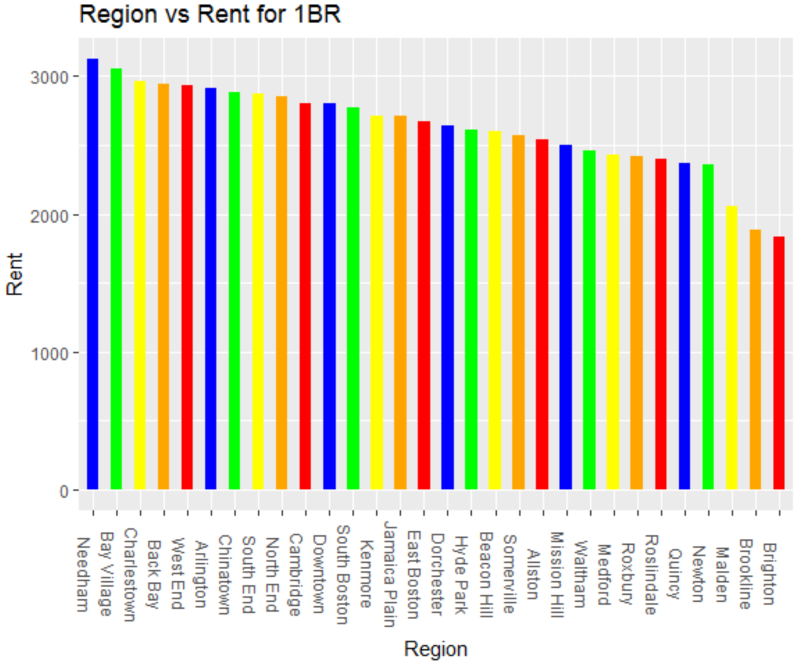
Data Retrieval and Analysis

- Most common requirements is 1BR, 2BR and 3BR apartments.
- So I have retrieved and visualized data for these three cases.
- Data is retrieved using sql query also containing joins and then plotted with ggplot2.

Retrieval and Analysis for 1BR

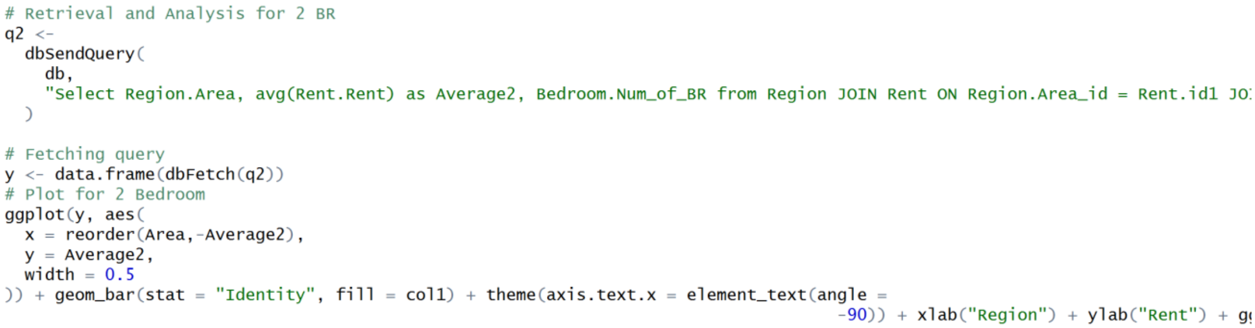


```
# Retrieval and Analysis for 1 BR
q1 <-
  dbSendQuery(
    db,
    "Select Region.Area, avg(Rent.Rent) as Average1, Bedroom.Num_of_BR from Region JOIN Rent ON Region.Area_id = Rent.id1 JOIN
  )
# Fetching query
x <- data.frame(dbFetch(q1))
# Colour vector to be used for plotting
col <- c("blue", "green", "yellow", "orange", "red")
col1 <- rep(col, 6)

# Loading package ggplot2
library(ggplot2)

# Plot for 1 Bedroom
ggplot(x, aes(
  x = reorder(Area,-Average1),
  y = Average1,
  width = 0.5
)) + geom_bar(stat = "Identity", fill = col1) + theme(axis.text.x = element_text(angle =
                                                                                   -90)) + xlab("Region") + ylab("Rent") + ggt
```

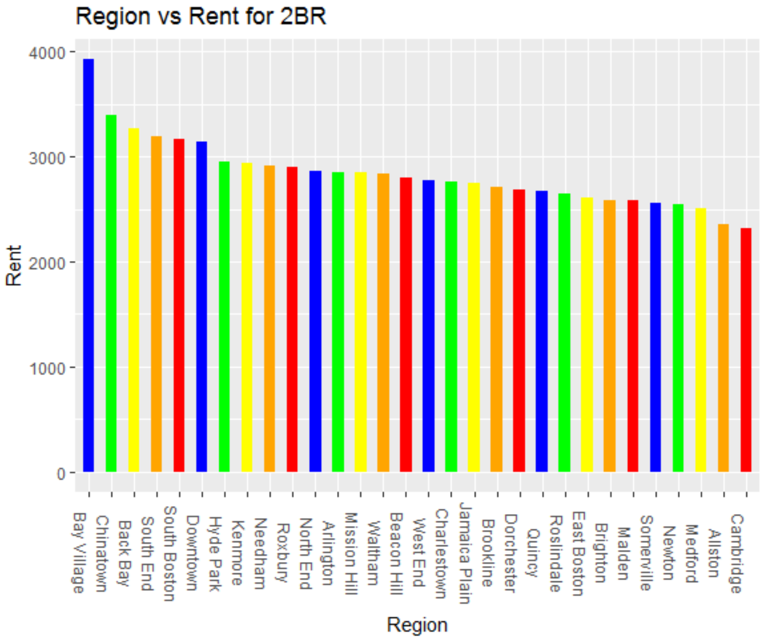


- Above plot shows average rent for 1BR in decreasing order.
- Needham being the region with highest rent and Brighton with lowest.

Retrieval and Analysis for 2BR

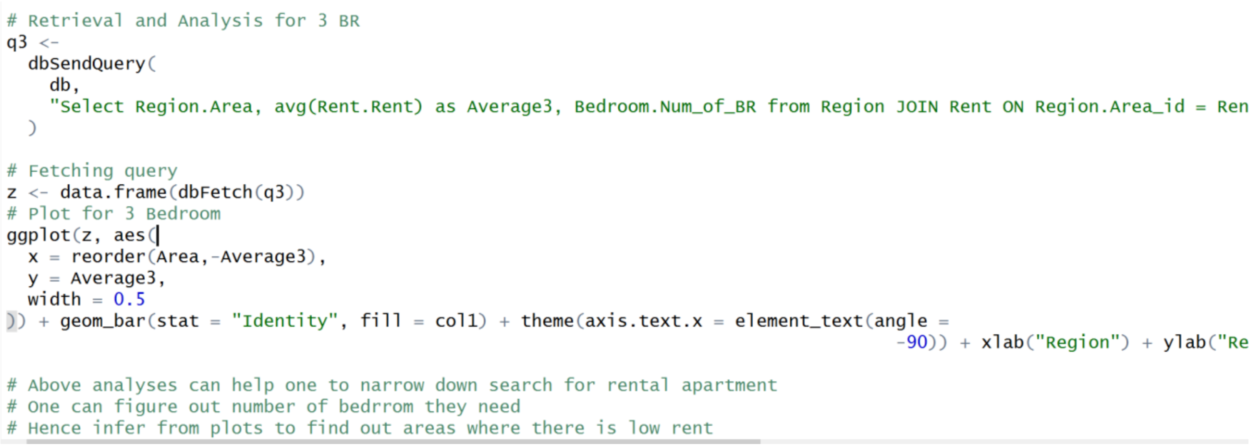


```
# Retrieval and Analysis for 2 BR
q2 <-
  dbSendQuery(
    db,
    "Select Region.Area, avg(Rent.Rent) as Average2, Bedroom.Num_of_BR from Region JOIN Rent ON Region.Area_id = Rent.id1 JO
  )

# Fetching query
y <- data.frame(dbFetch(q2))
# Plot for 2 Bedroom
ggplot(y, aes(
  x = reorder(Area,-Average2),
  y = Average2,
  width = 0.5
)) + geom_bar(stat = "Identity", fill = col1) + theme(axis.text.x = element_text(angle =
                                                                                  -90)) + xlab("Region") + ylab("Rent") + g
```

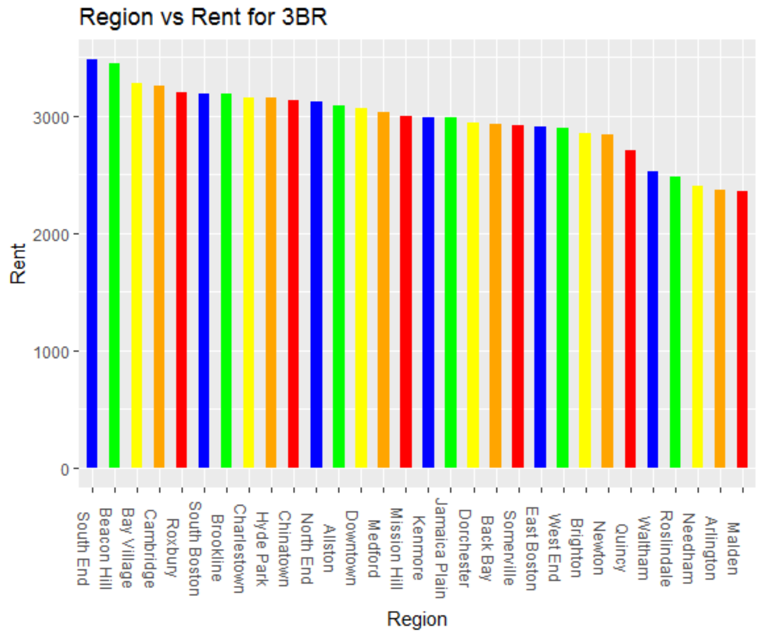


- Highest to lowest average rent region wise for 2 BR.

Retrieval and Analysis for 3 BR



```
# Retrieval and Analysis for 3 BR
q3 <-
  dbSendQuery(
    db,
    "Select Region.Area, avg(Rent.Rent) as Average3, Bedroom.Num_of_BR from Region JOIN Rent ON Region.Area_id = Ren
  )

# Fetching query
z <- data.frame(dbFetch(q3))
# Plot for 3 Bedroom
ggplot(z, aes(
  x = reorder(Area,-Average3),
  y = Average3,
  width = 0.5
)) + geom_bar(stat = "Identity", fill = col1) + theme(axis.text.x = element_text(angle =
                                                                                      -90)) + xlab("Region") + ylab("Re

# Above analyses can help one to narrow down search for rental apartment
# One can figure out number of bedrrom they need
# Hence infer from plots to find out areas where there is low rent
```



- Highest to lowest rent region wise for 3 BR.

✓ Above visualizations can be used to infer rents while looking for apartment to rent in Boston.

✓ If a person has decided apartment specifications such as number of bedrooms, he/she can narrow down search by using above plots.

✓ It will help pick region with low rent depending upon apartment specifications.

References

- Collecting, Storing and Retrieving Data Coursework.
- www.google.com
- www.wikipedia.com
- Stack Overflow