ASSIGNMENT FIVE

Nathan Chan
UC Davis, STA 141A
November 12, 2018
Professor Ulle

1. In brief: reading a post

The first function, named read_post, takes in the file name and directory of a single text file containing a Craigslist post and puts all the information in a data frame with one row.

2. In brief: reading all the posts in a directory

The second function, named read_all_posts, takes in the name of a directory and uses the read_post function to read each post in the directory.

3. More details about both functions

The function read_post creates a data frame containing a single row for each post. The function read_all_posts combines all the data frames with one row into a data frame containing information from the entire directory (again, one row is one file). I had to revise read_post several times so that it worked well with read_all_posts; I decided that it would be best for read_post to create the data frame row right away so all read_all_posts had to do was bind the data frames. I chose the columns of each data frame based on the information that was consistent on every single post, including the title, text, date, price, latitude, longitude, number of bedrooms and bathrooms, and square-footage.

Having now completed the assignment, my choice of rows and columns made it extremely convenient for further string processing, because everything was neatly organized in a data frame, which I could access very easily. Separating the title and the text was helpful, because they both included different information, and I could extract separately. Creating predefined columns for the separate data variables was also helpful, because I converted them into numbers and could easily use them for analysis.

4. Extract price from the title of the post

I wrote two functions. The function extract_rental_price takes a string, the title of the post, and returns the price. The function extract_all_prices takes a data frame and uses extract_rental_price to get a vector of all the prices. The vector of prices is then added to the data frame as a column.

Not all posts have prices in the title. There are two types of posts: either the post has the price in the title and a price attribute, or the post has neither a price in the title nor a price attribute. The price attribute at the price in the title will always match, because when Craigslist creates the title, the price in the title is taken directly from the price attribute. I assumed that the price is the first thing in the title, so even if the post has the price somewhere else in the title, I did not use it.

5. Extracting deposit from text; relationship between rental price and deposit

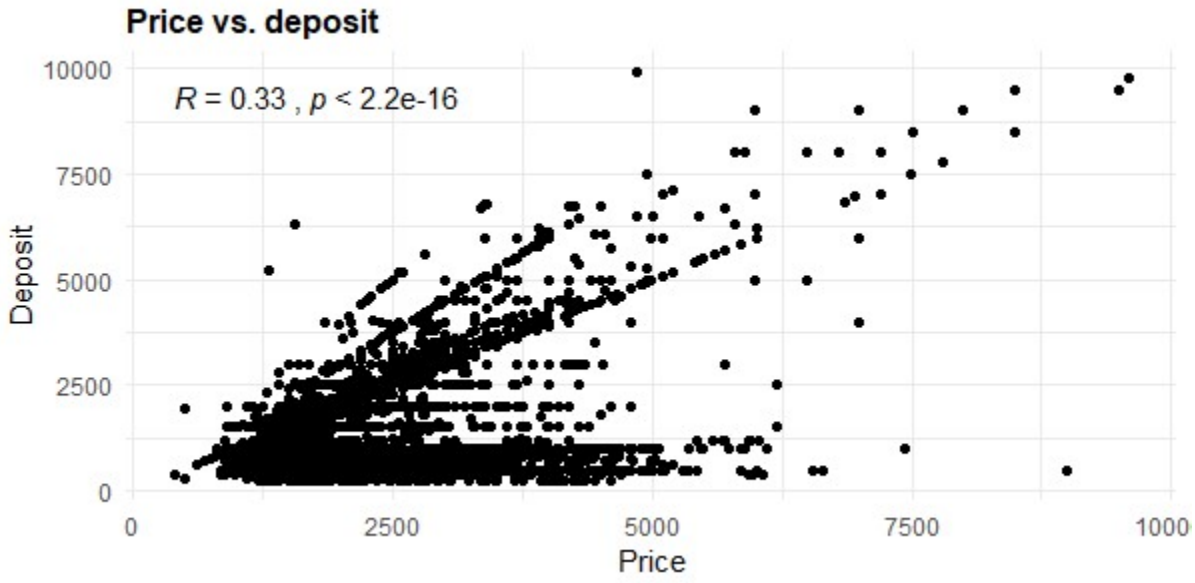
To extract the deposit amount from the text, I wrote two regular expressions that would determine whether a post had certain qualities, which would help extract the price. I checked for the word “deposit” and looked around it for the price. To get rid of pet deposits, I limited the range of the deposits, because pet deposits are cheaper than regular deposits.

After removing several outliers that couldn’t possibly be the price or the deposit amount, and are surely typos or are from the lack of capability of our algorithm (no way is the price $34,083,742 or $9,951,095, nor can a deposit or price be $0 or $1.)

The relationship between price and deposit is weak, and not very clear. While it may be common sense that as the apartments gets more expensive, the price of the deposit goes up, this is not the case, at least, for many of the apartments. It looks like many apartments charge the same deposit amount, no matter how much the apartment costs. This is the case even when forcing out pet deposits.

Why? It may be because the deposits are for repairs, and the cost of repairs does not necessarily follow the same market pattern as real estate. While the same apartment size may cost different in San Francisco compared to Sacramento, the cost of fixing a hold in the wall may be the same in each of these cities.

Here is the scatter plot:



With this strategy, I extracted around 60% of the deposits from all the posts that mentioned deposits. While not every single deposit is accurate, it is still a significant chunk, and can be improved upon with more time in the future.

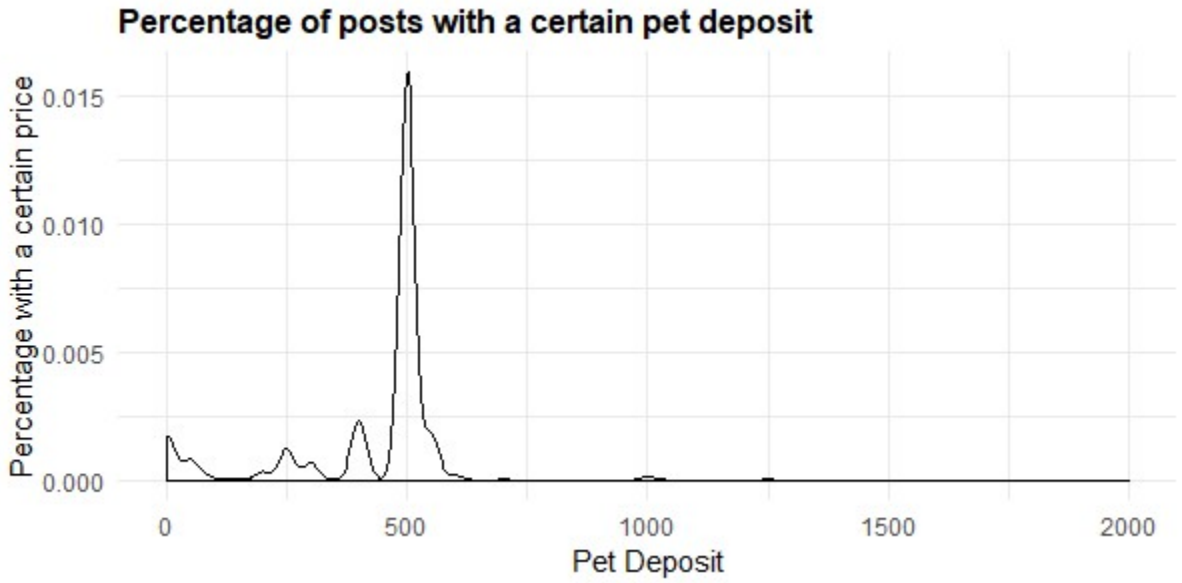
6. Extract whether an apartment allows pets: cats, dogs, both, or none

I wrote four regular expressions for each situation: the apartment allows only cats, only dogs, both cats and dogs, or neither cats nor dogs. I looked through a ton of random posts to look at how people indicate if they allow pets and what type they allow; for example, every mention of a “pet deposit” or “pet fee” would indicate that they allow all types of pets, or maybe they flat out say “no pets.” These common phrases make it easy to determine what an apartment allows.

Out of a total 45,000 posts, 25,000 posts mention “ pet” and 28,000 posts mention “ pet,” “ dog,” or “ cat.” Many of the 45,000 posts do not mention pets at all. From about 28,000 posts, I extracted about 23,000 pet policies, which is around 80% of the posts about pets.

Out of the posts I’ve seen in the data set, there was no mention of any sort of pet besides dogs and cats. This may be because most other animals are caged or have not been domesticated for the American household. For example, rabbits, snakes, fish, birds, and hamsters are all caged. Sometimes, it can be assumed; for example, if an apartments says “pets welcome” or allows both cats and dogs, they will likely allow all types of pets (since a fish is less messy than a dog, and messiness is probably the main concern of an apartment owner).

The following is the distribution of pet deposits. I used the same method of finding pet deposits as I did for regular deposits, except I included the word “pet” in front of deposit. This may exclude some pet deposits, but since we’re only looking at the distribution, a sample will work for now.



A significant number of pet deposits are about $500 dollars. Note that some apartments may not even have a pet deposit, even when they allow pets. The fact that all the pet deposits are around the same price tells us that the market for repairs from price has settled; everyone around California agrees that $500 is the proper price for a pet deposit.

7. Air conditioning and heating

Here is some analysis comparing air conditioning and heating:

	Given the apartment has air conditioning…	Given the apartment does not have air conditioning…
Percentage of apartments with heating	7,153 / 11,136 = 64.23%	10,294 / 34,709 = 29.66%
Percentage of apartments without heating	3,983 / 11,136 = 35.77%	24,415 / 34,709 = 70.34%

	Given the apartment has heating…	Given the apartment does not have heating…
Percentage of apartments with air conditioning	7,153 / 17,447 = 41.00%	3,983 / 28,398 = 14.03%
Percentage of apartments without air conditioning	10,294 / 17,447 = 59.00%	24,415 / 28,398 = 85.97%

It seems that air conditioning is the gold standard of an apartment. If the apartment has air conditioning, there will probably be heating, since air conditioning is probably more advanced than heating. If an apartment does not have air conditioning, it probably won't have a heater either.

On the other hand, if an apartment has heating, there's a good chance it won't have an air conditioner. If an apartment doesn't even have heating, then it would be very unlikely it would have a more advanced air condition system.

It seems that apartments have neither air conditioning nor heating, have both, or have heating. It is rarest that an apartment have air conditioning and no heating, since air conditioning, because it is a better technology, implies heating.

8. Hiding email addresses and phone numbers

Craigslist gives users the option to hide their contact info with a button titled "show contact info." In our text section of the posts in the data set, instead of being given the contact info, we will see the word "show contact info."

Doing a simple search of "show contact info" in the search bar, it looks like 34,832 out of 45,845 posts used the feature that replaced their contact info with the button. That means 76% of users used this feature, so it must be pretty popular and easily accessible.

9. R code appendix

```
# Assignment 5
# Nathan Chan
# UC Davis STA 141A
# Professor Ulle

library(stringr)
library(ggplot2)
library(ggpubr)
library(ggpmisc)
library(plyr)

read_post = function(file_name, directory_name) {
  # reads a single text file containing a single Craigslist post
  # takes in the directory name and the file name to access the file directly
  # gets a single vector of characters, each character representing one line
  # turn vector of characters into a row of the data frame

  # how to combine two strings
  # https://stackoverflow.com/questions/7201341/how-can-two-strings-be-concatenated

  path = paste(directory_name, file_name, sep = "/")

  file_contents = readLines(path) # vector of characters

  # combine all the vector of characters that are is part of the text description
  string = character(0)
  for (i in 2:(length(file_contents) - 7)) {
    string = paste(string, file_contents[i])
  }

  # create data frame using the vector of characters and the text description
  post = data.frame(
    title = file_contents[1],
    text = string,
    date_posted = strptime(str_remove(file_contents[length(file_contents) - 6], "Date
Posted: "),
                           "%B %d, %Y at %H:%M"),
    price = as.numeric(str_remove(file_contents[length(file_contents) - 5],
fixed("Price: $"))),
    latitude = as.numeric(str_remove(file_contents[length(file_contents) - 4],
"Latitude: ")),
    longitude = as.numeric(str_remove(file_contents[length(file_contents) - 3],
"Longitude: ")),
    bedrooms = as.numeric(str_remove(file_contents[length(file_contents) - 2],
"Bedrooms: ")),
    bathrooms = as.numeric(str_remove(file_contents[length(file_contents) - 1],
"Bathrooms: ")),
    sqft = as.numeric(str_remove(file_contents[length(file_contents)], "Sqft: "))
  )

  return(post) # return one row data frame
}
```

```
read_all_posts = function(directory_name) {
  # get names of all posts in a directory (one txt file = one post)
  # use read_post to read all the posts in the directory using the names of the posts
  # returns information in a data frame

  files = list.files(directory_name) # character vector
  # all_directory_contents = lapply(files, read_post, directory_name =
directory_name)
  # list of one row data frames

  # an empty data frame
  # https://stackoverflow.com/questions/10689055/create-an-empty-data-frame

  df_all_contents = data.frame(
    title = character(),
    text = character(),
    date_posted = character(),
    price = character(),
    latitude = character(),
    longitude = character(),
    bedrooms = character(),
    bathrooms = character(),
    sqft = character()
  )

  # combine all rows of data frames into one giant data frame
  # https://stackoverflow.com/questions/8169323/r-concatenate-two-dataframes

  # for (i in seq_along(all_directory_contents)) {
  #
  #   # access elements in a list
  #   # https://stackoverflow.com/questions/21091202/how-to-index-an-element-of-a-
list-object-in-r
  #
  #   df_all_contents = rbind(df_all_contents, all_directory_contents[[i]])
  # }

  # the above method is less efficient, because rbind needs to make a copy of the
data frame from the list
  # need to use a for loop

  # create a complete data frame with contents of all files in directory
  for (i in seq_along(files)) {
    df_all_contents = rbind(df_all_contents, read_post(files[i], directory_name =
directory_name))
  }

  return(df_all_contents) # return complete data frame
}

directories = list.files("messy")

# create an empty data frame (that will hold data frames from all directories)
df = data.frame(
```

```
  title = character(),
  text = character(),
  date_posted = character(),
  price = character(),
  latitude = character(),
  longitude = character(),
  bedrooms = character(),
  bathrooms = character(),
  sqft = character()
)

# # get a list of all the data frames from each directory
# list_df = lapply(directories, function(directory) {
#   path = paste("messy", directory, sep = "/")
#   return(read_all_posts(path))
# })
#
# # combine data frames from each directory into a single, complete data frame
# for (i in seq_along(list_df)) {
#   df = rbind(df, list_df[[i]])
# }

# the above method is less efficient, because rbind needs to make a copy of the data
frame from the list
# need to use a for loop

# create a complete data frame with information from all directories
for (i in seq_along(directories)) {
  directory_name = paste("messy", directories[i], sep = "/")
  directory_df = read_all_posts(directory_name)
  df = rbind(df, directory_df)
}

# 4.
# Goal: get rental price from title
# Path: read title, extract first word (which is always the price), add to end of
data frame

extract_rental_price = function(row) {
  # given a single string, read the title, and return price
  rental_price = str_split(row, " ", 2)
  rental_price = str_remove(rental_price[[1]][1], fixed("$"))
  rental_price = as.numeric(rental_price)
  return(rental_price)
}

extract_all_prices = function(df) {
  # given a data frame, extract prices from all rows, first column, return a vector
of the prices
  prices = sapply(df[, 1], extract_rental_price)
  return(prices)
}

# add vector of prices to df as the last column
title_price = extract_all_prices(df)
```

```
df = cbind(df, title_price)

# 5.
# Goal: extract the deposit from the text
# Path: Look at how different posts list their deposit, built test cases for them

# look at how many posts contain the word "deposit"
pattern = regex("deposit", ignore_case = TRUE)
has_deposit = str_detect(df$text, pattern)
table(has_deposit)
# there are ~18,000 posts with deposit in them

# see how the deposit amount is formatted
message(df$text[has_deposit][[1]])
message(df$text[has_deposit][[60]])

regex1 = regex("(deposit)[-: ]*([$]*[0-9][,.]*[0-9]*[,.]*[0-9]*)", ignore_case =
TRUE)
match1 = str_match(df$text, regex1)

regex2 = regex("([$]*[0-9][,.]*[0-9]*[,.]*[0-9]*)[ ]*(deposit)", ignore_case = TRUE)
match2 = str_match(df$text, regex2)

# combine match1 and match2
combined = cbind(match1[, 3], match2[, 2])

# see how many NA's we have with the two matches
num_na = rowSums(is.na(combined))
table(num_na)
# this tells us we have ~11,000 deposit matches out of 18,000

# gets first column
combined_final = combined[, 1]

# get second column if first column was NA
is_na = is.na(combined_final)
combined_final[is_na] = combined[is_na, 2]

combined_final = sapply(combined_final, function(x) {
  x = str_remove(x, fixed("$"))
  x = str_remove(x, fixed(","))
  x = as.numeric(x)
  return(x)
})

deposit = combined_final
df = cbind(df, deposit)

df2 = df
df2 = subset(df2, df2$price < 15000 & df2$price > 200)
df2 = subset(df2, df2$deposit < 10000 & df2$deposit > 200)

ggplot(df2, aes(x = price, y = deposit)) +
  geom_point() +
  labs(x = "Price", y = "Deposit", title = "Price vs. deposit") +
```

```
  geom_smooth(method = 'loess') +
  theme_minimal() +
  theme(plot.title = element_text(size = 12, face = "bold"), text =
element_text(family = "Helvetica")) +
  stat_cor(method = "pearson")

# 6.
# Goal: figure out if an apartment allows cats, dogs, both, or none
# Path: come up with regular expression combinations that would indicate whether an
apartments allows pets or not

df3 = df

pattern0 = regex("( pet)", ignore_case = TRUE)
pet_mentioned = str_detect(df$text, pattern0)
table(pet_mentioned)
# ~25,000 posts mention pets

pattern0 = regex("( pet| dog| cat)", ignore_case = TRUE)
pet_mentioned = str_detect(df$text, pattern0)
table(pet_mentioned)
# ~28,000 posts mention pets

pattern1 = regex("(pet[- ]*friendly)|(pet[- ]*fee)(pet[- ]*allowed)|(pet rent|pet
deposit|pets are allowed|
                 pet max|pets max|pet spa|pets welcome|dogs and cats allowed|pet
area|pet limit|cats & dogs|
                 pets ok|pets allowed|dogs and cats ok)|((dogs are ok)&(cats are
ok))", ignore_case = TRUE)
pattern2 = regex("(no pet|not pet|no animal|pets: no|no dogs or cats allowed)",
ignore_case = TRUE)
pattern3 = regex("(no dogs|no dog|not dogs|not dog|cats allowed|cats welcome)|(cat[-
]*friendly|cats are ok)", ignore_case = TRUE)
pattern4 = regex("(no cats|no cat|not cats|not cat|dog park|dogs allowed|dogs
welcome|dog run|dog spa)|
                 (dog[- ]*friendly|dogs are ok)", ignore_case = TRUE)

# check if all pets are allowed
pet_both = str_detect(df$text, pattern1)
table(pet_both)
pet_both = sapply(pet_both, function(x){
  if (x == TRUE) {
    return("both")
  } else {
    return(NA)
  }
})

pet_none = str_detect(df$text, pattern2)
table(pet_none)
pet_none = sapply(pet_none, function(x){
  if (x == TRUE) {
    return("none")
  } else {
    return(NA)
```

```
  }
})

pet_no_dog = str_detect(df$text, pattern3)
table(pet_no_dog)
pet_no_dog = sapply(pet_no_dog, function(x){
  if (x == TRUE) {
    return("cats")
  } else {
    return(NA)
  }
})

pet_no_cat = str_detect(df$text, pattern4)
table(pet_no_cat)
pet_no_cat = sapply(pet_no_cat, function(x){
  if (x == TRUE) {
    return("dogs")
  } else {
    return(NA)
  }
})

pet_combined = cbind(pet_both, pet_none)
pet_combined = cbind(pet_combined, pet_no_dog)
pet_combined = cbind(pet_combined, pet_no_cat)

pet_combined

# gets first column
pet_combined_final = pet_combined[, 1]

# get second column if item in first column was NA
is_na = is.na(pet_combined_final)
pet_combined_final[is_na] = pet_combined[is_na, 2]

# get third column if item in first column was NA
is_na = is.na(pet_combined_final)
pet_combined_final[is_na] = pet_combined[is_na, 3]

# get fourth column if item in first column was NA
is_na = is.na(pet_combined_final)
pet_combined_final[is_na] = pet_combined[is_na, 4]
pet_combined_final
pets = pet_combined_final

table(is.na(pets))

df3 = cbind(df, pets)

# get a df of only pets to do analysis
df_pets = subset(df3, df3$pets == "both" | df3$pets == "dogs" | df3$pets == "cats")

# use same method from deposit
```

```
regex1 = regex("(pet deposit)[-: ]*([$]*[0-9][,.]*[0-9]*[,.]*[0-9]*)", ignore_case =
TRUE)
match1 = str_match(df$text, regex1)

regex2 = regex("([$]*[0-9][,.]*[0-9]*[,.]*[0-9]*)[ ]*(pet deposit)", ignore_case =
TRUE)
match2 = str_match(df$text, regex2)

# combine match1 and match2
combined = cbind(match1[, 3], match2[, 2])

# see how many NA's we have with the two matches
num_na = rowSums(is.na(combined))
table(num_na)
# this tells us we have ~11,000 deposit matches out of 18,000

# gets first column
combined_final = combined[, 1]

# get second column if first column was NA
is_na = is.na(combined_final)
combined_final[is_na] = combined[is_na, 2]

combined_final = sapply(combined_final, function(x) {
  x = str_remove(x, fixed("$"))
  x = str_remove(x, fixed(","))
  x = as.numeric(x)
  return(x)
})

pet_deposit = combined_final
df_pet = cbind(df, pet_deposit)

ggplot(df_pet, aes(x = pet_deposit)) +
  geom_density() +
  labs(x = "Pet Deposit", y = "Percentage with a certain price",
       title = "Percentage of posts with a certain pet deposit") +
  theme_minimal() +
  theme(plot.title = element_text(size = 12, face = "bold"), text =
element_text(family = "Helvetica"))

# 6.
# Goal: Determine whether apartment has different types of heating

df4 = df

pattern1 = regex("( heater|heating| heat)(?=(fireplace|fire place|wood-burning|wood
burning))", ignore_case = TRUE)
pattern2 = regex("( heater|heating| heat)", ignore_case = TRUE)
pattern3 = regex("(fireplace|fire place|wood-burning|wood burning)", ignore_case =
TRUE)
pattern4 = regex("( )", ignore_case = TRUE) # don't mention either

both = str_detect(df$text, pattern1)
table(both)
```

```
both = sapply(both, function(x){
  if (x == TRUE) {
    return("both")
  } else {
    return(NA)
  }
})

heat = str_detect(df$text, pattern2)
table(heat)
heat = sapply(heat, function(x){
  if (x == TRUE) {
    return("heater")
  } else {
    return(NA)
  }
})

fireplace = str_detect(df$text, pattern3)
table(fireplace)
fireplace = sapply(fireplace, function(x){
  if (x == TRUE) {
    return("fireplace")
  } else {
    return(NA)
  }
})

none = str_detect(df$text, pattern4)
table(none)
none = sapply(none, function(x){
  if (x == TRUE) {
    return("neither")
  } else {
    return(NA)
  }
})

combined = cbind(both, heat)
combined = cbind(combined, fireplace)
combined = cbind(combined, none)

# gets first column
heating = combined[, 1]

# get second column if item in first column was NA
is_na = is.na(heating)
heating[is_na] = combined[is_na, 2]

# get third column if item in first column was NA
is_na = is.na(heating)
heating[is_na] = combined[is_na, 3]

# get fourth column if item in first column was NA
is_na = is.na(heating)
```

```
heating[is_na] = combined[is_na, 4]
heating

table(is.na(heating))

df4 = cbind(df4, heating)

# Objective: extract whether an apartment has air conditioning

pattern1 = regex("(air condition|air-condition)|( a[/]*c[!./:; ])", ignore_case =
TRUE)
pattern2 = regex("( )", ignore_case = TRUE)

ac = str_detect(df$text, pattern1)
table(ac)
ac = sapply(ac, function(x){
  if (x == TRUE) {
    return("yes")
  } else {
    return(NA)
  }
})

no_ac = str_detect(df$text, pattern2)
table(no_ac)
no_ac = sapply(no_ac, function(x){
  if (x == TRUE) {
    return("no")
  } else {
    return(NA)
  }
})

air_condition = cbind(ac, no_ac)

# gets first column
ac = air_condition[, 1]

# get second column if item in first column was NA
is_na = is.na(ac)
ac[is_na] = air_condition[is_na, 2]

df4 = cbind(df4, ac)

# Do apartments with heating also have air conditioning? And vice versa? Is one more
common than the other?

# apartments with air conditioning
with_ac = subset(df4, df4$ac == "yes")
nrow(with_ac) #11,136

# apartments heating given ac
with_ac_heating = subset(with_ac, with_ac$heating != "neither")
nrow(with_ac_heating) #7,153
```

```
# apartments without heating given ac
with_ac_no_heating = subset(with_ac, with_ac$heating == "neither")
nrow(with_ac_no_heating) #3,983

# apartments without air conditioning
without_ac = subset(df4, df4$ac == "no")
nrow(without_ac) #34,709

# apartments heating given no ac
without_ac_heating = subset(without_ac, without_ac$heating != "neither")
nrow(without_ac_heating) #10,294

# apartments without heating given no ac
without_ac_no_heating = subset(without_ac, without_ac$heating == "neither")
nrow(without_ac_no_heating) #24,415



# apartments with heating
with_heating = subset(df4, df4$heating != "neither")
nrow(with_heating) #17,447

# apartments with both
with_both = subset(df4, df4$ac == "yes" & df4$heating != "neither")
nrow(with_both) #7,153

# apartments without heating
without_heating = subset(df4, df4$heating == "neither")
nrow(without_heating)

# apartments without heating, with ac
without_heating_yes_ac = subset(without_heating, without_heating$ac == "yes")
nrow(without_heating_yes_ac)
```