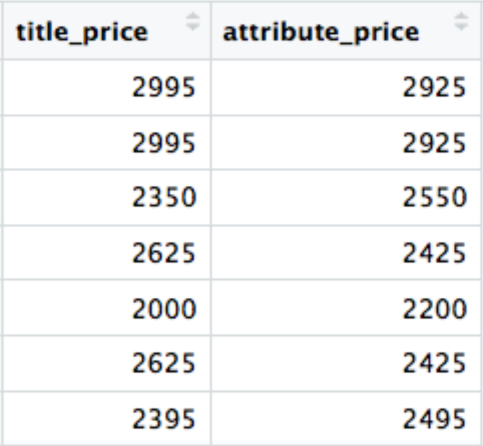
Mira Mastoras
STA141A Assignment 5

Read_posts and read_all_posts function design

Two functions were used to read all craigslist posts in from the messy_cl directory provided, and structure them into a data frame. The first, read_posts, takes the entire file path as an argument and returns a string containing all the text from the file. The second function, read_all_posts, takes the file path to the craigslist location directories inside of messy_cl, such as “losangeles” as an argument, and builds a data frame with all the files in that directory. Each post is a row, and there are two columns: one is the text and the other is the name of the directory such as “losangeles.” I chose to select the rows and columns this way because I was following the example used in the lecture 15 notes for the messy craigslist vehicles example, and it seemed like an appropriate way to structure the returned data frame. Read_all_posts uses read_posts by using sapply to apply read_posts onto every file in the outer directory.

When I reached problems 6-8, I realized a better way to solve the problem would have been to look ahead, and extract all the attributes I needed for these problems inside of my read_posts and read_all_posts functions. Basically, I should have called separate functions reading a single text file and extracting the attribute in my read_post functions. At this point, I was too nervous to alter my read_all_posts and read_posts functions as they were working well, so I wrote new functions to extract the necessary attributes and used lapply to apply them over the entire data frame, then appended the returned list of attributes to my posts_df. This method worked well, but was much more computationally cumbersome and time consuming than if I had looked ahead and incorporated the attribute extraction in the read_posts and read_all_posts functions.

Rental Prices



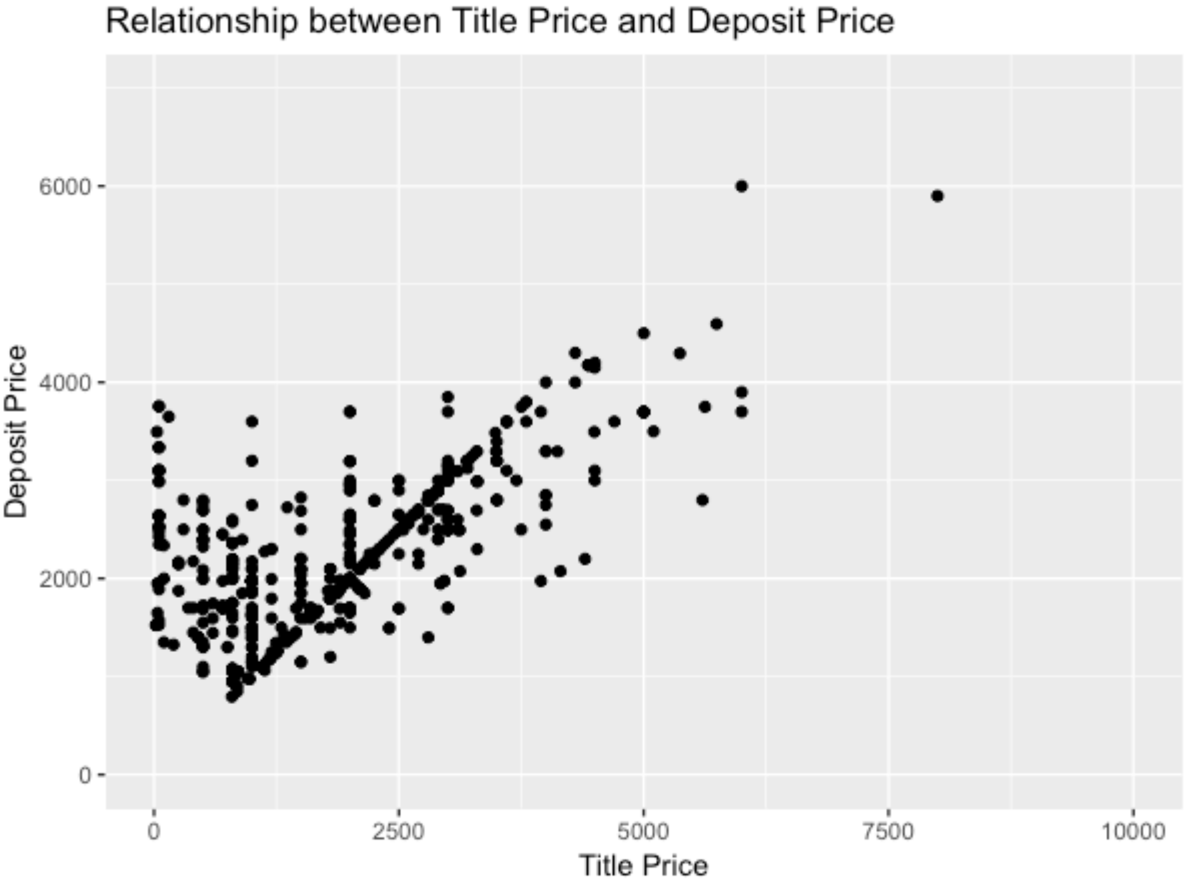
title_price	attribute_price
2995	2925
2995	2925
2350	2550
2625	2425
2000	2200
2625	2425
2395	2495

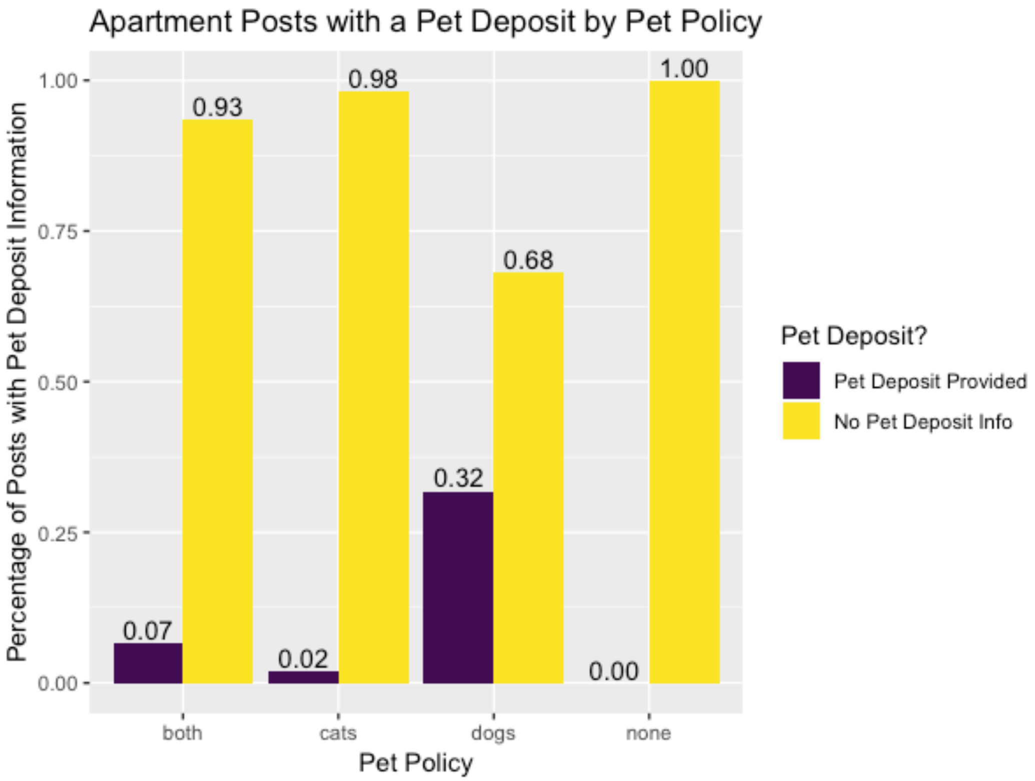
The majority of the posts (45669) have prices in their titles, while only 176 posts do not contain prices in their titles. None of these 176 posts contain missing values or NA’s in their title or text, therefore the poster simply didn’t include the price in the title. All 176 of the posts that were missing title prices were also missing attribute prices, meaning that the poster didn’t include price in either of these locations. Most likely such a high percentage of posts have prices in their titles because posters know that price is a huge factor for people searching for apartments, and the main attribute that they use to filter their apartment search results.

In addition, 180 posts out of the total 45845 are missing a price attribute. Only 7 posts had a different price listed in their price attribute and in their title. Looking at the differences between these posts, all seven of them have a difference of a single digit, indicating that the difference between title price and attribute price was caused by a typo on the part of the poster.

Deposit Prices:

There appears to be a modest positive relationship between deposit price and title price. Of all the posts, 8172 had a deposit number listed that was not a pet deposit. Outliers with a title price above $10,000 were excluded from the analysis. It appears that for most of the posts, as title price increases so does deposit price. This trend is particularly true for posts with a title price above $1200. For posts below $1200, the relationship between title price and deposit price is less clear, with more variability in the deposit price for any post at a given title price. In general however, higher title prices correspond to higher deposit prices, which makes sense because more expensive apartments would expect people to pay more for a deposit.

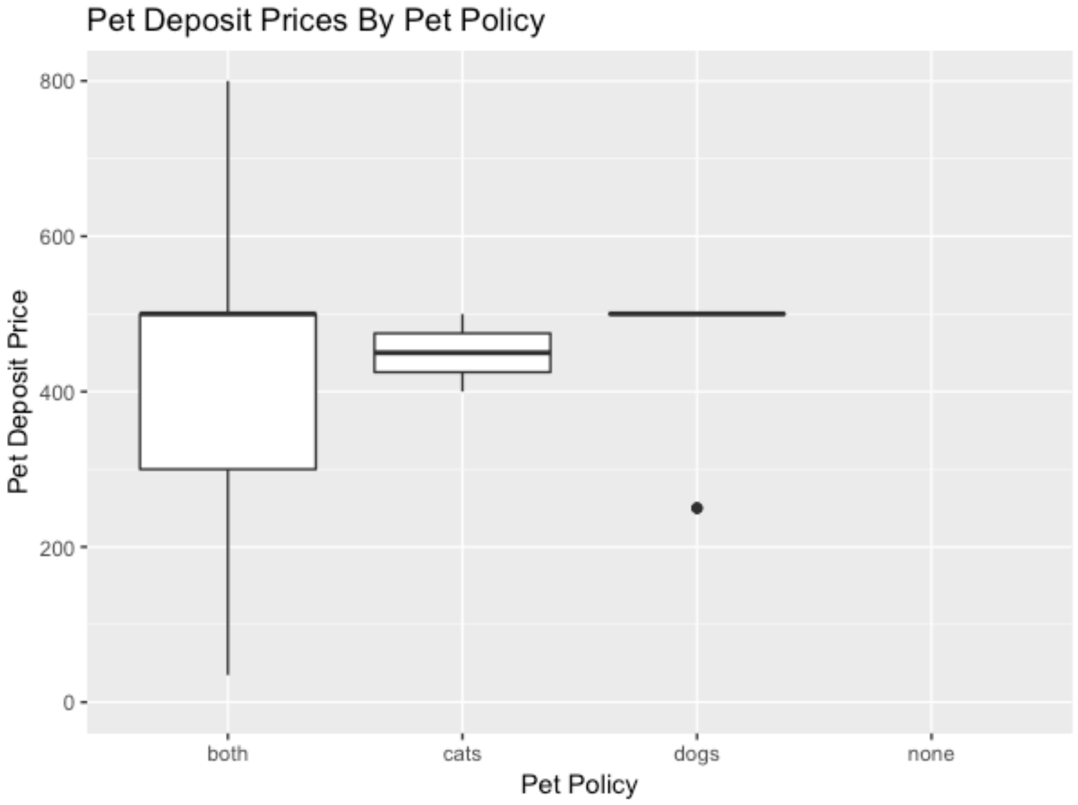




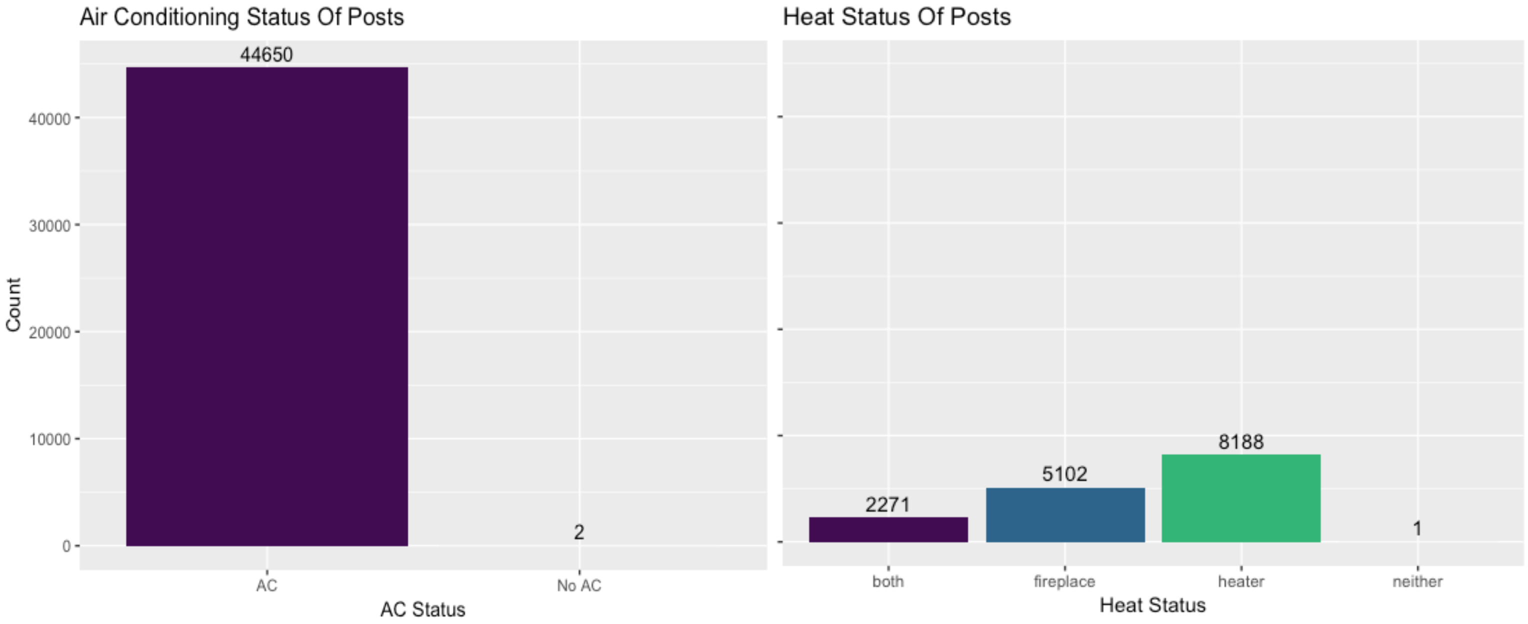
<u>Pets</u>

Out of 45845 apartment posts, I was able to extract a pet deposit value from 2108 posts. First, I examined whether the type of pet allowed had any influence over whether a pet deposit was required or provided. For apartments that only allow dogs, 32% of the posts provided a pet deposit value while the remaining 68% did not. This represents a much higher proportion of posts that provided a pet deposit value than can be seen in the other pet policy categories. For apartments that allow either cats or both dogs and cats, greater than 90% of the apartment posts did not provide a pet deposit value. Notably, it cannot be assumed that these apartments don’t have a pet deposit at all; they may have just forgotten to provide it in their post, or they may have provided it in a format that my extraction algorithm failed to account for.

Next, I examined whether certain pet policy categories resulted in a higher pet deposit price. The average deposit price appeared to be very similar for apartments with each type of pet policy (dogs only, cats only, or both dogs and cats). The average pet deposit for dogs and both dogs and cats was $500, and for cats only it was close to $450. The variation and spread of the pet deposit prices was strikingly different between the three categories. For cats, the data was evenly spread about the mean. For both cats and dogs, the mean was exactly at the upper IQR, indicating that there were a lot of high outliers pulling up the average. For dogs, there was no spread of the data, with all points clustered at the mean and only a few outliers (one shown). This is probably due to the fact that there were only 44 posts allowing only dogs. With such a small sample size, strange and skewed distributions are very likely to occur.

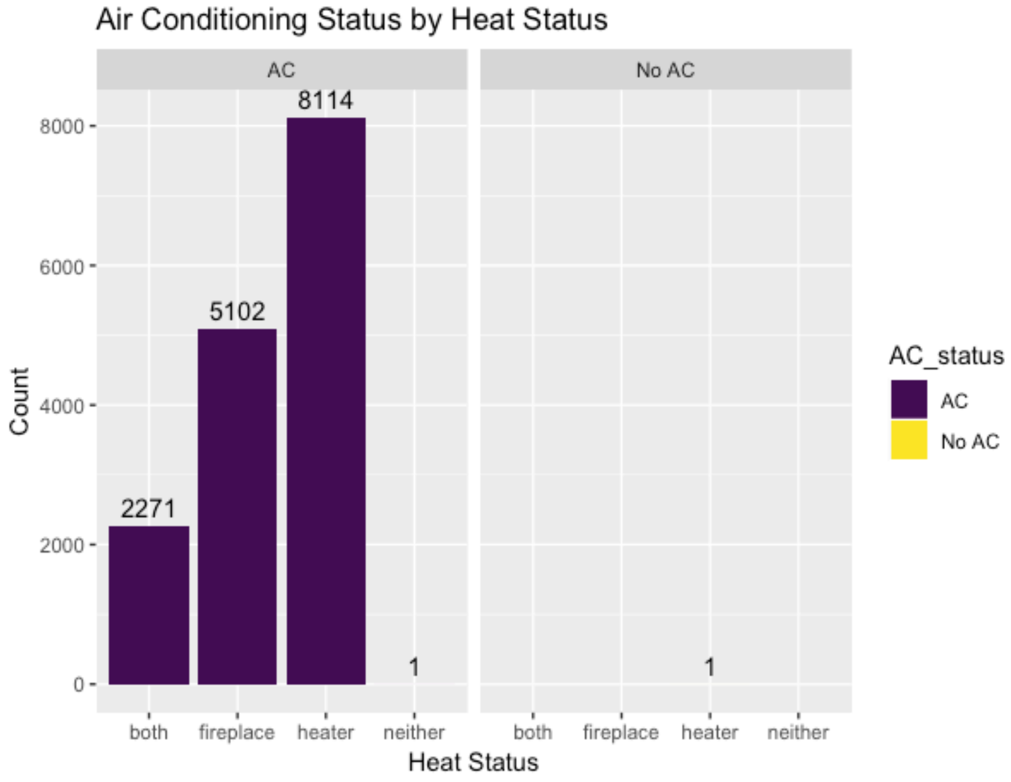


Some of the apartments do indeed allow other pets besides dogs or cats. For instance, I found an apartment in downtown Oakland which allows exotic animals such as “Birds (Cockatiels, Parrots & Macaws), Ferrets, Raccoon's, Reptiles (Snakes and Iguanas), Rabbits, Squirrels, and Skunks.” I discovered this by searching the posts for different animal names, such as “rabbits.” In fact, I found 173 posts which allow rabbits as a pet.

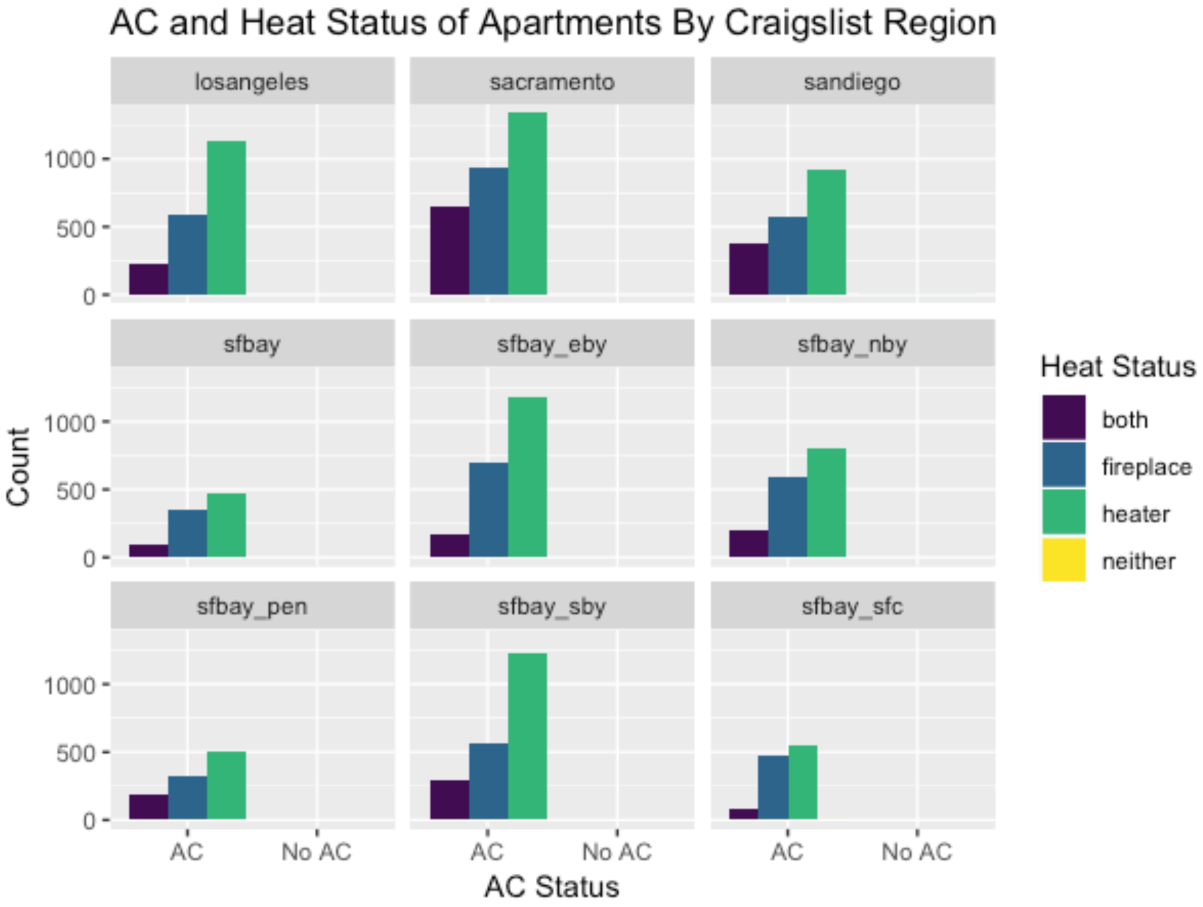


AC and Heat

Air conditioning is much more common than heating. Out of 45845 posts, 44650 had air conditioning, while only 2 specifically mentioned they did not. The rest did not mention AC and were designed “NA.” In comparison, only 8188 posts had a central heating system, 5102 had a fireplace, and 2271 had both. This trend makes sense when you consider that these apartments are all in California, where the climate is much hotter year round than in other places. In most places in California, an air conditioning is more important than a heating system, which is why the posters probably highlighted the presence of the air conditioning system over a heating system.



Apartments without air conditioning are much less likely to have heating than apartments with air conditioning. Only one apartment without AC had a central heat system. As you can see, none of the apartments without AC fell into the heat status category of “neither,” meaning they didn’t have a fireplace or a heater. This is because to satisfy the requirement for the “neither” category, the post had to mention that they did not have any heating system. Most apartments left this information out, and thus fell into the NA category. As the “NA” apartments provided no information about their heating system, we cannot make conclusions about them. For the apartments that did have AC, a higher amount of them also contained a heater (8114) than contained a fireplace(5102) or both a heater and a fireplace(2271).



Next, I was curious as to whether apartments in different regions had different distributions of heating and air conditioning units. This would make sense to me, because different regions would have slightly different climates which may impact whether the apartment would like to highlight the presence of these temperature-control appliances. Based on craigslist website region, there is not a huge difference in the presence of these different appliances. Apartments with AC units in the SF-south bay region are more likely to have a heater compared with apartments in other regions. In addition, apartments in San Francisco with AC are pretty much equally likely to have either a fireplace or a heater, whereas in other regions apartments with AC are significantly more likely to contain a heater than a fireplace. This could be that a greater majority of apartments in the city of San Francisco are older and contain more fireplaces than modern heating systems. This analysis is limited in that the regions examined are quite broad. A more detailed analysis parsing apartment posts by more distinct geographical regions, such as county, may provide more interesting results.

Hidden Contact Info

My original strategy when examining whether many people choose to hide their contact information from web scrapers was to search the posts for "@" symbols, and identify whether these were associated with personal email addresses. Only 294 posts contained valid email addresses, and many of these were repeats from apartment posts posted by the same rental company. Only 3 of the email addresses appeared to be personal email addresses, and not the email address of the apartment complex or rental company. Using the same strategy I extracted 1302 posts which contained phone numbers, some of which are for the rental company's and not personal ones. Out of 45845 posts, 1302 is such a small fraction that it also supports that most people use the hide contact info feature. I also noticed that 34,832 posts contained the phrase "show contact info", which indicated to me that this was the way the craigslist website hid the email addresses and phone numbers from web scrapers: by having them hidden underneath a link that you would have to click on to view them. There are 45845 posts in total, and since only 3 showed personal email addresses and 34,832 selected to hide their contact info underneath the "show contact info" link, I conclude that the majority of posts opt for this choice.

Sources:

https://stackoverflow.com/questions/17164715/how-to-remove-a-level-of-lists-from-a-list-of-lists
https://piazza.com/class/jmf7qwk0sf03ya?cid=339
https://www.wired.com/2008/08/four-regular-expressions-to-check-email-addresses/
https://piazza.com/class/jmf7qwk0sf03ya?cid=463

Code:

```
# Mira Mastoras
# STA141A HW 5

library(ggplot2)

# Question 1
library(stringr)
read_posts = function(file) { # parameter "file" is full file path
  text = readLines(file)
  text = str_c(text, collapse = '\n') # return value is a string with the text from the file
}

#test for read_posts
```

```
desc = read_posts("/Users/miramastoras/sta141a/messy/losangeles/_ant_apa_d_1-bd-1-bd-water-paid-utility_6718641721.txt")
desc

# Question 2

read_all_posts = function(directory) { # parameter is full path to craigslist location directory ie "losangeles"
  files = list.files(directory, full.names = T )
  posts = sapply(files, read_posts)
  newposts = data.frame(text = posts, region = basename(directory))
}

# gets list of files from messy directory
area_list = list.files(messy_cl, full.names = TRUE)
area_list
#applies the read_all_posts function to each directory
posts = lapply(area_list, read_all_posts)
head(posts)
#combines list of data frames in "posts" into a data frame of data frames
posts_df = do.call(rbind, posts)
str(posts_df)

# remove duplicate rows from DF
posts_df[!duplicated(posts_df$text), ]

# Question 4 -- Rental Prices

test = posts_df$text[[1]]
message(test)

# extract price from title and store it back in posts_df

result = str_split_fixed(posts_df$text, "\n", 2)
posts_df$title = result[, 1]
posts_df$text = result[, 2]

str(posts_df)

#gets title price from each post
title_price = lapply(posts_df$title, str_match, "\\$[0-9]+")
head(title_price)
new_title_price = unlist(title_price, recursive=FALSE)
new_title_price = lapply(new_title_price, str_remove_all, "[\\$]")
new_title_price = unlist(new_title_price)
new_title_price
new_title_price = as.numeric(new_title_price)

#checking the new title price Df matches the posts
head(title_price_df)
nrow(posts_df)
nrow(title_price_df)


# extract price from the attribute:

attribute_price = lapply(posts_df$text, str_extract, regex("Price:\\s\\$[0-9,]+", ignore_case =
TRUE))
attribute_price_new = lapply(attribute_price, str_split, regex("Price: " , ignore_case = TRUE))
# removing nested lists
attribute_1 = unlist(attribute_price_new, recursive=FALSE )
attribute_2 = unlist(attribute_1, recursive=FALSE )

attribute_price = attribute_2[!attribute_2 == ""]
attribute_price = lapply(attribute_price, str_remove_all, "[\\$]")
attribute_price = unlist(attribute_price)

attribute_price = as.numeric(attribute_price)

shadow = is.na(attribute_price)
table(shadow)

#adding prices to DF
posts_df$attribute_price = attribute_price
posts_df$title_price = new_title_price

posts_df$price_diff = posts_df$title_price - posts_df$attribute_price

#getting rid of the NA's:

shadow = !(is.na(posts_df$price_diff))
shadow
posts_na_rm = posts_df[shadow,]
nrow(posts_na_rm)

#subset based on those with price difference btw title and attribute
```

```
price_is_diff = posts_na_rm[!posts_na_rm$price_diff == "0", ]
price_is_diff[ select =c(posts_na_rm$title_price, posts_na_rm$attribute_price)]

#subset data based on just those two columns
subset(price_is_diff, select = c(title_price, attribute_price))

#Do all of the titles have prices?

shadow = is.na(new_title_price)
table(shadow) # 176 do not have prices in title - they are NA

shadow_na_title = (is.na(posts_df$title_price)) # find posts with na in their title
posts_just_na = posts_df[shadow_na_title,] # subset by posts with na in title
nrow(posts_just_na)
colnames(posts_just_na)

#none of these have NA in their text
table(is.na(posts_just_na$text))
table(is.na(posts_just_na$attribute_price))
posts_just_na$title[1]


# Question 5-- deposit price

contains_deposit_TF = str_detect(posts_df$text, regex("deposit", ignore_case = T)) #how many
have deposit in them? 18303

#subset by posts with the word deposit
contains_dep_df = posts_df[contains_deposit_TF,]

#subset by all posts without pet deposits, reg ex code from Piazza
no_pets = str_detect(contains_dep_df$text,
regex("(pets?|dogs?|cats?|animals?)(friendly|allowed|additional|.*)(deposit|fee)",ignore_case =
TRUE))

only_correct_dep = contains_dep_df[!no_pets,]
nrow(only_correct_dep) #8172 posts have apartment deposit info

#now extracting the deposit price

deposit_price = str_match(only_correct_dep$text, regex("deposit\\s\\$([0-9,]+)", ignore_case =
TRUE ))
num_dep_price = deposit_price[,2]
deposit_price_clean = str_remove(deposit_price[,2], ",")
```

```
table(is.na(deposit_price_clean))
deposit_price_clean = as.numeric(deposit_price_clean)

deposit_price

only_correct_dep$deposit_price = deposit_price_clean #extracted 608 prices

# graph looking at relationship btw title and deposit
ggplot(only_correct_dep, aes( y = title_price, x = deposit_price)) +
  geom_point() + ylim(0,7000) +
  xlim(0, 10000) +

  labs(title = "Relationship between Title Price and Deposit Price", x = "Title Price", y = "Deposit
Price")

# looking at attribute price & deposit , looks the same

ggplot(only_correct_dep, aes( y = attribute_price, x = deposit_price, alpha = 0.3)) +
  geom_point() + ylim(0,7000) +
  xlim(0, 10000) +
  labs(title = "Relationship between Attribute Price and Deposit Price", x = "Title Price", y =
"Deposit Price")

# random code
table(is.na(deposit_price))
only_correct_dep$deposit_price = deposit_price
table(is.na(only_correct_dep$deposit_price))
shadow_1 = ! (is.na(deposit_price))
newnew = posts_df[shadow_1,]
newnew$text[1]
posts_df[shadow_1,]


# Question 6:

# extracting pets allowed feature

parse_pets = function(text) {
  status = NA
  pets_sent = str_extract(text, regex("((\\w+ ){0,5})pets? ((\\w+ ){0,5})|((\\w+ ){0,5})dogs? ((\\w+
){0,5})|((\\w+ ){0,5}) cats?((\\w+ ){0,5})", ignore_case = T)) # regex from piazza
  if (!is.na(pets_sent)) {
    both_yes = str_detect(pets_sent,  regex("(pets?\\s?|animals?\\s?|dogs? and cats?\\s?|cats?
and dogs?)\\s?( friendly | allowed | yes | welcome .*)", ignore_case = T))
```

```
    cat_yes = str_detect(pets_sent, regex(" cats?\\s(allowed|okay|deposit)|\\s?no dogs?|\\s?dogs
not allowed\\s?", ignore_case = T))
    dog_yes = str_detect(pets_sent, regex("dogs?\\s(allowed|okay|deposit)|\\s?no cats?|\\s?cats
not allowed\\s?", ignore_case = T))
    no_pets = str_detect(pets_sent, regex("\\s?no pets?|\\s?no pets? allowed\\s?|\\s?pets not
allowed\\s|\\s?pets? prohibited\\s", ignore_case = T))
    if (both_yes == T) {
      status = "both"
      return(status)}
    if (cat_yes == T){
      status = "cats"
      return(status)}
    if (dog_yes == T) {
      status = "dogs"
      return(status)}
    if (no_pets == T) {
      status = "none"
      return(status)}
    return(status)
  }
  return(status)
}

#testing function as i build it
old_regex = "^.*\\b( pets?|animals?|dogs?|cats?.)\\b.*$"

test_str = "-Dogs and Cats Welcome (select apartments). "
test_patt = regex("(pets?\\s|animals?\\s|dogs? and cats?\\s|cats? and dogs?)\\s( friendly |
allowed | yes | welcome .*)", ignore_case = T)

str_extract(tester, regex("(pets?\\s?|animals?\\s?|dogs? and cats?\\s?|cats? and dogs?)\\s?(
friendly | allowed | yes | welcome .*)", ignore_case = T))

tester = posts_df$text[800]
str_detect(test_str, test_patt)
posts_df$text[8000]
parse_pets(tester)
debug(parse_pets)

# use lapply to build a list matching the DF of categories

pet_policy = lapply(posts_df$text, parse_pets)
pet_policy_2 = unlist(pet_policy)
table(pet_policy_2) # it worked! thank god!
```

```
# here is where i append the categories for pet policy back onto the original data frame
posts_df$pet_policy = pet_policy_2

posts_df$pet_policy
# extracting pet deposit

find_pet_deposit = function (text) {
  deposit_num = NA
  pet_dep_after = str_match(text, regex("(\\s?pets?\\s?|\\s?cats?\\s?|\\s?dogs?\\s?)[A-Za-z
:]+(deposit)[A-Za-z :]+[$]([0-9]+)", ignore_case = T))
  pet_dep_before = str_match(text, regex("[$]([0-9]+)[A-Za-z
:]+(\\s?pets?\\s?|\\s?cats?\\s?|\\s?dogs?\\s?)[A-Za-z :]+(deposit)", ignore_case = T))
  if (!is.na(pet_dep_before[1])){
    deposit_num = as.numeric(pet_dep_before[2])
    return(deposit_num)
  }
  if (!is.na(pet_dep_after[1])){
    deposit_num = as.numeric(pet_dep_after[4])
    return(deposit_num)
  }
  return(deposit_num)
}

#testing this function
find_pet_deposit(posts_df$text[224])
posts_df$text[224]
debug(find_pet_deposit)
str_detect(posts_df$text, "pet deposit")



str_901 = "$500 per pet deposit"

str_match(str_901, regex("[$]([0-9]+)[A-Za-z :]+(\\s?pets?\\s?|\\s?cats?\\s?|\\s?dogs?\\s?)[A-Za-
z :]+(deposit)", ignore_case = T))

str_match_all(str_ex, regex("\\s?pets?\\s?| cats?\\s?| dogs?\\s?(\\$([0-
9,]+))?\\s?deposit//s?(:|\\s)?\\$([0-9,]+)", ignore_case = T))
str_match("cat deposit $500", regex("(pets?\\s?| cats?\\s?| dogs?\\s?)(deposit)[A-Za-z :]+[$]([0-
9]+)", ignore_case = T))

pet_dep = str_match("deposit pet $500 pet deposit", regex("[$]([0-9]+)( pets?\\s?| cats?\\s?|
dogs?\\s?)(deposit)", ignore_case = T))
```

```
as.numeric(pet_dep[2])
str_extract(posts_df$text[992], str_ex)

pet_dep_after = str_match("pet $500", regex("(\\bpets?\\s?|\\bcats?\\s?|\\bdogs?\\s?)(deposit)[A-Za-z :]+[$]([0-9]+)", ignore_case = T))
pet_dep_after
posts_df$text[992]
str_ex = "Dogs and Cats are welcome with a $500 pet deposit per pet and a monthly pet rent of $50. "


# lapply over data set as before
pet_deposit = lapply(posts_df$text, find_pet_deposit)
pet_deposit_2 = unlist(pet_deposit)
posts_df$pet_deposit = pet_deposit_2
table(pet_deposit_2)

#how many pet deposits did i isolate? 2108 numbers
how_many_did_i_get = posts_df[!is.na(pet_deposit_2),]
nrow(how_many_did_i_get)


#graphing pet deposit information

# first: between the dogs, cats, none and both, which are more likely to have a pet deposit?

posts_df$detect_deposit = is.na(posts_df$pet_deposit)
detect_dep_df = as.data.frame(table(posts_df$detect_deposit, posts_df$pet_policy))

detect_dep_df_split = split(detect_dep_df, detect_dep_df$Var2)

#getting sums for all categories, prob could have used lapply here?
detect_dep_df_split$both$sum = sum(detect_dep_df_split$both$Freq)
detect_dep_df_split$dogs$sum = sum(detect_dep_df_split$dogs$Freq)
detect_dep_df_split$cats$sum = sum(detect_dep_df_split$cats$Freq)
detect_dep_df_split$none$sum = sum(detect_dep_df_split$none$Freq)

#getting props for all categories
detect_dep_df_split$both$new = detect_dep_df_split$both$Freq / detect_dep_df_split$both$sum
detect_dep_df_split$dogs$new = detect_dep_df_split$dogs$Freq / detect_dep_df_split$dogs$sum
```

```
detect_dep_df_split$cats$new = detect_dep_df_split$cats$Freq /
detect_dep_df_split$cats$sum
detect_dep_df_split$none$new = detect_dep_df_split$none$Freq /
detect_dep_df_split$none$sum
detect_dep_df_split


detect_dep = as.data.frame(do.call(rbind, detect_dep_df_split))
detect_dep
ggplot(detect_dep, aes(x = Var2, y =new, fill = factor(Var1, labels = c("Pet Deposit Provided",
"No Pet Deposit Info")))) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "Apartment Posts with a Pet Deposit by Pet Policy", x = "Pet Policy", y =
"Percentage of Posts with Pet Deposit Information", fill = "Pet Deposit?")+
  scale_fill_viridis_d() +
  geom_text(aes(label=sprintf("%0.2f", round(new, digits = 2))),
position=position_dodge(width=0.9), vjust=-0.25)

# now, how to the actual deposit prices differ between the categories

price_by_cat = aggregate(posts_df$pet_deposit, by=list(Category=posts_df$pet_policy),
FUN=sum, na.rm = T)
price_by_cat = as.data.frame(price_by_cat)
price_by_cat$total = (table(posts_df$pet_policy))
price_by_cat = as.data.frame(price_by_cat)
price_by_cat
price_by_cat$avg_dep_price = price_by_cat$x / price_by_cat$total
price_by_cat

ggplot(price_by_cat, aes(x = Category, y = avg_dep_price, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Pet Deposit Prices By Pet Policy", x = "Pet Policy", y = "Average Pet Deposit
Price") +
  scale_fill_viridis_d() +
  geom_text(aes(label=sprintf("%0.2f", round(avg_dep_price, digits = 2))),
position=position_dodge(width=0.9), vjust=-0.25)

ggplot(posts_df[!is.na(posts_df$pet_policy),], aes(x = pet_policy, y = pet_deposit )) +
  geom_boxplot() + ylim(0,800) +
  labs(title = "Pet Deposit Prices By Pet Policy", x = "Pet Policy", y = "Pet Deposit Price")

table(posts_df$pet_policy)

new = posts_df[is.na(posts_df$pet_policy),]
```

```
table(new$pet_deposit)

# unusual pets

lol = str_detect(posts_df$text, regex("rabbit", ignore_case = TRUE))
rabbits = posts_df[lol,]
rabbits$text


#Question 7

# function to extract air conditioning

has_ac = function(text) {
  ac_status = NA
  ac_yes = str_extract(text, regex("\\s?air conditioning\\s?|\\s?central air\\s?|\\s?AC\\s?",
ignore_case = T))
  ac_no = str_extract(text, regex("\\s?no AC\\s?|\\s?no central air\\s?|\\s?no air
conditioning\\s?"))
  if (!is.na(ac_yes)) {
    ac_status = "AC"
  }
  if (!is.na(ac_no)){
    ac_status = "No AC"
  }
  return(ac_status)
}

# testing the function
table(str_detect(posts_df$text, "\\s?no AC\\s?|\\s?no central air\\s?|\\s?no air conditioning\\s?"))

ac_yes = str_detect(posts_df$text, ac_pattern)

str_extract(posts_df$text[865], regex("\\s?air conditioning\\s?|\\s?central air\\s?|\\s?AC\\s?"))
posts_df$text[745]
has_ac(posts_df$text[745])

#lapply over the data set
AC_status = lapply(posts_df$text, has_ac)

AC_status_2 = unlist(AC_status)
posts_df$AC_status = AC_status_2
table(AC_status_2)
```

```
# Now extracting the heating variable

has_heat = function(text) {
  heat_status = NA
  heat_yes = str_extract(text, regex("\\s?heating\\s?|\\s?heater\\s?|\\s?central
heat\\s?|\\s?heat\\s?(^heated pool)", ignore_case = T))
  fire_yes = str_extract(text, regex("\\s?fireplace\\s?|\\s?wood burning stove\\s?", ignore_case =
T))
  neither = str_extract(text, regex("\\s?no fireplace\\s?"))
  if (!is.na(heat_yes)) {
    heat_status = "heater"
  }
  if (!is.na(fire_yes)) {
    heat_status = "fireplace"
  }
  if(!is.na(neither)){
    heat_status = "neither"
  }
  if(!is.na(heat_yes) & !is.na(fire_yes)) {
    heat_status = "both"
  }
  return(heat_status)
}

#testing the heat function
table(str_detect(posts_df$text, "fireplace not"))
posts_df$text[517]

heat_pattern = regex("heating | heater | wood burning stove | fireplace | central heat| heat",
ignore_case = T)
posts_df$heat = str_extract_all(posts_df$text, heat_pattern)

# now use lapply to extract

heat_status = lapply(posts_df$text, has_heat)

heat_status_2 = unlist(heat_status)
posts_df$heat_status = heat_status_2
table(heat_status_2)

# now the graphs: is air conditioning more common than heating?

comparing_AC_Heat = as.data.frame(table(posts_df$heat_status, posts_df$AC_status))
AC_table =table(posts_df$heat_status, posts_df$AC_status)
```

```
AC_table

#heat graph
ggplot(posts_df[!is.na(heat_status),], aes(x = heat_status, fill = heat_status)) +
  geom_bar(position = position_dodge()) +
  geom_text(stat = "count",aes(label=..count..), position=position_dodge(width=0.9), vjust=-0.5)+
  labs(title = "Heat Status Of Posts", x = "Heat Status", y = "Count") +
  scale_fill_viridis_d() + ylim(0,45000) + guides(fill = FALSE) + facet_wrap(region~.)



#AC graph
ggplot(posts_df[!is.na(AC_status),], aes(x = AC_status, fill = AC_status)) +
  geom_bar(position = position_dodge()) +
  geom_text(stat = "count",aes(label=..count..), position=position_dodge(width=0.9), vjust=-0.5)+
  labs(title = "Air Conditioning Status Of Posts", x = "AC Status", y = "Count") +
  scale_fill_viridis_d()+ ylim(0,45000) +guides(fill = FALSE)

# heat & AC together
ggplot(posts_df[!is.na(heat_status) & !is.na(AC_status),], aes(x = heat_status, fill = AC_status))
+
  geom_bar(position = position_dodge()) +
  geom_text(stat = "count",aes(label=..count..), position=position_dodge(width=0.9), vjust=-0.5)
+
  facet_wrap(AC_status ~ .) +
  labs(title = "Air Conditioning Status by Heat Status", x = "Heat Status", y = "Count") +
  scale_fill_viridis_d()

ggplot(posts_df[!is.na(heat_status) & !is.na(AC_status),], aes(fill = heat_status, x = AC_status))
+
  geom_bar(position = position_dodge()) +
  geom_text(stat = "count",aes(label=..count..), position=position_dodge(width=0.5), vjust=-0.1)
+
  facet_wrap(heat_status~.)



# heat and AC by location:
ggplot(posts_df[!is.na(heat_status),], aes(x = heat_status, fill = heat_status)) +
  geom_bar(position = position_dodge()) +
  geom_text(stat = "count",aes(label=..count..), position=position_dodge(width=0.5), vjust=-
0.24)+
  labs(title = "Heat Status Of Posts", x = "Heat Status", y = "Count") +
  scale_fill_viridis_d() + ylim(0,1500) + guides(fill = FALSE) + facet_wrap(region~.)
```

```
ggplot(posts_df[!is.na(AC_status),], aes(x = AC_status, fill = AC_status)) +
  geom_bar(position = position_dodge()) +
  geom_text(stat = "count",aes(label=..count..), position=position_dodge(width=0.9), vjust=-0.5)+
  labs(title = "Air Conditioning Status Of Posts", x = "AC Status", y = "Count") +
  scale_fill_viridis_d()+guides(fill = FALSE) + facet_wrap(region~.) +
  ylim(0, 8000)

ggplot(posts_df[!is.na(heat_status) & !is.na(AC_status) ,], aes(fill = heat_status, x = AC_status))
+
  geom_bar(position = position_dodge()) +
  facet_wrap(region~.) + labs(title = "AC and Heat Status of Apartments By Craigslist Region", x
= "AC Status", y = "Count", fill = "Heat Status") +
  scale_fill_viridis_d()

#Question 8

#detecting email addresses:

#regular expression did not detect emails
str_detect(posts_df$text, regex("\\w+\\@\\.w{3}", ignore_case = T))

#detecting an @ sign sandwiched between two word characters - 308 of them
table(str_detect(posts_df$text, regex("\\w\\@\\w", ignore_case = T)))
the_at = str_detect(posts_df$text, regex("\\w\\@\\w", ignore_case = T))

whats_the_at = str_extract(posts_df$text, regex("[A-Za-z]{4}\\@\\w", ignore_case = T))

whats_the_at[!is.na(whats_the_at)]

table(whats_the_at)
whats_at = posts_df[whats_the_AT,]

#detecting phone numbers:

table(str_detect(posts_df$text, regex("\\(?\\d{3}\\)?[.-]? *\\d{3}[.-]? *[.-]?\\d{4}", ignore_case = T)))

has_phone_subset = posts_df[has_phone_num,]
has_phone_subset$text
# which posts contain "show contact info"
hide_nums = str_detect(posts_df$text, regex("show contact info", ignore_case = T))
table(hide_nums)

hidden_nums = posts_df[hide_nums,]
```