6 Boston Housing

Xuhui Ying

08/18/2022

Background

You have been hired by the tax authority of the City of Boston to asses Tax Assessments. Your task is to create a model to predict the av_total (assessed value) of properties in the greater Boston area.

READ THIS

keep your code organized, I'm not giving you required steps you need to figure out how to build a regression model, explore the data, partition the data etc. this is just an outline of how i'd approach this problem, you can choose to do something different. this needs to be your own work!!

Libraries

load your libraries

```
options(warn = -1)
library(ggplot2)
library(corrplot)
library(MASS)
library(skimr)
library(readr)
library(dplyr)
library(tidyverse)
library(janitor)
library(statsr)
library(PerformanceAnalytics)
library(modelr)
library(broom)
library(reshape2)

options(scipen = 999) # turns off scientific notation
```

Import

boston.csv zips.csv

I'd use clean names but it's up to you…

```
boston <- read_csv("boston.csv") %>% clean_names()

head(boston)
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1812296230	2136	Y	321200	10288	1992	0	1681	1.0	R
2006736000	2132	Y	894400	10148	1900	2016	3024	2.5	R
2009442000	2132	Y	387700	8512	1920	0	1160	2.0	R
1704953000	2124	Y	450500	3187	1900	2001	1868	2.0	R
1812269079	2136	Y	325100	10088	1971	1975	1534	1.0	R
2011986000	2132	Y	408800	3500	1960	1987	1632	2.0	R

6 rows | 1-10 of 28 columns

```
zips <- read_csv("zips.csv") %>% clean_names()

head(zips)
```

zip	population	pop_density	median_income	city_state
<chr>	<dbl>	<dbl>	<dbl>	<chr>
02132	36314	13251	75446	Cambridge, MA
02124	47783	15913	48841	Dorchester Center, MA
02131	29826	11505	66735	Roslindale, MA
02136	28488	6207	58890	Hyde Park, MA
02130	35401	10618	75730	Jamaica Plain, MA
02138	36314	13251	75446	Cambridge, MA

6 rows

Explore AV_TOTAL

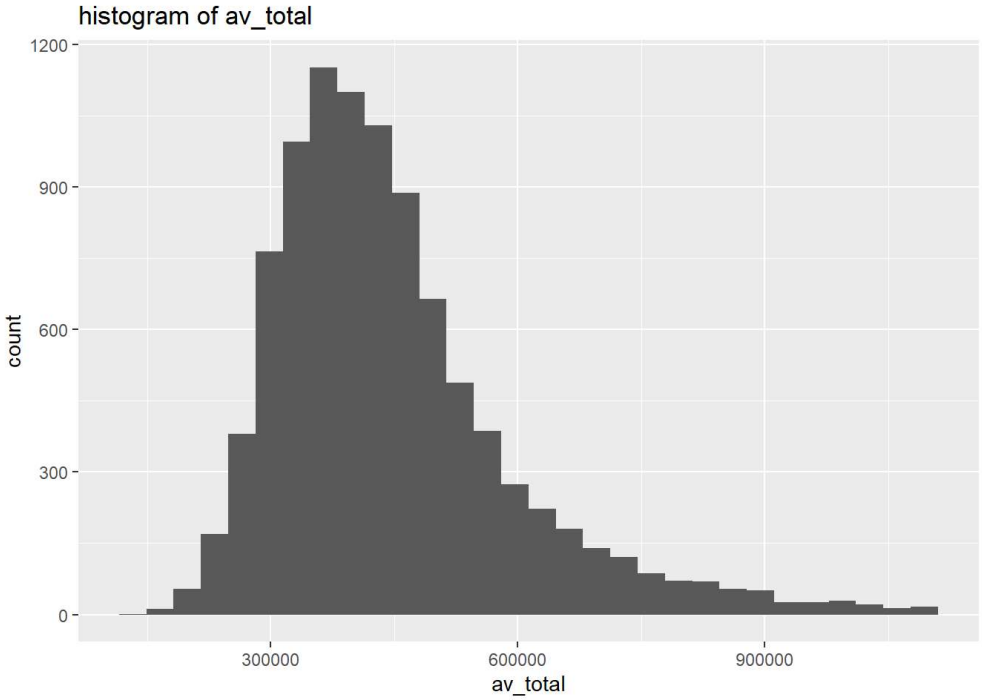
what's the average av_total?

1. make a histogram of av_total
2. make a box plot of av_total

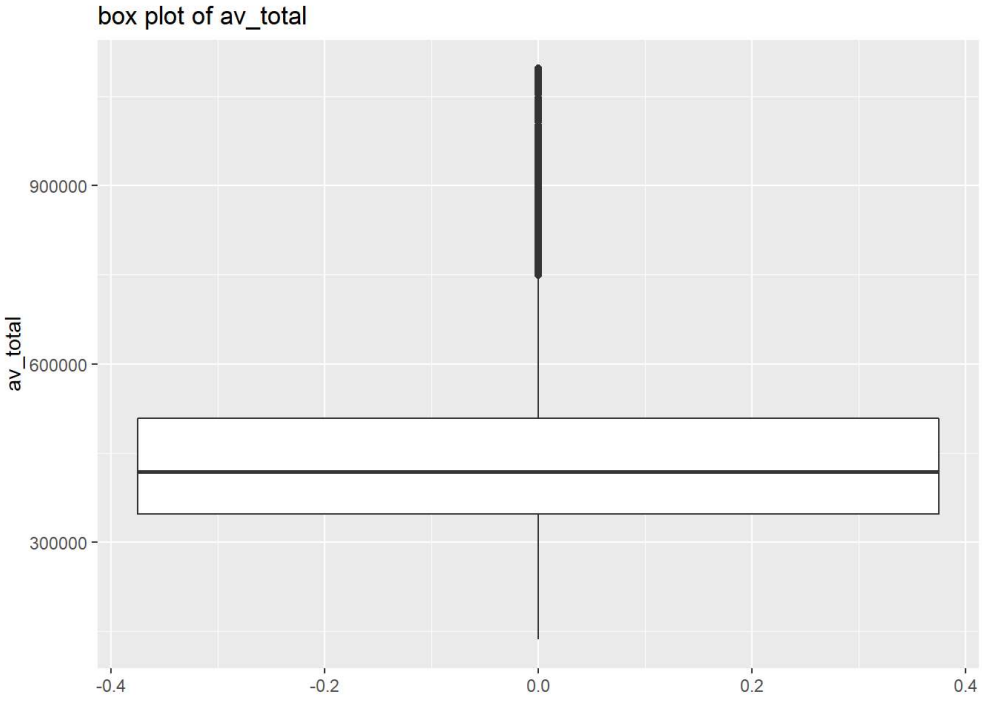
```
#sprintf("the average of av_total = %f", mean(boston$av_total, na.rm = T))
#sprintf("the median of av_total = %f", median(boston$av_total, na.rm = T))

summary(boston$av_total)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  136100  347750  418000  447391  507850 1097100
```

```
boston %>%
  ggplot(aes(x=av_total)) +
  geom_histogram() +
  labs(title = "histogram of av_total",
       xlab = "av_total",
       ylab = "count")
```




```
boston %>%
  ggplot(aes(av_total)) +
  geom_boxplot() +
  coord_flip() +
  labs(title = "box plot of av_total",
    x = "av_total")
```



Transform

there are a number of helpful transformations you can make but here is what i'd minamaly do:

1. join boston to zips on zipcode = zip,

- note zip is character you'll need to convert it to an integer.

2. create a home age variable using some form of logic

- IF yr_remod > yr_built THEN age = 2020 - yr_remod
- ELSE age = 2020 - yr_built

```
zips$zip <- as.numeric(zips$zip)

data <- boston %>%
  inner_join(zips, by = c("zipcode"="zip")) %>%
  mutate(home_age = if_else(yr_remod > yr_built, 2020-yr_remod, 2020-yr_built))

head(data)
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1812296230	2136	Y	321200	10288	1992	0	1681	1.0	R
2006736000	2132	Y	894400	10148	1900	2016	3024	2.5	R
2009442000	2132	Y	387700	8512	1920	0	1160	2.0	R
1704953000	2124	Y	450500	3187	1900	2001	1868	2.0	R
1812269079	2136	Y	325100	10088	1971	1975	1534	1.0	R
2011986000	2132	Y	408800	3500	1960	1987	1632	2.0	R

6 rows | 1-10 of 33 columns

```
train_prep <- data

head(train_prep)
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1812296230	2136	Y	321200	10288	1992	0	1681	1.0	R
2006736000	2132	Y	894400	10148	1900	2016	3024	2.5	R
2009442000	2132	Y	387700	8512	1920	0	1160	2.0	R
1704953000	2124	Y	450500	3187	1900	2001	1868	2.0	R
1812269079	2136	Y	325100	10088	1971	1975	1534	1.0	R
2011986000	2132	Y	408800	3500	1960	1987	1632	2.0	R

6 rows | 1-10 of 33 columns

```
test_prep <- data

head(test_prep)
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1812296230	2136	Y	321200	10288	1992	0	1681	1.0	R
2006736000	2132	Y	894400	10148	1900	2016	3024	2.5	R
2009442000	2132	Y	387700	8512	1920	0	1160	2.0	R
1704953000	2124	Y	450500	3187	1900	2001	1868	2.0	R
1812269079	2136	Y	325100	10088	1971	1975	1534	1.0	R
2011986000	2132	Y	408800	3500	1960	1987	1632	2.0	R

6 rows | 1-10 of 33 columns

Explore Categorical Variables

I'd do some kind of null analysis and frequency analysis, what variables can I exclude?

```
# Remove columns with lots of nulls
null_count <- function(c){
  # function to calculate the null count per column
  sum(is.na(c))
}

# create a table of column values by count
res_00 <- data %>%
  summarize(across(1:as.numeric(ncol(data)),null_count)) %>%
  pivot_longer(cols=1:as.numeric(ncol(data)), names_to ="column", values_to="null_count") %>%
  mutate(null_pct = null_count / nrow(data))

res_00
```

column <chr>	null_count <int>	null_pct <dbl>
pid	0	0.0000000000
zipcode	0	0.0000000000
own_occ	0	0.0000000000
av_total	0	0.0000000000
land_sf	3	0.0003163556
yr_built	0	0.0000000000
yr_remod	349	0.0368026996
living_area	0	0.0000000000
num_floors	0	0.0000000000
structure_class	0	0.0000000000

1-10 of 33 rows Previous 1 2 3 4 Next

```
# make a table of columns to drop
drop_cols <- res_00 %>%
  filter(null_pct > 0.99) %>%
  dplyr::select(1)

drop_cols # 0×1的dataset
```

0 rows

```
# filter out the junk columns
data_filtered <- data %>%
  dplyr::select(!drop_cols$column)

#data_filtered

data_filtered %>% skim()
```

Data summary

Name	Piped data
Number of rows	9483

Number of columns	33
_____	
Column type frequency:	
character	15
numeric	18
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
own_occ	0	1	1	1	0	2	0
structure_class	0	1	1	1	0	1	0
r_bldg_styl	0	1	2	2	0	16	0
r_roof_typ	0	1	1	1	0	7	0
r_ext_fin	0	1	1	1	0	11	0
r_bth_style	0	1	1	1	0	4	0
r_kitch_style	0	1	1	1	0	4	0
r_heat_typ	0	1	1	1	0	7	0
r_ac	0	1	1	1	0	3	0
r_ext_cnd	0	1	1	1	0	5	0
r_ovrall_cnd	0	1	1	1	0	4	0
r_int_cnd	0	1	1	1	0	5	0
r_int_fin	0	1	1	1	0	3	0
r_view	0	1	1	1	0	5	0
city_state	0	1	13	21	0	5	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
pid	0	1.00	1851668065.43	188346205.10	702991020	1805562500.00	1900547000.0	2004602500	2012270020	
zipcode	0	1.00	2131.40	3.71	2124	2131.00	2132.0	2132	2136	
av_total	0	1.00	447390.95	146367.23	136100	347750.00	418000.0	507850	1097100	
land_sf	3	1.00	5901.87	2701.33	864	4309.75	5297.5	6703	49350	
yr_built	0	1.00	1933.77	29.80	1710	1910.00	1933.0	1955	2016	
yr_remod	349	0.96	668.51	944.03	0	0.00	0.0	1997	2016	
living_area	0	1.00	1656.05	547.62	403	1297.00	1550.0	1903	8623	
num_floors	0	1.00	1.72	0.45	1	1.50	2.0	2	3	
r_total_rms	0	1.00	7.08	1.55	3	6.00	7.0	8	17	
r_bdrms	0	1.00	3.33	0.92	1	3.00	3.0	4	9	
r_full_bth	0	1.00	1.34	0.55	1	1.00	1.0	2	6	
r_half_bth	0	1.00	0.56	0.53	0	0.00	1.0	1	3	
r_kitch	0	1.00	1.02	0.12	1	1.00	1.0	1	3	

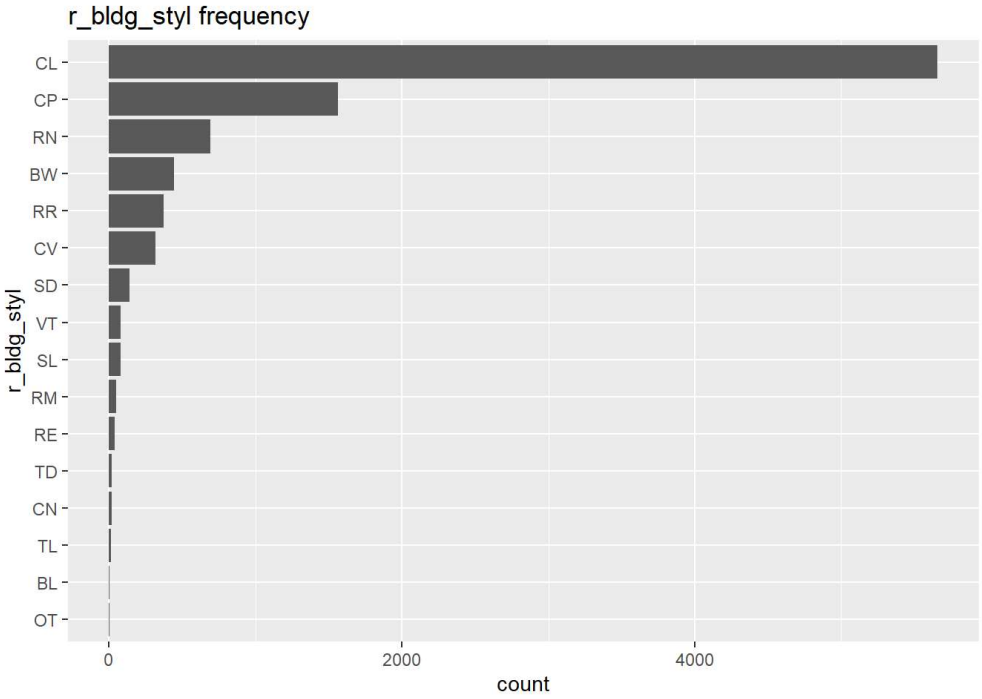
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
r_fplace	0	1.00	0.58	0.62	0	0.00	1.0	1	5	
population	0	1.00	34908.54	6357.25	28488	29826.00	35401.0	36314	47783	
pop_density	0	1.00	11389.22	3294.12	6207	10618.00	11505.0	13251	15913	
median_income	0	1.00	65865.77	9777.06	48841	58890.00	66735.0	75446	75730	
home_age	349	0.96	62.90	39.72	4	23.00	65.0	95	310	

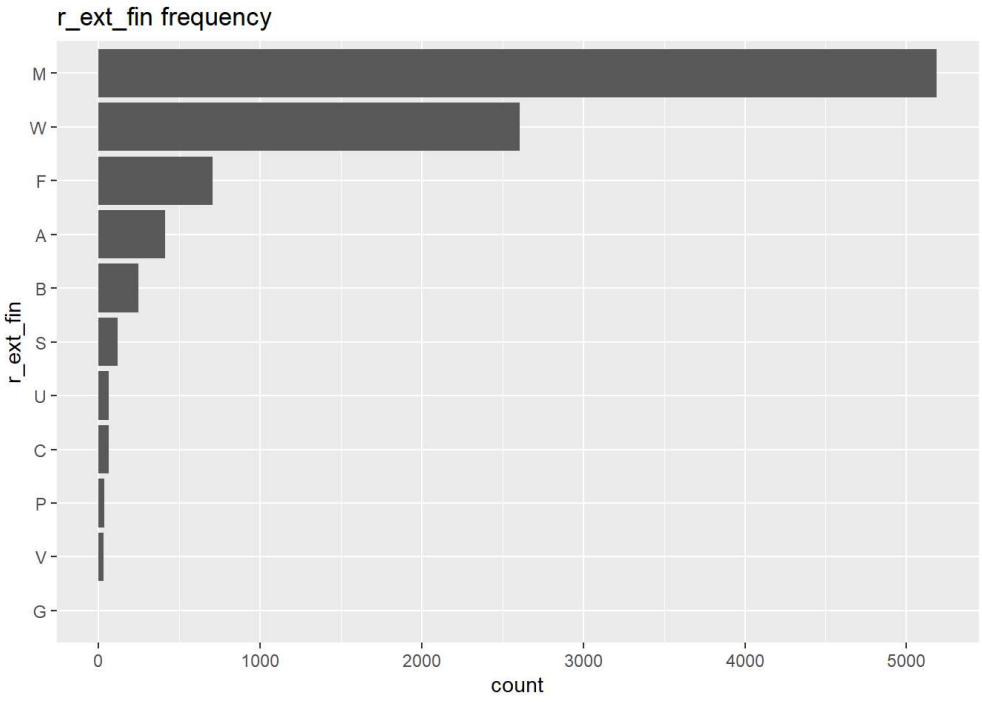
```
# Frequency Analysis
freq_long <- data_filtered %>%
  pivot_longer(cols = is.character, names_to = "column", values_to = "value") %>%
  dplyr::select(column, value) %>%
  group_by(column) %>%
  summarise(count = n(),
            n_miss = sum(is.na(value)),
            n_distinct = n_distinct(value)) %>%
  mutate(count = count - n_miss, pct_distinct = round(n_distinct / count, digits = 4)) %>%
  arrange(-pct_distinct)

freq_long
```

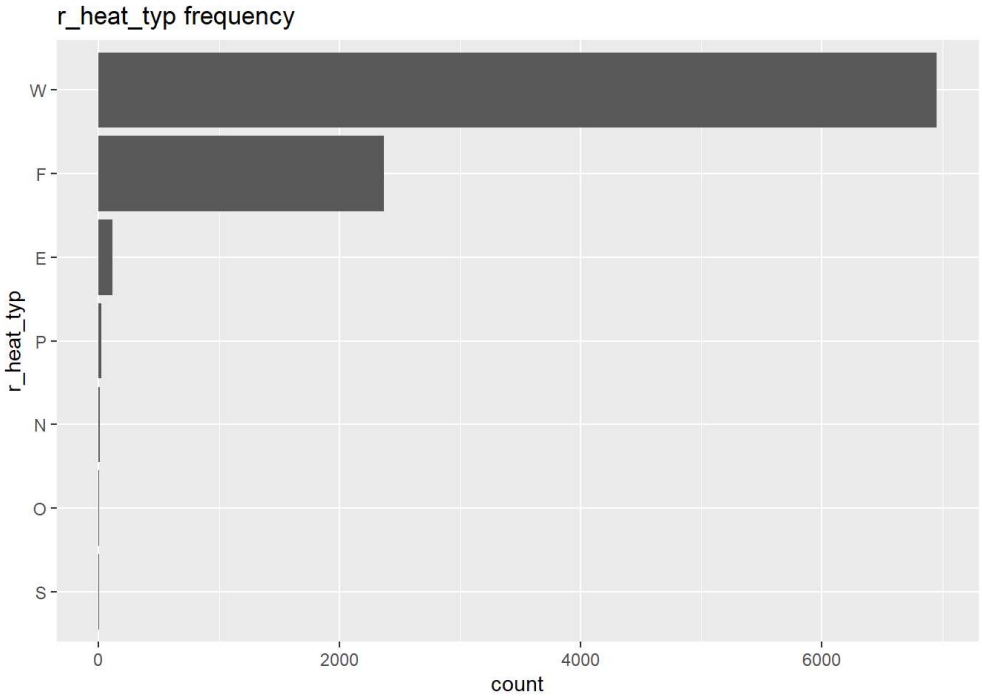
column	count	n_miss	n_distinct	pct_distinct
<chr>	<int>	<int>	<int>	<dbl>
r_bldg_styl	9483	0	16	0.0017
r_ext_fin	9483	0	11	0.0012
r_heat_typ	9483	0	7	0.0007
r_roof_typ	9483	0	7	0.0007
city_state	9483	0	5	0.0005
r_ext_cnd	9483	0	5	0.0005
r_int_cnd	9483	0	5	0.0005
r_view	9483	0	5	0.0005
r_bth_style	9483	0	4	0.0004
r_kitch_style	9483	0	4	0.0004

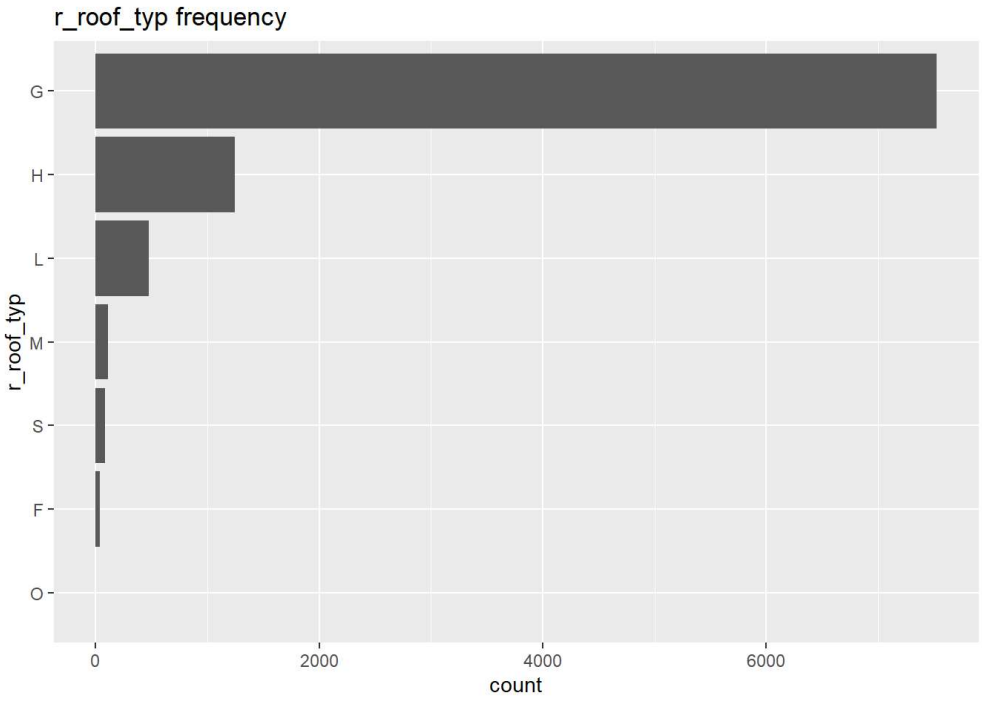
1-10 of 15 rows Previous 1 2 Next

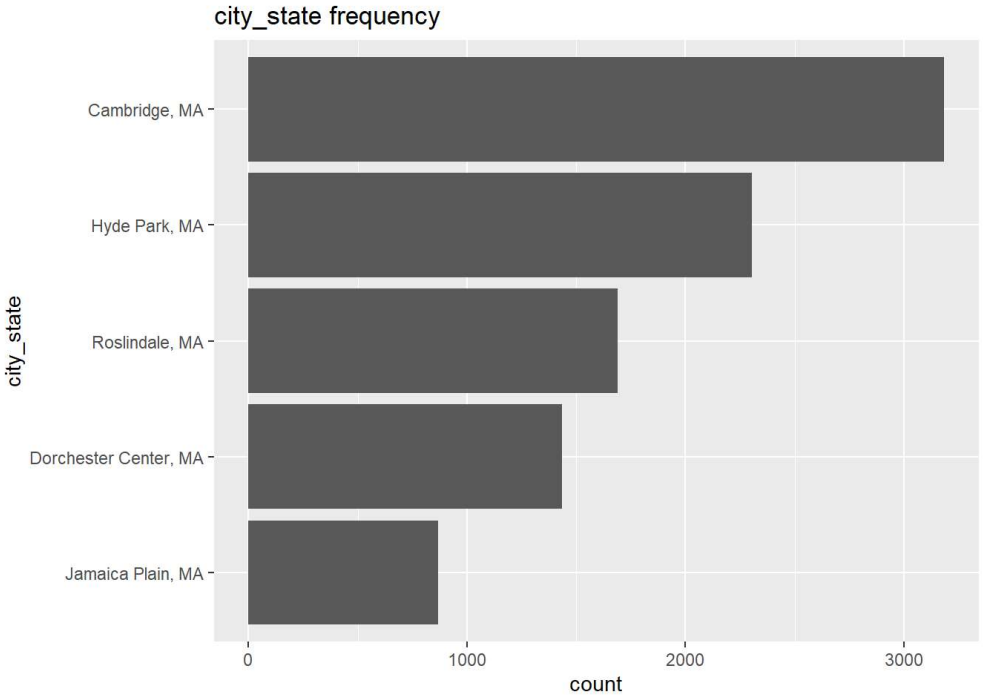
```
numeric_category_cols <- freq_long %>%
  filter(n_distinct < 30)

for (c in numeric_category_cols$column){
#  print(c)
  P <- data_filtered %>%
    count(!!as.name(c), sort=TRUE) %>%
    ggplot(aes(y=reorder(!!as.name(c),n), x=n)) +
    geom_col() +
    labs(title = paste(c,"frequency"), x="count", y=c)
  print(P)
}
```

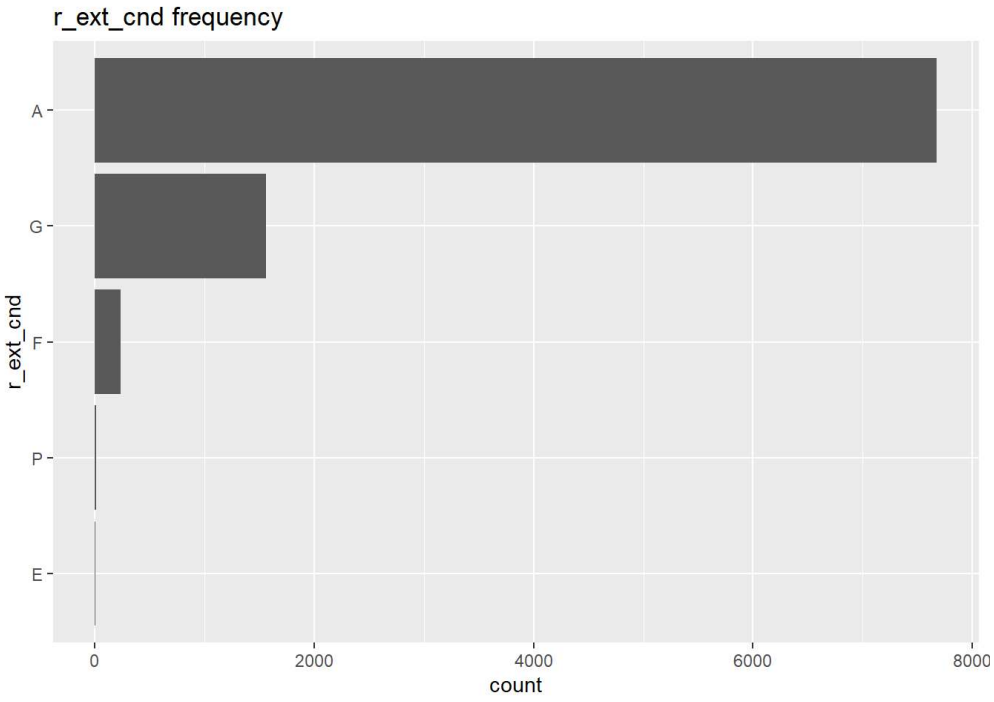

r_bldg_styl frequency
CL
CP
RN
BW
RR
CV
SD
VT
SL
RM
RE
TD
CN
TL
BL
OT
r_bldg_styl
0
2000
4000
count

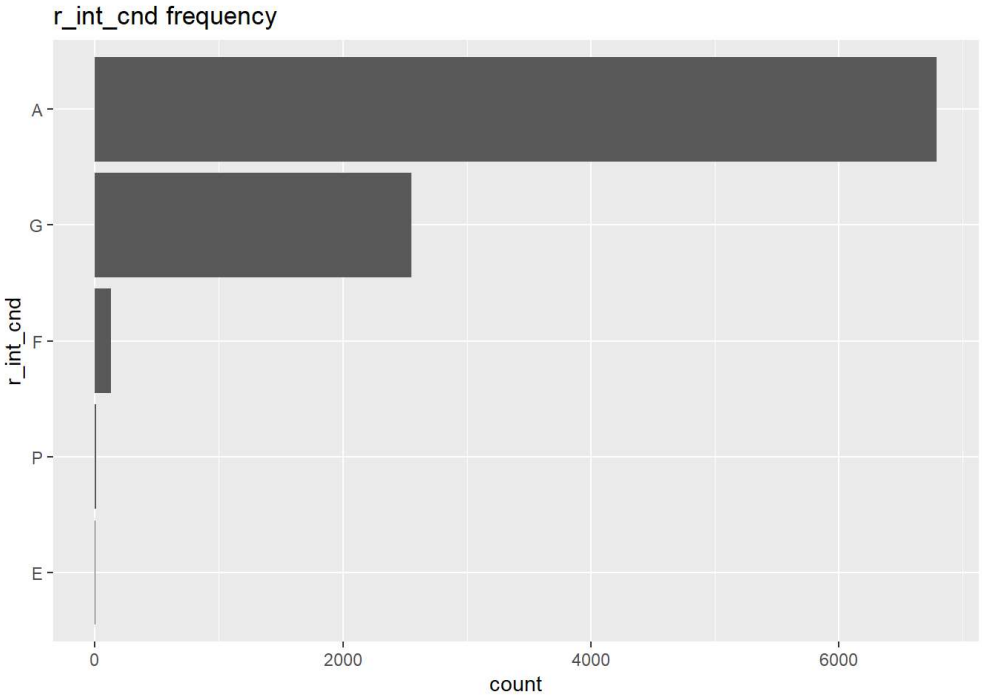

r_ext_fin frequency
M
W
F
A
B
S
U
C
P
V
G
r_ext_fin
0
1000
2000
3000
4000
5000
count

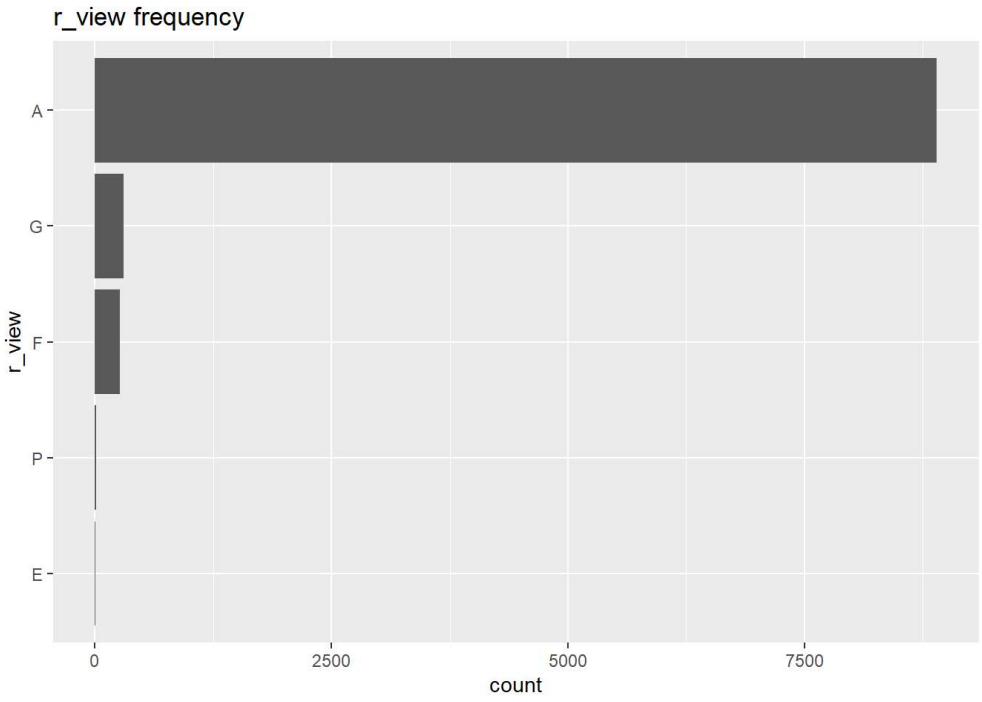

r_heat_typ frequency
W
F
E
P
N
O
S
r_heat_typ
0
2000
4000
6000
count

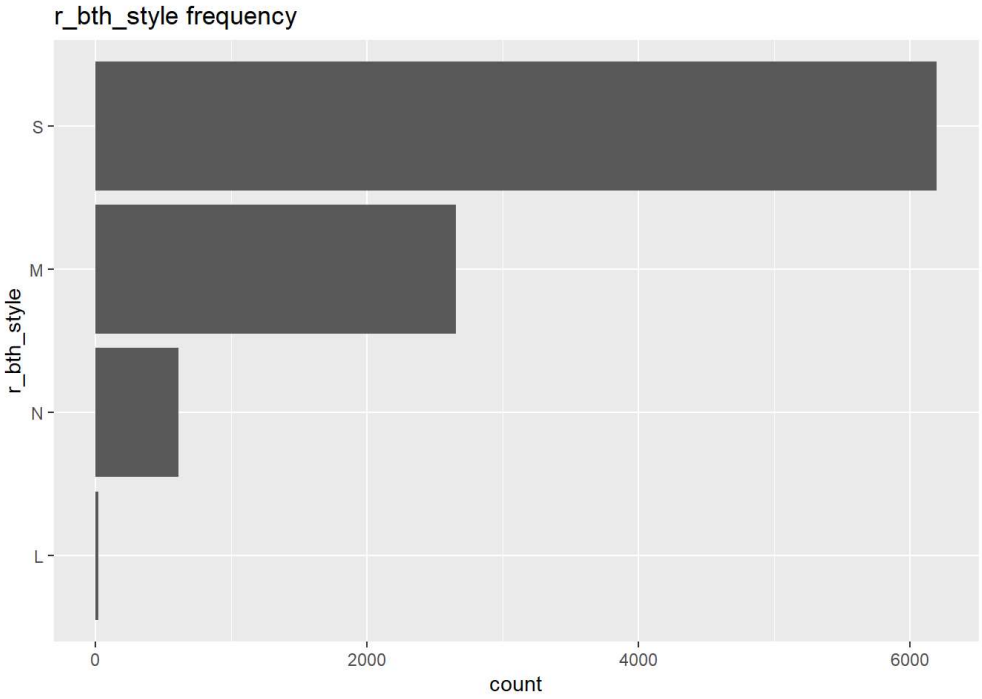

r_roof_typ frequency
G
H
L
M
S
F
O
r_roof_typ
0
2000
4000
6000
count

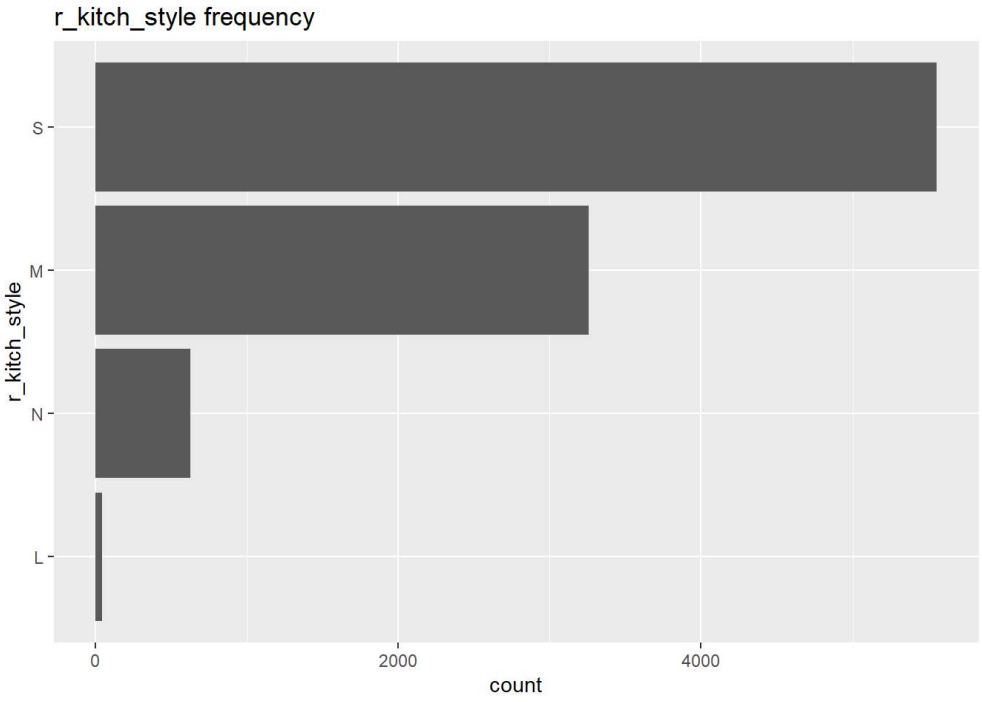

city_state frequency
Cambridge, MA
Hyde Park, MA
Roslindale, MA
Dorchester Center, MA
Jamaica Plain, MA
city_state
0
1000
2000
3000
count

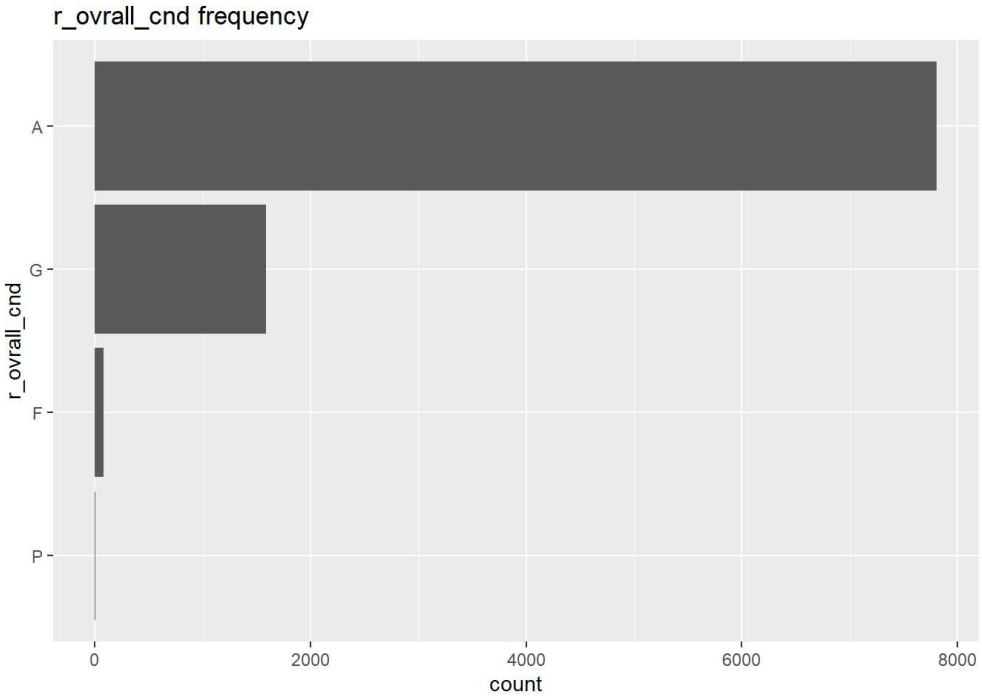

r_ext_cnd frequency
A
G
F
P
E
r_ext_cnd
0
2000
4000
6000
8000
count

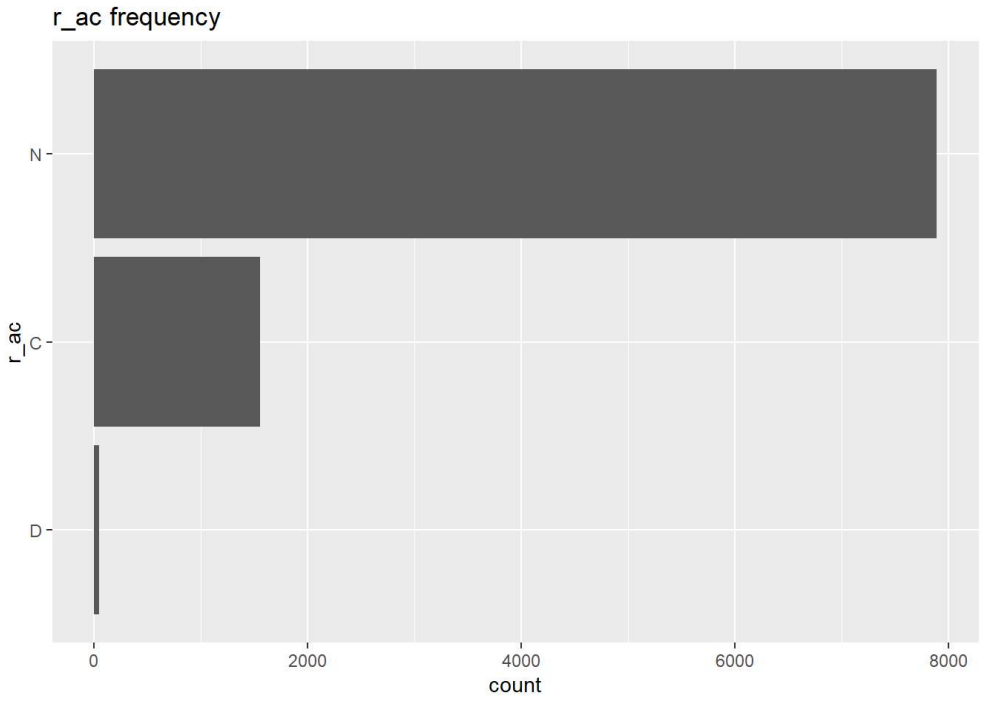

r_int_cnd frequency
A
G
F
P
E
r_int_cnd
0
2000
4000
6000
count

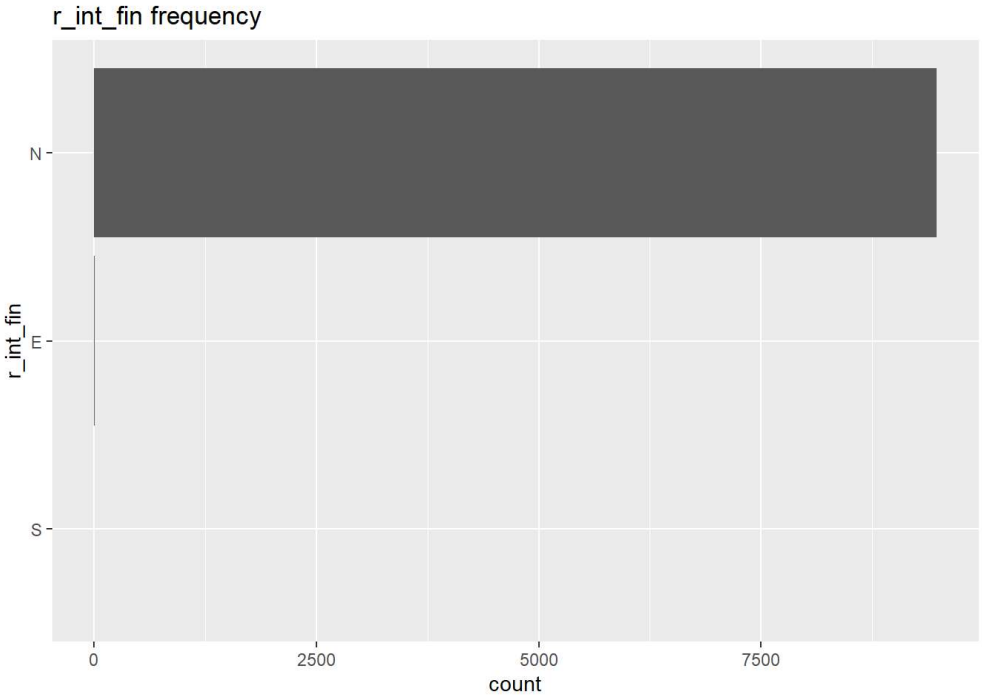

r_view frequency
A
G
F
P
E
r_view
0
2500
5000
7500
count

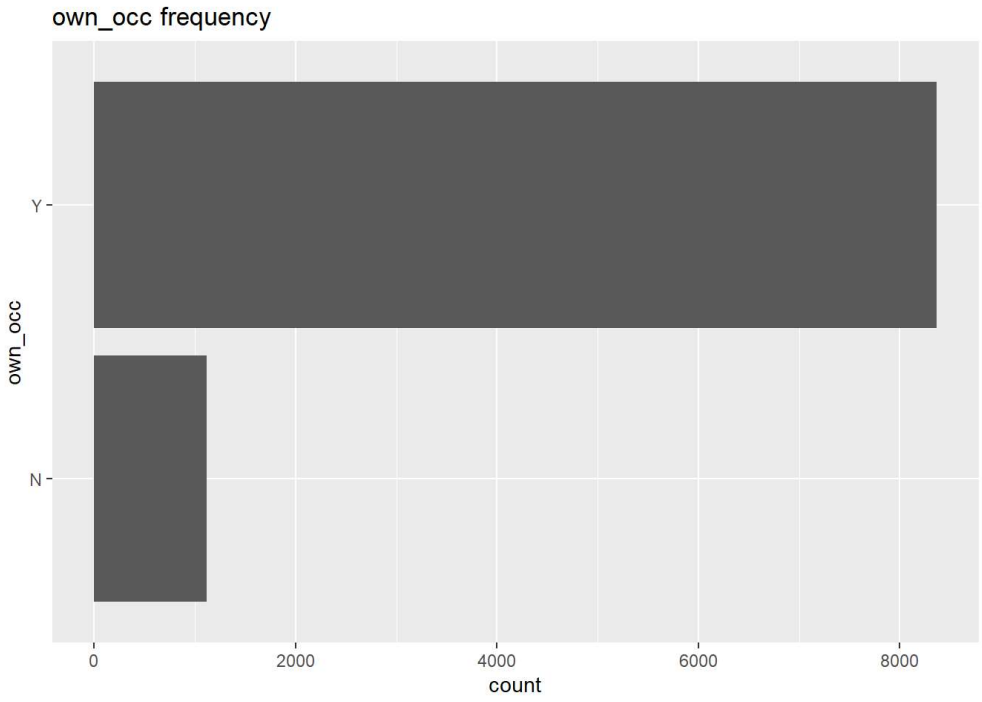

r_bth_style frequency
S
M
N
L
r_bth_style
0
2000
4000
6000
count

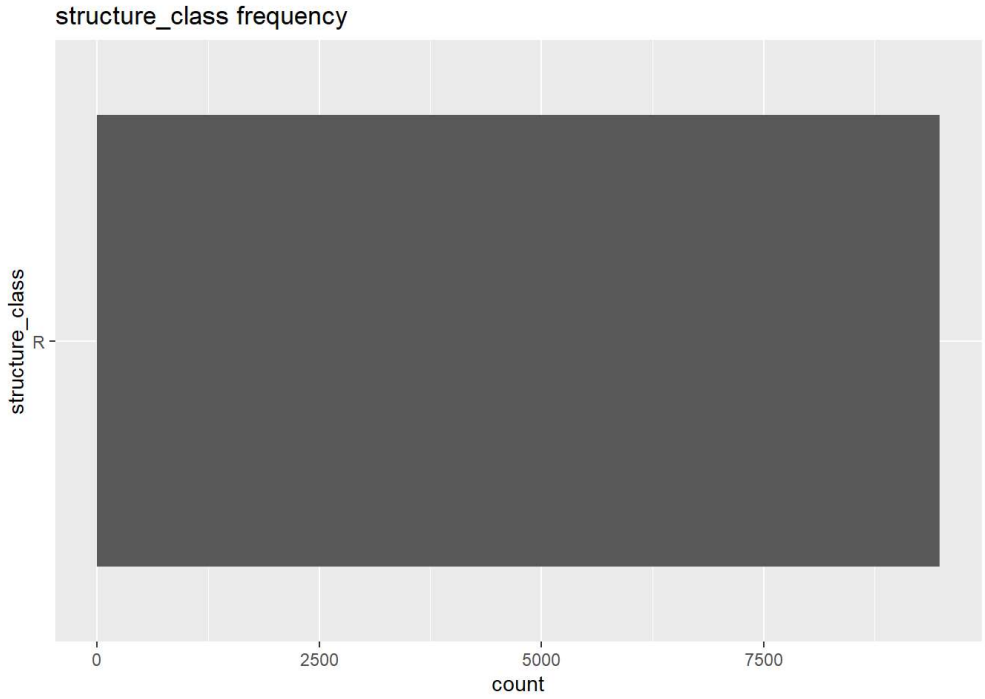

r_kitch_style frequency
S
M
N
L
r_kitch_style
0
2000
4000
count


r_ovrall_cnd frequency
A
G
F
P
r_ovrall_cnd
0
2000
4000
6000
8000
count


r_ac frequency
N
C
D
r_ac
0
2000
4000
6000
8000
count


r_int_fin frequency
N
E
S
r_int_fin
0
2500
5000
7500
count


own_occ frequency
Y
N
own_occ
0
2000
4000
6000
8000
count




```
# drop: structure_class (because n_distinct == 1)
```

Explore Numeric Variables

I'd do some kind of descriptive statistics analysis, what variables can I exclude?

```
data_numeric <- data_filtered %>%
  select_if(is.numeric)

head(data_numeric)
```

pid <dbl>	zipcode <dbl>	av_total <dbl>	land_sf <dbl>	yr_built <dbl>	yr_remod <dbl>	living_area <dbl>	num_floors <dbl>	r_total_rms <dbl>	r_bdrms <dbl>
1812296230	2136	321200	10288	1992	0	1681	1.0	6	3
2006736000	2132	894400	10148	1900	2016	3024	2.5	8	5
2009442000	2132	387700	8512	1920	0	1160	2.0	5	3
1704953000	2124	450500	3187	1900	2001	1868	2.0	7	4
1812269079	2136	325100	10088	1971	1975	1534	1.0	7	3
2011986000	2132	408800	3500	1960	1987	1632	2.0	6	3

6 rows | 1-10 of 18 columns

```
data_numeric %>% skim()
```

Data summary

Name	Piped data
Number of rows	9483
Number of columns	18
_____	
Column type frequency:	
numeric	18

Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
pid	0	1.00	1851668065.43	188346205.10	702991020	1805562500.00	1900547000.0	2004602500	2012270020	
zipcode	0	1.00	2131.40	3.71	2124	2131.00	2132.0	2132	2136	
av_total	0	1.00	447390.95	146367.23	136100	347750.00	418000.0	507850	1097100	
land_sf	3	1.00	5901.87	2701.33	864	4309.75	5297.5	6703	49350	
yr_built	0	1.00	1933.77	29.80	1710	1910.00	1933.0	1955	2016	
yr_remod	349	0.96	668.51	944.03	0	0.00	0.0	1997	2016	
living_area	0	1.00	1656.05	547.62	403	1297.00	1550.0	1903	8623	
num_floors	0	1.00	1.72	0.45	1	1.50	2.0	2	3	
r_total_rms	0	1.00	7.08	1.55	3	6.00	7.0	8	17	
r_bdrms	0	1.00	3.33	0.92	1	3.00	3.0	4	9	
r_full_bth	0	1.00	1.34	0.55	1	1.00	1.0	2	6	
r_half_bth	0	1.00	0.56	0.53	0	0.00	1.0	1	3	
r_kitch	0	1.00	1.02	0.12	1	1.00	1.0	1	3	
r_fplace	0	1.00	0.58	0.62	0	0.00	1.0	1	5	
population	0	1.00	34908.54	6357.25	28488	29826.00	35401.0	36314	47783	
pop_density	0	1.00	11389.22	3294.12	6207	10618.00	11505.0	13251	15913	
median_income	0	1.00	65865.77	9777.06	48841	58890.00	66735.0	75446	75730	
home_age	349	0.96	62.90	39.72	4	23.00	65.0	95	310	

```
descriptive_stats <- data_numeric %>%
  pivot_longer(cols = is.numeric, names_to = "column", values_to="value") %>%
  dplyr::select(column, value) %>%
  group_by(column) %>%
  summarise(count = n(),
            n_miss = sum(is.na(value)),
            n_distinct = n_distinct(value),
            mean = round(mean(value, na.rm = TRUE),2),
            median = median(value, na.rm = TRUE),
            min = min(value, na.rm = TRUE),
            max = max(value, na.rm = TRUE),
            sd = sd(value, na.rm = TRUE),
            )

descriptive_stats
```

column	count	n_miss	n_distinct	mean	median	min	max	sd
<chr>	<int>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
av_total	9483	0	4241	447390.95	418000.0	136100	1097100	146367.2260687
home_age	9483	349	168	62.90	65.0	4	310	39.7175113
land_sf	9483	3	4345	5901.87	5297.5	864	49350	2701.3276055
living_area	9483	0	2091	1656.05	1550.0	403	8623	547.6241388

column	count	n_miss	n_distinct	mean	median	min	max	sd
<chr>	<int>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
median_income	9483	0	5	65865.77	66735.0	48841	75730	9777.0583456
num_floors	9483	0	5	1.72	2.0	1	3	0.4479729
pid	9483	0	9483	1851668065.43	1900547000.0	702991020	2012270020	188346205.0984228
pop_density	9483	0	5	11389.22	11505.0	6207	15913	3294.1159391
population	9483	0	5	34908.54	35401.0	28488	47783	6357.2458841
r_bdrms	9483	0	9	3.33	3.0	1	9	0.9166145

1-10 of 18 rows Previous 1 2 Next

Correlations

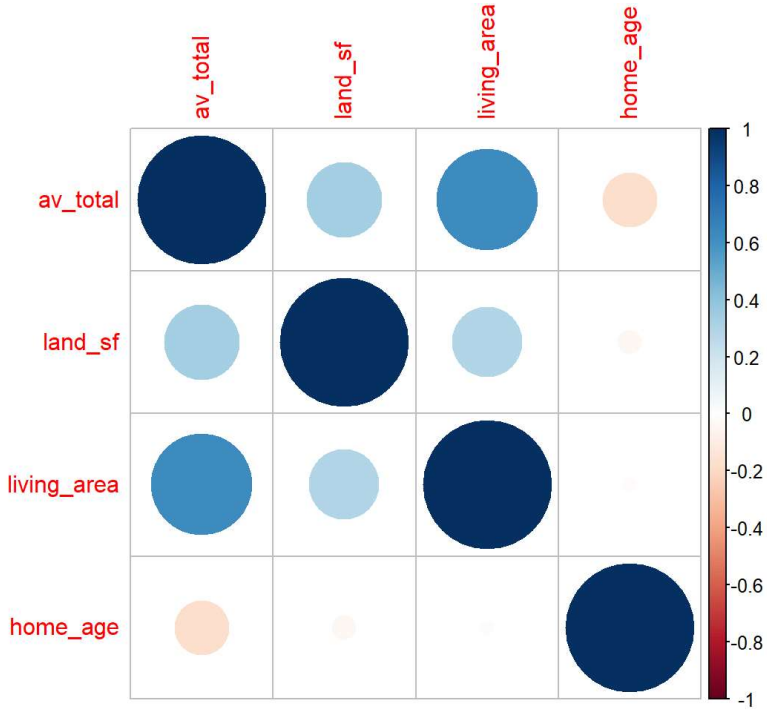
1. create a correlation matrix of key numeric varaibles like: av_total, land_sf, living_area, and age.

hint: you need to deal with missing values

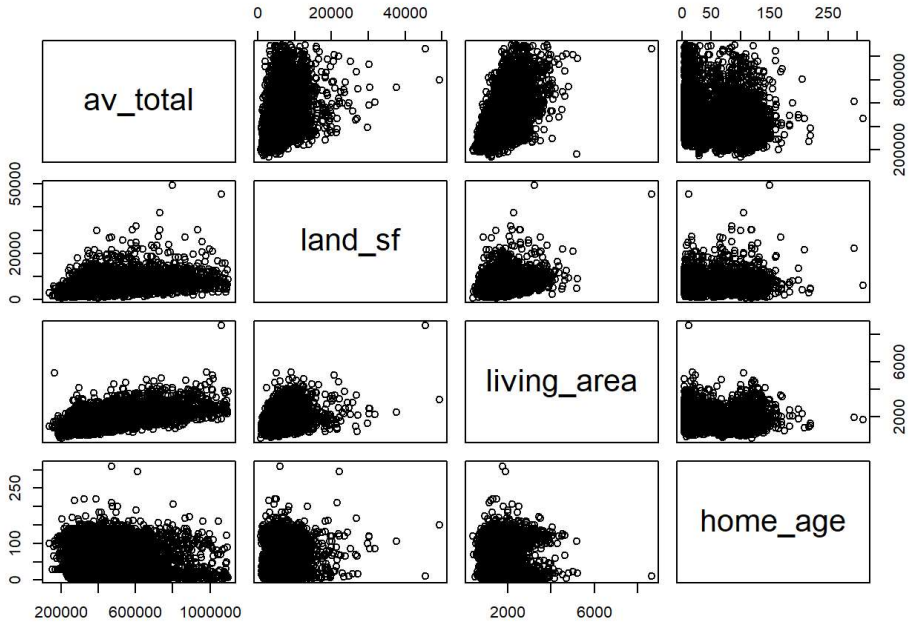
```
cor_analysis <- data %>%
  na.omit() %>%
  dplyr::select(av_total, land_sf, living_area, home_age) %>%
  cor() %>%
  melt() %>% #turn it into a dataframe
  arrange(desc(value))

cor_analysis_1 <- data %>%
  na.omit() %>%
  dplyr::select(av_total, land_sf, living_area, home_age)

cormat <- cor(cor_analysis_1)
round(cormat, 2)
```

```
##             av_total land_sf living_area home_age
## av_total        1.00    0.34        0.62    -0.18
## land_sf         0.34    1.00        0.30    -0.03
## living_area     0.62    0.30        1.00    -0.01
## home_age       -0.18   -0.03       -0.01     1.00
```

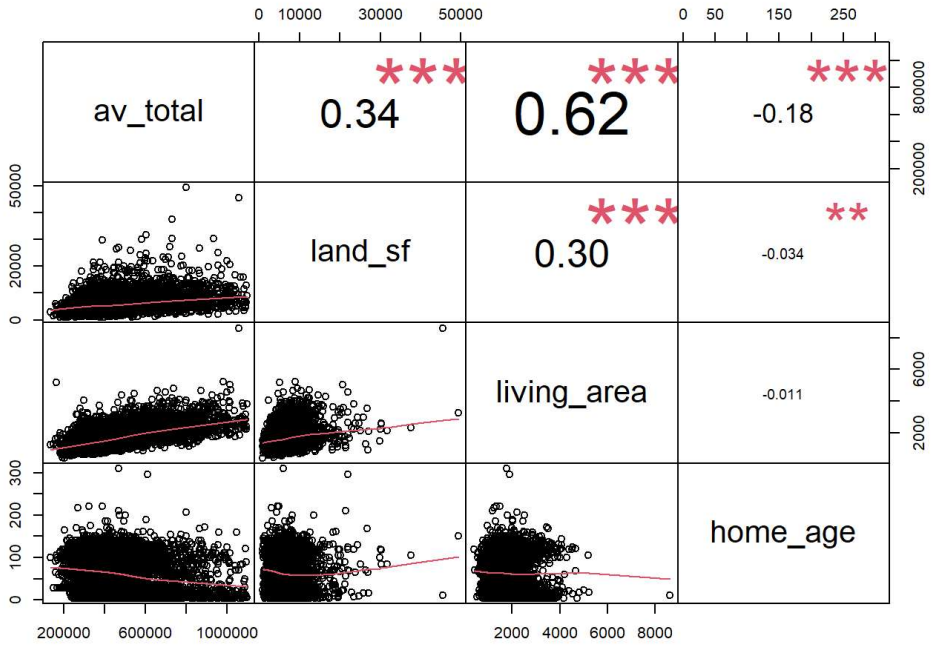
```
corrplot(cormat)
```



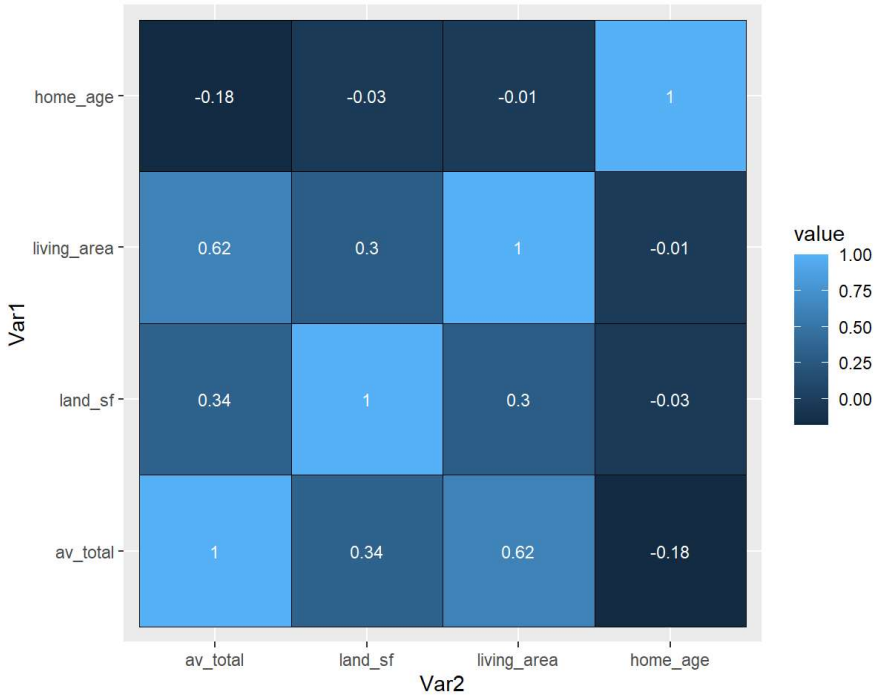

```
pairs(cor_analysis_1[1:4])
```




```
chart.Correlation(cor_analysis_1,histogram=FALSE,pch=4)
```




```
cor_analysis %>%
  ggplot(aes(Var2, Var1, fill = value)) +
  geom_tile(color = "black")+ geom_text(aes(label = round(value,2)), color = "white", size = 3) +
  coord_fixed()
```

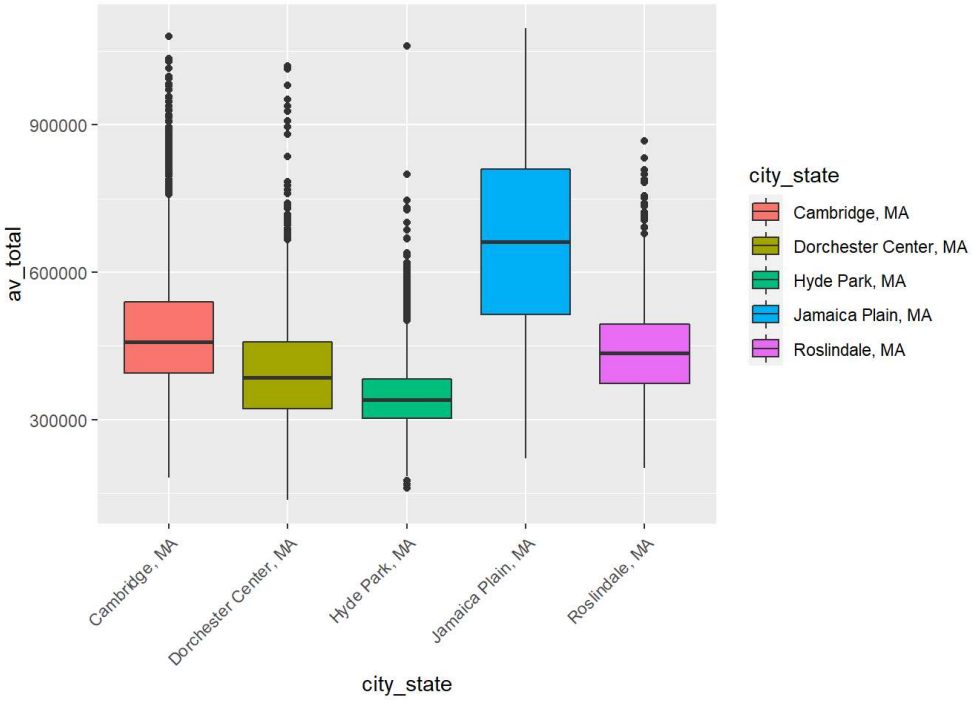


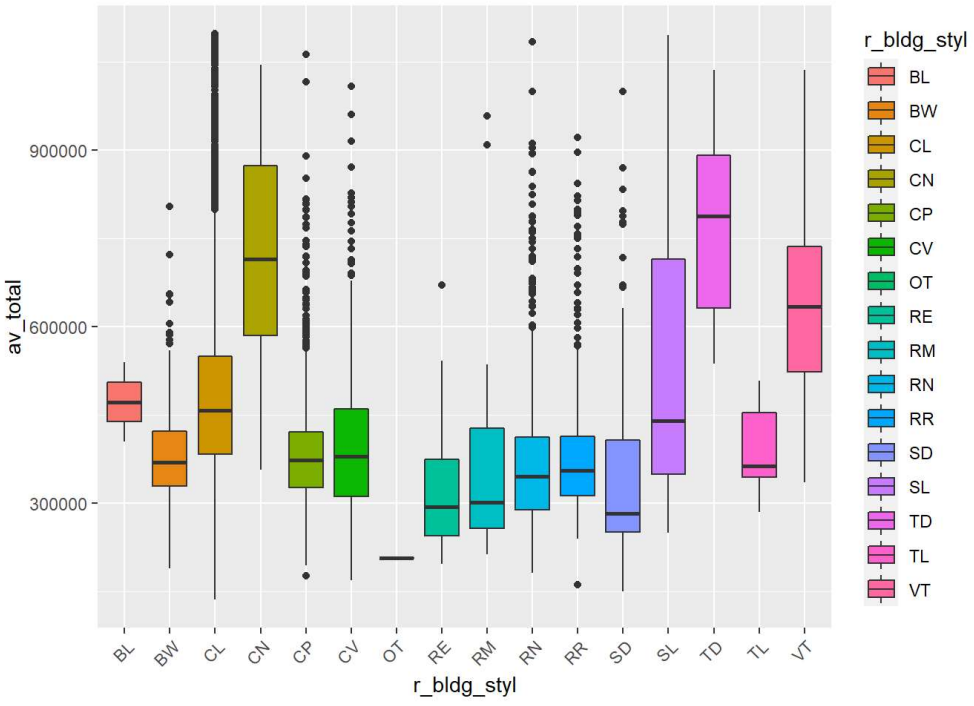
Explore Categorical Predictors

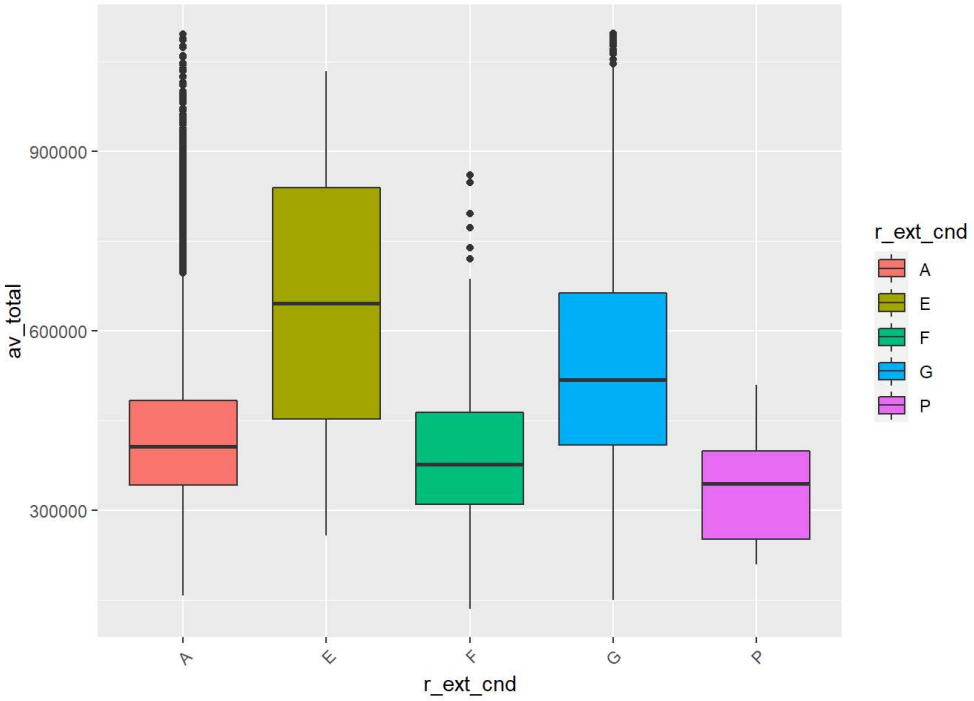
find 4 categorical variables are likely to be useful in predicting home prices?

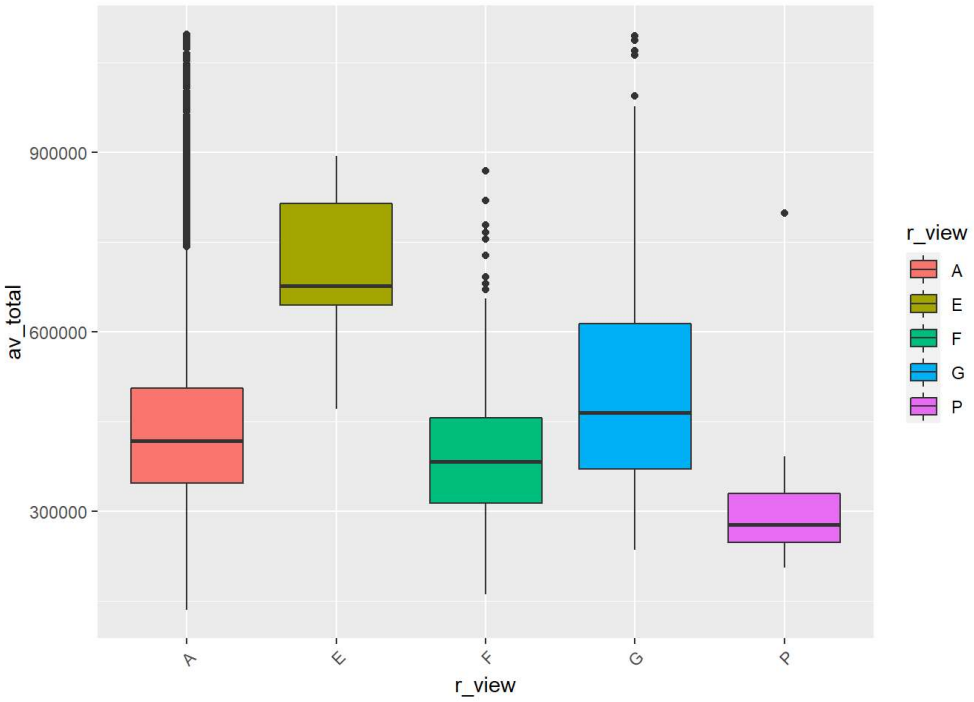
- use a chart comparing the category with av_total,
- a useful variable will have differences in the mean of av_total
- for example a boxplot of zipcode vs av_total is telling.

```
data_filtered$zipcode <- as.factor(data_filtered$zipcode)

useful_categorical <- c('city_state', 'r_bldg_styl', 'r_ext_cnd', 'r_view')

for (c in useful_categorical){
    plt <- data_filtered %>%
        ggplot(aes(x = !!as.name(c), y = av_total, fill = !!as.name(c))) +
        geom_boxplot() +
        theme(axis.text.x=element_text(angle=45, hjust=1))
    print(plt)
}
```


av_total
900000
600000
300000
Cambridge, MA
Dorchester Center, MA
Hyde Park, MA
Jamaica Plain, MA
Roslindale, MA
city_state
city_state
Cambridge, MA
Dorchester Center, MA
Hyde Park, MA
Jamaica Plain, MA
Roslindale, MA


av_total
900000
600000
300000
BL
BW
CL
CN
CP
CV
OT
RE
RM
RN
RR
SD
SL
TD
TL
VT
r_bldg_styl
r_bldg_styl
BL
BW
CL
CN
CP
CV
OT
RE
RM
RN
RR
SD
SL
TD
TL
VT






```
# 4 categorical variables: city_state, r_bldg_styl, r_ext_cnd, r_view
```

Prepare your data

1. select the following columns

- pid
- av_total
- age
- land_sf
- living_area
- num_floors
- population
- median_income
- city_state

PLUS your 4 character columns you think will be useful

2. Convert character columns to factors

- hint: mutate_at(c("var1", ...), as.factor)

```
# 4 categorical variables: city_state, r_bldg_styl, r_ext_cnd, r_view
data_model <-
  data_filtered %>%
  na.omit() %>%
  dplyr::select(pid, av_total, home_age, land_sf, living_area, num_floors, population, median_income, city_state, r_bldg_styl, r_ext_cnd, r_view)

category <- c('city_state', 'r_bldg_styl', 'r_ext_cnd', 'r_view')
data_model <- data_model %>% mutate_at(category, as.factor)

head(data_model)
```

pid	av_total	home_age	land_sf	living_area	num_floors	population	median_income	city_state
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1812296230	321200	28	10288	1681	1.0	28488	58890	Hyde Park, MA
2006736000	894400	4	10148	3024	2.5	36314	75446	Cambridge, MA
2009442000	387700	100	8512	1160	2.0	36314	75446	Cambridge, MA
1704953000	450500	19	3187	1868	2.0	47783	48841	Dorchester Center, MA
1812269079	325100	45	10088	1534	1.0	28488	58890	Hyde Park, MA
2011986000	408800	33	3500	1632	2.0	36314	75446	Cambridge, MA

6 rows | 1-9 of 12 columns

1. Partition your data 70/30 (train / test split)

1. split your data set into 70% training and 30% test
2. print out the % of each data set

```
set.seed(1234)

9483 * 0.7
```

```
## [1] 6638.1
```

```
sample <- sample.int(n=9483,size=floor(9483*0.7))

# sample

x_train <- train_prep[sample,]

x_train
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1806401000	2131	Y	424100	5000	1955	1975	1772	2.0	R
1812268073	2136	Y	324300	10099	1955	0	1534	1.0	R
1803750000	2136	Y	500300	4122	1894	2003	3460	1.5	R
2011520000	2132	Y	221850	4300	1920	0	960	1.0	R
2012056000	2132	Y	427800	8411	1960	0	1590	1.0	R
1812023060	2136	Y	351600	3631	1962	0	1392	2.0	R
2008310000	2132	Y	427300	5943	1965	0	1518	2.0	R

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1905021000	2130	Y	455400	2797	1910	0	1282	2.0	R
1605304000	2124	Y	411200	3834	1910	0	1680	2.0	R
1810481020	2136	Y	480200	9388	1875	NA	3242	2.0	R

1-10 of 6,638 rows | 1-10 of 33 columns Previous 1 2 3 4 5 6 ... 664 Next

```
x_test <- test_prep[-sample,]

x_test
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1704953000	2124	Y	450500	3187	1900	2001	1868	2.0	R
2011986000	2132	Y	408800	3500	1960	1987	1632	2.0	R
1902260006	2130	Y	794300	6007	1962	2001	2146	2.0	R
1805011001	2131	Y	432600	5000	1910	0	1772	2.0	R
1803911000	2136	Y	294900	6400	1933	0	1532	2.0	R
1904957000	2130	Y	499300	4020	1926	0	1768	1.5	R
1811225000	2136	Y	229500	4275	1955	1979	864	1.0	R
2004380000	2132	Y	590700	8652	1935	2008	2072	2.0	R
1102092000	2130	Y	483700	2812	1900	0	1778	2.0	R
2001711000	2132	Y	626200	8492	1900	0	2680	2.0	R

1-10 of 2,845 rows | 1-10 of 33 columns Previous 1 2 3 4 5 6 ... 285 Next

```
pct_train <- nrow(x_train)/nrow(train_prep)
sprintf("%1.4f%%", 100*pct_train)
```

```
## [1] "69.9989%"
```

```
pct_test <- nrow(x_test)/nrow(test_prep)
sprintf("%1.4f%%", 100*pct_test)
```

```
## [1] "30.0011%"
```

2. Train model 1

for example: model_1 <- lm(av_total ~ living_area + age + num_floors, data=train)

```
model_1 <- lm(av_total ~ living_area + home_age + num_floors, data=x_train)

model_1
```

```
##
## Call:
## lm(formula = av_total ~ living_area + home_age + num_floors,
##     data = x_train)
##
## Coefficients:
## (Intercept)  living_area     home_age   num_floors
##    188494.1        157.5       -626.0      21967.8
```

```
summary(model_1)
```

```
##
## Call:
## lm(formula = av_total ~ living_area + home_age + num_floors,
##     data = x_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -820342  -63838   -5325   49710  566656
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 188494.067   6046.984  31.172 < 0.0000000000000002 ***
## living_area    157.464      2.908  54.147 < 0.0000000000000002 ***
## home_age      -626.006     34.557 -18.115 < 0.0000000000000002 ***
## num_floors   21967.810   3539.447   6.207       0.000000000576 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 109300 on 6377 degrees of freedom
##   (257 observations deleted due to missingness)
## Multiple R-squared:  0.4238, Adjusted R-squared:  0.4235
## F-statistic:  1563 on 3 and 6377 DF,  p-value: < 0.00000000000000022
```

3. Train model 2

for example: model_2 <- lm(av_total ~ living_area + age + num_floors + , data=train)

```
model_2 <- MASS::stepAIC(model_1, direction="both")
```

```
## Start:  AIC=148062
## av_total ~ living_area + home_age + num_floors
##
##               Df      Sum of Sq             RSS    AIC
## <none>                           76125584580855 148062
## - num_floors   1   459850843156  76585435424011 148098
## - home_age     1  3917343401836  80042927982692 148380
## - living_area  1 34999009209507 111124593790362 150474
```

```
summary(model_2)
```

```
##
## Call:
## lm(formula = av_total ~ living_area + home_age + num_floors,
##     data = x_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -820342  -63838   -5325   49710  566656
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 188494.067   6046.984  31.172 < 0.0000000000000002 ***
## living_area    157.464      2.908  54.147 < 0.0000000000000002 ***
## home_age      -626.006     34.557 -18.115 < 0.0000000000000002 ***
## num_floors   21967.810   3539.447   6.207       0.000000000576 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 109300 on 6377 degrees of freedom
##   (257 observations deleted due to missingness)
## Multiple R-squared:  0.4238, Adjusted R-squared:  0.4235
## F-statistic:  1563 on 3 and 6377 DF,  p-value: < 0.00000000000000022
```

```
model_2 <- lm(formula = av_total ~ home_age + land_sf + living_area + num_floors + population + median_income + city_state + r_bldg_styl + r_ext_
    cnd + r_view, data=x_train)
summary(model_2)
```

```
##
## Call:
## lm(formula = av_total ~ home_age + land_sf + living_area + num_floors +
##     population + median_income + city_state + r_bldg_styl + r_ext_cnd +
##     r_view, data = x_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -625070  -36522    2087   36608  366962
##
## Coefficients: (2 not defined because of singularities)
##                                    Estimate    Std. Error t value
## (Intercept)                     -2798896.0706    89658.7438 -31.217
## home_age                            -416.2128       22.6463 -18.379
## land_sf                               10.0827        0.3557  28.345
## living_area                          120.9130        2.1783  55.509
## num_floors                         -14771.2101     5365.9354  -2.753
## population                           -139.8129        3.0259 -46.205
## median_income                         106.8486        2.1855  48.890
## city_stateDorchester Center, MA   4351511.5315    93499.7055  46.540
## city_stateHyde Park, MA            567561.2838    13609.3884  41.704
## city_stateJamaica Plain, MA                NA           NA      NA
## city_stateRoslindale, MA                   NA           NA      NA
## r_bldg_stylBW                       51401.2222    68922.4729   0.746
## r_bldg_stylCL                       92722.0339    69030.4884   1.343
## r_bldg_stylCN                      147087.0472    71493.9277   2.057
## r_bldg_stylCP                       63103.9780    68872.5426   0.916
## r_bldg_stylCV                       46735.0965    69076.1584   0.677
## r_bldg_stylOT                      -30635.3751   100046.2143  -0.306
## r_bldg_stylRE                       58880.0256    70290.6196   0.838
## r_bldg_stylRM                       43973.6264    70238.3708   0.626
## r_bldg_stylRN                       42083.2689    68902.8787   0.611
## r_bldg_stylRR                       -1831.5591    68942.9326  -0.027
## r_bldg_stylSD                       26773.4828    69569.1050   0.385
## r_bldg_stylSL                       68584.5030    69496.5204   0.987
## r_bldg_stylTD                      207415.4209    71599.0975   2.897
## r_bldg_stylTL                       56927.0408    72165.5102   0.789
## r_bldg_stylVT                      163644.3600    69710.9269   2.347
## r_ext_cndE                           2127.2005    69327.7733   0.031
## r_ext_cndF                         -43128.6937     5722.5396  -7.537
## r_ext_cndG                          54294.6658     2521.5720  21.532
## r_ext_cndP                        -145376.9450    26070.5580  -5.576
## r_viewE                             97974.8598    34504.5401   2.839
## r_viewF                            -32899.8597     5315.7860  -6.189
## r_viewG                             23435.4507     4947.9477   4.736
## r_viewP                            -40015.6826    23073.8697  -1.734
##                                      Pr(>|t|)
## (Intercept)                     < 0.0000000000000002 ***
## home_age                        < 0.0000000000000002 ***
## land_sf                         < 0.0000000000000002 ***
## living_area                     < 0.0000000000000002 ***
## num_floors                                0.00593 **
## population                      < 0.0000000000000002 ***
## median_income                   < 0.0000000000000002 ***
## city_stateDorchester Center, MA < 0.0000000000000002 ***
## city_stateHyde Park, MA         < 0.0000000000000002 ***
## city_stateJamaica Plain, MA                      NA
## city_stateRoslindale, MA                         NA
## r_bldg_stylBW                             0.45583
## r_bldg_stylCL                             0.17925
## r_bldg_stylCN                             0.03969 *
## r_bldg_stylCP                             0.35957
## r_bldg_stylCV                             0.49870
```

```
## r_bldg_styl0T                              0.75945
## r_bldg_stylRE                              0.40225
## r_bldg_stylRM                              0.53130
## r_bldg_stylRN                              0.54138
## r_bldg_stylRR                              0.97881
## r_bldg_stylSD                              0.70036
## r_bldg_stylSL                              0.32374
## r_bldg_stylTD                              0.00378 **
## r_bldg_stylTL                              0.43023
## r_bldg_stylVT                              0.01893 *
## r_ext_cndE                                 0.97552
## r_ext_cndF                      0.000000000000055 ***
## r_ext_cndG                    < 0.0000000000000002 ***
## r_ext_cndP                      0.000000025582599 ***
## r_viewE                                    0.00453 **
## r_viewF                         0.000000000642898 ***
## r_viewG                         0.000002222796748 ***
## r_viewP                                    0.08292 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 68780 on 6348 degrees of freedom
##   (258 observations deleted due to missingness)
## Multiple R-squared:  0.7727, Adjusted R-squared:  0.7716
## F-statistic: 696.1 on 31 and 6348 DF,  p-value: < 0.00000000000000022
```

4. MAKE PREDICTIONS

make predictions on training and test for each model

for example, do this 4 times:

train$model_1_pred <- predict(model1,train)

or use https://modelr.tidyverse.org/reference/add_predictions.html (https://modelr.tidyverse.org/reference/add_predictions.html)

add_predictions to do the same thing

```
# -- apply the models
training_1 <- x_train
training_1 <- add_predictions(training_1, model_1, var="ppg_prediction")

test_1 <- x_test
test_1 <- add_predictions(test_1, model_1, var="ppg_prediction")

training_1
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1806401000	2131	Y	424100	5000	1955	1975	1772	2.0	R
1812268073	2136	Y	324300	10099	1955	0	1534	1.0	R
1803750000	2136	Y	500300	4122	1894	2003	3460	1.5	R
2011520000	2132	Y	221850	4300	1920	0	960	1.0	R
2012056000	2132	Y	427800	8411	1960	0	1590	1.0	R
1812023060	2136	Y	351600	3631	1962	0	1392	2.0	R
2008310000	2132	Y	427300	5943	1965	0	1518	2.0	R
1905021000	2130	Y	455400	2797	1910	0	1282	2.0	R
1605304000	2124	Y	411200	3834	1910	0	1680	2.0	R

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1810481020	2136	Y	480200	9388	1875	*NA*	3242	2.0	R

1-10 of 6,638 rows | 1-10 of 34 columns Previous **1** 2 3 4 5 6 ... 664 Next

```
test_1
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1704953000	2124	Y	450500	3187	1900	2001	1868	2.0	R
2011986000	2132	Y	408800	3500	1960	1987	1632	2.0	R
1902260006	2130	Y	794300	6007	1962	2001	2146	2.0	R
1805011001	2131	Y	432600	5000	1910	0	1772	2.0	R
1803911000	2136	Y	294900	6400	1933	0	1532	2.0	R
1904957000	2130	Y	499300	4020	1926	0	1768	1.5	R
1811225000	2136	Y	229500	4275	1955	1979	864	1.0	R
2004380000	2132	Y	590700	8652	1935	2008	2072	2.0	R
1102092000	2130	Y	483700	2812	1900	0	1778	2.0	R
2001711000	2132	Y	626200	8492	1900	0	2680	2.0	R

1-10 of 2,845 rows | 1-10 of 34 columns Previous **1** 2 3 4 5 6 ... 285 Next

```
training_2 <- x_train
training_2 <- add_predictions(training_2, model_2, var="ppg_prediction")

test_2 <- x_test
test_2 <- add_predictions(test_2, model_2, var="ppg_prediction")

training_2
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1806401000	2131	Y	424100	5000	1955	1975	1772	2.0	R
1812268073	2136	Y	324300	10099	1955	0	1534	1.0	R
1803750000	2136	Y	500300	4122	1894	2003	3460	1.5	R
2011520000	2132	Y	221850	4300	1920	0	960	1.0	R
2012056000	2132	Y	427800	8411	1960	0	1590	1.0	R
1812023060	2136	Y	351600	3631	1962	0	1392	2.0	R
2008310000	2132	Y	427300	5943	1965	0	1518	2.0	R
1905021000	2130	Y	455400	2797	1910	0	1282	2.0	R
1605304000	2124	Y	411200	3834	1910	0	1680	2.0	R
1810481020	2136	Y	480200	9388	1875	*NA*	3242	2.0	R

1-10 of 6,638 rows | 1-10 of 34 columns Previous **1** 2 3 4 5 6 ... 664 Next

```
test_2
```

pid	zipcode	own_occ	av_total	land_sf	yr_built	yr_remod	living_area	num_floors	structure_class
<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1704953000	2124	Y	450500	3187	1900	2001	1868	2.0	R
2011986000	2132	Y	408800	3500	1960	1987	1632	2.0	R
1902260006	2130	Y	794300	6007	1962	2001	2146	2.0	R
1805011001	2131	Y	432600	5000	1910	0	1772	2.0	R
1803911000	2136	Y	294900	6400	1933	0	1532	2.0	R
1904957000	2130	Y	499300	4020	1926	0	1768	1.5	R
1811225000	2136	Y	229500	4275	1955	1979	864	1.0	R
2004380000	2132	Y	590700	8652	1935	2008	2072	2.0	R
1102092000	2130	Y	483700	2812	1900	0	1778	2.0	R
2001711000	2132	Y	626200	8492	1900	0	2680	2.0	R

1-10 of 2,845 rows | 1-10 of 34 columns Previous 1 2 3 4 5 6 ... 285 Next

5. Calculate Evaluatation Metrics

use modelr package or do it by hand but you'll want to calculate for both training and test datasets for each of your two models, you need to be able to explain what these metrics mean. is a large RMSE good or bad? is a large RSQUARE good or bad, how do you interpret RSQUARE? mse() rmse() mae() rsquare()

https://modelr.tidyverse.org/reference/index.html (https://modelr.tidyverse.org/reference/index.html)

```
table <- data.frame(
    row.names = c("training_1", "test_1", "training_2", "test_2"),
    mse = c(mse(model_1,training_1), mse(model_1,test_1), mse(model_2,training_2), mse(model_2,test_2)),
    rmse = c(rmse(model_1,training_1), rmse(model_1,test_1), rmse(model_2,training_2), rmse(model_2,test_2)),
    mae = c(mae(model_1,training_1), mae(model_1,test_1), mae(model_2,training_2), mae(model_2,test_2)),
    rsquare = c(rsquare(model_1,training_1), rsquare(model_1,test_1), rsquare(model_2,training_2), rsquare(model_2,test_2))
)

table %>% write_csv("table.csv")
```

6. Which PREDICTIONS did great, over and underestimated av_total?

using only your TEST partition what are the top 10 houses 1. that your best linear regression did the best predicting residual closest to zero 2. that your worst linear regression overestimated av_total

3. that your worst linear regression underestimated av_total

```
residual <- test_2 %>%
  na.omit() %>%
  mutate(compare = if_else(ppg_prediction<av_total, 'underestimated', 'overestimated'), error = av_total - ppg_prediction, abs_error = abs(av_tot
         al - ppg_prediction)) %>%
  dplyr::select(av_total, ppg_prediction, compare, error, abs_error)


residual %>%
  slice_min(order_by = abs_error, n = 10) %>%
  dplyr::select(av_total, ppg_prediction, compare, abs_error)
```

av_total	ppg_prediction	compare	abs_error
<dbl>	<dbl>	<chr>	<dbl>
332900	332871.4	underestimated	28.64844
410100	410146.2	overestimated	46.24147

av_total	ppg_prediction	compare	abs_error
<dbl>	<dbl>	<chr>	<dbl>
315700	315608.8	underestimated	91.18818
308100	308217.2	overestimated	117.19738
369100	369284.1	overestimated	184.12290
475600	475400.7	underestimated	199.34223
458500	458712.9	overestimated	212.88591
402800	402584.6	underestimated	215.35275
611300	611552.9	overestimated	252.86260
330800	330508.9	underestimated	291.11620

1-10 of 10 rows

```
residual %>%
  slice_min(order_by = error, n = 10) %>%
  dplyr::select(av_total, ppg_prediction, compare, error)
```

av_total	ppg_prediction	compare	error
<dbl>	<dbl>	<chr>	<dbl>
601464	987444.7	overestimated	-385980.7
483700	852004.2	overestimated	-368304.2
739300	1024842.3	overestimated	-285542.3
262700	526076.0	overestimated	-263376.0
300500	551806.8	overestimated	-251306.8
314600	548303.1	overestimated	-233703.1
680800	907142.0	overestimated	-226342.0
295900	519296.0	overestimated	-223396.0
321300	540277.5	overestimated	-218977.5
318080	534920.2	overestimated	-216840.2

1-10 of 10 rows

```
residual %>%
  slice_max(order_by = error, n = 10) %>%
  dplyr::select(av_total, ppg_prediction, compare, error)
```

av_total	ppg_prediction	compare	error
<dbl>	<dbl>	<chr>	<dbl>
1088200	730257.9	underestimated	357942.1
1023600	711703.4	underestimated	311896.6
1085616	776037.6	underestimated	309578.4
1083700	795510.9	underestimated	288189.1
1078000	801611.5	underestimated	276388.5
777500	521162.6	underestimated	256337.4
1057500	808343.8	underestimated	249156.2
909000	661579.3	underestimated	247420.7

av_total	ppg_prediction	compare	error
<dbl>	<dbl>	<chr>	<dbl>
1090500	845793.7	underestimated	244706.3
906000	665268.4	underestimated	240731.6

1-10 of 10 rows

your notebook should knit from beginning to end, and should be your own work!!!