Challenge 3 R Notebook

Eagle Xuhui Ying

11/8/2022

- Library
- DATA
- Histogram Target
- Explore numerics
- Explore character variables
- homes built in the 1990s, tend to have higher home values.
- Partition our data 70/30 PLUS make K-Fold Cross Validation
- Recipe
- Linear Reg Setup
- random forest
- Evaluate the random forest Model
- XGBoost Model Buiding
- XGB Tuning
- Final Fit XGB
- VIP
- Evaluate the XGBoost BEST Model
- Best Worst Predicitons
- KAGGLE

Library

```
options(scipen = 999) # turns off scientific notation
library(tidymodels)
library(tidyverse)
library(janitor)
library(vip)
library(skimr)
```

DATA

import data

```
boston <- read_csv("boston_train.csv") %>% clean_names()
kaggle <- read_csv("boston_holdout.csv") %>% clean_names()
zips   <- read_csv("zips.csv") %>% clean_names()

boston %>% skim()
```

Data summary

Name	Piped data
Number of rows	9959
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	10	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	5	0
r_int_cnd	0	1	1	1	0	5	0
r_int_fin	0	1	1	1	0	2	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	71093.60	41011.82	10	35780.0	71180	106575	142240	
zipcode	0	1.00	2131.45	3.69	2124	2131.0	2132	2132	2136	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
av_total	0	1.00	448563.64	147761.17	134800	347100.0	418700	510150	1097100	
land_sf	0	1.00	5936.18	2954.01	920	4319.0	5320	6756	107158	
yr_built	0	1.00	1933.40	35.28	0	1910.0	1931	1955	2016	
yr_remod	347	0.97	675.71	946.59	0	0.0	0	1997	2016	
living_area	0	1.00	1659.00	545.95	332	1300.0	1554	1914	8623	
num_floors	0	1.00	1.73	0.45	1	1.5	2	2	3	
r_total_rms	0	1.00	7.11	1.55	3	6.0	7	8	17	
r_bdrms	0	1.00	3.34	0.92	0	3.0	3	4	9	
r_full_bth	0	1.00	1.35	0.56	1	1.0	1	2	6	
r_half_bth	0	1.00	0.56	0.55	0	0.0	1	1	10	
r_kitch	0	1.00	1.02	0.14	1	1.0	1	1	3	
r_fplace	0	1.00	0.60	0.62	0	0.0	1	1	5	
zip	0	1.00	2131.45	3.69	2124	2131.0	2132	2132	2136	
population	0	1.00	34871.56	6299.30	28488	29826.0	35401	36314	47783	
pop_density	0	1.00	11368.99	3293.58	6207	10618.0	11505	13251	15913	
median_income	0	1.00	65984.07	9749.72	48841	58890.0	66735	75446	75730	

Histogram Target

The first plot shows us that the data are right-skewed; there are more inexpensive houses than expensive ones. When modeling this type of outcome, a strong argument can be made that the price should be log-transformed. The advantages of this type of transformation are that no houses would be predicted with negative sale prices and that errors in predicting expensive houses will not have an undue influence on the model. you can deal with this in the recipe

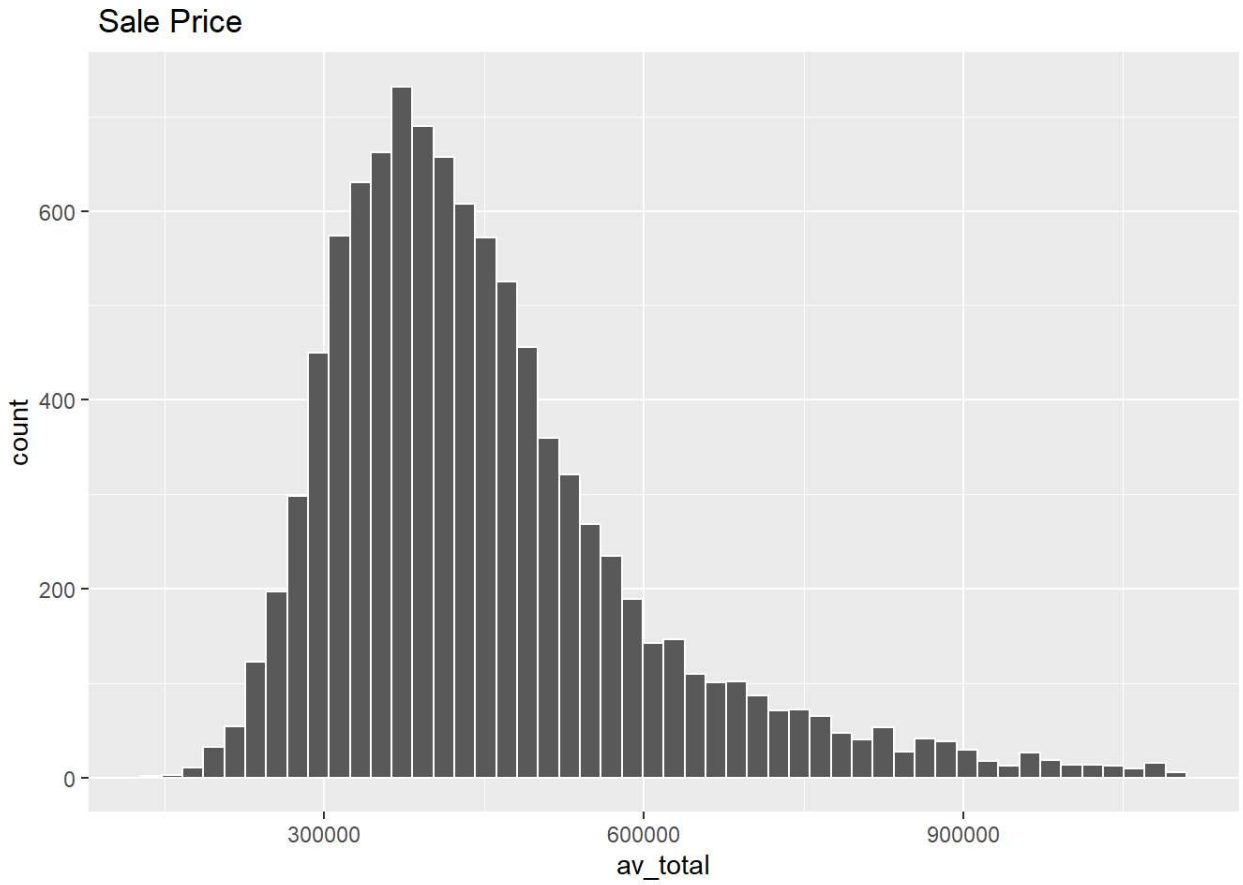
```
median(boston$av_total)
```

```
## [1] 418700
```

```
mean(boston$av_total)
```

```
## [1] 448563.6
```

```
ggplot(boston, aes(x = av_total)) +
  geom_histogram(bins = 50, col= "white") +
  labs(title=" Sale Price")
```

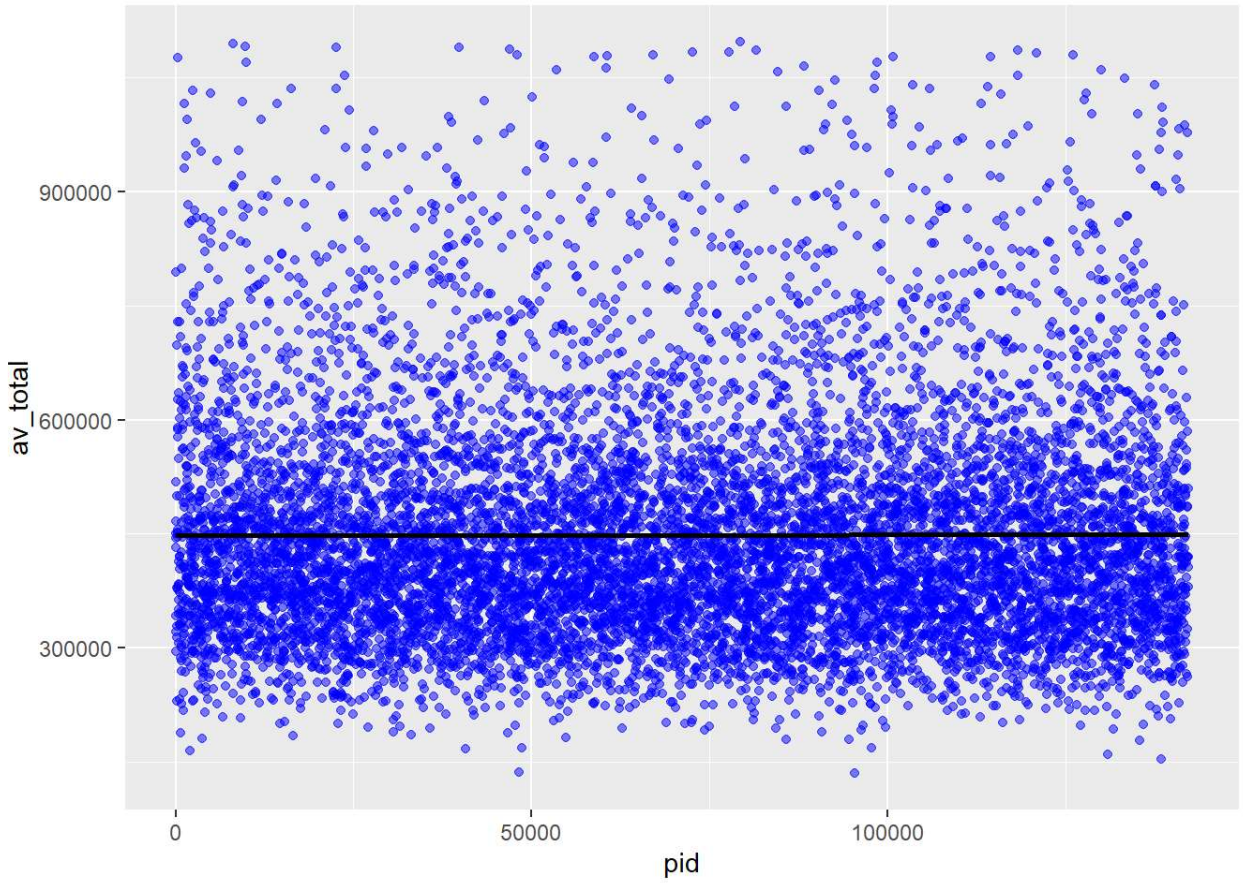



```
ggplot(boston, aes(x = av_total)) +
  geom_histogram(bins = 50, col= "white") +
  scale_x_log10() +
  labs(title="Histogram Log of Sale Price")
```

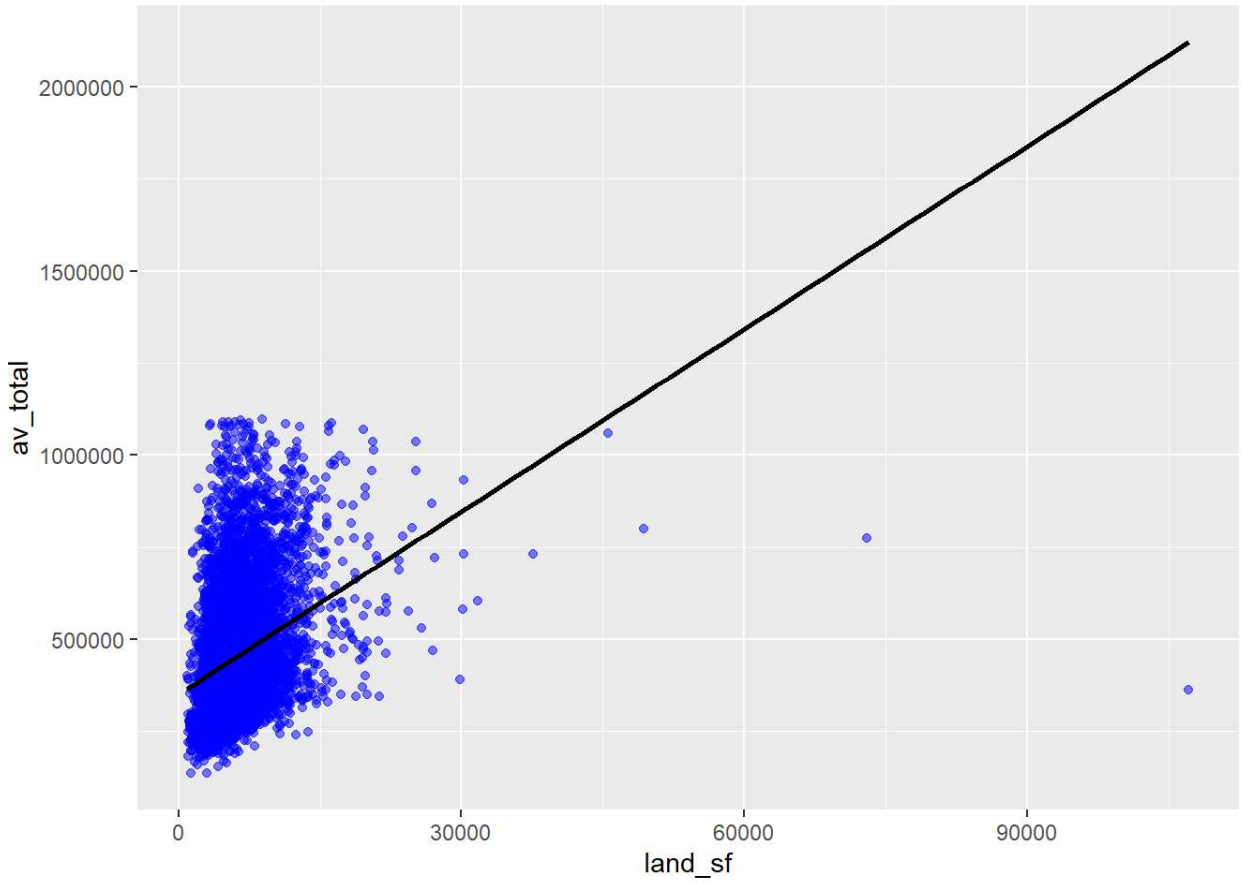
Explore numerics

numeric variables: pid, land_sf, yr_built, yr_remod, living_area, num_floors, r_total_rms, r_bdrms, r_full_bth, r_half_bth, r_kitch, r_fplace, population, pop_density, median_income

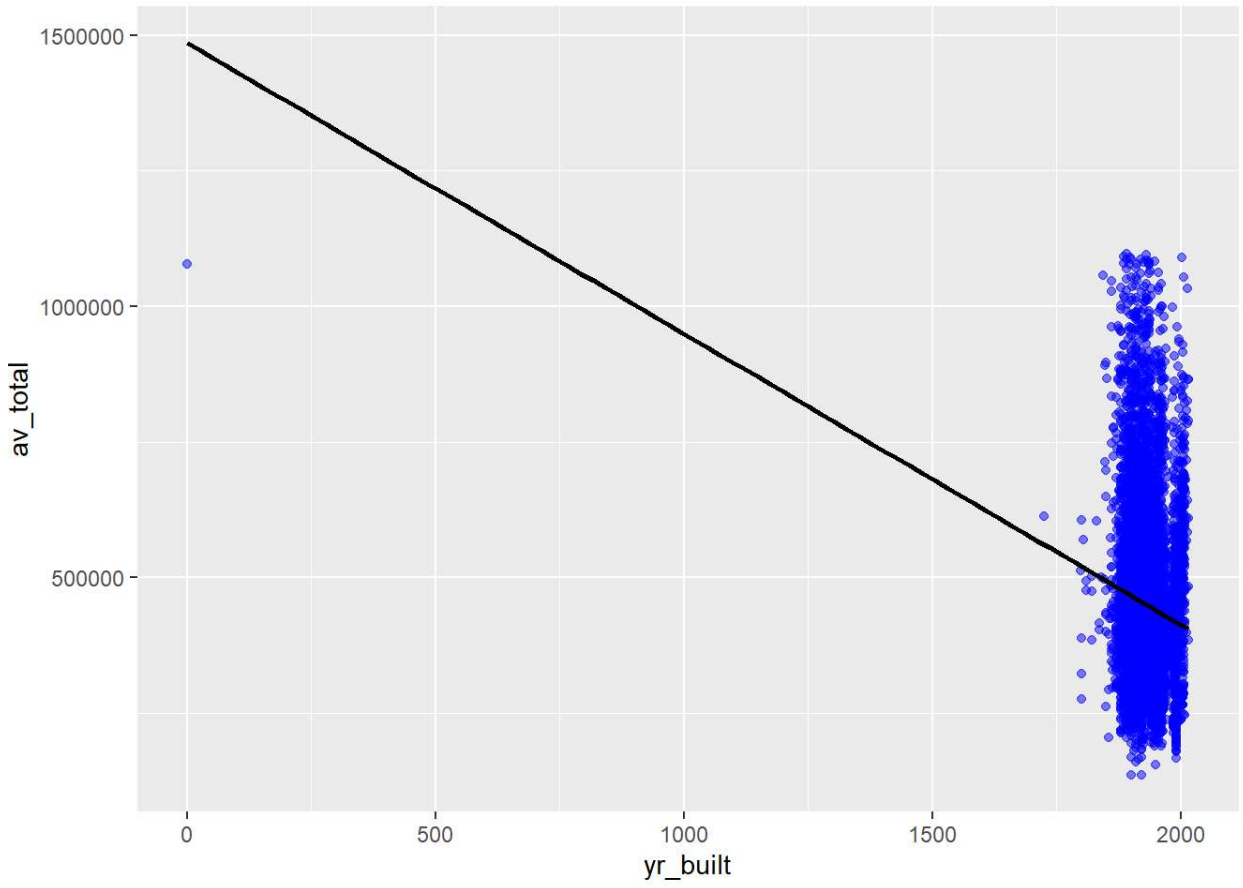
```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=pid, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



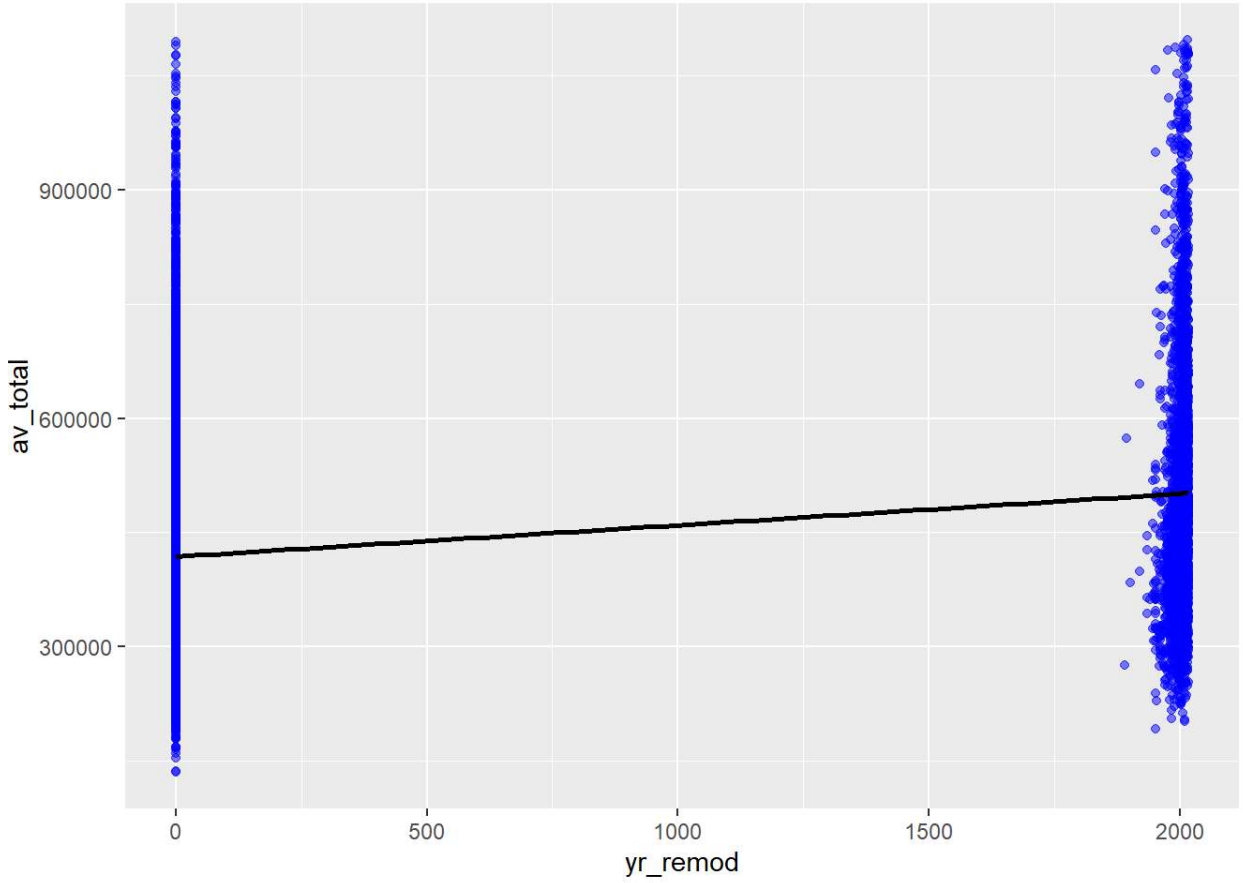

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=land_sf, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



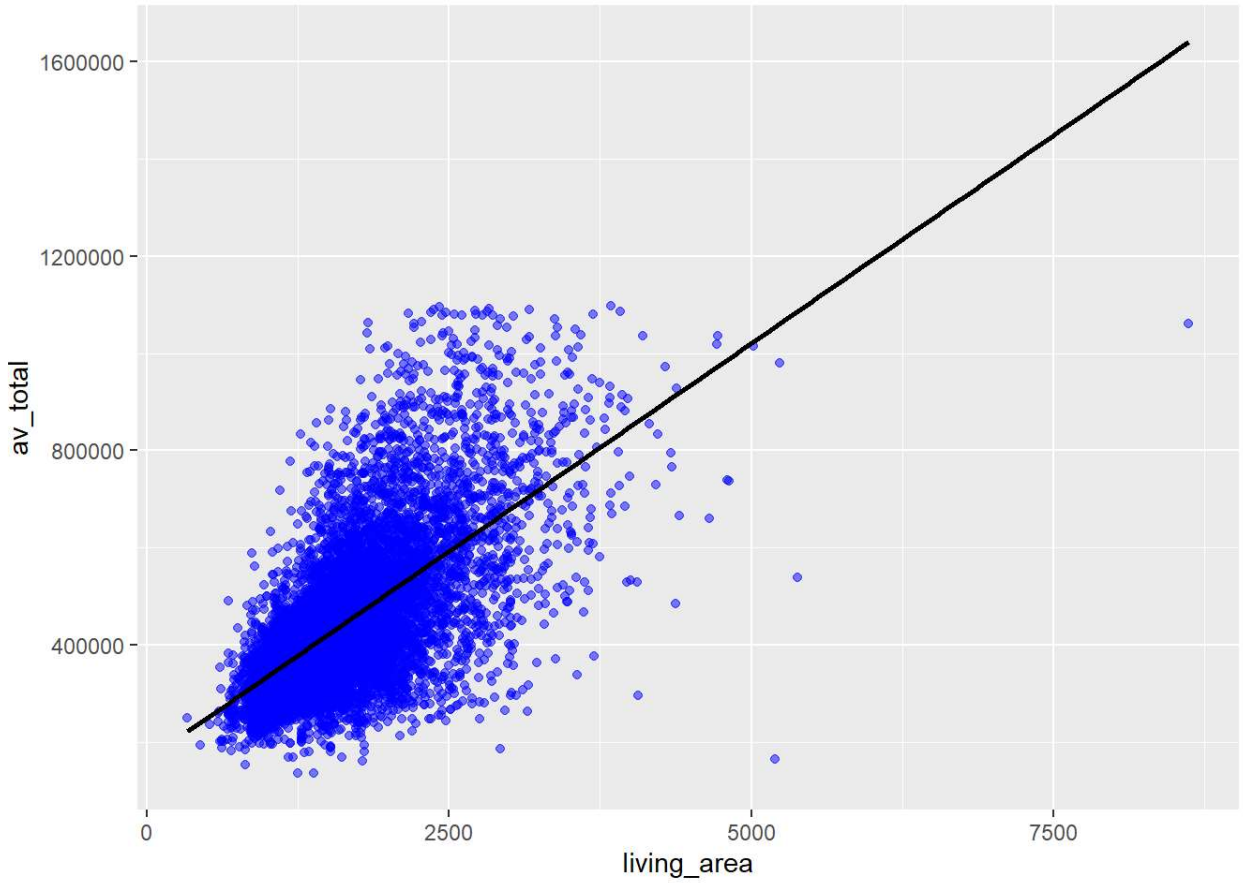

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=yr_built, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



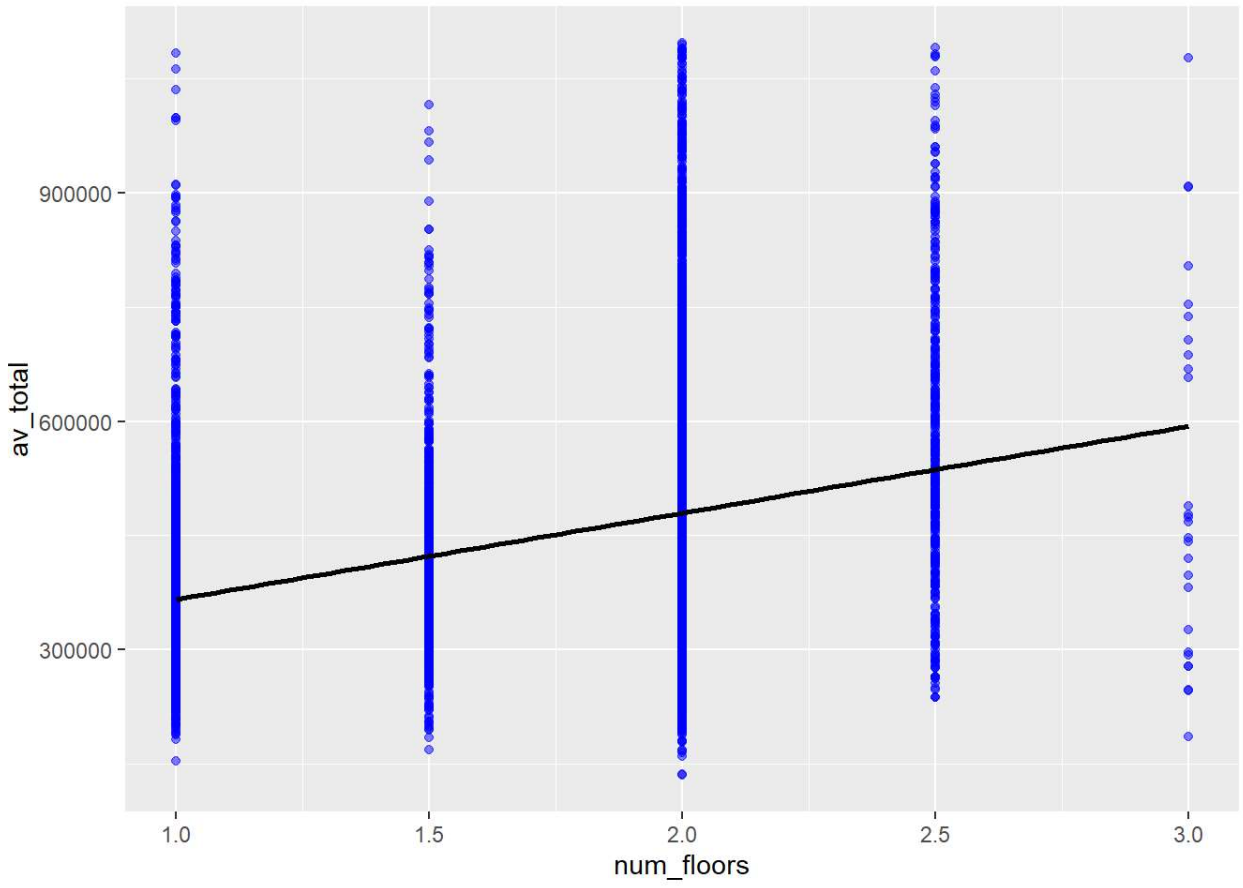
```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=yr_remod, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



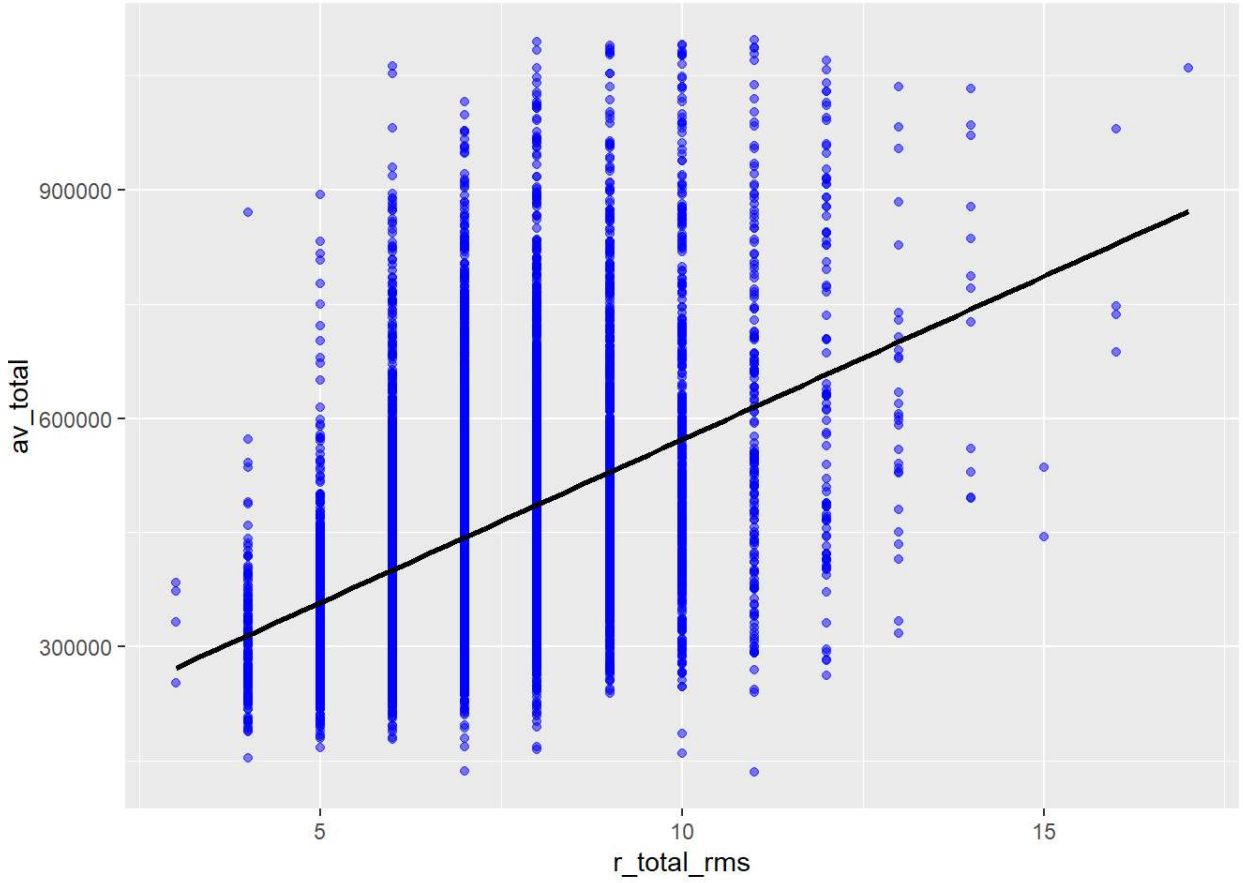

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=living_area, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



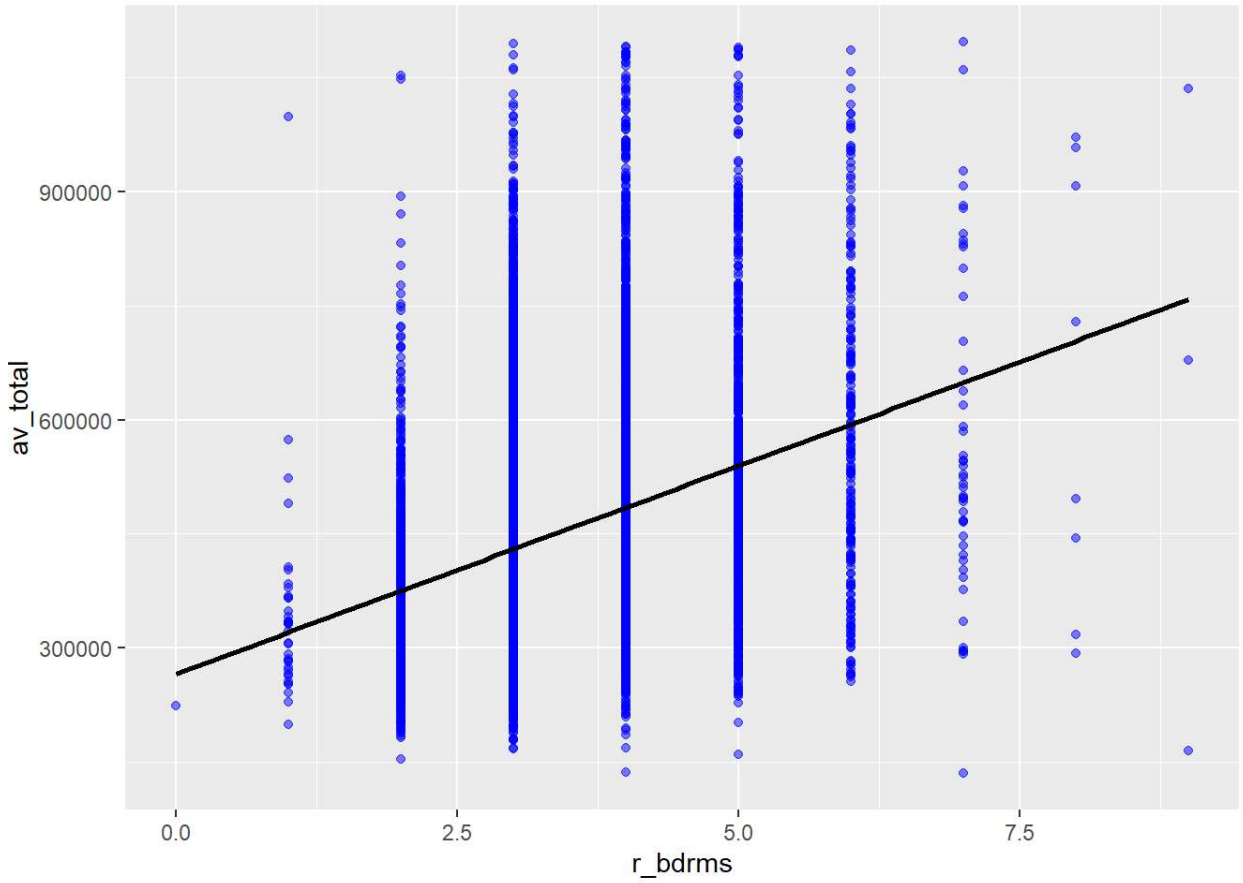

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=num_floors, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



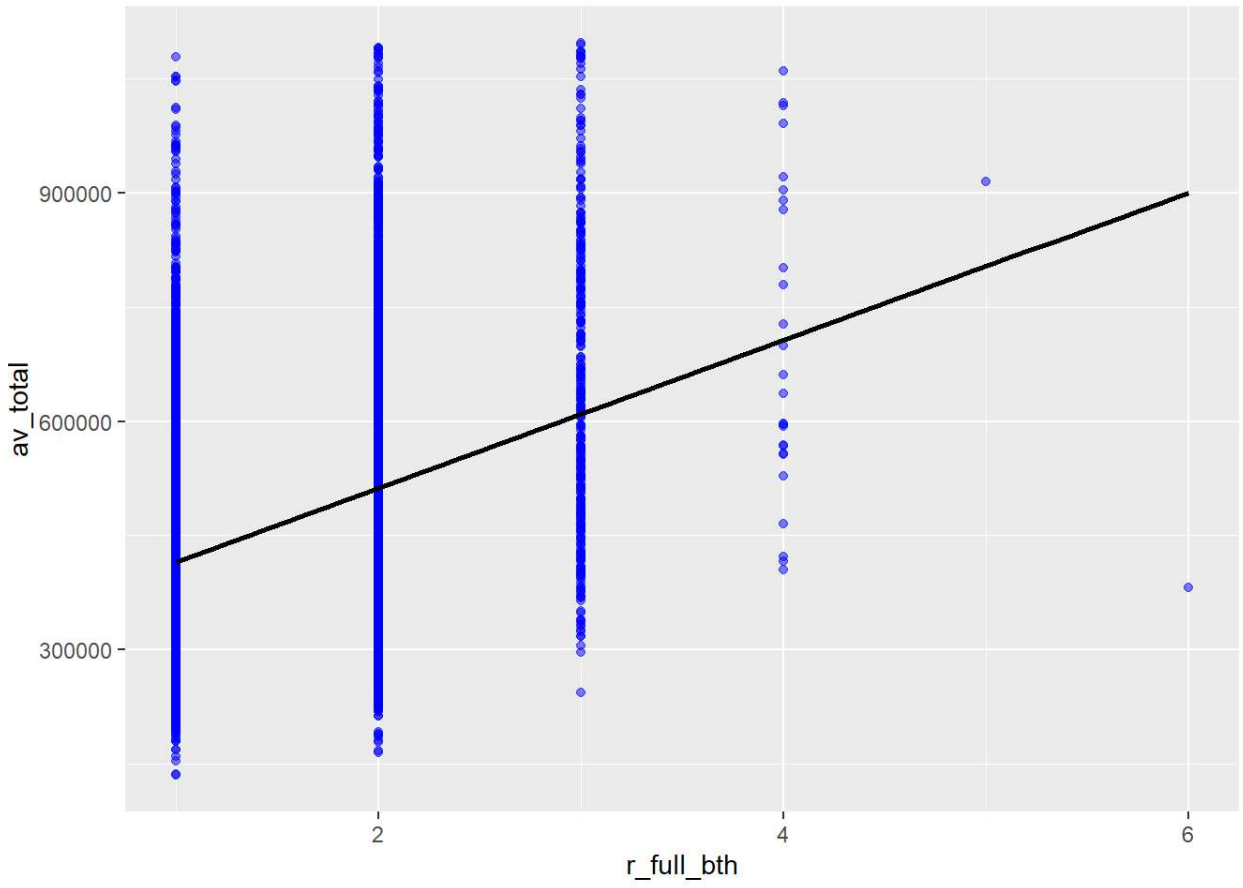
```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=r_total_rms, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



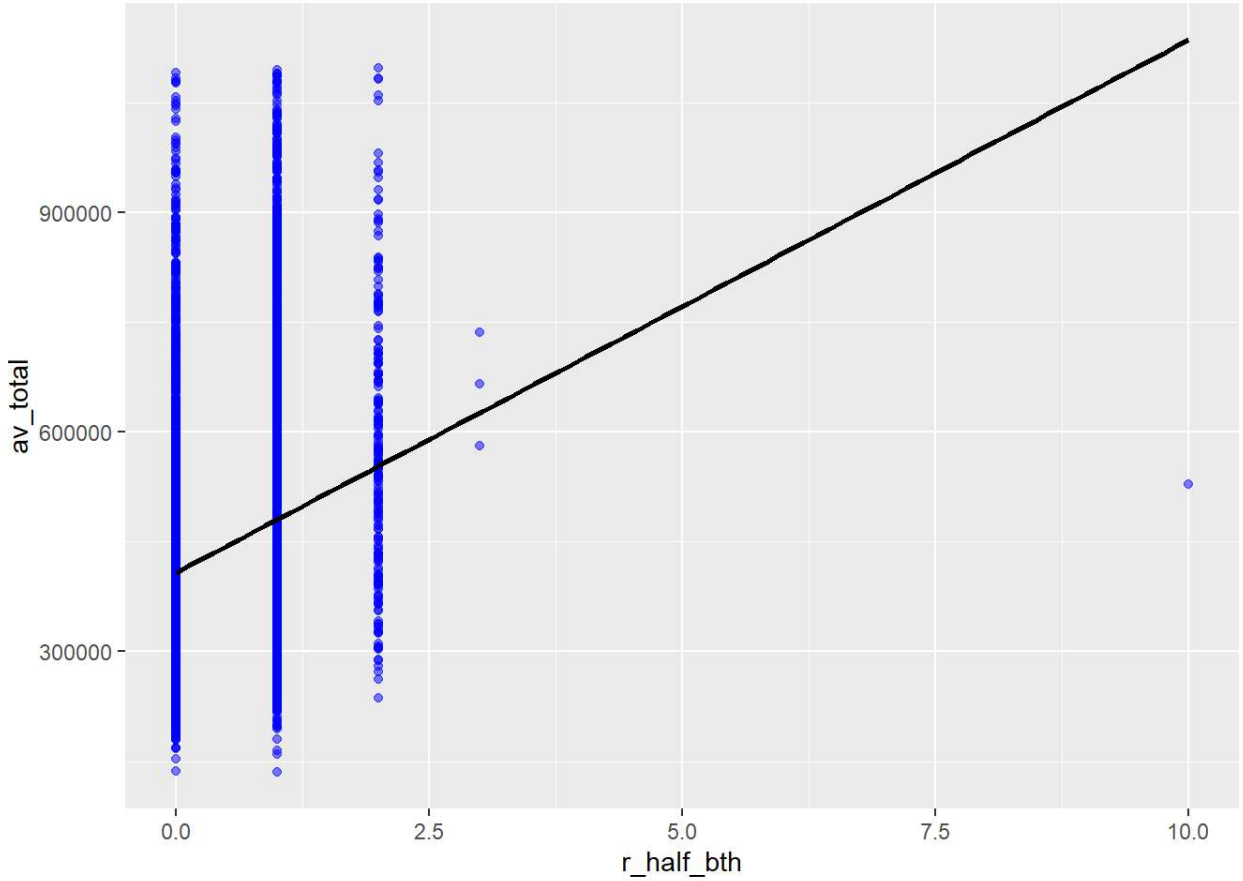

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=r_bdrms, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



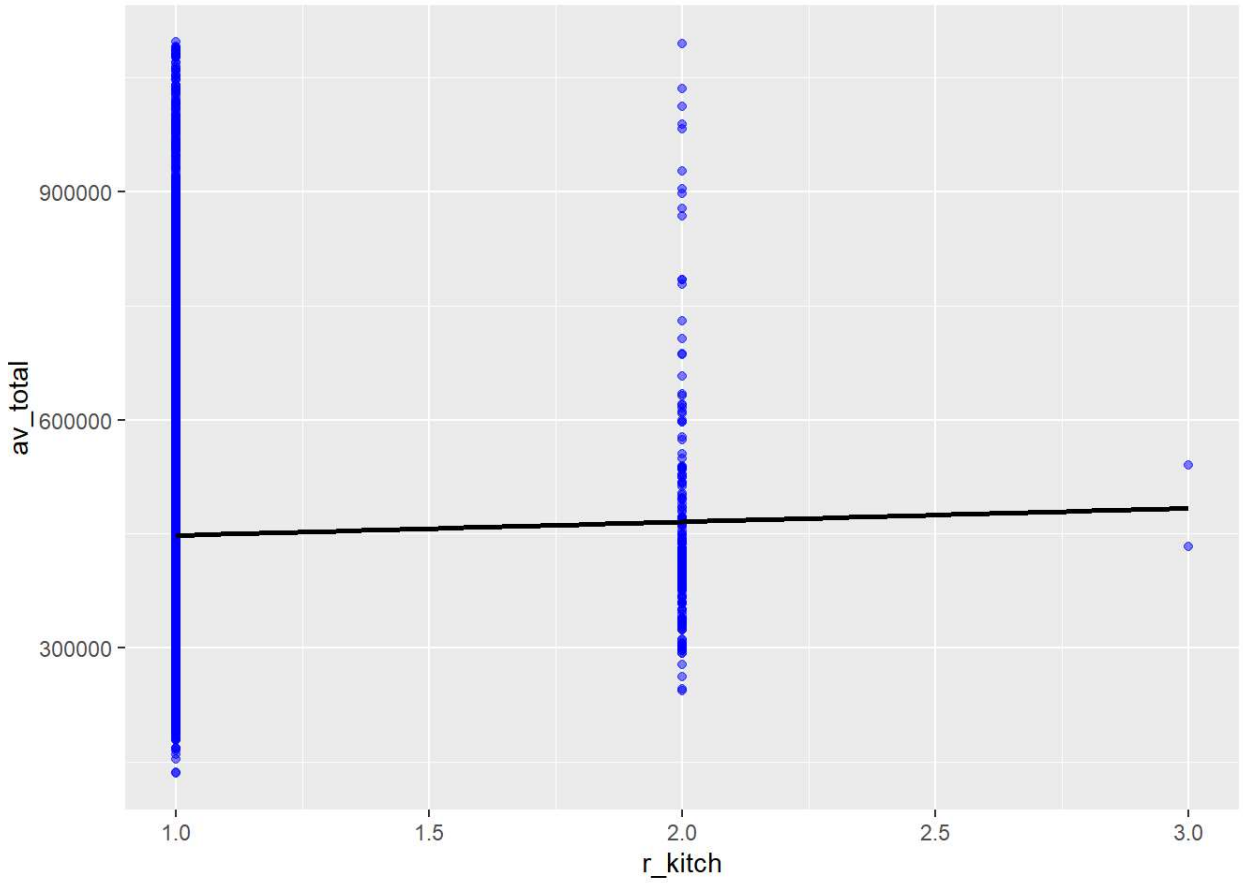

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=r_full_bth, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



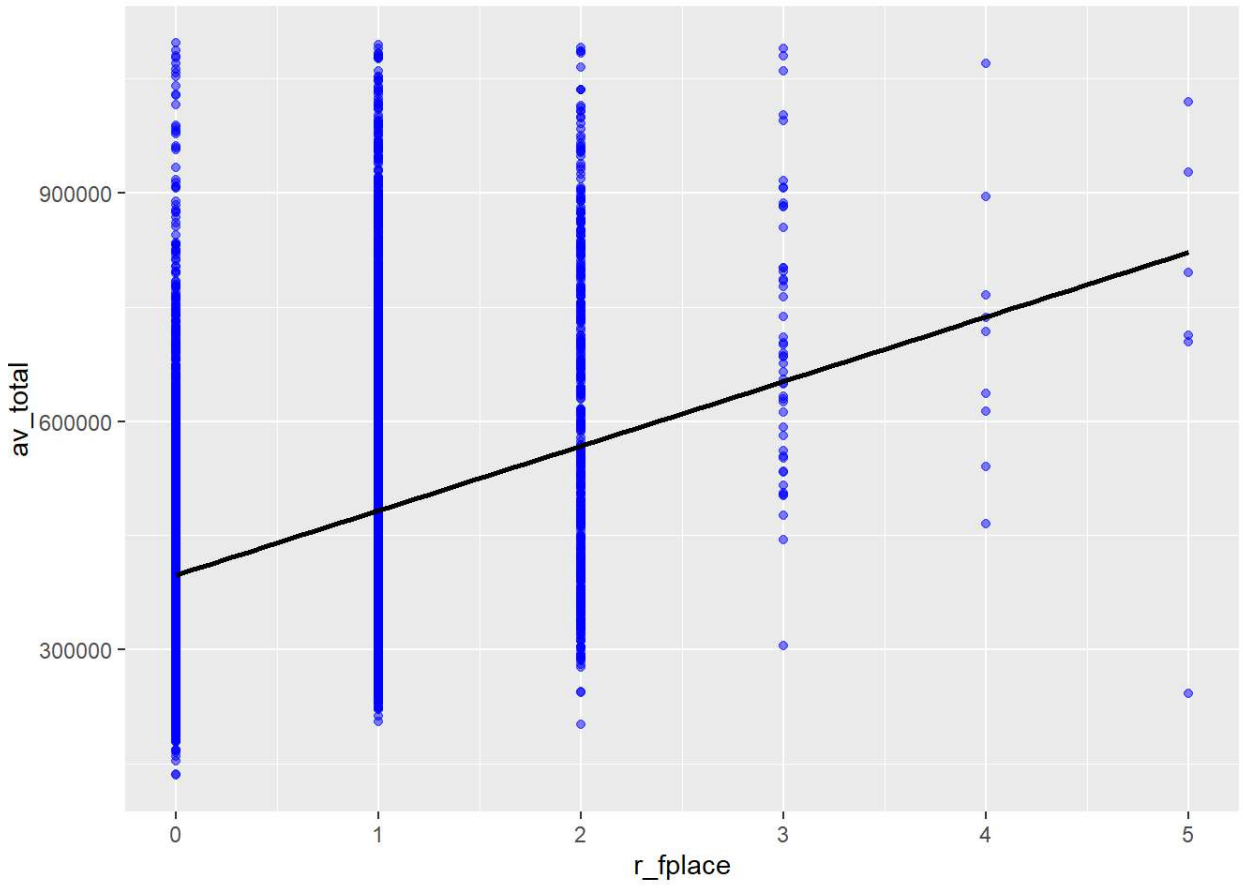
```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=r_half_bth, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



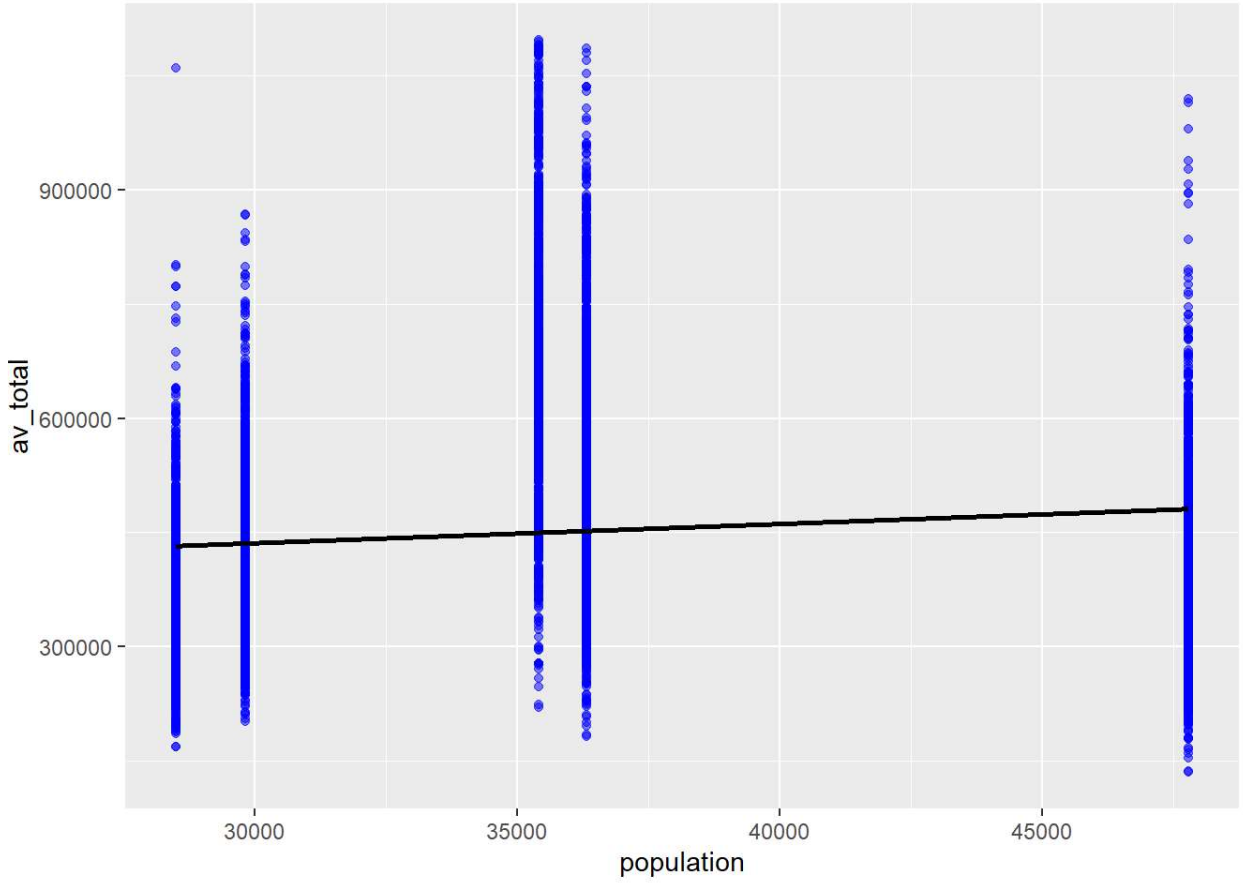

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=r_kitch, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



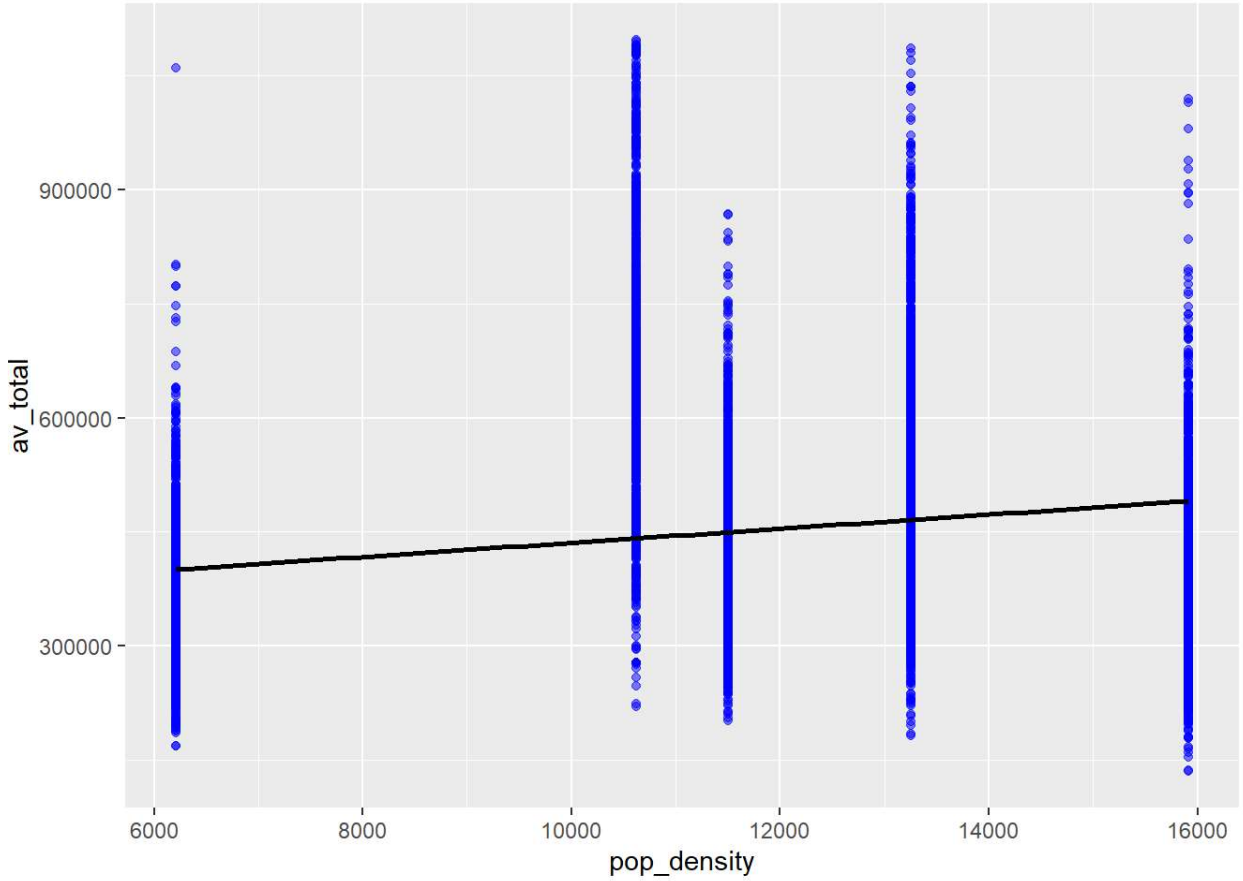

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=r_fplace, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



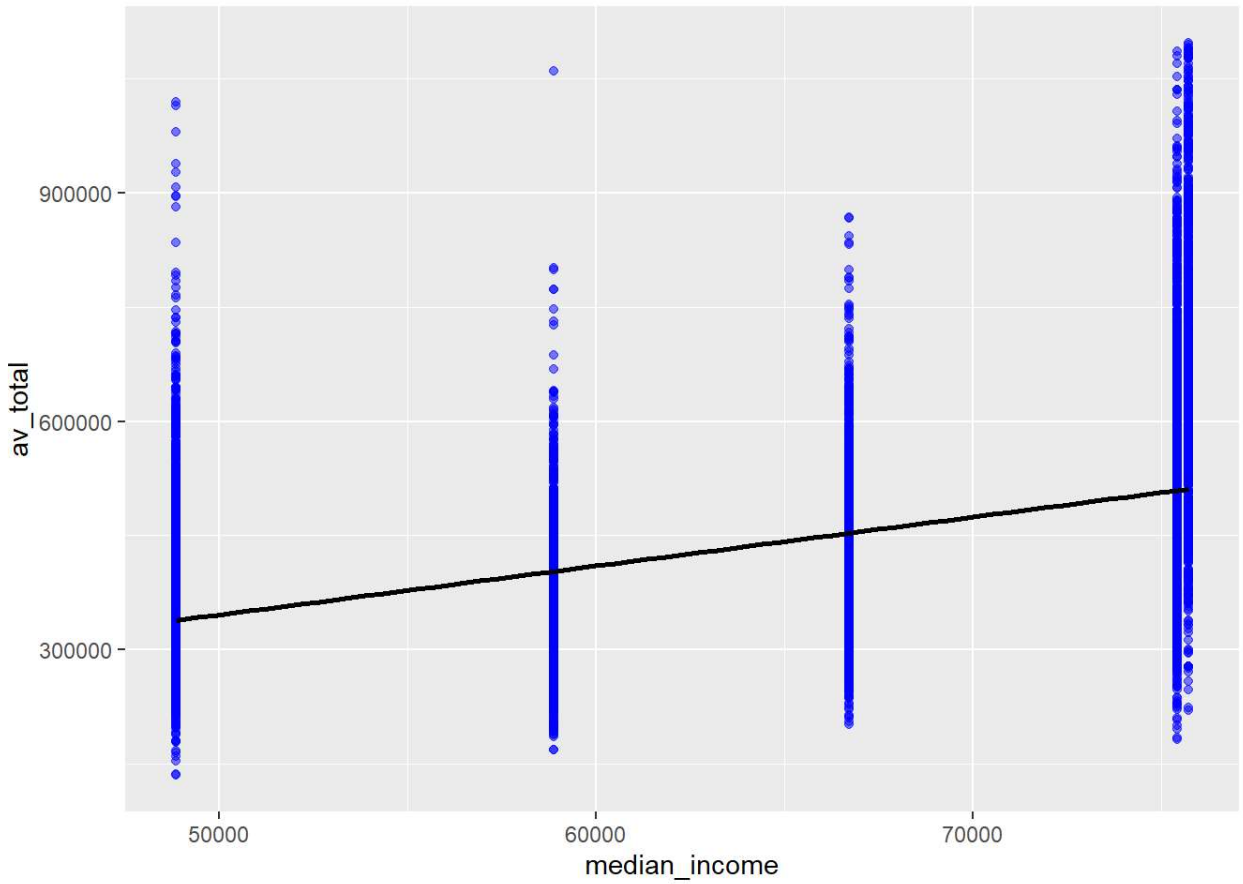
```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=population, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



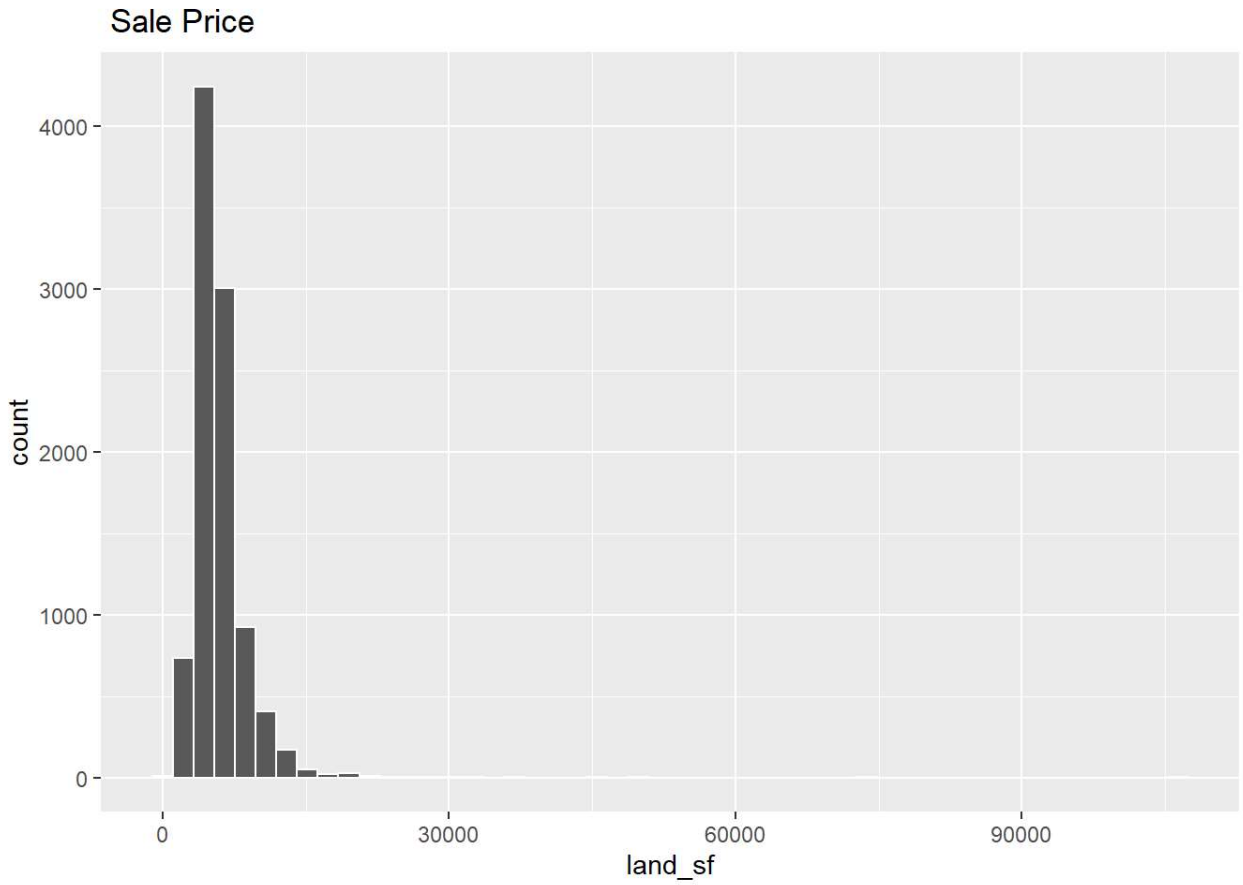

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=pop_density, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```

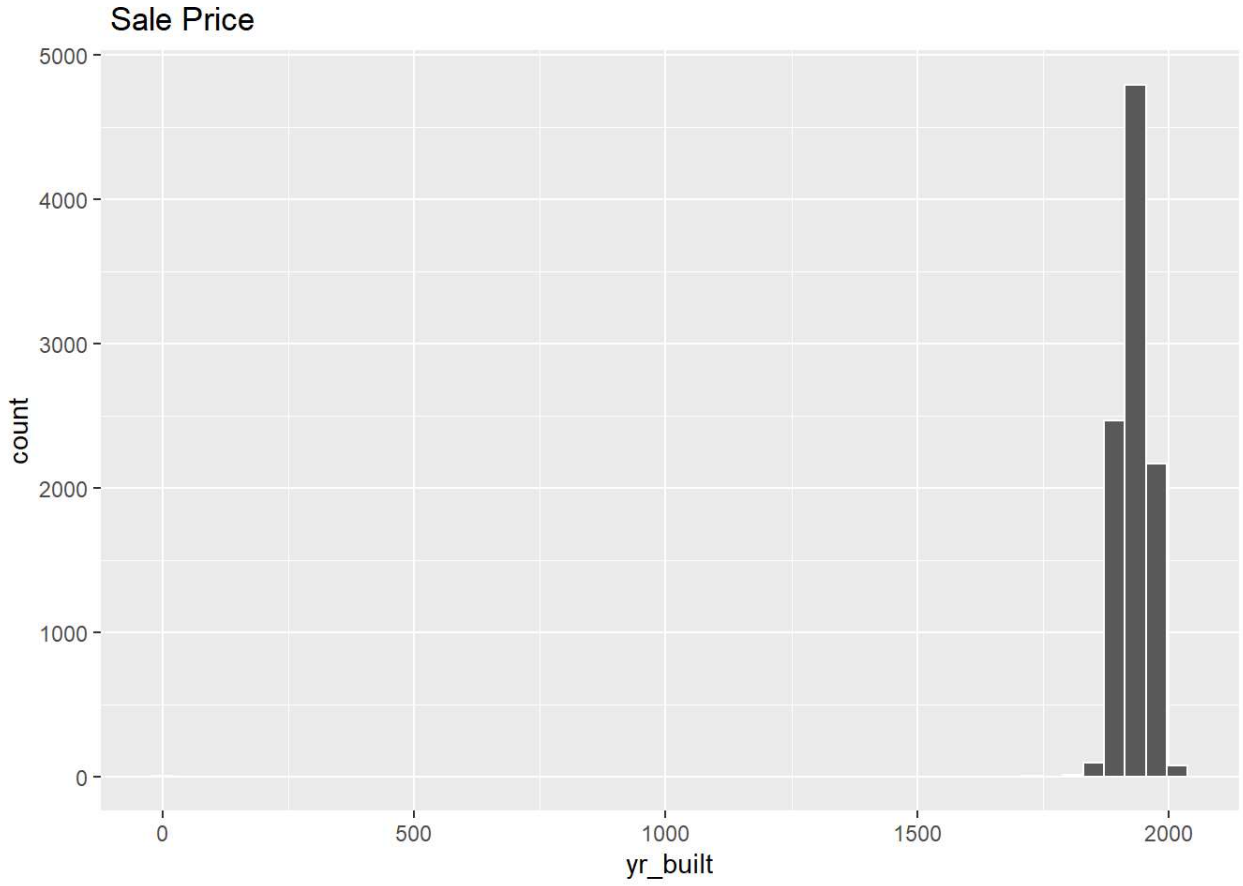


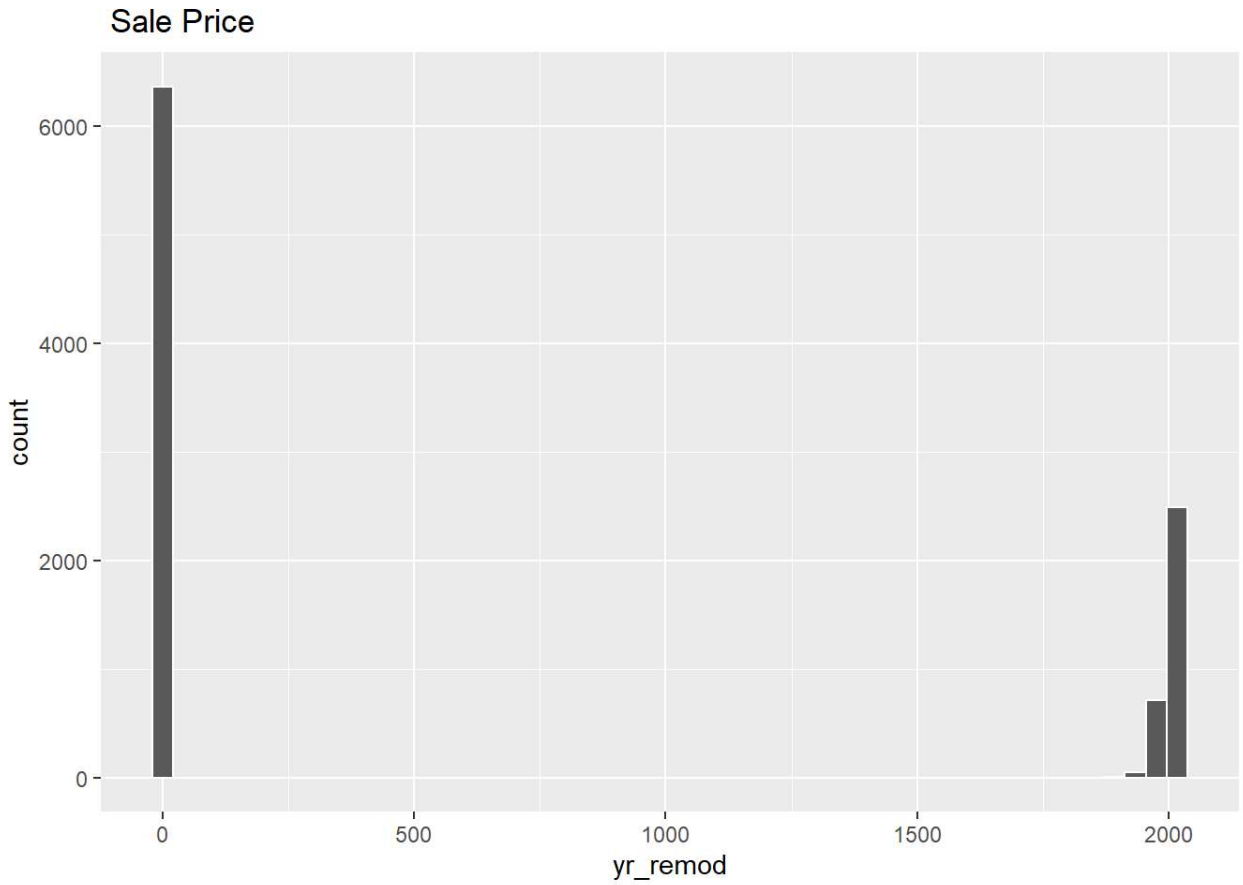

```
ggplot(data = boston[!is.na(boston$av_total),], aes(x=median_income, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```

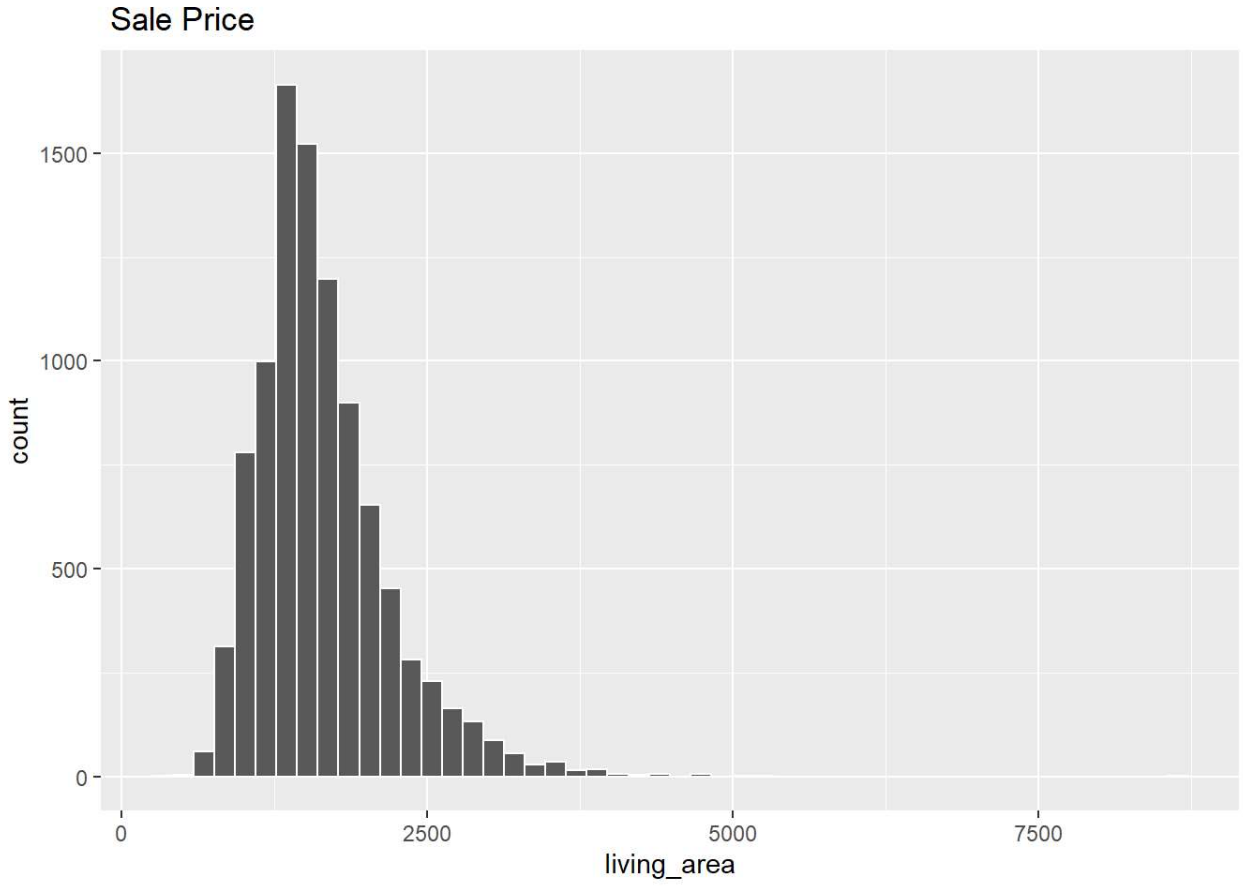


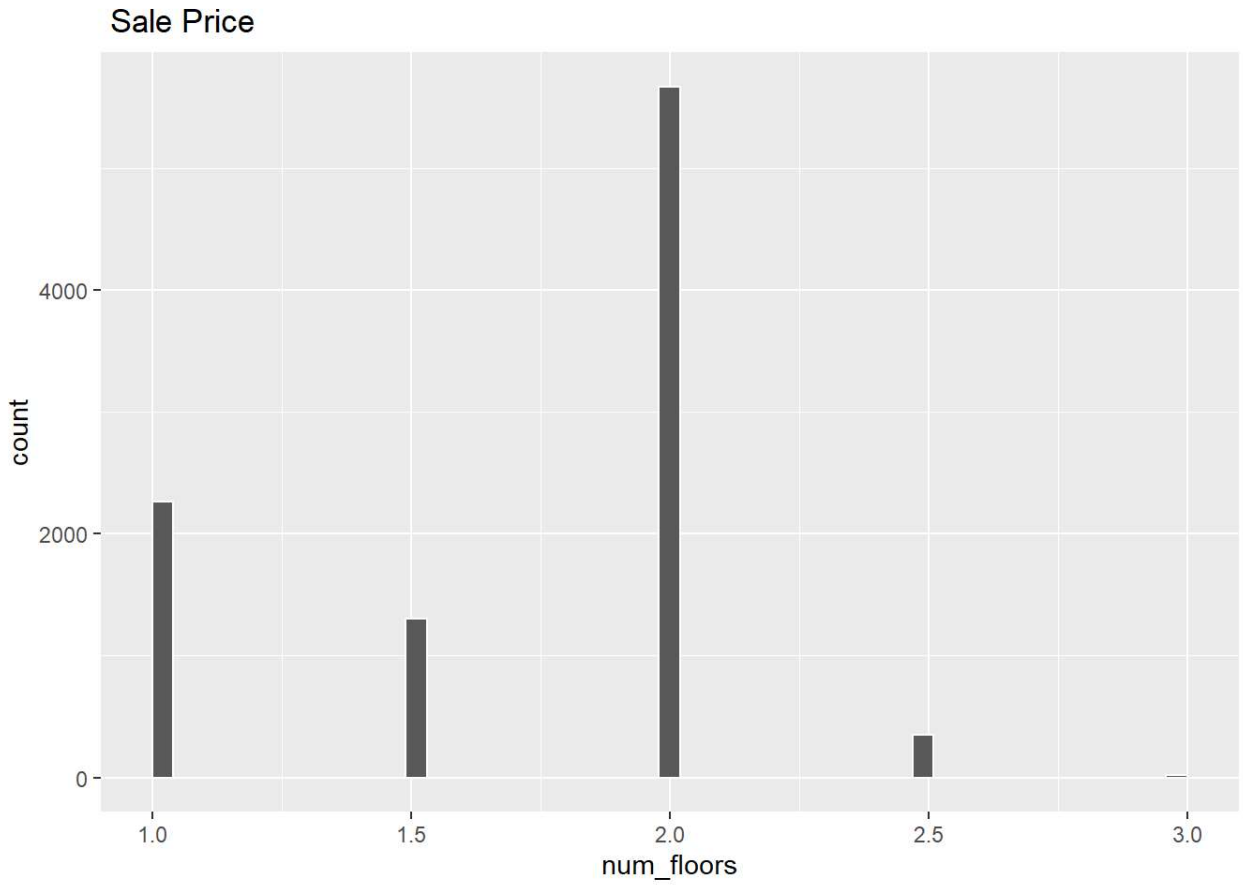
```
histogram <- function(m){
  options(scipen = 999)
  boston %>%
  na.omit() %>%
  ggplot(aes(x = !!as.name(m))) +
  geom_histogram(bins = 50, col= "white") +
  labs(title=" Sale Price")
}

numerics <- c('land_sf', 'yr_built', 'yr_remod', 'living_area', 'num_floors', 'r_total_rms', 'r_bdrms', 'r_full_bth',
         'r_half_bth', 'r_kitch', 'r_fplace', 'population', 'pop_density', 'median_income')


for (c in numerics){
    print(histogram(c))
}
```

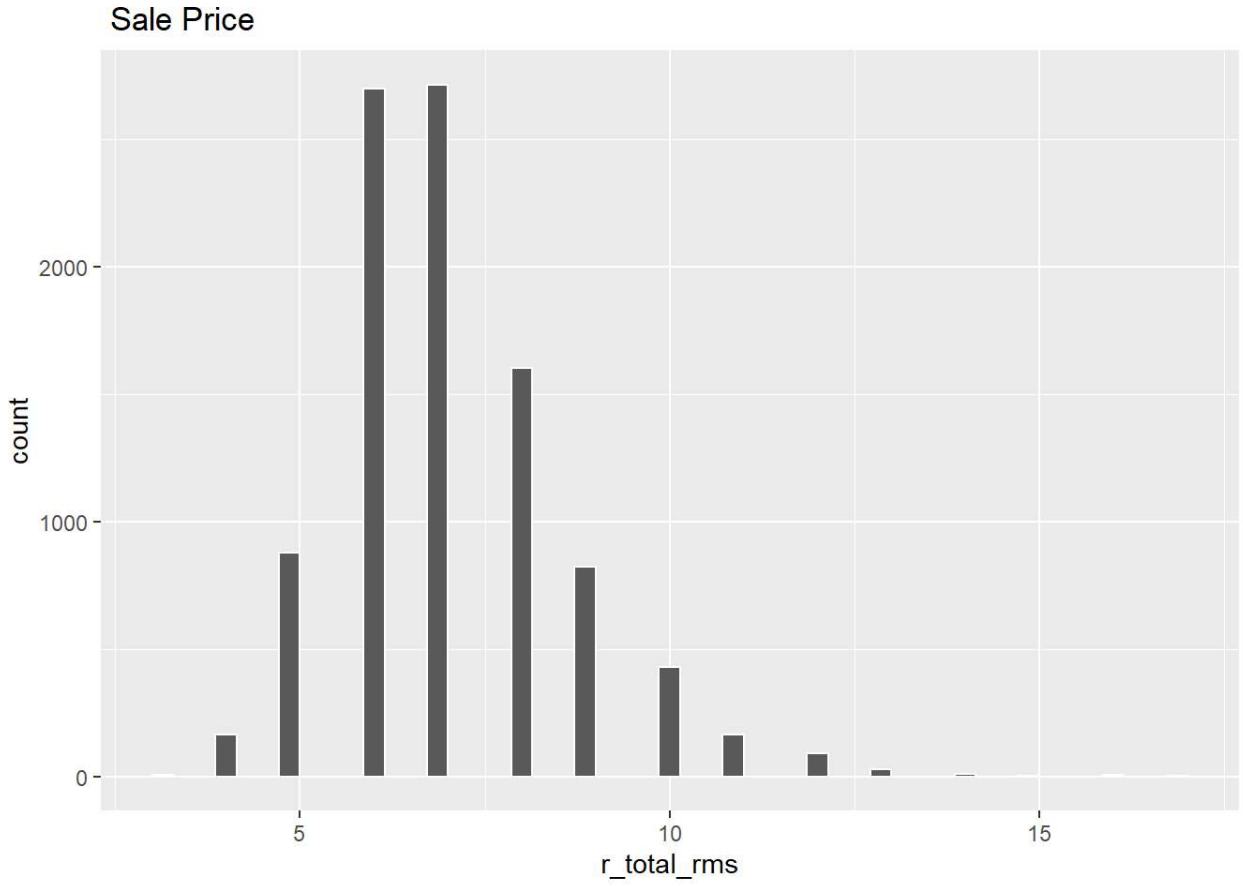

Sale Price
count
land_sf
0
1000
2000
3000
4000
0
30000
60000
90000

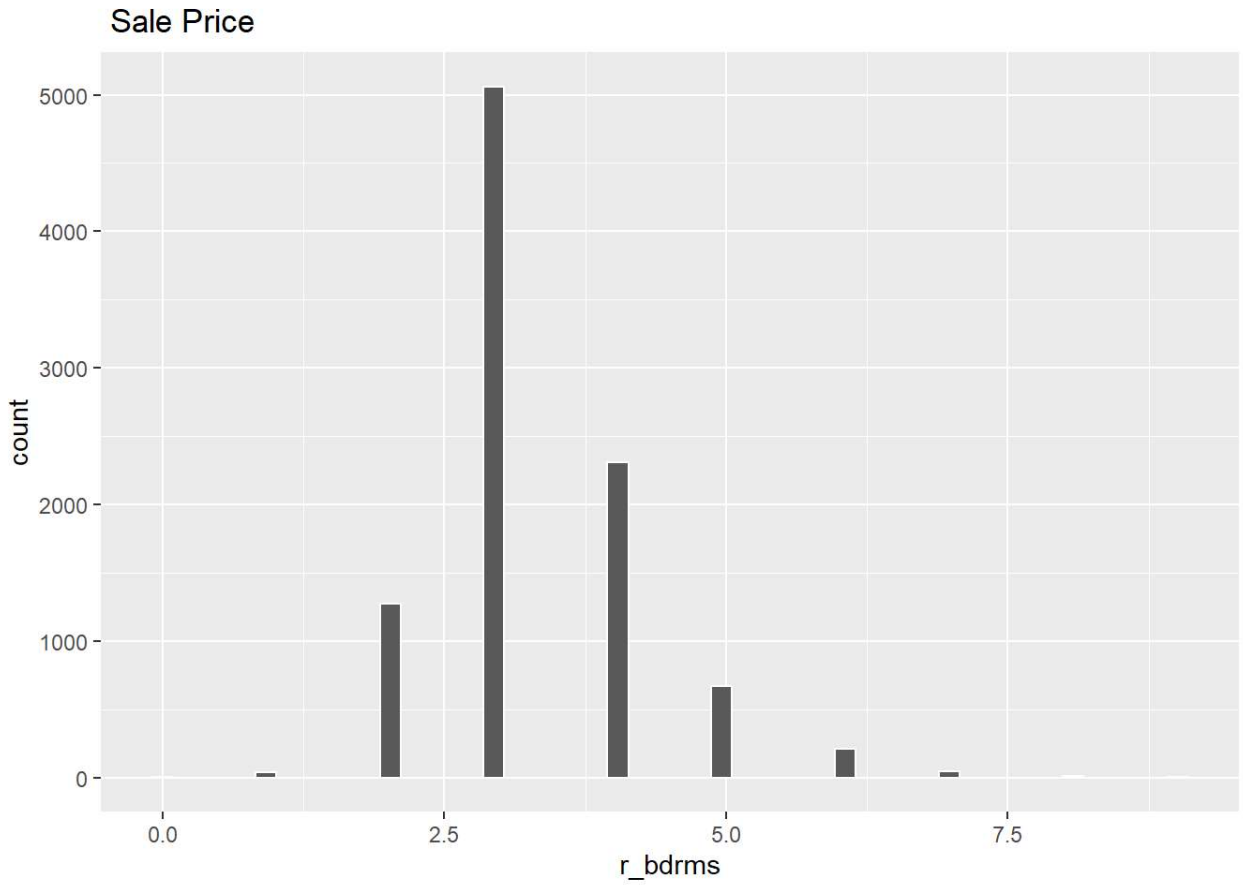

Sale Price
count
yr_built
0
1000
2000
3000
4000
5000
0
500
1000
1500
2000

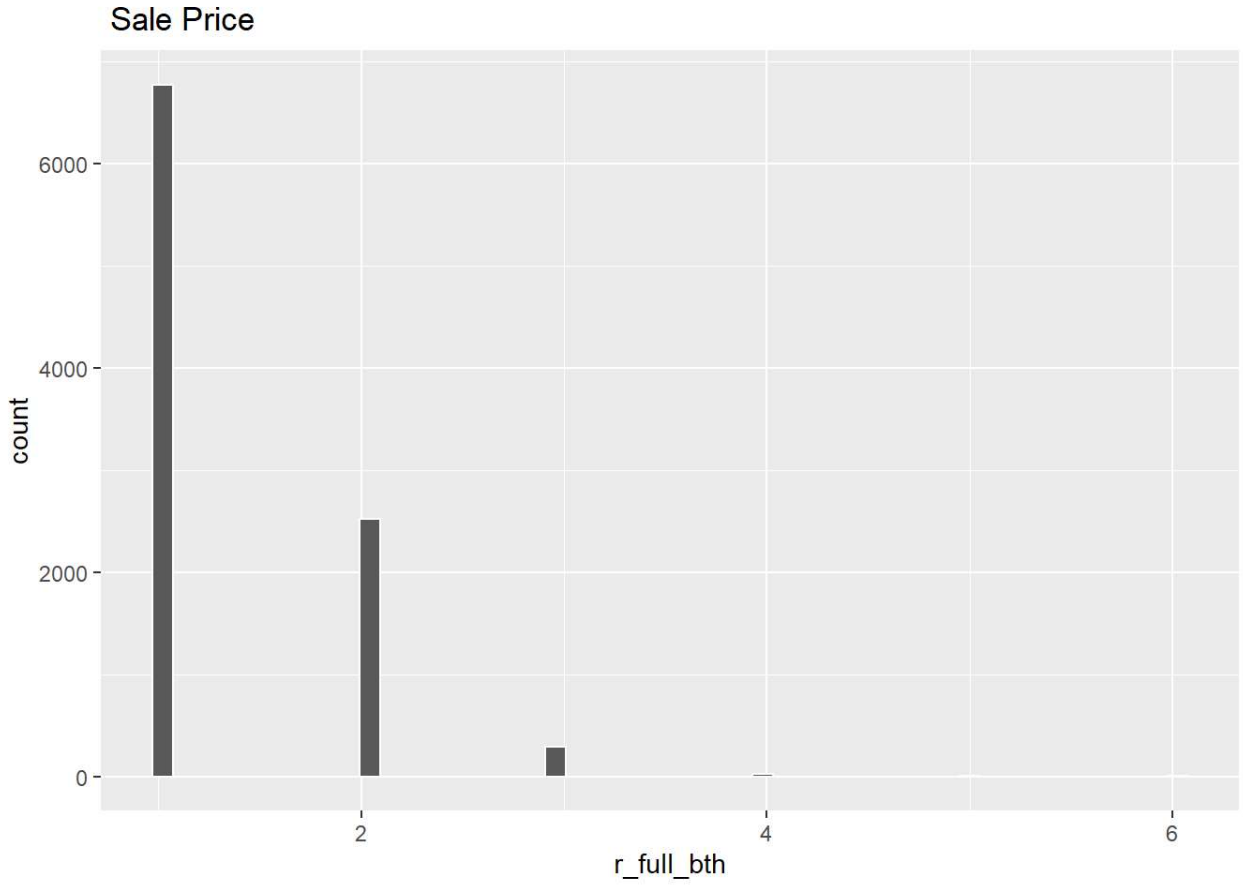

Sale Price
6000
4000
2000
0
count
0
500
1000
1500
2000
yr_remod

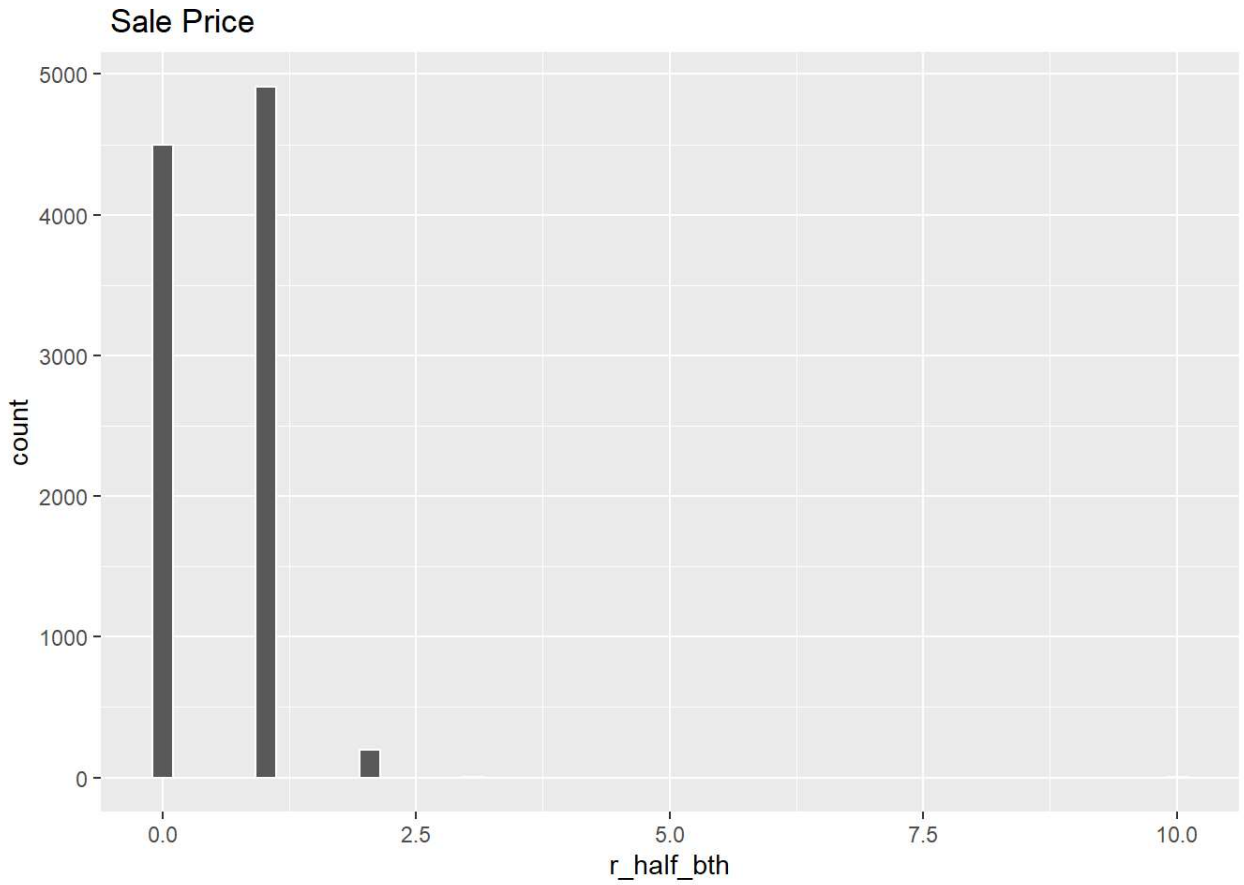

Sale Price
1500
1000
500
0
count
0
2500
5000
7500
living_area

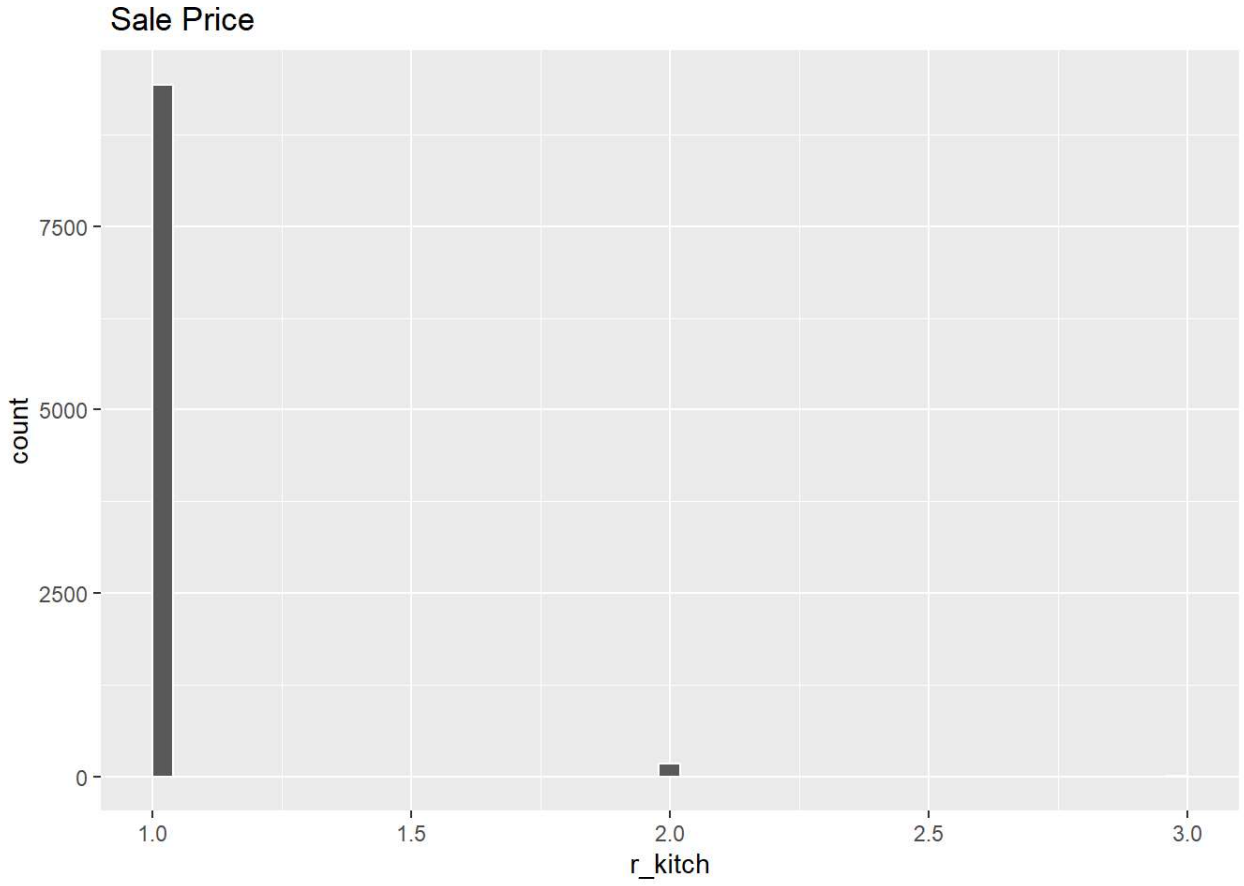

Sale Price
count
4000
2000
0
1.0
1.5
2.0
2.5
3.0
num_floors

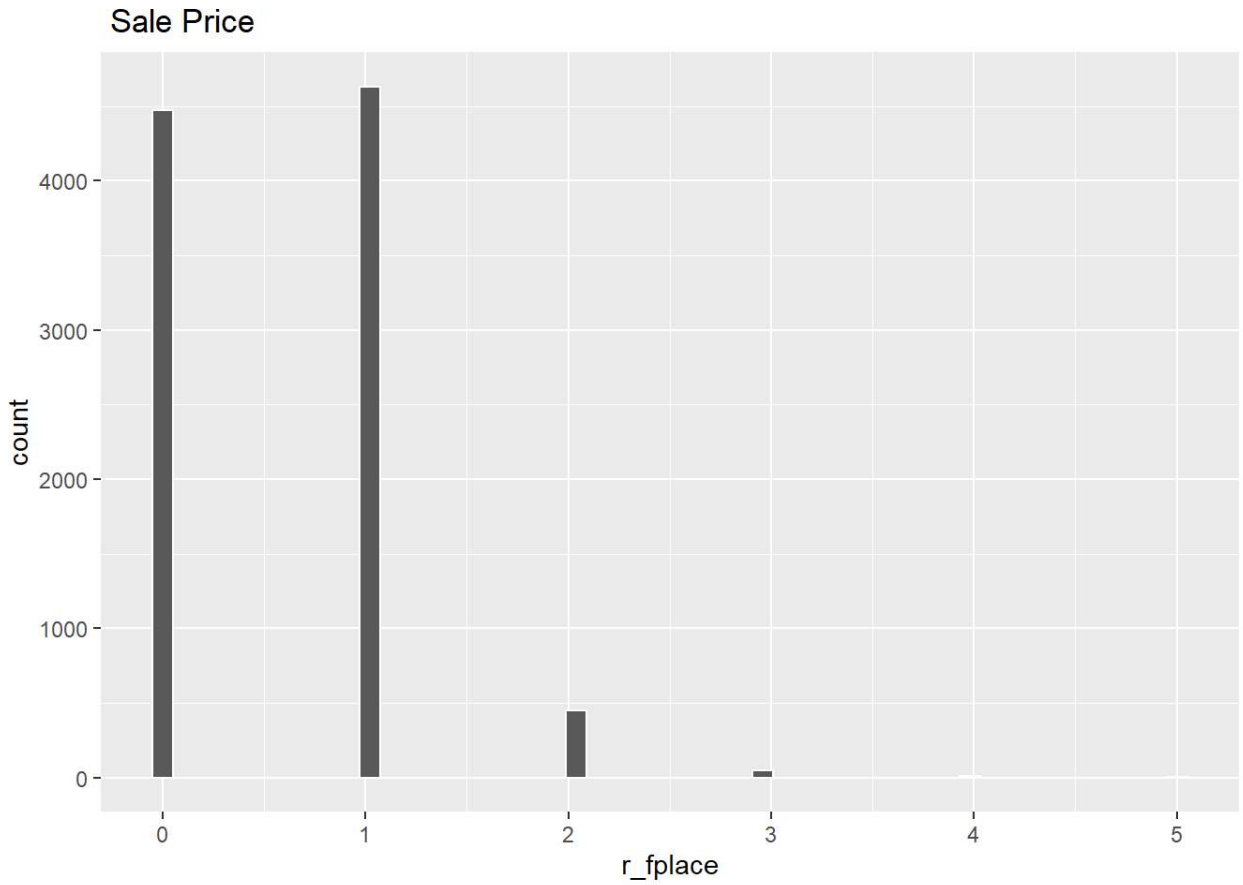

Sale Price
count
2000
1000
0
5
10
15
r_total_rms

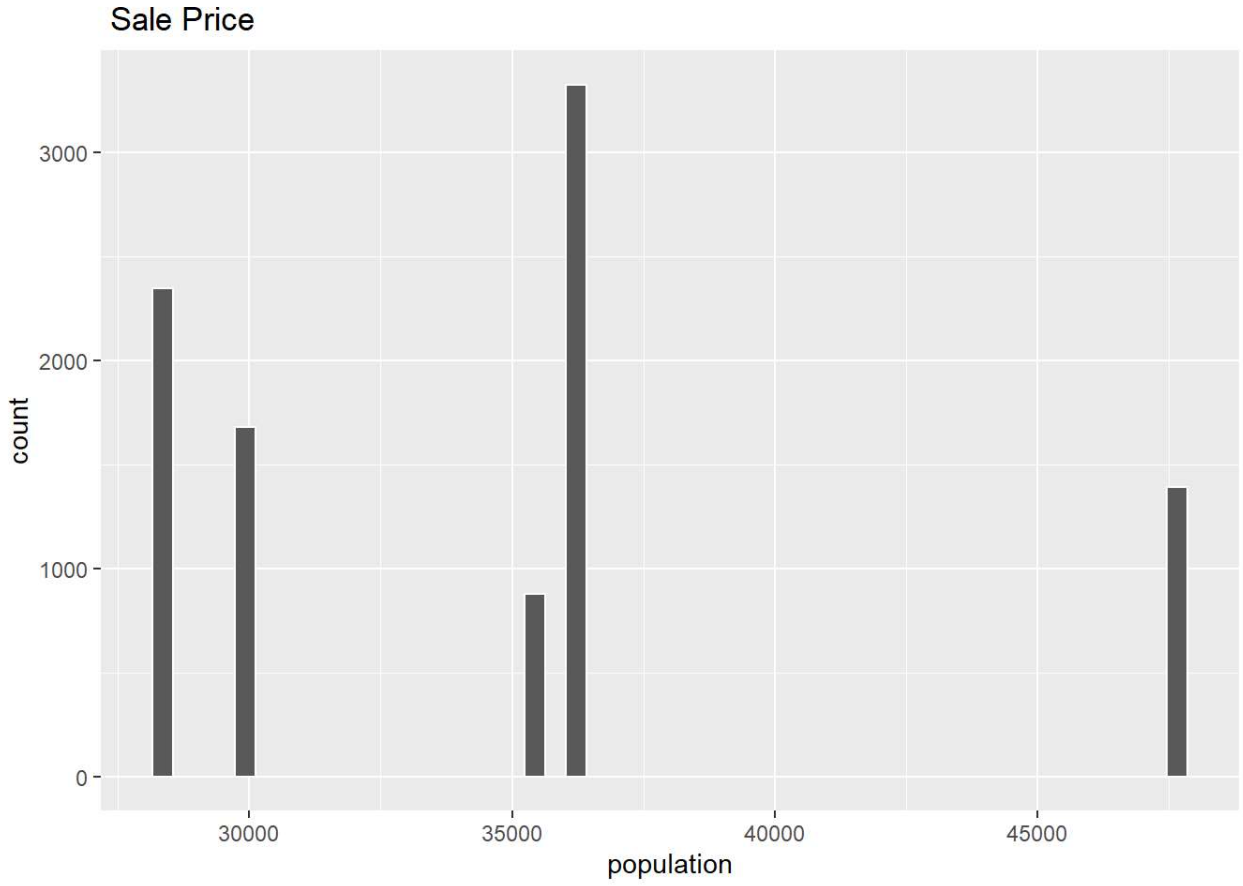

Sale Price
count
r_bdrms
0
1000
2000
3000
4000
5000
0.0
2.5
5.0
7.5

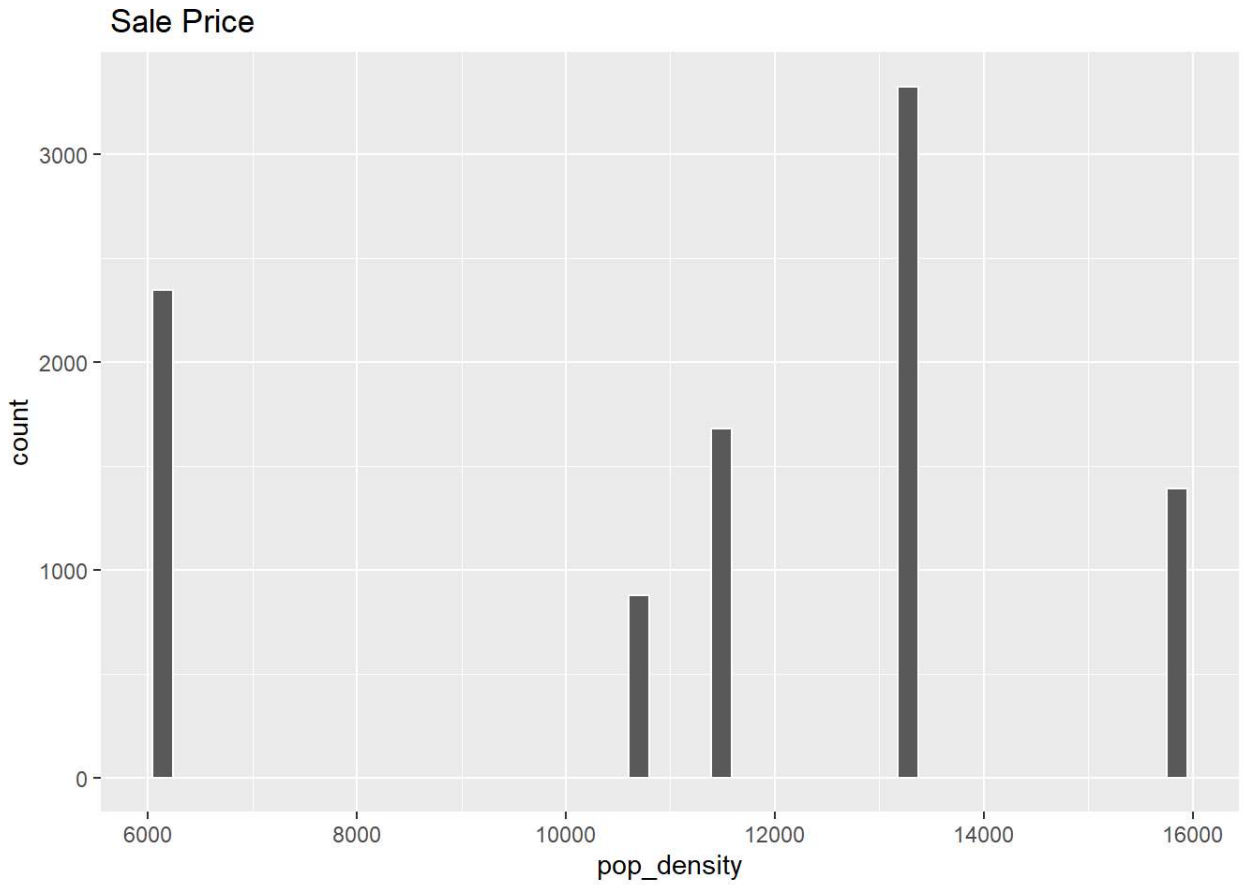

Sale Price
count
r_full_bth
0
2000
4000
6000
2
4
6

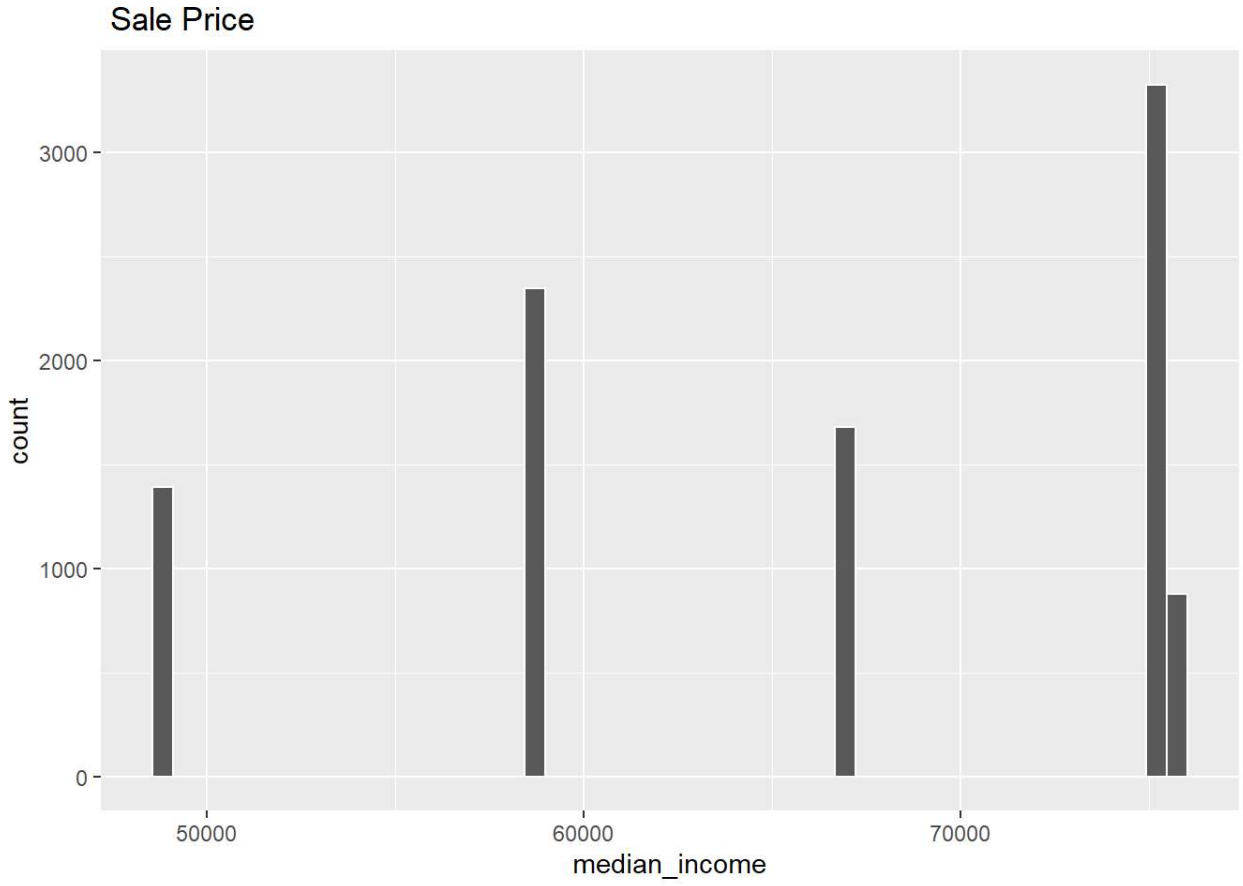

Sale Price
count
5000
4000
3000
2000
1000
0
0.0
2.5
5.0
7.5
10.0
r_half_bth


Sale Price
count
7500
5000
2500
0
1.0
1.5
2.0
2.5
3.0
r_kitch


Sale Price
count
4000
3000
2000
1000
0
0
1
2
3
4
5
r_fplace


Sale Price
count
3000
2000
1000
0
30000
35000
40000
45000
population

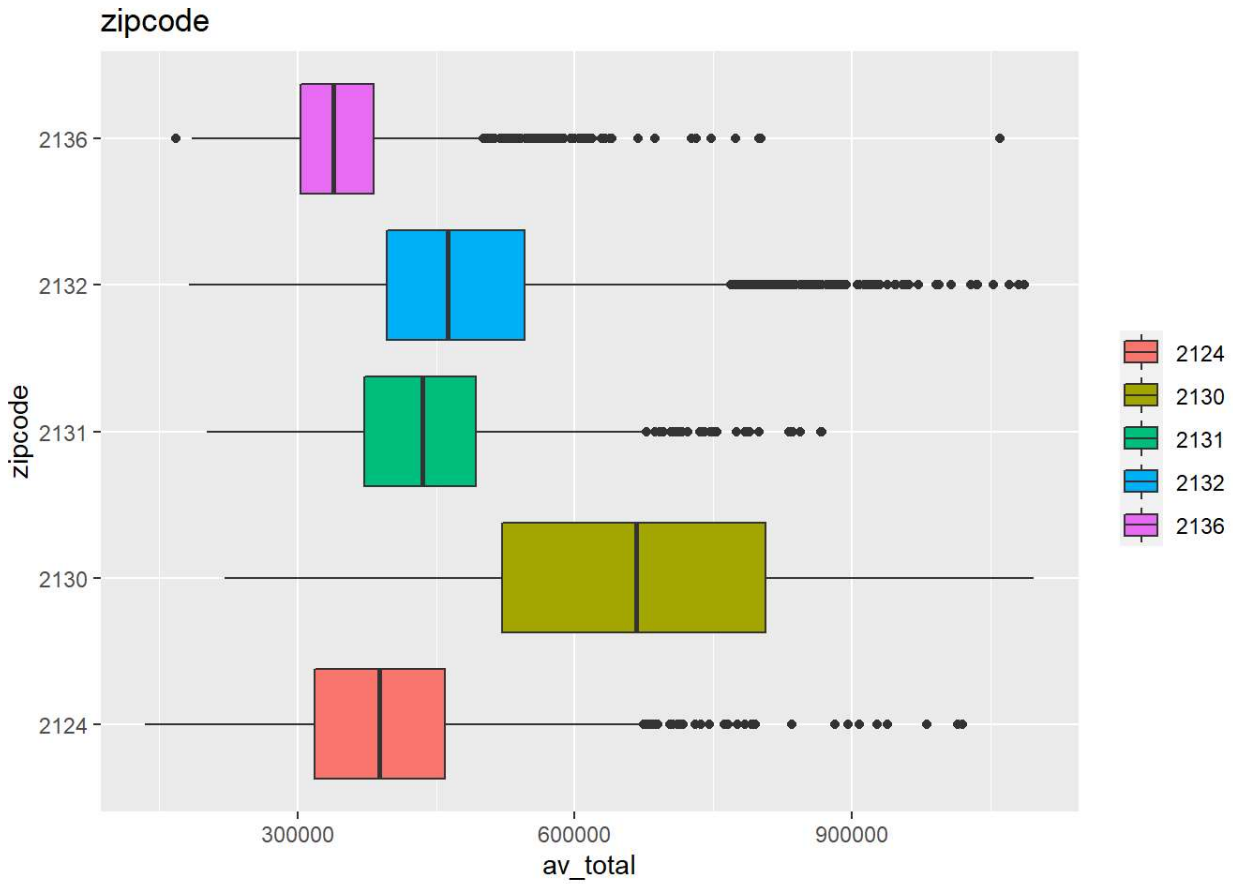




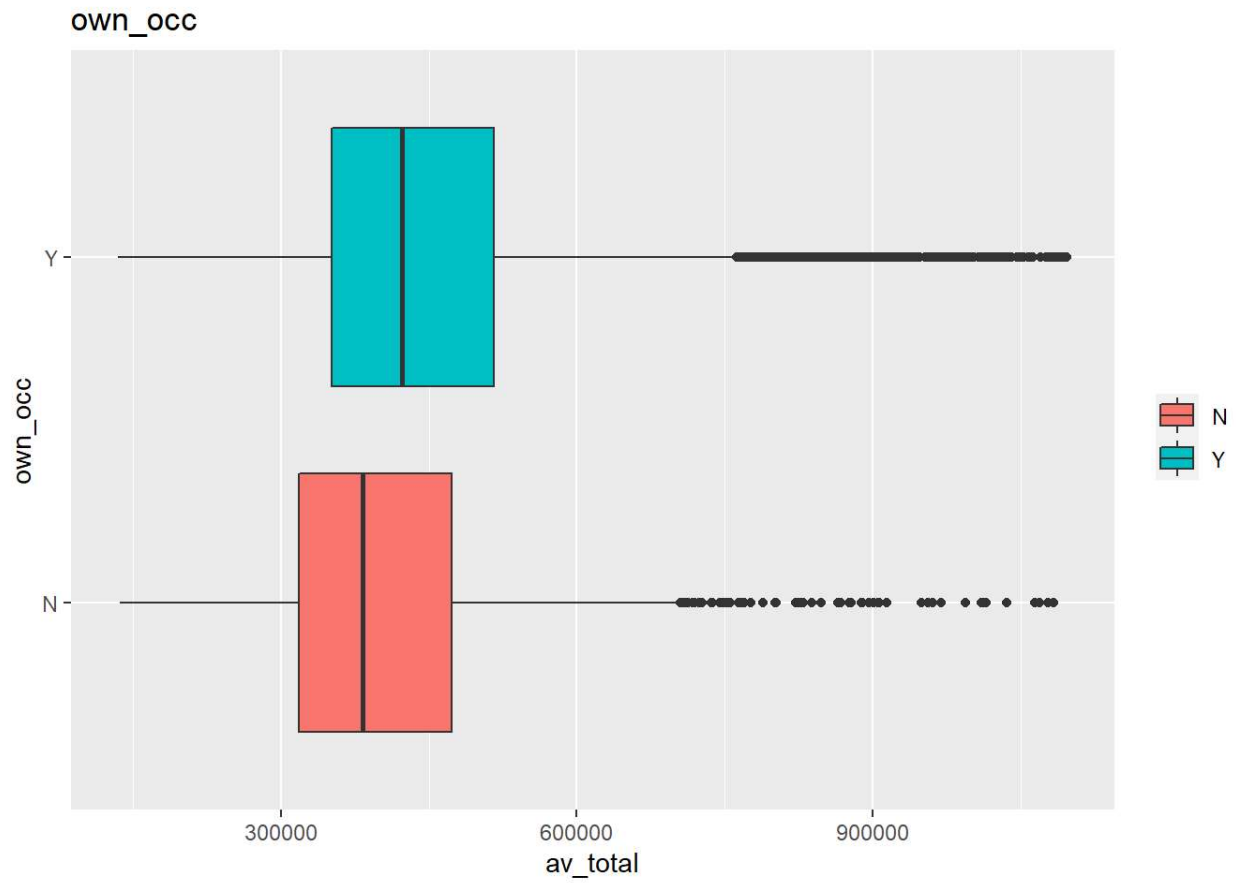
Explore character variables

categorical variables: zipcode, own_occ, structure_class, r_bldg_styl, r_roof_typ, r_ext_fin, r_bth_style, r_kitch_style, r_heat_typ, r_ac, r_ext_cnd, r_ovrall_cnd, r_int_cnd, r_int_fin, r_view, zip, city_state

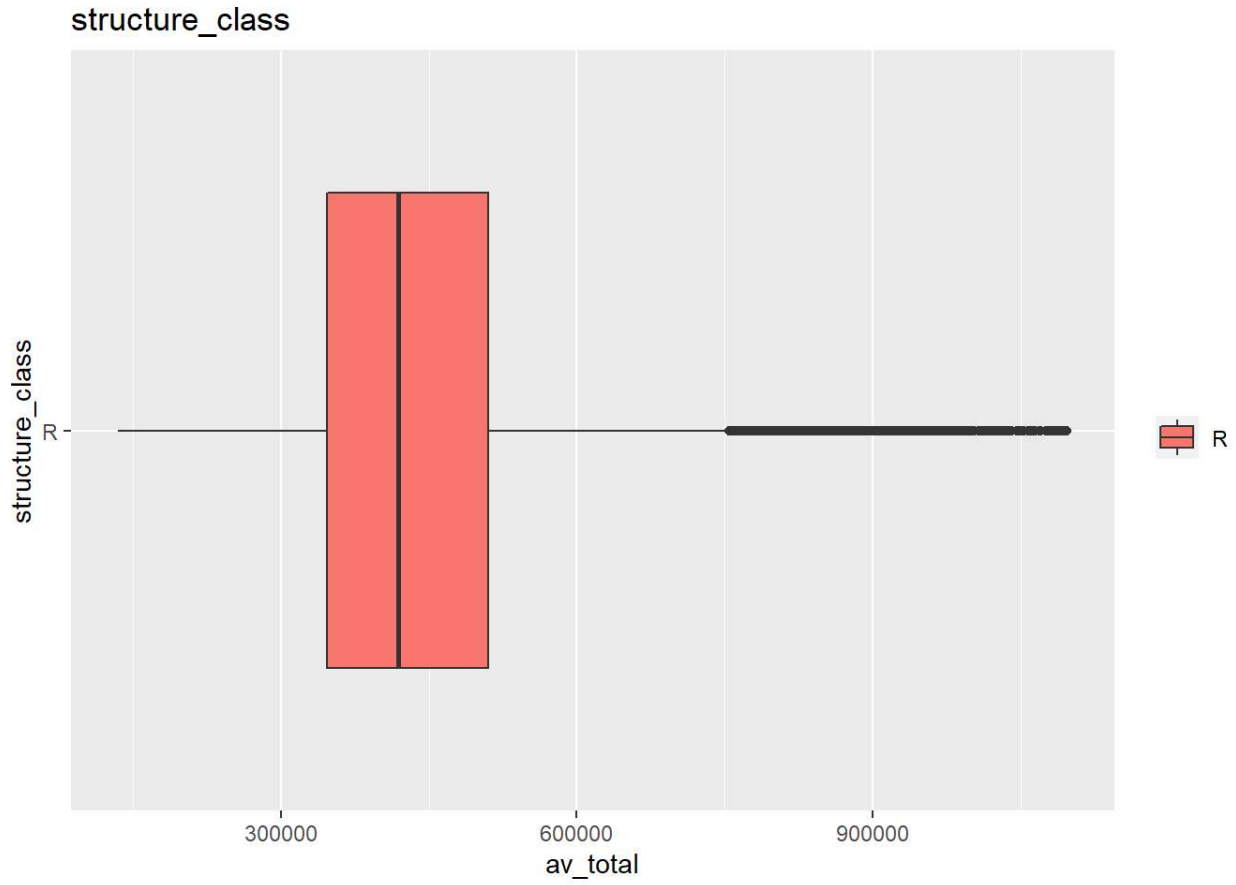
```
ggplot(boston, aes(x=av_total, y=as.factor(zipcode), fill=as.factor(zipcode))) +
  geom_boxplot() + labs(title = 'zipcode', x = 'av_total', y = 'zipcode') + theme(legend.title = element_blank())
```



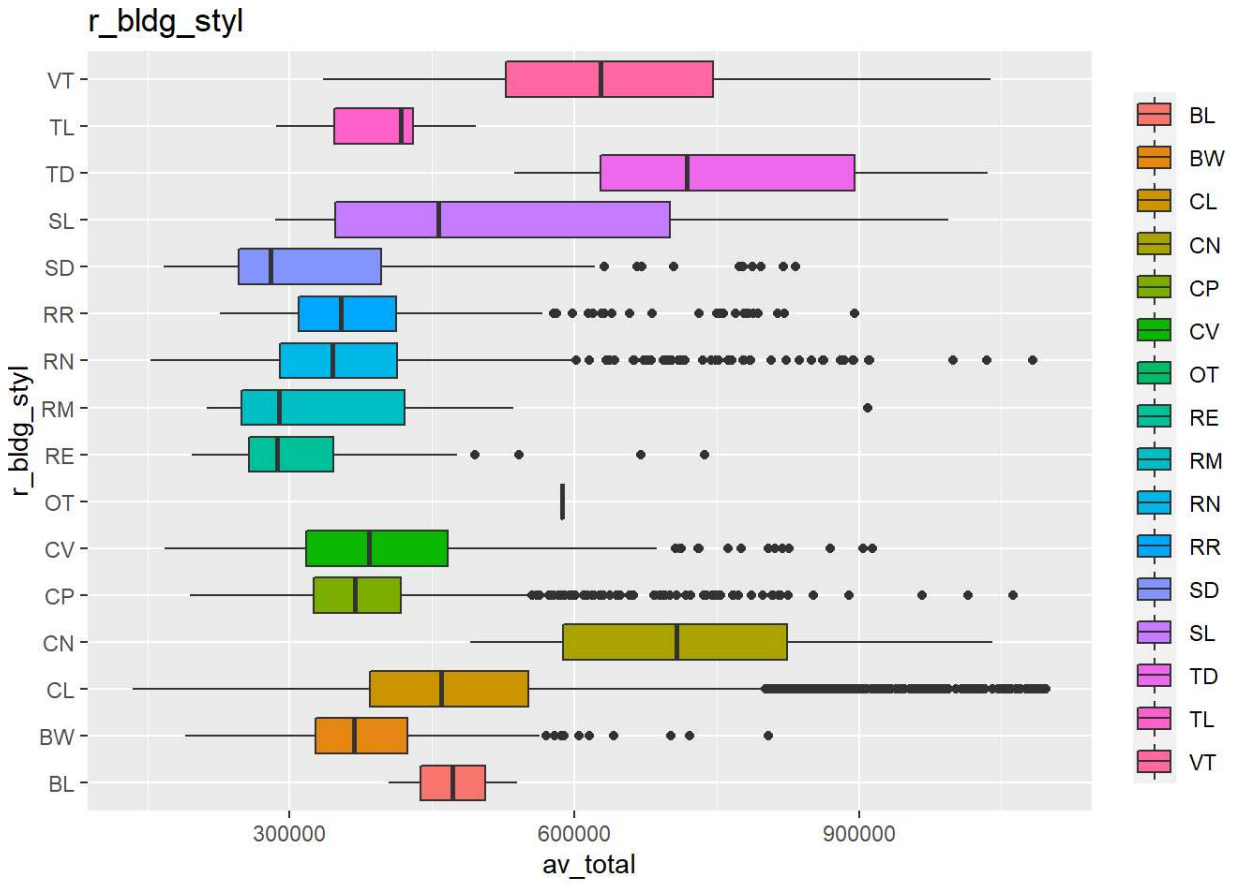

```
ggplot(boston, aes(x=av_total, y=as.factor(own_occ), fill=as.factor(own_occ))) +
  geom_boxplot() + labs(title = 'own_occ', x = 'av_total', y = 'own_occ') + theme(legend.title = element_blank())
```



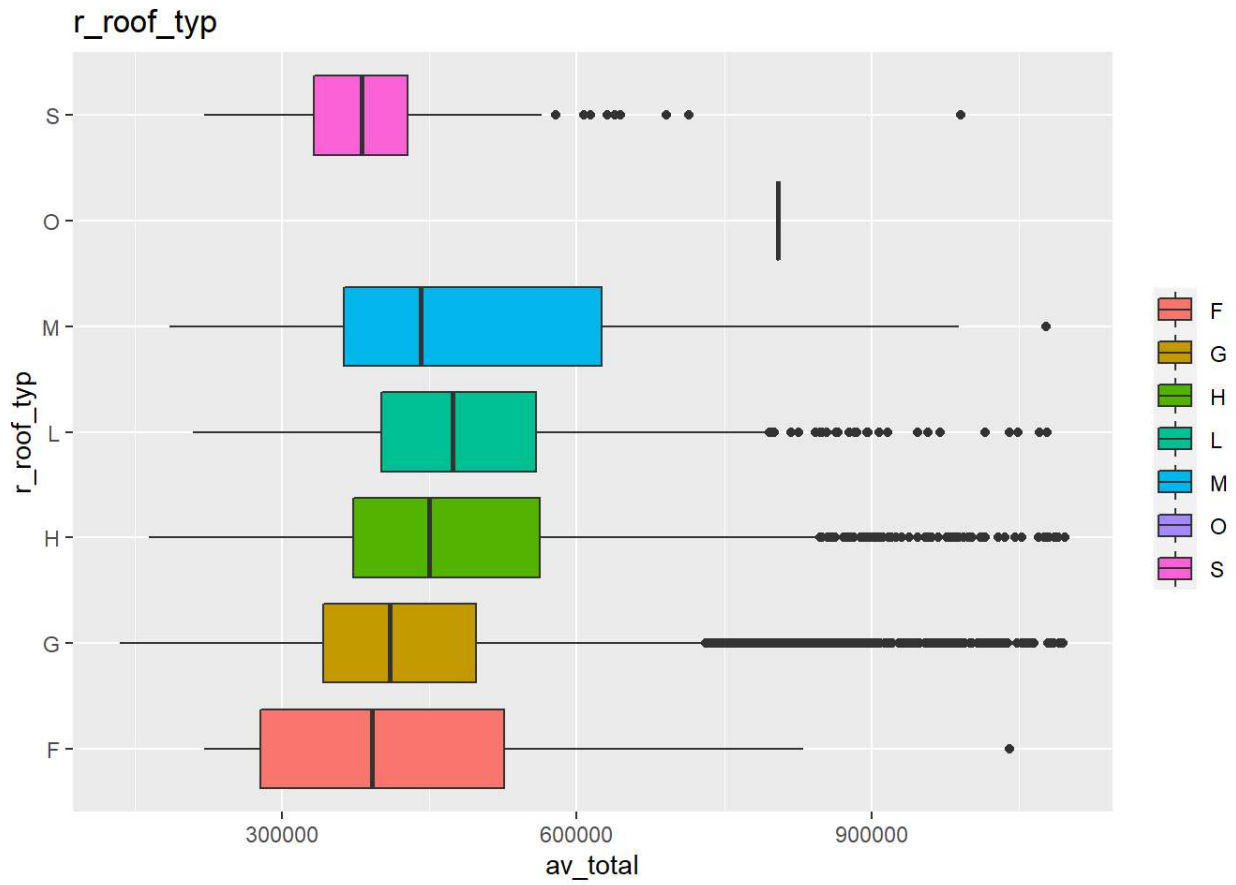

```
ggplot(boston, aes(x=av_total, y=as.factor(structure_class), fill=as.factor(structure_class))) +
  geom_boxplot() + labs(title = 'structure_class', x = 'av_total', y = 'structure_class') + theme(legend.title = elem
        ent_blank())
```



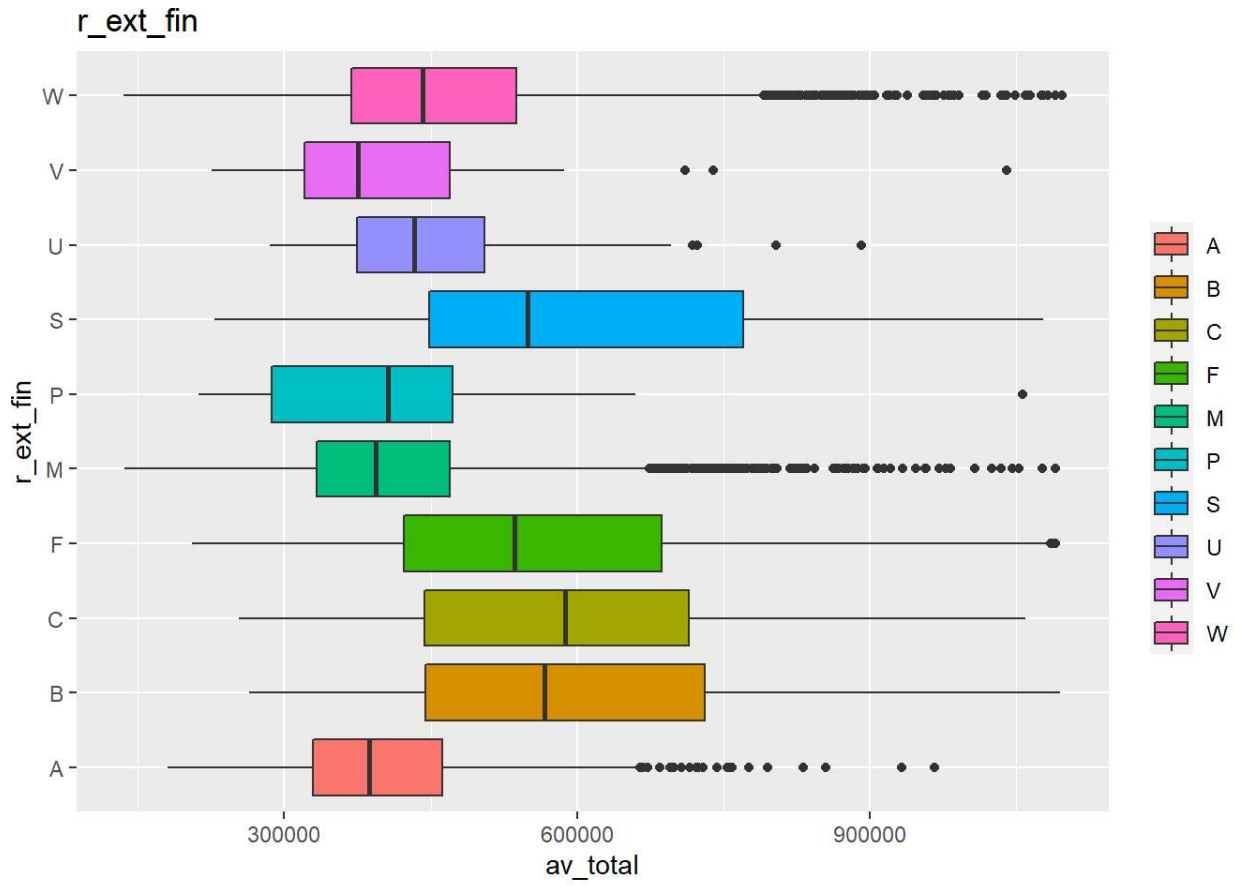
```
ggplot(boston, aes(x=av_total, y=as.factor(r_bldg_styl), fill=as.factor(r_bldg_styl))) +
  geom_boxplot() + labs(title = 'r_bldg_styl', x = 'av_total', y = 'r_bldg_styl') + theme(legend.title = element_blan
        k())
```



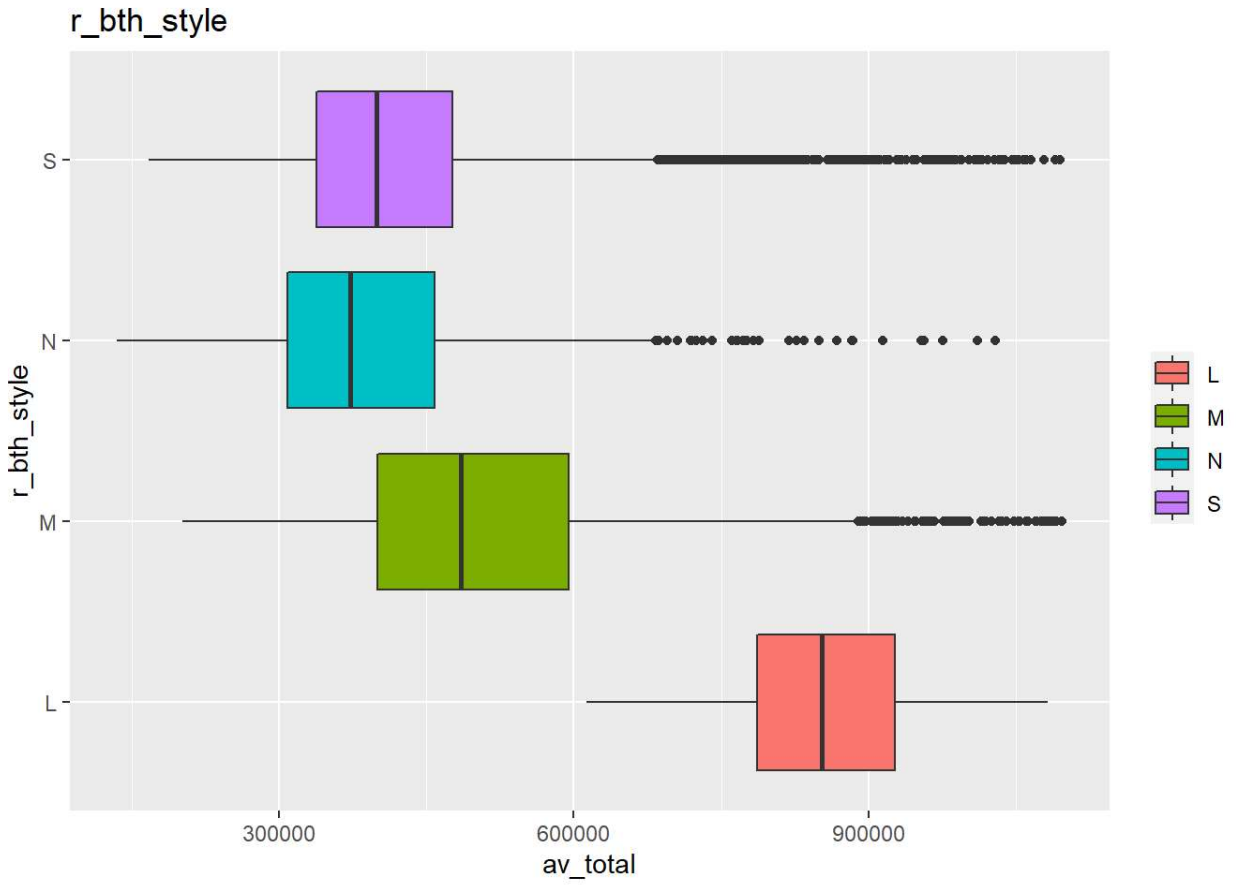

```
ggplot(boston, aes(x=av_total, y=as.factor(r_roof_typ), fill=as.factor(r_roof_typ))) +
  geom_boxplot() + labs(title = 'r_roof_typ', x = 'av_total', y = 'r_roof_typ') + theme(legend.title = element_blank
        ())
```



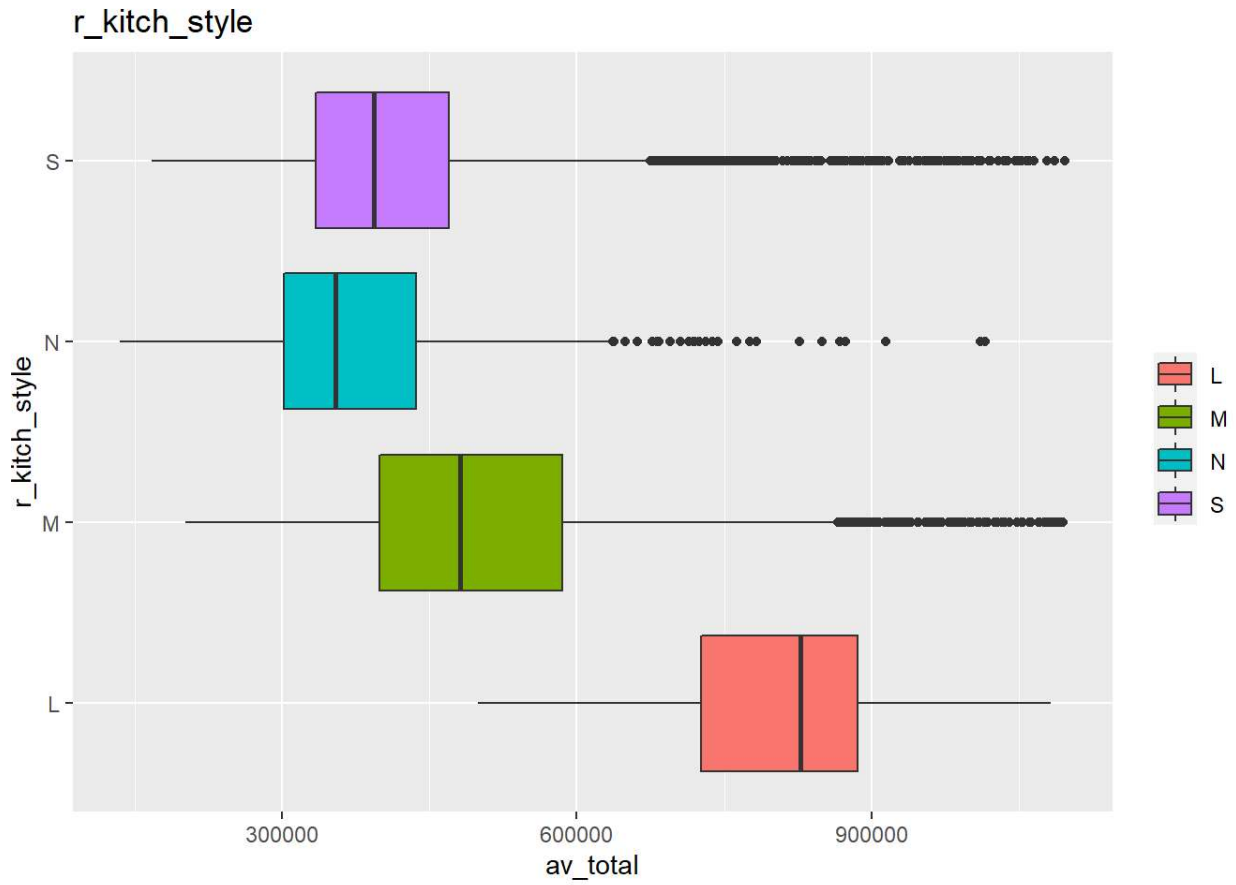

```
ggplot(boston, aes(x=av_total, y=as.factor(r_ext_fin), fill=as.factor(r_ext_fin))) +
  geom_boxplot() + labs(title = 'r_ext_fin', x = 'av_total', y = 'r_ext_fin') + theme(legend.title = element_blank())
```



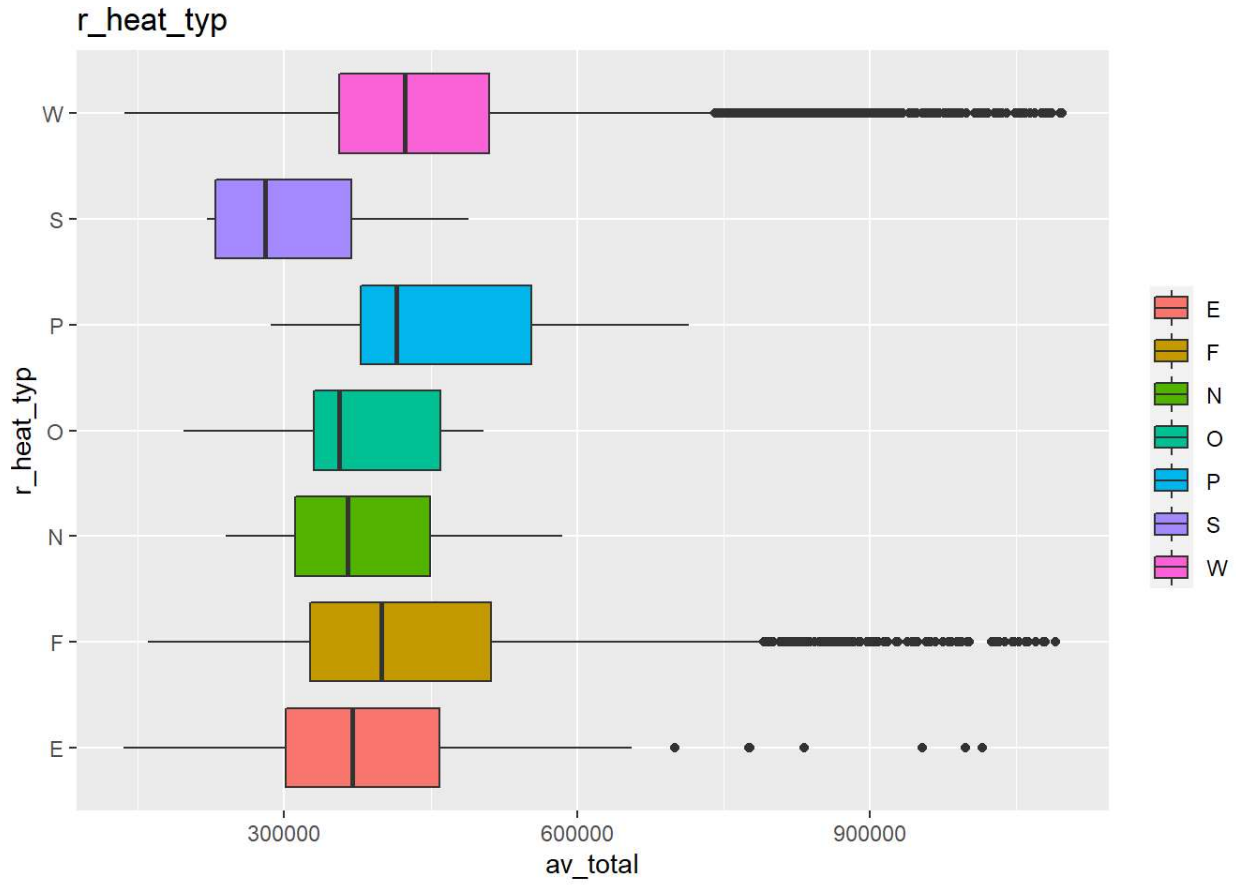
```
ggplot(boston, aes(x=av_total, y=as.factor(r_bth_style), fill=as.factor(r_bth_style))) +
  geom_boxplot() + labs(title = 'r_bth_style', x = 'av_total', y = 'r_bth_style') + theme(legend.title = element_blan
        k())
```



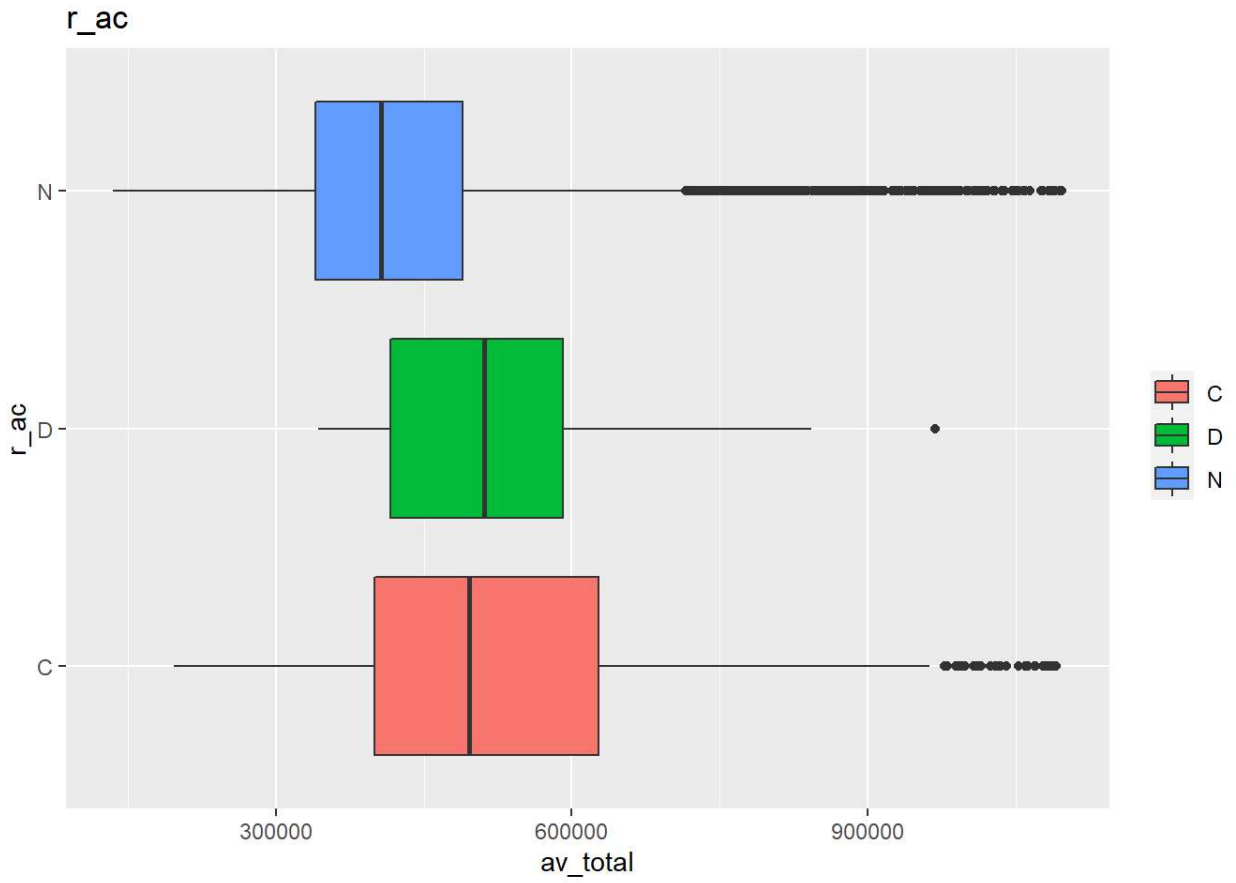

```
ggplot(boston, aes(x=av_total, y=as.factor(r_kitch_style), fill=as.factor(r_kitch_style))) +
  geom_boxplot() + labs(title = 'r_kitch_style', x = 'av_total', y = 'r_kitch_style') + theme(legend.title = element_
        blank())
```



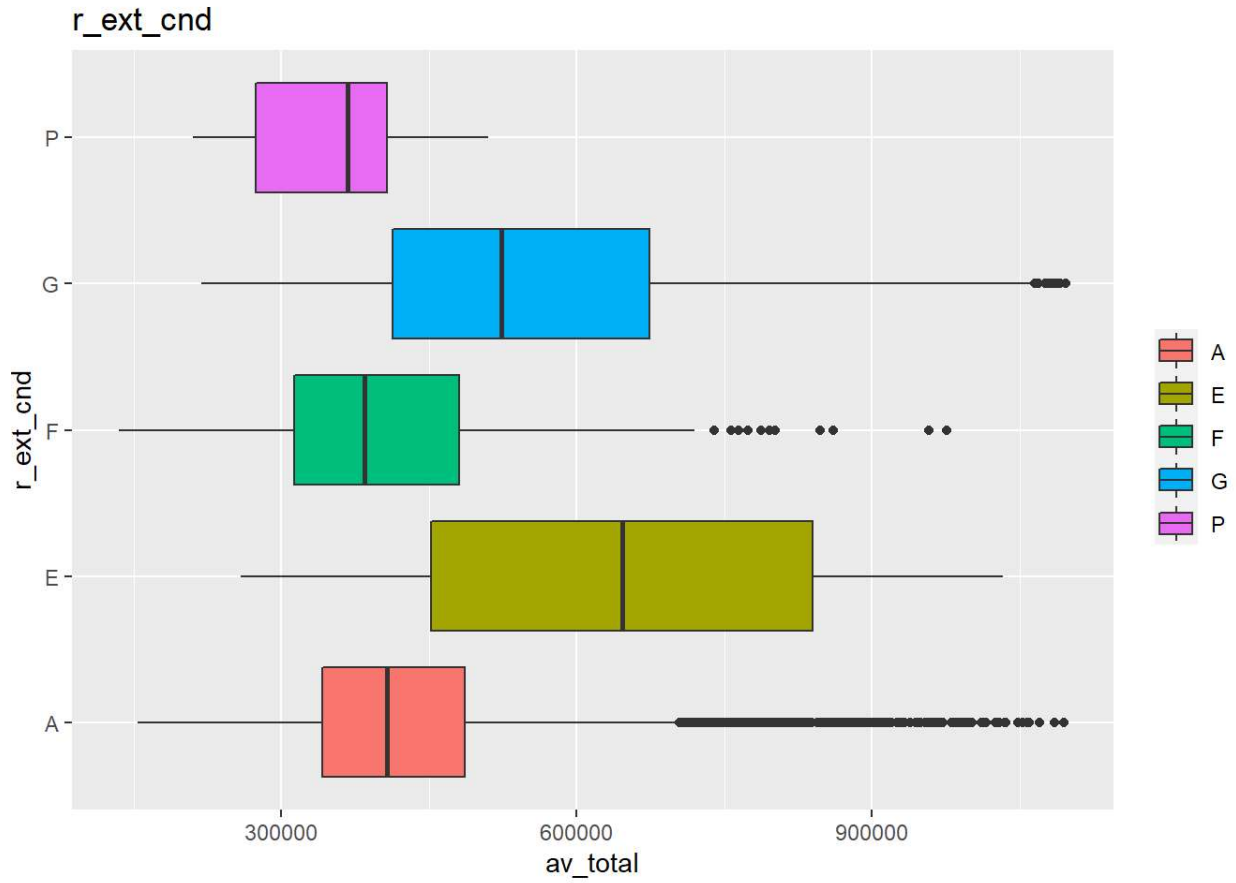

```
ggplot(boston, aes(x=av_total, y=as.factor(r_heat_typ), fill=as.factor(r_heat_typ))) +
  geom_boxplot() + labs(title = 'r_heat_typ', x = 'av_total', y = 'r_heat_typ') + theme(legend.title = element_blank
        ())
```



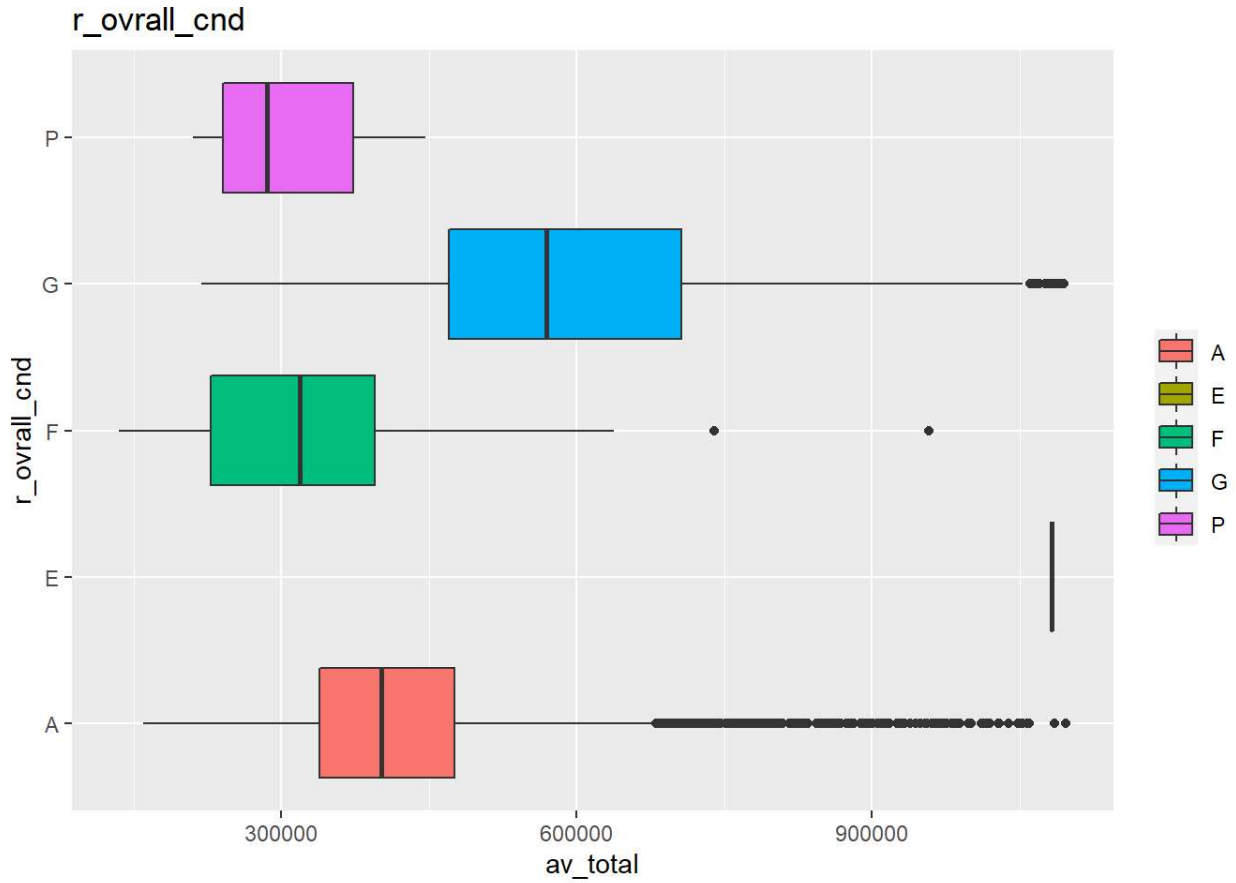
```
ggplot(boston, aes(x=av_total, y=as.factor(r_ac), fill=as.factor(r_ac))) +
  geom_boxplot() + labs(title = 'r_ac', x = 'av_total', y = 'r_ac') + theme(legend.title = element_blank())
```



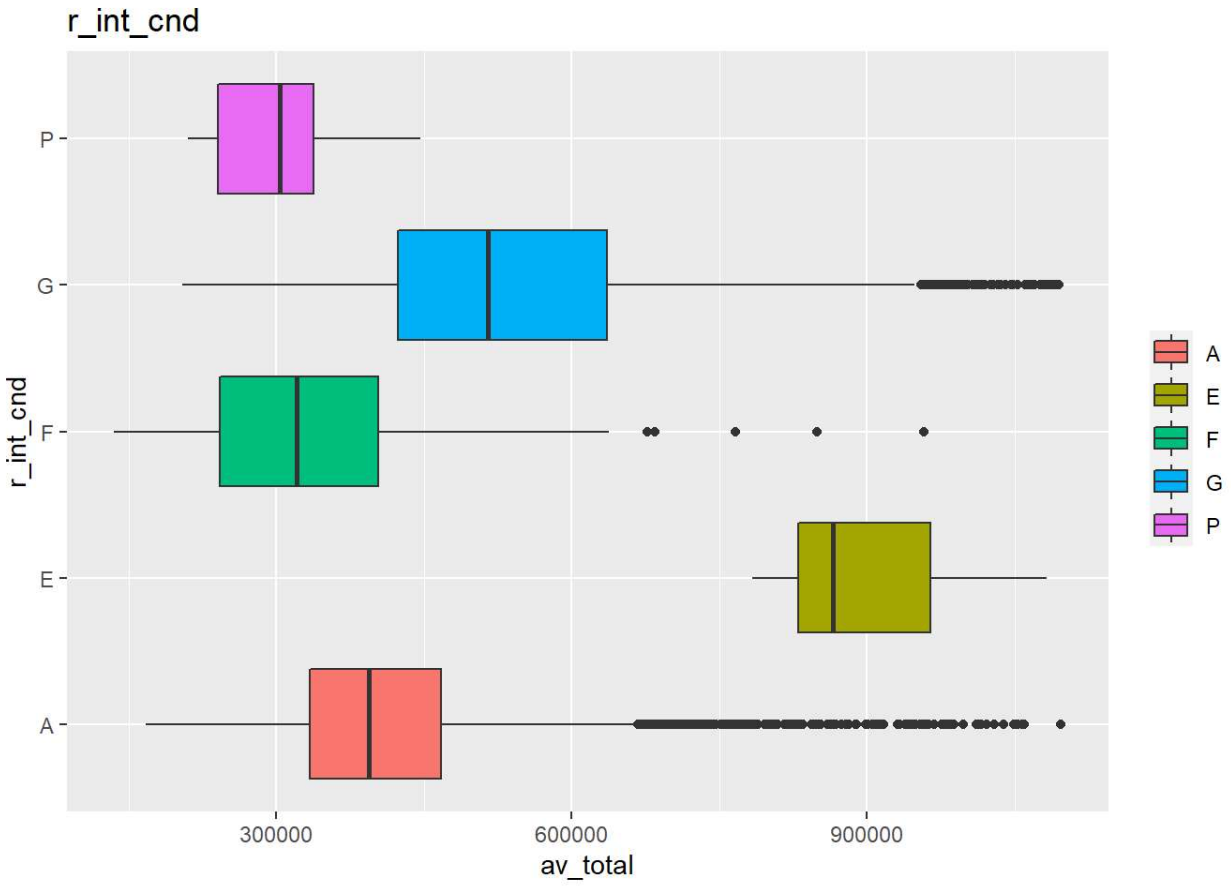

```
ggplot(boston, aes(x=av_total, y=as.factor(r_ext_cnd), fill=as.factor(r_ext_cnd))) +
  geom_boxplot() + labs(title = 'r_ext_cnd', x = 'av_total', y = 'r_ext_cnd') + theme(legend.title = element_blank())
```



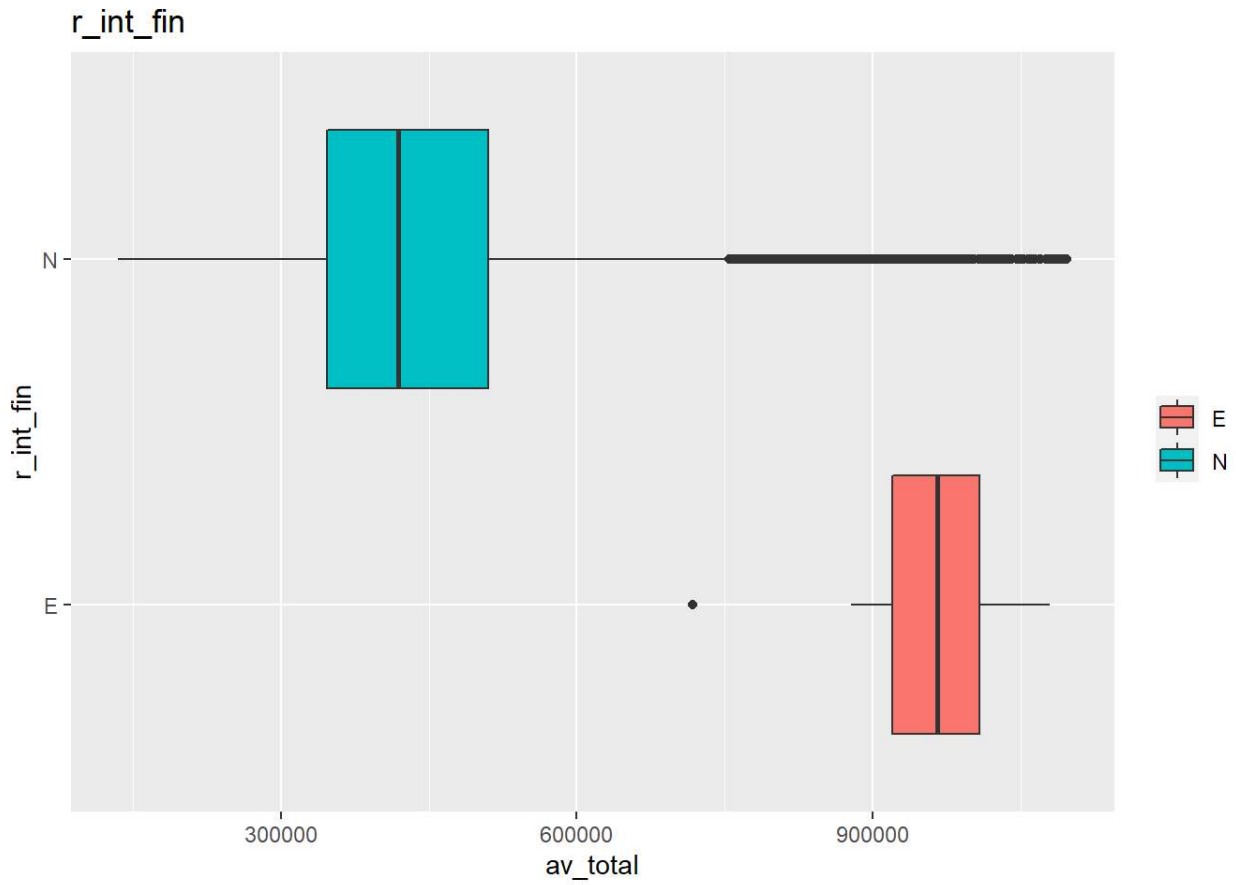

```
ggplot(boston, aes(x=av_total, y=as.factor(r_ovrall_cnd), fill=as.factor(r_ovrall_cnd))) +
  geom_boxplot() + labs(title = 'r_ovrall_cnd', x = 'av_total', y = 'r_ovrall_cnd') + theme(legend.title = element_bl
        ank())
```



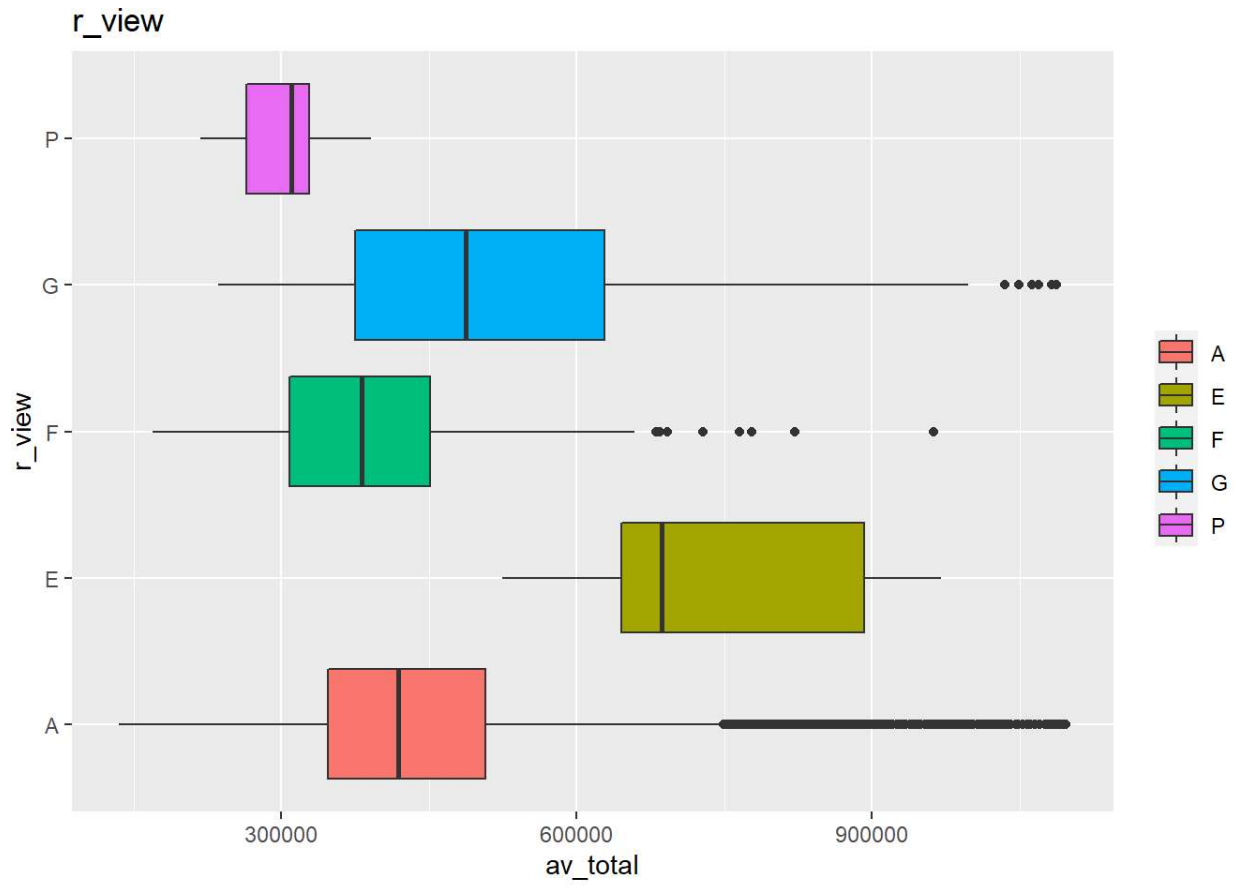
```
ggplot(boston, aes(x=av_total, y=as.factor(r_int_cnd), fill=as.factor(r_int_cnd))) +
  geom_boxplot() + labs(title = 'r_int_cnd', x = 'av_total', y = 'r_int_cnd') + theme(legend.title = element_blank())
```



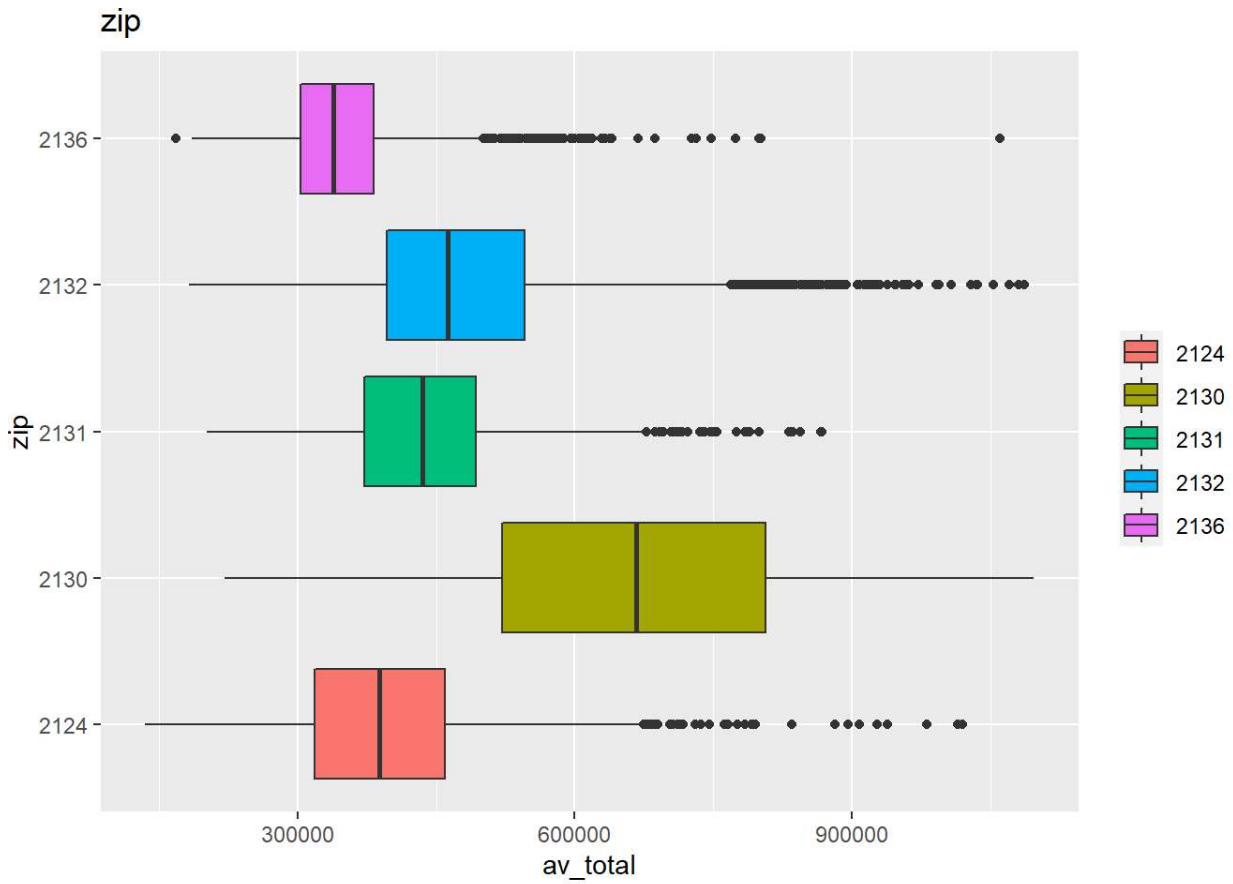

```
ggplot(boston, aes(x=av_total, y=as.factor(r_int_fin), fill=as.factor(r_int_fin))) +
  geom_boxplot() + labs(title = 'r_int_fin', x = 'av_total', y = 'r_int_fin') + theme(legend.title = element_blank())
```



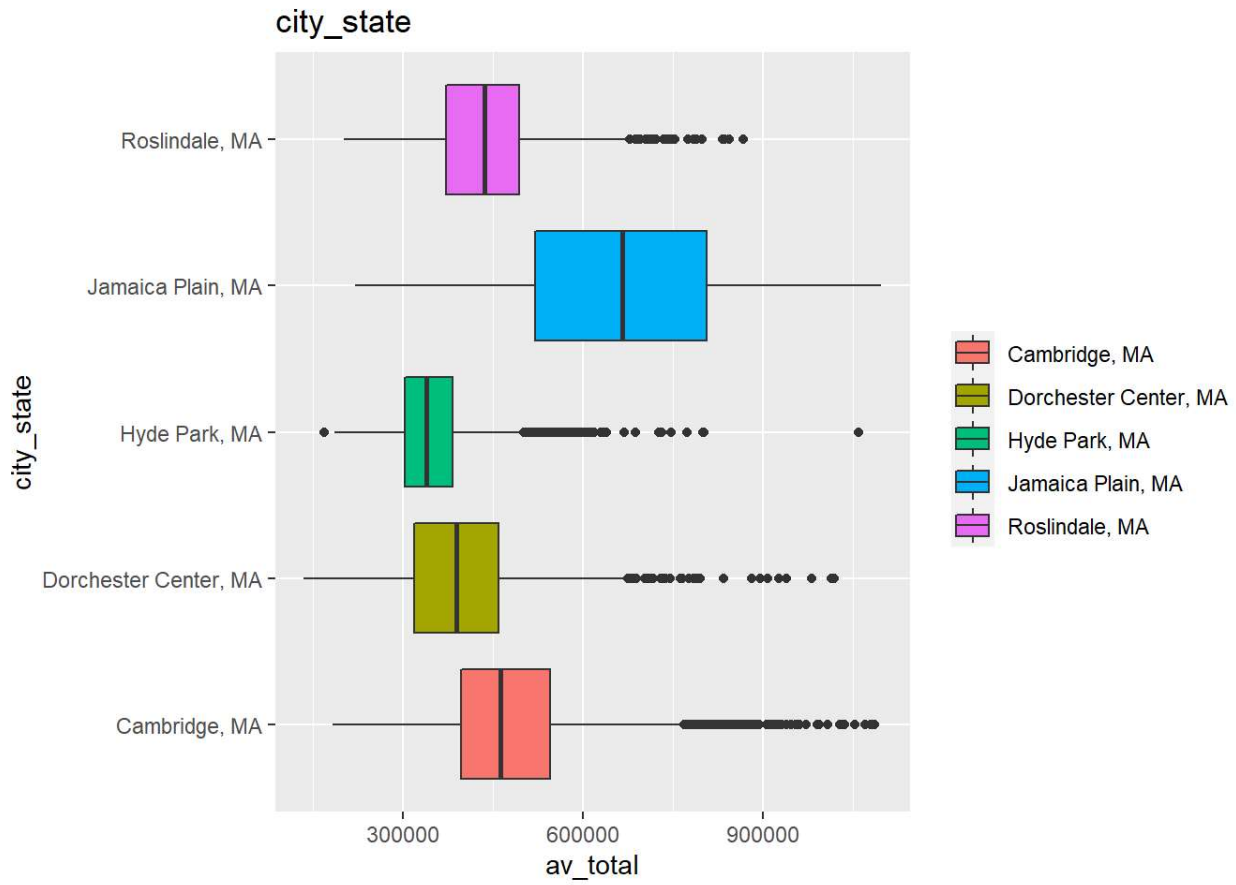

```
ggplot(boston, aes(x=av_total, y=as.factor(r_view), fill=as.factor(r_view))) +
  geom_boxplot() + labs(title = 'r_view', x = 'av_total', y = 'r_view') + theme(legend.title = element_blank())
```



```
ggplot(boston, aes(x=av_total, y=as.factor(zip), fill=as.factor(zip))) +
  geom_boxplot() + labs(title = 'zip', x = 'av_total', y = 'zip') + theme(legend.title = element_blank())
```

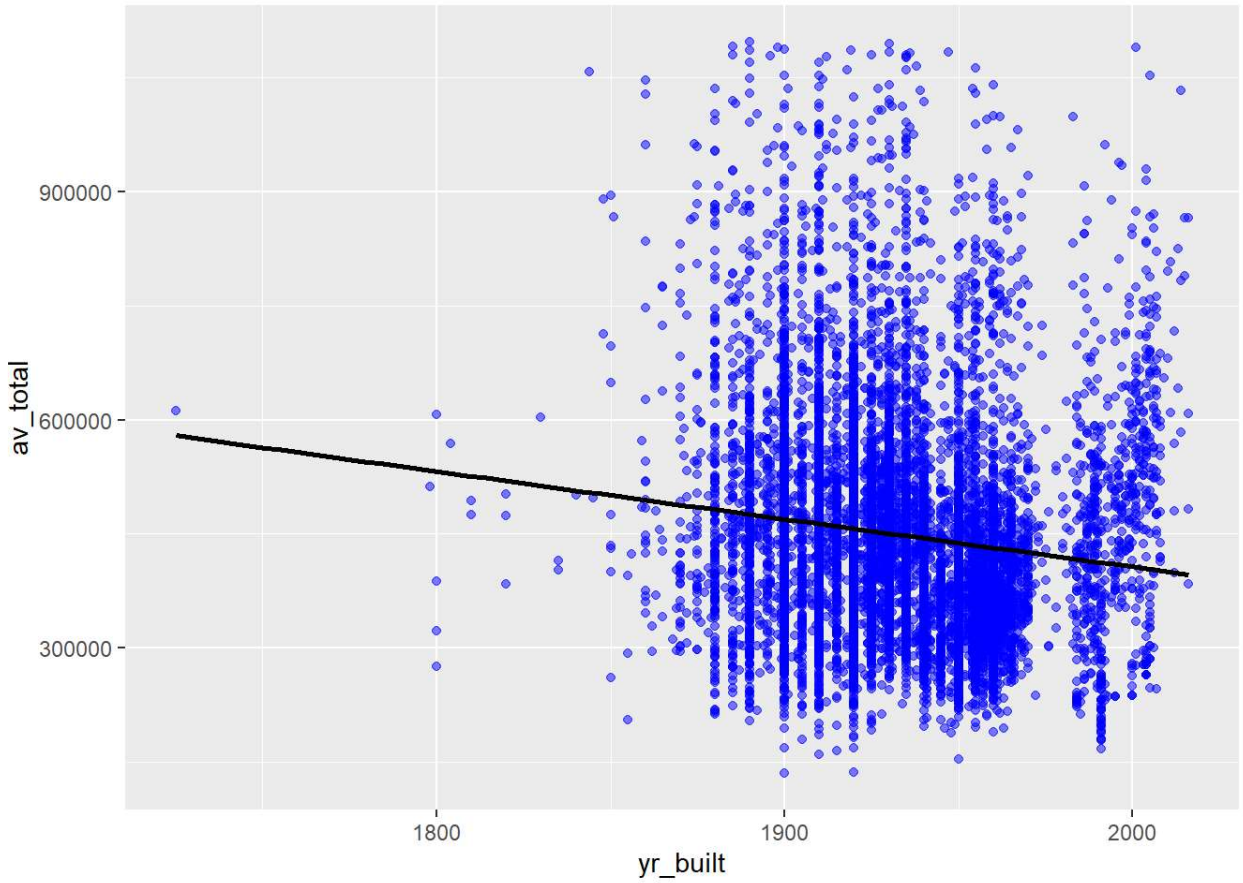



```
ggplot(boston, aes(x=av_total, y=as.factor(city_state), fill=as.factor(city_state))) +
  geom_boxplot() + labs(title = 'city_state', x = 'av_total', y = 'city_state') + theme(legend.title = element_blank
        ())
```



homes built in the 1990s, tend to have higher home values.

```
new_data <- boston %>% filter(yr_built != 0)

ggplot(data = new_data[!is.na(new_data$av_total),], aes(x=yr_built, y=av_total))+
  geom_point(col='blue',alpha=0.5) + geom_smooth(method = 'lm', se=FALSE, color='black', aes(group=1))
```



Partition our data 70/30 PLUS make K-Fold Cross Validation

Split the data 70 % train, 30% test, then make a 5 or 10 fold dataset from the test set.

```
# Save the split information for an 70/30 split of the data
bsplit <- initial_split(boston, prop = 0.75)
train <- training(bsplit)
test  <- testing(bsplit)

# Kfold cross validation
kfold_splits <- vfold_cv(train, v=5)
```

Recipe

Write your recipe out using formula, here i've make a simple recipe with 5 predictors, I recomend avoiding using AV_TOTAL ~ .

```
# write out the formula
boston_recipe <-
  recipe(av_total ~ land_sf + yr_built + living_area + num_floors + r_total_rms + r_bdrms + r_full_bth + r_half_bth +
         r_kitch + r_fplace + own_occ + r_bldg_styl + r_roof_typ + r_ext_fin + r_bth_style + r_kitch_style + r_heat_
         typ + r_ac + r_ext_cnd + r_ovrall_cnd + r_int_cnd + r_int_fin + r_view + city_state, data = train) %>%
  step_mutate(age = 2022 - yr_built ) %>%
  step_rm(yr_built) %>%
  step_impute_median(all_numeric_predictors()) %>% # missing values numeric
  step_novel(all_nominal_predictors()) %>% # new factor levels
  step_unknown(all_nominal_predictors()) %>% # missing values
  step_dummy(all_nominal_predictors(), one_hot = TRUE) %>%
  step_nzv(all_predictors()) #step_nzv creates a specification of a recipe step that will potentially remove variable
         s that are highly sparse and unbalanced


## Check the recipe results m
bake(boston_recipe %>% prep(),train %>% sample_n(1000))
```

land_sf	living_area	num_floors	r_total_rms	r_bdr...	r_full_bth	r_half_bth	r_fplace	av_total	a..
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
4648	1768	1.5	6	3	1	1	0	515600	23
8087	1248	1.5	6	3	1	1	0	318700	117
8469	1075	1.0	7	3	1	0	0	306600	33
5890	1923	2.5	8	3	1	1	0	368000	117
6000	2455	2.5	9	5	1	0	1	562300	102
6822	982	1.5	6	3	1	0	0	308900	132
3575	1536	2.0	6	3	1	1	1	597300	21
4857	2291	2.0	10	6	1	1	1	682300	107
8257	1680	2.0	7	4	1	0	1	533600	95
4080	1118	1.0	6	3	1	0	0	350500	95

1-10 of 1,000 rows | 1-10 of 43 columns Previous **1** 2 3 4 5 6 ... 100 Next

Linear Reg Setup

Linear regression there is really nothing to tune unless you want to get fancy. this is your baseline model that you should compare your work against.

```
lm_model <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

lm_wflow <- workflow() %>%
  add_recipe(boston_recipe) %>%
  add_model(lm_model) %>%
  fit(train)

tidy(lm_wflow) %>%
  mutate_if(is.numeric,round,4)
```

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	199203.9400	24298.4023	8.1982	0.0000
land_sf	7.6365	0.2772	27.5481	0.0000
living_area	95.9989	2.3708	40.4924	0.0000
num_floors	-1758.2342	3252.0867	-0.5406	0.5888
r_total_rms	3278.1982	789.3928	4.1528	0.0000
r_bdrms	-2253.8479	1194.2959	-1.8872	0.0592
r_full_bth	17546.6845	1656.3731	10.5934	0.0000
r_half_bth	19265.0393	1570.1083	12.2699	0.0000
r_fplace	20156.2075	1350.9293	14.9203	0.0000
age	219.4432	23.0477	9.5213	0.0000

1-10 of 43 rows Previous **1** 2 3 4 5 Next

```
lm_wflow %>%
  pull_workflow_fit() %>%
  tidy()%>%
  mutate_if(is.numeric,round,4)
```

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	199203.9400	24298.4023	8.1982	0.0000
land_sf	7.6365	0.2772	27.5481	0.0000
living_area	95.9989	2.3708	40.4924	0.0000
num_floors	-1758.2342	3252.0867	-0.5406	0.5888
r_total_rms	3278.1982	789.3928	4.1528	0.0000

term	estimate	std.error	statistic	p.value
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
r_bdrms	-2253.8479	1194.2959	-1.8872	0.0592
r_full_bth	17546.6845	1656.3731	10.5934	0.0000
r_half_bth	19265.0393	1570.1083	12.2699	0.0000
r_fplace	20156.2075	1350.9293	14.9203	0.0000
age	219.4432	23.0477	9.5213	0.0000

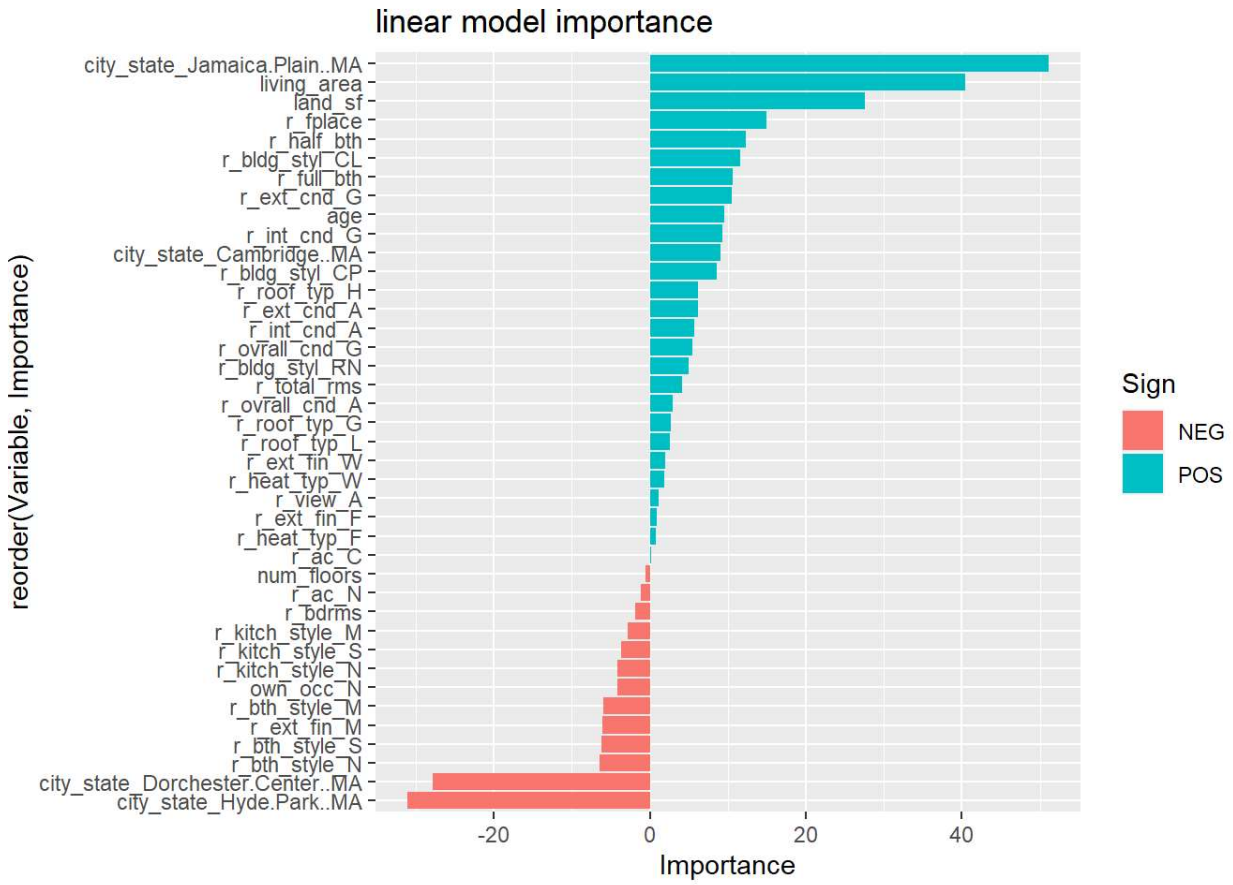
1-10 of 43 rows Previous 1 2 3 4 5 Next

```
lm_wflow %>%
  pull_workflow_fit() %>%
  vi() %>%
  mutate(Importance = if_else(Sign == "NEG", -Importance, Importance))
```

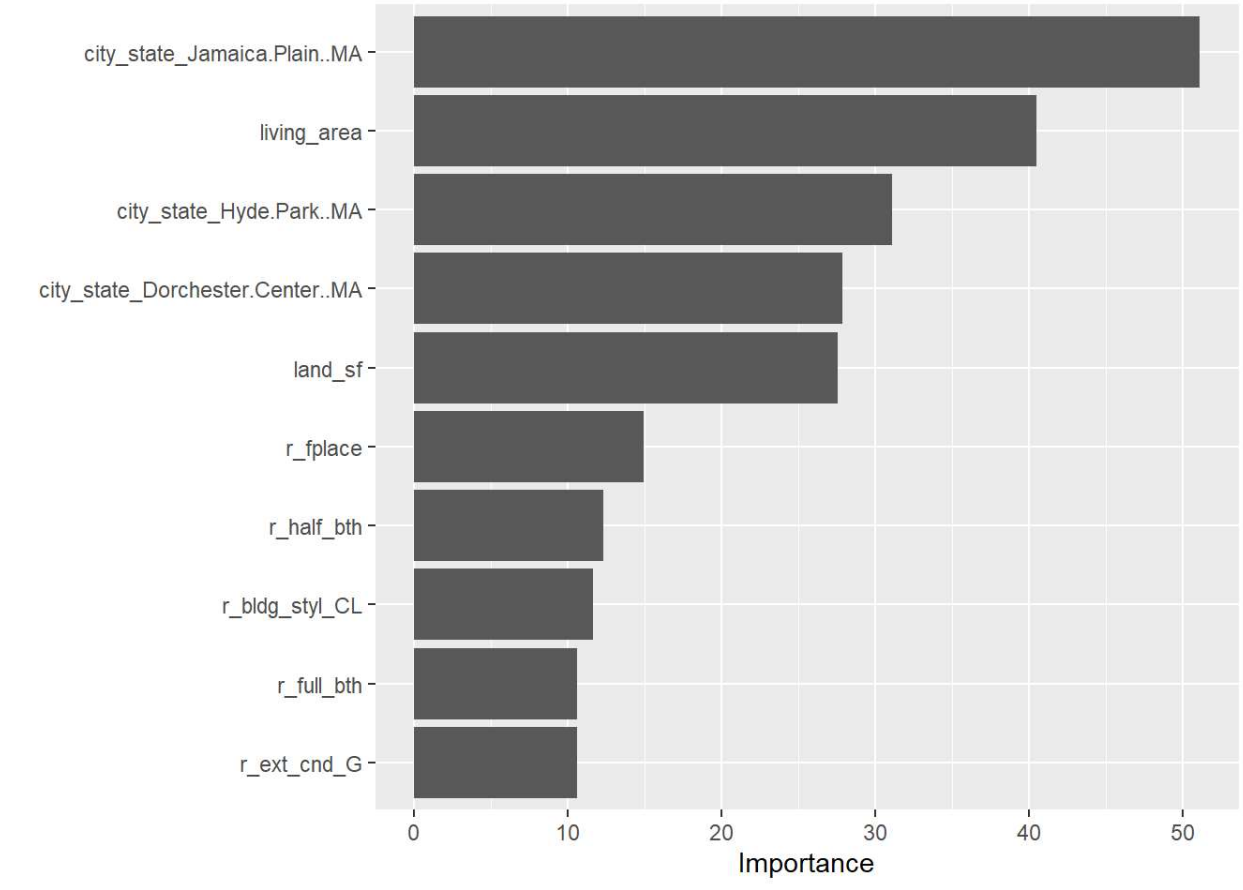
Variable	Importance	Sign
<chr>	<dbl>	<chr>
city_state_Jamaica.Plain..MA	51.0864827	POS
living_area	40.4923731	POS
city_state_Hyde.Park..MA	-31.1039542	NEG
city_state_Dorchester.Center..MA	-27.8395606	NEG
land_sf	27.5480532	POS
r_fplace	14.9202533	POS
r_half_bth	12.2698792	POS
r_bldg_styl_CL	11.6040723	POS
r_full_bth	10.5934371	POS
r_ext_cnd_G	10.5714688	POS

1-10 of 40 rows Previous 1 2 3 4 Next

```
lm_wflow %>%
  pull_workflow_fit() %>%
  vi() %>%
  mutate(Importance = if_else(Sign == "NEG", -Importance, Importance)) %>%
  ggplot(aes(reorder(Variable, Importance), Importance, fill=Sign)) +
  geom_col() + coord_flip() + labs(title="linear model importance")
```




```
lm_wflow %>%
  pull_workflow_fit() %>%
  vip()
```



```
bind_cols(
  predict(lm_wflow,train, type="numeric"), train) %>%
  mutate(part = "train") -> score_lm_train

bind_cols(
  predict(lm_wflow,test), test) %>% mutate(part = "test") -> score_lm_test

bind_rows(score_lm_train, score_lm_test) %>%
  group_by(part) %>%
  metrics(av_total,.pred) %>%
  pivot_wider(id_cols = part, names_from = .metric, values_from = .estimate)
```

part	rmse	rsq	mae
<chr>	<dbl>	<dbl>	<dbl>
test	60905.34	0.8300582	44316.78
train	63653.64	0.8144978	45146.54

2 rows

random forest

```
rf_model <- rand_forest(trees=tune()) %>%
   set_engine("ranger", num.threads = 5, max.depth = 10, importance="permutation") %>%
  set_mode("regression")

rf_wflow <-workflow() %>%
  add_recipe(boston_recipe) %>%
  add_model(rf_model)

rf_search_res <- rf_wflow %>%
  tune_bayes(
    resamples = kfold_splits,
    # Generate five at semi-random to start
    initial = 5,
    iter = 50,
    # How to measure performance?
    metrics = metric_set(rmse, rsq),
    control = control_bayes(no_improve = 5, verbose = TRUE)
  )
```

```
##
```

```
## ❯  Generating a set of 5 initial parameter results
```

```
## Warning: package 'ranger' was built under R version 4.2.1
```

```
## ✓ Initialization complete
```

```
##
```

```
##
```

```
## ── Iteration 1 ──────────────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=61400 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1995 candidates
```

```
## i Predicted candidates
```

```
## i trees=800
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=61420 (+/-503)
```

```
##
```

```
## ─── Iteration 2 ──────────────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=61400 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1994 candidates
```

```
## i Predicted candidates
```

```
## i trees=1049
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ♥ Newest results:    rmse=61360 (+/-418)
```

```
##
```

```
## ── Iteration 3 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=61360 (@iter 2)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1993 candidates
```

```
## i Predicted candidates
```

```
## i trees=2000
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=61420 (+/-432)
```

```
##
```

```
## ── Iteration 4 ──────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=61360 (@iter 2)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1992 candidates
```

```
## i Predicted candidates
```

```
## i trees=1084
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=61480 (+/-463)
```

```
##
```

```
## ─── Iteration 5 ──────────────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:     rmse=61360 (@iter 2)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1991 candidates
```

```
## i Predicted candidates
```

```
## i trees=401
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=61610 (+/-463)
```

```
##
```

```
## ─── Iteration 6 ──────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=61360 (@iter 2)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1990 candidates
```

```
## i Predicted candidates
```

```
## i trees=405
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=61480 (+/-328)
```

```
##
```

```
## ─── Iteration 7 ──────────────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=61360 (@iter 2)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 1989 candidates
```

```
## i Predicted candidates
```

```
## i trees=281
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=61710 (+/-411)
```

```
## ! No improvement for 5 iterations; returning current results.
```

Evaluate the random forest Model

```
rf_search_res
```

splits	id	.metrics	.notes	.iter
<list>	<chr>	<list>	<list>	<int>
<S3: vfold_split>	Fold1	<tibble[,5]>	<tibble[,3]>	0
<S3: vfold_split>	Fold2	<tibble[,5]>	<tibble[,3]>	0

splits	id	.metrics	.notes	.iter
<list>	<chr>	<list>	<list>	<int>
<S3: vfold_split>	Fold3	<tibble[,5]>	<tibble[,3]>	0
<S3: vfold_split>	Fold4	<tibble[,5]>	<tibble[,3]>	0
<S3: vfold_split>	Fold5	<tibble[,5]>	<tibble[,3]>	0
<S3: vfold_split>	Fold1	<tibble[,5]>	<tibble[,3]>	1
<S3: vfold_split>	Fold2	<tibble[,5]>	<tibble[,3]>	1
<S3: vfold_split>	Fold3	<tibble[,5]>	<tibble[,3]>	1
<S3: vfold_split>	Fold4	<tibble[,5]>	<tibble[,3]>	1
<S3: vfold_split>	Fold5	<tibble[,5]>	<tibble[,3]>	1

1-10 of 40 rows Previous 1 2 3 4 Next

```
lowest_rf_rmse <- rf_search_res %>%
  select_best("rmse")

rf_wflow <- finalize_workflow(
  rf_wflow, lowest_rf_rmse) %>%
  fit(train)

bind_cols(
  predict(rf_wflow,train), train) %>%
  metrics(av_total,.pred)
```

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rmse	standard	48284.9197974
rsq	standard	0.9134734
mae	standard	36770.9753597

3 rows

```
bind_cols(
  predict(rf_wflow,test), test) %>%
  metrics(av_total,.pred)
```

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rmse	standard	60165.0797564
rsq	standard	0.8578297

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
mae	standard	44242.0888595

3 rows

XGBoost Model Buiding

Here we want to TUNE our XGB model using the Bayes method.

```
xgb_model <- boost_tree(trees=tune(),
                        learn_rate = tune(),
                        tree_depth = tune()) %>%
  set_engine("xgboost",
             importance="permutation") %>%
  set_mode("regression")


xgb_wflow <-workflow() %>%
  add_recipe(boston_recipe) %>%
  add_model(xgb_model)


xgb_search_res <- xgb_wflow %>%
  tune_bayes(
    resamples = kfold_splits,
    # Generate five at semi-random to start
    initial = 5,
    iter = 50,
    # How to measure performance?
    metrics = metric_set(rmse, rsq),
    control = control_bayes(no_improve = 5, verbose = TRUE)
  )
```

```
##
```

```
## ❯  Generating a set of 5 initial parameter results
```

```
## Warning: package 'xgboost' was built under R version 4.2.2
```

```
## ✓ Initialization complete
```

```
##
```

```
##
```

```
## ── Iteration 1 ──────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=54770 (@iter 0)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=1375, tree_depth=9, learn_rate=0.0213
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ♥ Newest results:    rmse=54630 (+/-541)
```

```
##
```

```
## ─── Iteration 2 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:     rmse=54630 (@iter 1)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=28, tree_depth=8, learn_rate=0.00101
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=459600 (+/-1340)
```

```
##
```

```
## ─── Iteration 3 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=54630 (@iter 1)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=1397, tree_depth=6, learn_rate=0.0709
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ♥ Newest results:    rmse=53490 (+/-452)
```

```
##
```

```
## ─── Iteration 4 ──────────────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=53490 (@iter 3)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=652, tree_depth=4, learn_rate=0.0171
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=54350 (+/-668)
```

```
##
```

```
## ─── Iteration 5 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=53490 (@iter 3)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=11, tree_depth=6, learn_rate=0.113
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=142400 (+/-943)
```

```
##
```

```
## ─── Iteration 6 ──────────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=53490 (@iter 3)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=1988, tree_depth=14, learn_rate=0.00306
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=56060 (+/-596)
```

```
##
```

```
## ─── Iteration 7 ──────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=53490 (@iter 3)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=2000, tree_depth=7, learn_rate=0.316
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=57280 (+/-460)
```

```
##
```

```
## ─── Iteration 8 ──────────────────────────────────────────────────────────────────────────────────────────────────────
```

```
##
```

```
## i Current best:      rmse=53490 (@iter 3)
```

```
## i Gaussian process model
```

```
## ✓ Gaussian process model
```

```
## i Generating 5000 candidates
```

```
## i Predicted candidates
```

```
## i trees=435, tree_depth=8, learn_rate=0.312
```

```
## i Estimating performance
```

```
## i Fold1: preprocessor 1/1
```

```
## ✓ Fold1: preprocessor 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1
```

```
## ✓ Fold1: preprocessor 1/1, model 1/1
```

```
## i Fold1: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold2: preprocessor 1/1
```

```
## ✓ Fold2: preprocessor 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1
```

```
## ✓ Fold2: preprocessor 1/1, model 1/1
```

```
## i Fold2: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold3: preprocessor 1/1
```

```
## ✓ Fold3: preprocessor 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1
```

```
## ✓ Fold3: preprocessor 1/1, model 1/1
```

```
## i Fold3: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold4: preprocessor 1/1
```

```
## ✓ Fold4: preprocessor 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1
```

```
## ✓ Fold4: preprocessor 1/1, model 1/1
```

```
## i Fold4: preprocessor 1/1, model 1/1 (predictions)
```

```
## i Fold5: preprocessor 1/1
```

```
## ✓ Fold5: preprocessor 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1
```

```
## ✓ Fold5: preprocessor 1/1, model 1/1
```

```
## i Fold5: preprocessor 1/1, model 1/1 (predictions)
```

```
## ✓ Estimating performance
```

```
## ⓧ Newest results:    rmse=56970 (+/-550)
```

```
## ! No improvement for 5 iterations; returning current results.
```

XGB Tuning

Evaluate the tuning efforts

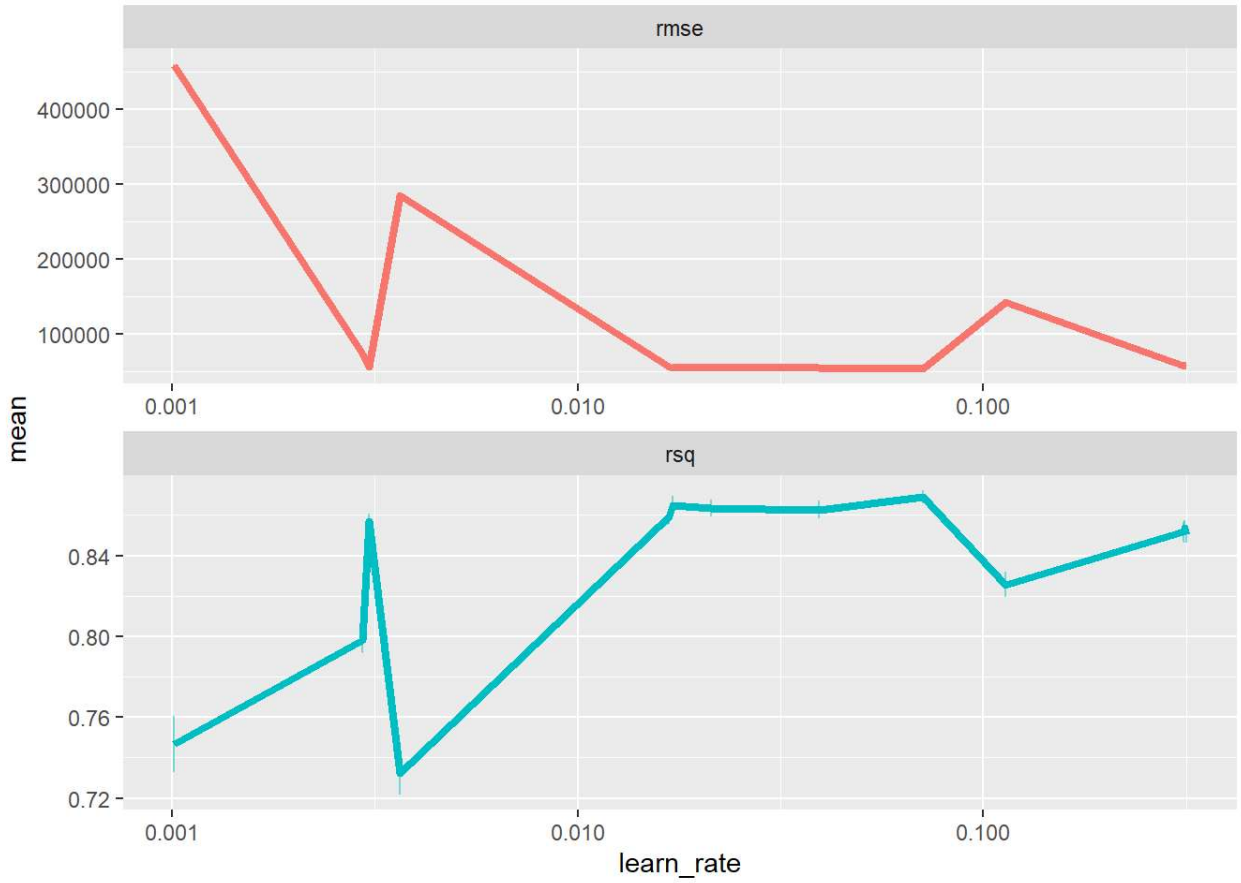
```
# Experiments
xgb_search_res %>%
  collect_metrics()  %>%
  filter(.metric == "rmse")
```

trees	tree_depth	learn_rate	.metric	.estimator	mean	n	std_err	.config	.iter
<int>	<int>	<dbl>	<chr>	<chr>	<dbl>	<int>	<dbl>	<chr>	<int>
1280	2	0.002955155	rmse	standard	73442.67	5	499.2830	Preprocessor1_Model1	0
147	4	0.003647472	rmse	standard	284642.56	5	1089.0727	Preprocessor1_Model2	0
1021	7	0.312691604	rmse	standard	56565.62	5	240.7886	Preprocessor1_Model3	0
1816	9	0.039383400	rmse	standard	54767.94	5	666.2453	Preprocessor1_Model4	0
414	14	0.016751867	rmse	standard	55595.54	5	499.8470	Preprocessor1_Model5	0

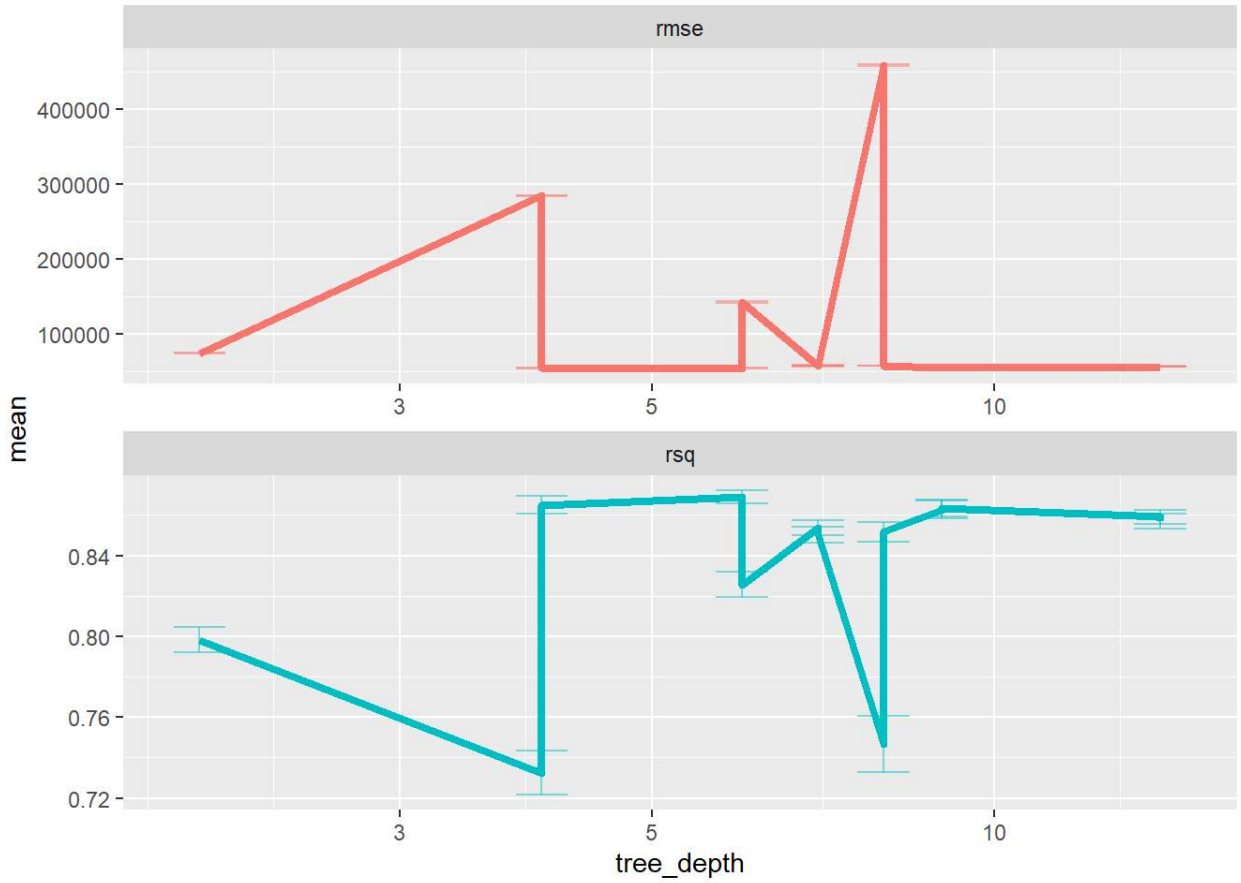
trees	tree_depth	learn_rate	.metric	.estimator	mean	n	std_err	.config	.iter
<int>	<int>	<dbl>	<chr>	<chr>	<dbl>	<int>	<dbl>	<chr>	<int>
1375	9	0.021311448	rmse	standard	54630.08	5	541.2904	Iter1	1
28	8	0.001010679	rmse	standard	459584.31	5	1342.7195	Iter2	2
1397	6	0.070930012	rmse	standard	53494.01	5	452.4693	Iter3	3
652	4	0.017142271	rmse	standard	54353.73	5	667.8986	Iter4	4
11	6	0.113286188	rmse	standard	142428.64	5	942.6405	Iter5	5

1-10 of 13 rows Previous **1** 2 Next

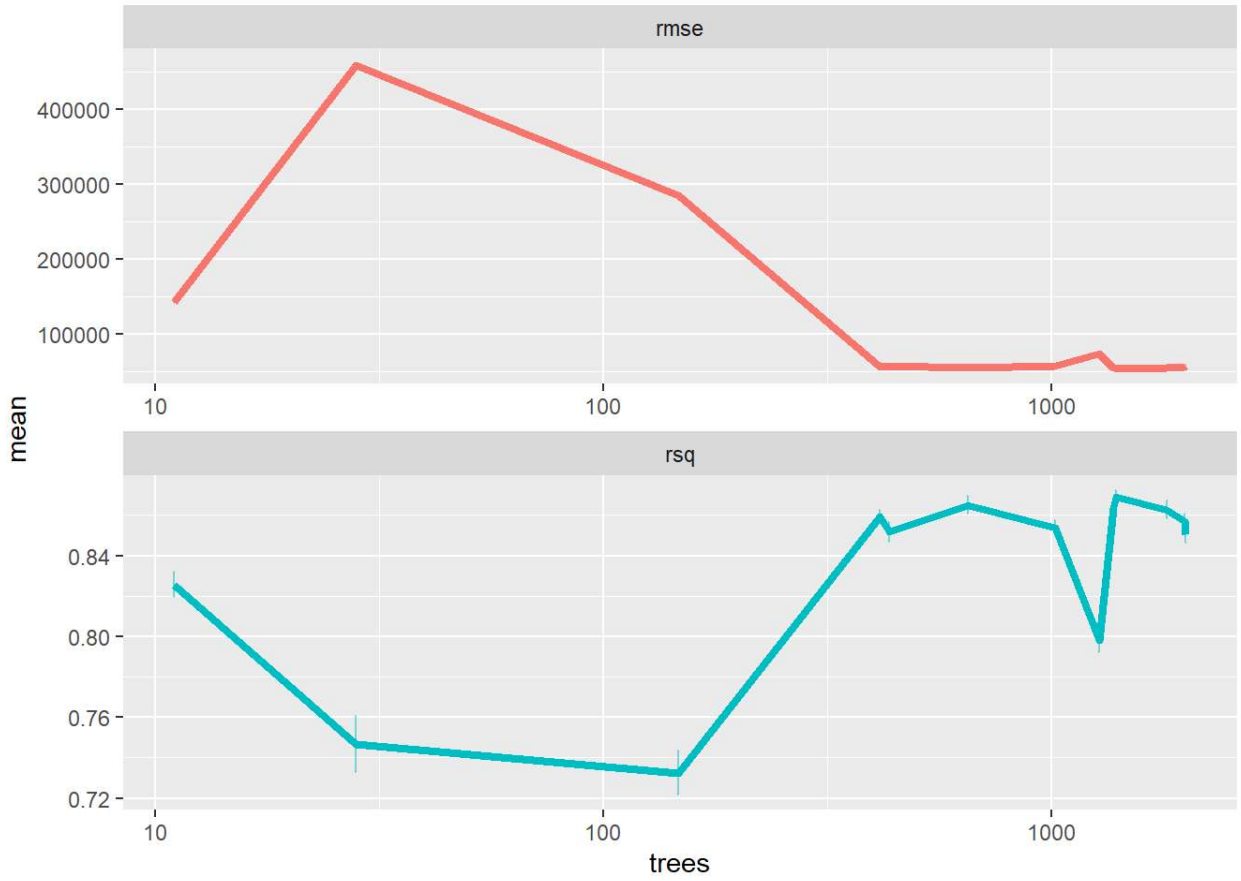
```
# Graph of learning rate
xgb_search_res %>%
  collect_metrics() %>%
  ggplot(aes(learn_rate, mean, color = .metric)) +
  geom_errorbar(aes(
    ymin = mean - std_err,
    ymax = mean + std_err
  ),
  alpha = 0.5
  ) +
  geom_line(size = 1.5) +
  facet_wrap(~.metric, scales = "free", nrow = 2) +
  scale_x_log10() +
  theme(legend.position = "none")
```




```
# graph of tree depth
xgb_search_res %>%
  collect_metrics() %>%
  ggplot(aes(tree_depth, mean, color = .metric)) +
  geom_errorbar(aes(
    ymin = mean - std_err,
    ymax = mean + std_err
  ),
  alpha = 0.5
  ) +
  geom_line(size = 1.5) +
  facet_wrap(~.metric, scales = "free", nrow = 2) +
  scale_x_log10() +
  theme(legend.position = "none")
```




```
# graph of number of trees
xgb_search_res %>%
  collect_metrics() %>%
  ggplot(aes(trees, mean, color = .metric)) +
  geom_errorbar(aes(
    ymin = mean - std_err,
    ymax = mean + std_err
  ),
  alpha = 0.5
  ) +
  geom_line(size = 1.5) +
  facet_wrap(~.metric, scales = "free", nrow = 2) +
  scale_x_log10() +
  theme(legend.position = "none")
```



Final Fit XGB

Finally fit the XGB model using the best set of parameters

```
lowest_xgb_rmse <- xgb_search_res %>%
  select_best("rmse")

lowest_xgb_rmse
```

trees	tree_depth	learn_rate	.config
<int>	<int>	<dbl>	<chr>
1397	6	0.07093001	Iter3

1 row

```
xgb_wflow <- finalize_workflow(
  xgb_wflow, lowest_xgb_rmse
) %>%
  fit(train)
```

```
## [23:11:09] WARNING: amalgamation/../src/learner.cc:627:
## Parameters: { "importance" } might not be used.
##
##   This could be a false alarm, with some parameters getting used by language bindings but
##   then being mistakenly passed down to XGBoost core, or some parameter actually being used
##   but getting flagged wrongly here. Please open an issue if you find any such cases.
```

VIP

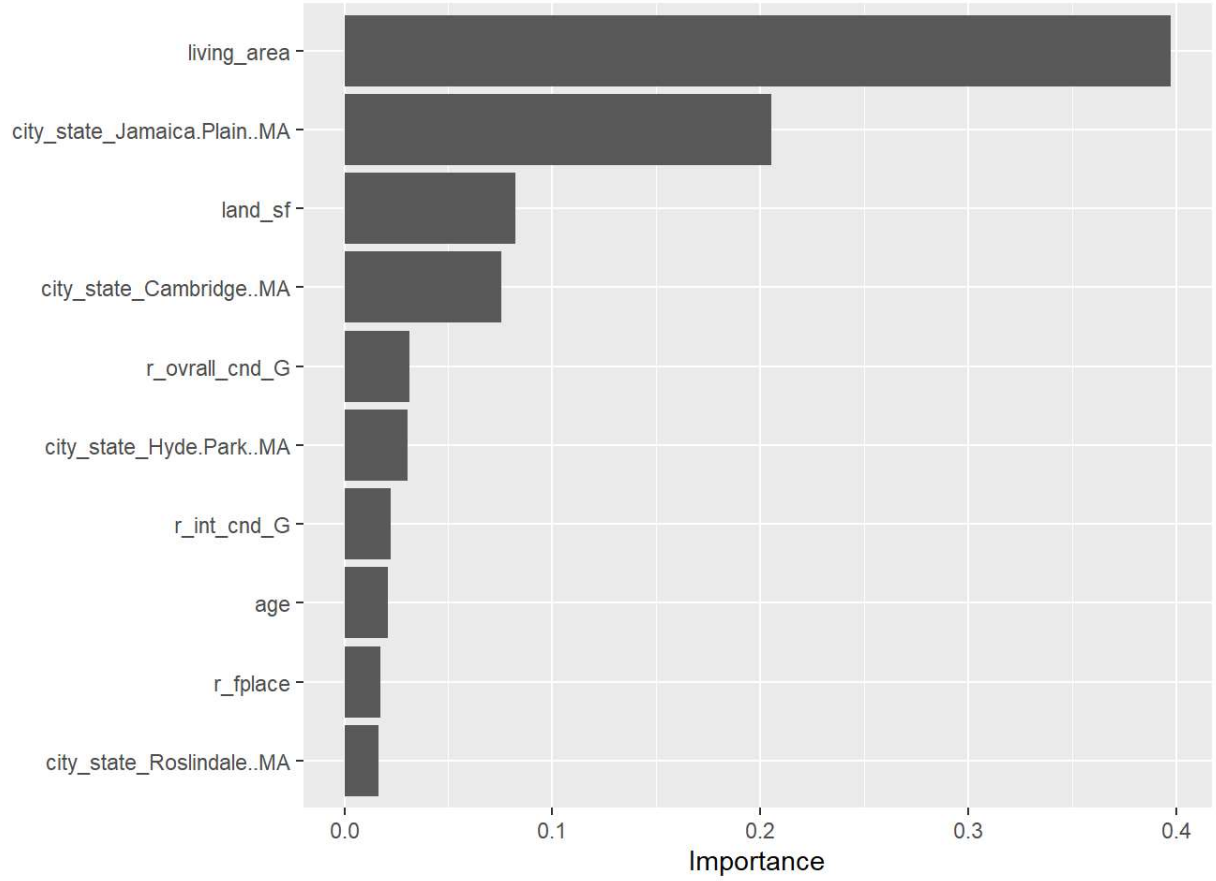
What variables are important

```
xgb_wflow %>%
  extract_fit_parsnip() %>%
  vi()
```

Variable <chr>	Importance <dbl>
living_area	0.3974402811
city_state_Jamaica.Plain..MA	0.2055939397
land_sf	0.0822588374
city_state_Cambridge..MA	0.0753727077
r_ovrall_cnd_G	0.0312103293
city_state_Hyde.Park..MA	0.0305150053
r_int_cnd_G	0.0223912452
age	0.0210724030
r_fplace	0.0172735329
city_state_Roslindale..MA	0.0162015743

1-10 of 41 rows Previous **1** 2 3 4 5 Next

```
xgb_wflow %>%
  extract_fit_parsnip() %>%
  vip()
```



Evaluate the XGBoost BEST Model

```
bind_cols(
  predict(xgb_wflow,train), train) %>%
  metrics(av_total,.pred)
```

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rmse	standard	12434.9140993
rsq	standard	0.9930585
mae	standard	8776.5723135

3 rows

```
bind_cols(
  predict(xgb_wflow,test), test) %>%
  metrics(av_total,.pred)
```

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rmse	standard	51351.9668516
rsq	standard	0.8790632

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
mae	standard	36941.6668486

3 rows

Best Worst Predicitons

You should have one best and two worst predictions

1. the properties that you under-estimate the value of
2. the properties that you over-estimate the value of
3. the properties that are your best-estimate

```
# best estimate
bind_cols(predict(xgb_wflow,test),test) %>%
  mutate(error = av_total - .pred,
         abs_error = abs(error)) %>%
  slice_min(order_by = abs_error,n=10) -> best_estimate

best_estimate %>% dplyr::select(.pred, av_total, error, abs_error)
```

.pred	av_total	error	abs_error
<dbl>	<dbl>	<dbl>	<dbl>
406090.4	406100	9.59375	9.59375
483884.5	483900	15.46875	15.46875
322675.7	322700	24.28125	24.28125
340452.2	340400	-52.25000	52.25000
383894.9	383800	-94.93750	94.93750
253114.6	253000	-114.60938	114.60938
440279.4	440400	120.59375	120.59375
577451.4	577600	148.62500	148.62500
398585.1	398400	-185.12500	185.12500
420673.8	420900	226.18750	226.18750

1-10 of 10 rows

```
best_estimate %>% summarize(mean(error), mean(av_total), mean(yr_built))
```

mean(error)	mean(av_total)	mean(yr_built)
<dbl>	<dbl>	<dbl>
9.782813	402720	1932.3

1 row

```
# worst estimate
bind_cols(predict(xgb_wflow,test),test) %>%
  mutate(error = av_total - .pred,
         abs_error = abs(error)) %>%
  slice_max(order_by = abs_error,n=10) -> worst_estimate

worst_estimate %>% dplyr::select(.pred, av_total, error, abs_error)
```

.pred	av_total	error	abs_error
<dbl>	<dbl>	<dbl>	<dbl>
759531.9	1062620	303088.1	303088.1
872352.4	583500	-288852.4	288852.4
691308.4	944600	253291.6	253291.6
903434.3	657900	-245534.3	245534.3
610301.6	835500	225198.4	225198.4
513501.7	732300	218798.3	218798.3
803286.8	1019300	216013.2	216013.2
508135.6	296400	-211735.6	211735.6
826324.6	616700	-209624.6	209624.6
649955.3	859500	209544.7	209544.7

1-10 of 10 rows

```
worst_estimate %>% summarize(mean(error), mean(av_total), mean(yr_built))
```

mean(error)	mean(av_total)	mean(yr_built)
<dbl>	<dbl>	<dbl>
47018.74	760832	1924.4

1 row

KAGGLE

```
bind_cols(predict(xgb_wflow,kaggle),kaggle) %>%
  select(pid,av_total = .pred) %>%
  write_csv("challenge_3_kaggle.csv")
```