Kaggle Competition: How much for your Airbnb?

Zichen (Zoe) Huang

Dec 3, 2018

- Read data
- Combine train and test datasets
- Data cleansing, build what is needed
    - Understand data
    - Impute missing value
    - Explore Dataset Variables
    - Data Preparation
    - Transform those values with high skewness-
    - Split combined full dataset into train and test datasets
- Further operation on variables left now
- Model building and evaluation

Read data

Read data including **analysisData** and **scoringData**.

```
# set working direction
setwd('/Users/Zoe/Desktop/Columbia University /Courses/APAN5200/Assignment/Kaggle Com
petition/Raw data')
data = read.csv('analysisData.csv')
testing = read.csv('scoringData.csv')
```

Combine train and test datasets

Deal with train dataset by removing record of **Uruguay** and records the price of which equals to zero. Then combine train and test datasets so that we can operate on these two datasets in the same way altogether.

```
# Remove record of Uruguay which is not useful for a model to predict house price in
NY
data <- data[-4568,]
# Deal with price_zero
sum(data$price == 0)
```

```
## [1] 25
```

```
data<-data[!(data$price == 0),]
# Remove the taget variable from train data and combine train and test datasets
SalePrice = data$price
data <- data[,-61]
fulldata <- rbind(data,testing)
```

Data cleansing, build what is needed

Understand data

```
# Before imputing the missing value, take a look at the whole dataset to understand t
he structure
str(fulldata)
```

```
## 'data.frame':    36402 obs. of  95 variables:
##  $ id                               : int  20091785 3710661 15055244 19640913 11888
948 11769831 7803475 13935360 17045788 21274450 ...
##  $ listing_url                      : Factor w/ 36428 levels "https://www.airbnb.co
m/rooms/10000070",..: 13151 20892 5764 12284 1646 1523 26109 4331 8635 15526 ...
##  $ scrape_id                        : num  2.02e+13 2.02e+13 2.02e+13 2.02e+13 2.02
e+13 ...
##  $ last_scraped                     : Factor w/ 3 levels "2018-03-04","2018-03-05",
..: 2 1 1 2 2 2 2 1 1 2 ...
##  $ name                             : Factor w/ 35898 levels ""," 1 Bed Apt in Utop
ic Williamsburg ",..: 10360 24298 12267 10152 6743 14867 14895 5136 10925 7910 ...
##  $ summary                          : Factor w/ 34209 levels "","         Great sun
ny second floor apartment in Brooklyn, Bedstuy. The building is a new construction wi
th eleva"| __truncated__,..: 1817 15683 300 23834 5039 9333 11545 4505 26480 13652 ..
.
##  $ space                            : Factor w/ 26420 levels "","   About your room
is a private room located on the first floor this is a 3 bedroom fully equipped apart
ment ju"| __truncated__,..: 1 17454 3637 1 19215 13283 7499 1 10218 8643 ...
##  $ description                      : Factor w/ 35931 levels "","         Great sun
ny second floor apartment in Brooklyn, Bedstuy. The building is a new construction wi
th eleva"| __truncated__,..: 1926 16412 332 24973 5303 9772 12126 4744 27748 14311 ..
.
##  $ experiences_offered              : Factor w/ 1 level "none": 1 1 1 1 1 1 1 1 1 1
...
##  $ neighborhood_overview            : Factor w/ 21821 levels "","  I love this tren
dy area because of all the restaurant options!!!!  and being so close to NYC, only a
15 min t"| __truncated__,..: 16564 10876 16969 1 1 10906 9379 1 8682 7833 ...
```

```
## $ notes : Factor w/ 14552 levels "",""," **Please fill-out your profile, preferably with a photo before you book. I appreciate knowing a bit about my g"| __truncated__,..: 1 1071 7930 1 1 1 1479 1 1 4793 ...
## $ transit : Factor w/ 23338 levels ""," #15A Bus Stop is two blocks away, 20 min bus ride to #7 Subway Main Street Station in Flushing(Chinatown). Nea"| __truncated__,..: 8884 18487 1875 1 1 13912 7479 1 16992 2321 ...
## $ access : Factor w/ 21193 levels "","  Accessible to bus, train lines, and taxi serivices.  Deliciously located near Harlem's \"Restaurant Row\". Ta"| __truncated__,..: 16871 6200 12711 1 3820 1 3842 1 13088 8680 ...
## $ interaction : Factor w/ 20926 levels "","  I often interact with guest while cleaning the common space & bathroom!  I love to keep this space tidy!!! Al"| __truncated__,..: 3793 4384 12091 1 7095 1 1278 1 13649 7534 ...
## $ house_rules : Factor w/ 22134 levels ""," - No illegal drugs in house or illegal actions. No parties allowed, Or extra guests without previous approval."| __truncated__,..: 1 7271 3373 1 1 1 10903 1 61 67 ...
## $ thumbnail_url : logi  NA NA NA NA NA NA ...
## $ medium_url : logi  NA NA NA NA NA NA ...
## $ picture_url : Factor w/ 36418 levels "https://a0.muscache.com/im/pictures/00046f21-eba2-4490-9e51-20f30a6b1507.jpg?aki_policy=large",..: 8705 11325 7112 775 24475 25958 1531 18349 27206 27140 ...
## $ xl_picture_url : logi  NA NA NA NA NA NA ...
## $ host_id : int  142871086 18930170 1732527 129627362 63399183 10555698 25847311 80560845 23878336 153896340 ...
## $ host_url : Factor w/ 30136 levels "https://www.airbnb.com/users/show/1000014",..: 4468 7986 7310 3076 19905 750 10850 22488 9989 5585 ...
## $ host_name : Factor w/ 9705 levels " Valéria","'Cil",..: 2277 7691 2214 2561 1369 6849 4921 7068 656 2117 ...
## $ host_since : Factor w/ 3088 levels "2008-04-21","2008-09-06",..: 2836 1739 842 2757 2339 1511 1906 2442 1854 2909 ...
## $ host_location : Factor w/ 1121 levels "","  Brooklyn, NY  ",..: 616 616 789 616 616 616 616 616 616 896 ...
## $ host_about : Factor w/ 19234 levels "","","\n","\n\n",..: 6679 8201 3324 1 1 3556 1 1 3020 1 ...
## $ host_response_time : Factor w/ 5 levels "a few days or more",..: 25 3 3 3 2 2 2 5 2 ...
## $ host_response_rate : Factor w/ 76 levels "0%","10%","100%",..: 76 3 3 3 37 76 76 76 3 76 ...
## $ host_acceptance_rate : Factor w/ 1 level "N/A": 1 1 1 1 1 1 1 1 1 1 ...
## $ host_is_superhost : Factor w/ 2 levels "f","t": 1 2 2 1 1 1 1 1 1 1 ...
## $ host_thumbnail_url : Factor w/ 30068 levels "https://a0.muscache.com/defaults/user_pic-50x50.png?v=3",..: 11704 5001 17718 3886 11417 15813 7604 3049 19118 12516 ...
## $ host_picture_url : Factor w/ 30068 levels "https://a0.muscache.com/defaults/user_pic-225x225.png?v=3",..: 11704 5001 17718 3886 11417 15813 7604 3049 19118 12516 ...
```

```
##  $ host_neighbourhood              : Factor w/ 361 levels "","Adams Point",..: 134
81 325 1 300 69 301 117 18 264 ...
##  $ host_listings_count             : int  1 3 1 1 1 1 1 1 5 1 ...
##  $ host_total_listings_count       : int  1 3 1 1 1 1 1 1 5 1 ...
##  $ host_verifications              : Factor w/ 520 levels "['email', 'amex', 'revi
ews', 'kba']",..: 111 381 174 194 381 381 377 381 377 414 ...
##  $ host_has_profile_pic            : Factor w/ 2 levels "f","t": 2 2 2 2 2 2 2 2 2
2 ...
##  $ host_identity_verified          : Factor w/ 2 levels "f","t": 1 2 2 1 2 2 2 2 2
1 ...
##  $ street                          : Factor w/ 267 levels " Brooklyn, NY, United S
tates",..: 181 153 55 153 153 55 153 153 35 153 ...
##  $ neighbourhood                   : Factor w/ 200 levels "","Allerton",..: 96 54
193 185 179 46 180 86 12 158 ...
##  $ neighbourhood_cleansed          : Factor w/ 214 levels "Allerton","Arden Height
s",..: 57 59 207 200 195 49 196 92 76 172 ...
##  $ neighbourhood_group_cleansed    : Factor w/ 5 levels "Bronx","Brooklyn",..: 4 3
2 3 3 2 3 3 1 3 ...
##  $ city                            : Factor w/ 263 levels ""," Brooklyn",..: 173 1
45 39 145 145 39 145 145 32 145 ...
##  $ state                           : Factor w/ 8 levels "","CA","New York",..: 6 6
6 6 6 6 6 6 6 6 ...
##  $ zipcode                         : Factor w/ 190 levels "","07002","07093",..: 1
52 31 109 35 44 114 25 33 81 17 ...
##  $ market                          : Factor w/ 20 levels "","Adirondacks",..: 13 1
3 13 13 13 13 13 13 13 13 ...
##  $ smart_location                  : Factor w/ 267 levels " Brooklyn, NY",..: 181
153 54 153 153 54 153 153 34 153 ...
##  $ country_code                    : Factor w/ 2 levels "US","UY": 1 1 1 1 1 1 1 1
1 1 ...
##  $ country                         : Factor w/ 2 levels "United States",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ latitude                        : num  40.8 40.8 40.7 40.9 40.8 ...
##  $ longitude                       : num  -73.9 -73.9 -74 -73.9 -74 ...
##  $ is_location_exact               : Factor w/ 2 levels "f","t": 2 1 2 1 2 2 2 2 2
2 ...
##  $ property_type                   : Factor w/ 37 levels "Aparthotel","Apartment",
..: 2 2 2 2 2 2 2 2 20 2 ...
##  $ room_type                       : Factor w/ 3 levels "Entire home/apt",..: 2 2
1 2 1 1 1 1 2 1 ...
##  $ accommodates                    : int  1 2 2 2 2 2 5 5 3 2 ...
##  $ bathrooms                       : num  1 1 1.5 1 1 1 1 1 3 1 ...
##  $ bedrooms                        : int  1 1 1 1 1 1 2 2 1 0 ...
##  $ beds                            : int  1 1 1 1 1 1 5 3 2 1 ...
##  $ bed_type                        : Factor w/ 5 levels "Airbed","Couch",..: 5 5 5
5 5 5 5 5 5 5 ...
##  $ amenities                       : Factor w/ 33990 levels "{\"Air conditioning\"
```

```
,\"Buzzer/wireless intercom\",Heating,\"Smoke detector\",Essentials,Shampoo,Hangers,\
"Hair dryer\",Iron}",..: 27214 7963 18253 23334 19244 23817 7767 18872 3028 557 ...
##  $ square_feet                  : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ weekly_price                 : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ monthly_price                : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ security_deposit             : int  NA NA NA NA NA NA 250 100 100 0 ...
##  $ cleaning_fee                 : int  NA 50 30 20 60 NA 200 30 20 100 ...
##  $ guests_included              : int  1 2 1 1 1 1 1 1 2 1 ...
##  $ extra_people                 : int  0 0 0 0 10 0 0 0 15 0 ...
##  $ minimum_nights               : int  1 2 6 2 12 2 30 4 3 1 ...
##  $ maximum_nights               : int  1125 99 14 1125 1125 1125 60 21 31 1125
...
##  $ calendar_updated             : Factor w/ 66 levels "1 week ago","10 months a
go",..: 56 25 26 38 60 2 54 56 53 14 ...
##  $ has_availability             : Factor w/ 1 level "t": 1 1 1 1 1 1 1 1 1 1 ..
.
##  $ availability_30              : int  0 3 0 12 4 0 27 0 22 3 ...
##  $ availability_60              : int  0 8 0 30 16 0 57 0 52 3 ...
##  $ availability_90              : int  0 24 0 30 46 0 87 0 82 3 ...
##  $ availability_365             : int  0 74 0 182 244 0 362 0 357 22 ...
##  $ calendar_last_scraped        : Factor w/ 3 levels "2018-03-04","2018-03-05",
..: 2 1 1 2 2 2 1 1 1 2 ...
##  $ number_of_reviews            : int  3 101 19 8 4 10 16 4 37 1 ...
##  $ first_review                 : Factor w/ 2495 levels "2008-10-13","2009-03-1
2",..: 2179 1103 1875 2155 1702 1692 1507 1809 2028 2333 ...
##  $ last_review                  : Factor w/ 1361 levels "2011-03-28","2011-05-2
3",..: 1085 1255 1233 1236 1213 1011 874 1087 1237 1238 ...
##  $ review_scores_rating         : int  90 93 100 89 87 100 95 95 93 100 ...
##  $ review_scores_accuracy       : int  9 10 10 9 10 10 10 10 9 10 ...
##  $ review_scores_cleanliness    : int  8 9 10 10 9 10 9 10 9 10 ...
##  $ review_scores_checkin        : int  10 10 10 9 10 10 10 9 10 10 ...
##  $ review_scores_communication  : int  10 10 10 9 10 10 10 8 10 10 ...
##  $ review_scores_location       : int  10 8 10 9 10 10 10 8 9 10 ...
##  $ review_scores_value          : int  10 9 10 9 10 10 9 9 9 10 ...
##  $ requires_license             : Factor w/ 1 level "f": 1 1 1 1 1 1 1 1 1 1 ..
.
##  $ license                      : logi  NA NA NA NA NA NA ...
##  $ jurisdiction_names           : Factor w/ 4 levels "","{\"SAN FRANCISCO\"}",.
.: 1 1 1 1 1 1 1 1 1 1 ...
##  $ instant_bookable             : Factor w/ 2 levels "f","t": 2 2 1 2 1 1 1 1 2
2 ...
##  $ is_business_travel_ready     : Factor w/ 2 levels "f","t": 1 1 1 1 1 1 1 1 1
1 ...
##  $ cancellation_policy          : Factor w/ 5 levels "flexible","moderate",..:
1 2 2 1 2 1 3 1 3 3 ...
##  $ require_guest_profile_picture : Factor w/ 2 levels "f","t": 1 1 1 1 1 1 1 1 1
1 ...
```

```
##  $ require_guest_phone_verification: Factor w/ 2 levels "f","t": 1 1 1 1 1 1 1 1 1
1 ...
##  $ calculated_host_listings_count  : int  1 3 1 1 1 1 1 1 5 1 ...
##  $ reviews_per_month               : num  0.42 2.33 1.1 1 0.17 0.43 0.54 0.21 3.03
0.48 ...
```

```
# Look at column names
names(fulldata)
```

```
##  [1] "id"                           "listing_url"
##  [3] "scrape_id"                    "last_scraped"
##  [5] "name"                         "summary"
##  [7] "space"                        "description"
##  [9] "experiences_offered"          "neighborhood_overview"
## [11] "notes"                        "transit"
## [13] "access"                       "interaction"
## [15] "house_rules"                  "thumbnail_url"
## [17] "medium_url"                   "picture_url"
## [19] "xl_picture_url"               "host_id"
## [21] "host_url"                     "host_name"
## [23] "host_since"                   "host_location"
## [25] "host_about"                   "host_response_time"
## [27] "host_response_rate"           "host_acceptance_rate"
## [29] "host_is_superhost"            "host_thumbnail_url"
## [31] "host_picture_url"             "host_neighbourhood"
## [33] "host_listings_count"          "host_total_listings_count"
## [35] "host_verifications"           "host_has_profile_pic"
## [37] "host_identity_verified"       "street"
## [39] "neighbourhood"                "neighbourhood_cleansed"
## [41] "neighbourhood_group_cleansed" "city"
## [43] "state"                        "zipcode"
## [45] "market"                       "smart_location"
## [47] "country_code"                 "country"
## [49] "latitude"                     "longitude"
## [51] "is_location_exact"            "property_type"
## [53] "room_type"                    "accommodates"
## [55] "bathrooms"                    "bedrooms"
## [57] "beds"                         "bed_type"
## [59] "amenities"                    "square_feet"
## [61] "weekly_price"                 "monthly_price"
## [63] "security_deposit"             "cleaning_fee"
## [65] "guests_included"              "extra_people"
## [67] "minimum_nights"               "maximum_nights"
## [69] "calendar_updated"             "has_availability"
## [71] "availability_30"              "availability_60"
## [73] "availability_90"              "availability_365"
```

```
## [75] "calendar_last_scraped"            "number_of_reviews"
## [77] "first_review"                     "last_review"
## [79] "review_scores_rating"             "review_scores_accuracy"
## [81] "review_scores_cleanliness"        "review_scores_checkin"
## [83] "review_scores_communication"      "review_scores_location"
## [85] "review_scores_value"              "requires_license"
## [87] "license"                          "jurisdiction_names"
## [89] "instant_bookable"                 "is_business_travel_ready"
## [91] "cancellation_policy"              "require_guest_profile_picture"
## [93] "require_guest_phone_verification" "calculated_host_listings_count"
## [95] "reviews_per_month"
```

Impute missing value

Looking at the number of missing values and taking a closer look at the data description, we can see that NA does not always mean that the values are missing. e.g.: In case of the feature *"beds","cleaning_fee","security_deoposit"*, NA just means that there is *"no beds"/"no cleaning_fee"/"no security_deoposit"* in the room. Since I will use advanced tree, I just code such NA values to say -1.

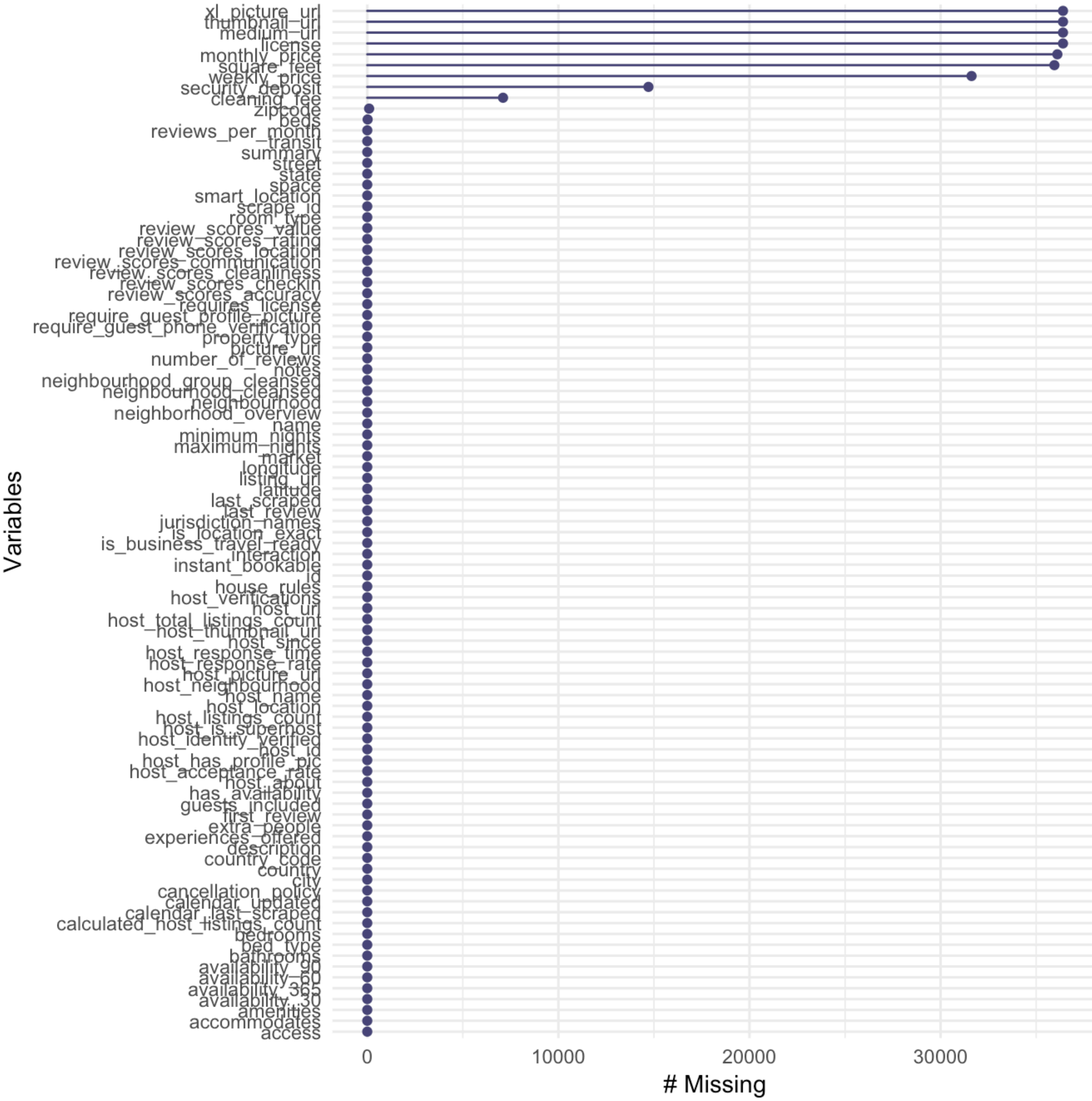
```
# finding missing data
sort(sapply(fulldata, function(x) {sum(is.na(x))}), decreasing = F)
```

```
##                               id                      listing_url
##                                0                                0
##                        scrape_id                     last_scraped
##                                0                                0
##                             name                          summary
##                                0                                0
##                            space                      description
##                                0                                0
##              experiences_offered            neighborhood_overview
##                                0                                0
##                            notes                          transit
##                                0                                0
##                           access                      interaction
##                                0                                0
##                      house_rules                      picture_url
##                                0                                0
##                          host_id                         host_url
##                                0                                0
##                        host_name                       host_since
##                                0                                0
##                    host_location                       host_about
##                                0                                0
##               host_response_time               host_response_rate
```

```
##                            0                              0
##         host_acceptance_rate              host_is_superhost
##                            0                              0
##           host_thumbnail_url               host_picture_url
##                            0                              0
##           host_neighbourhood            host_listings_count
##                            0                              0
##    host_total_listings_count             host_verifications
##                            0                              0
##         host_has_profile_pic         host_identity_verified
##                            0                              0
##                       street                  neighbourhood
##                            0                              0
##       neighbourhood_cleansed   neighbourhood_group_cleansed
##                            0                              0
##                         city                          state
##                            0                              0
##                       market                 smart_location
##                            0                              0
##                 country_code                        country
##                            0                              0
##                     latitude                      longitude
##                            0                              0
##            is_location_exact                  property_type
##                            0                              0
##                    room_type                   accommodates
##                            0                              0
##                    bathrooms                       bedrooms
##                            0                              0
##                     bed_type                      amenities
##                            0                              0
##              guests_included                   extra_people
##                            0                              0
##               minimum_nights                 maximum_nights
##                            0                              0
##             calendar_updated               has_availability
##                            0                              0
##              availability_30                availability_60
##                            0                              0
##              availability_90               availability_365
##                            0                              0
##        calendar_last_scraped              number_of_reviews
##                            0                              0
##                 first_review                    last_review
##                            0                              0
##         review_scores_rating         review_scores_accuracy
##                            0                              0
```

```
##        review_scores_cleanliness             review_scores_checkin
##                                0                                 0
##      review_scores_communication            review_scores_location
##                                0                                 0
##              review_scores_value                  requires_license
##                                0                                 0
##               jurisdiction_names                  instant_bookable
##                                0                                 0
##         is_business_travel_ready               cancellation_policy
##                                0                                 0
##    require_guest_profile_picture require_guest_phone_verification
##                                0                                 0
##   calculated_host_listings_count                 reviews_per_month
##                                0                                 1
##                             beds                           zipcode
##                               19                                97
##                     cleaning_fee                  security_deposit
##                             7103                             14712
##                     weekly_price                       square_feet
##                            31618                             35951
##                    monthly_price                     thumbnail_url
##                            36108                             36402
##                       medium_url                    xl_picture_url
##                            36402                             36402
##                          license
##                            36402
```

```
# Visualize missing values of data
gg_miss_var(fulldata)
```


xl_picture_url
thumbnail_url
medium_url
license
monthly_price
square_feet
weekly_price
security_deposit
cleaning_fee
zipcode
beds
reviews_per_month
transit
summary
street
state
space
smart_location
scrape_id
room_type
review_scores_value
review_scores_rating
review_scores_location
review_scores_communication
review_scores_cleanliness
review_scores_checkin
review_scores_accuracy
requires_license
require_guest_profile_picture
require_guest_phone_verification
property_type
picture_url
number_of_reviews
notes
neighbourhood_group_cleansed
neighbourhood_cleansed
neighbourhood
neighborhood_overview
name
minimum_nights
maximum_nights
market
longitude
listing_url
latitude
last_scraped
last_review
jurisdiction_names
is_location_exact
is_business_travel_ready
interaction
instant_bookable
id
house_rules
host_verifications
host_url
host_total_listings_count
host_thumbnail_url
host_since
host_response_time
host_response_rate
host_picture_url
host_neighbourhood
host_name
host_location
host_listings_count
host_is_superhost
host_identity_verified
host_id
host_has_profile_pic
host_acceptance_rate
host_about
has_availability
guests_included
first_review
extra_people
experiences_offered
description
country_code
country
city
cancellation_policy
calendar_updated
calendar_last_scraped
calculated_host_listings_count
bedrooms
bed_type
bathrooms
availability_90
availability_60
availability_365
availability_30
amenities
accommodates
access
Variables
0
10000
20000
30000
Missing

```
# Remove variables with only missing value as well as zipcode which I will use anothe
r variable to represent location information:
variables_to_romove0 <- c("xl_picture_url","thumbnail_url","medium_url","license","zi
pcode")
fulldata <- fulldata[,!colnames(fulldata) %in% variables_to_romove0, drop=F]
# For these categorical variables with high propotion of missing value, I use whether
there is a value or not to differentiate them by coding missing value to be 0, and ot
her values to be 1.
## monthly_price
fulldata <- fulldata %>%
  mutate(monthly_price_new = case_when(
    monthly_price > 0 ~ 'with monthly price',
    TRUE ~ "without monthly price"))
fulldata$monthly_price <- NULL
## square_feet
fulldata <- fulldata %>%
  mutate(square_feet_new = case_when(
    square_feet > 0 ~ 'with square feet',
    TRUE ~ "without square feet"))
fulldata$square_feet <- NULL
## weekly_price
fulldata <- fulldata %>%
  mutate(weekly_price_new = case_when(
    weekly_price > 0 ~ 'with weekly price',
    TRUE ~ "without weekly price"))
fulldata$weekly_price <- NULL
## security_deposit
fulldata <- fulldata %>%
  mutate(security_deposit_new = case_when(
    security_deposit > 0 ~ 'with security deposit',
    TRUE ~ "without security deposit"))
fulldata$security_deposit <- NULL
# Convert character columns to factor, filling NA values with "missing"
for (col in colnames(fulldata)){
  if (typeof(fulldata[,col]) == "character"){
    new_col = fulldata[,col]
    new_col[is.na(new_col)] = "missing"
    fulldata[col] = as.factor(new_col)
  }
}
# Fill remaining NA values with -1
fulldata[is.na(fulldata)] = -1
```

Explore Dataset Variables

1. Variance of each variable: If any variable has zero variance, then we would consider removing that feature.

```
# Take a look at the variance of each variable
nearZeroVar(data, saveMetrics = TRUE)
```

```
##                                freqRatio percentUnique zeroVar   nzv
## id                              1.000000  1.000000e+02   FALSE FALSE
## listing_url                     1.000000  1.000000e+02   FALSE FALSE
## scrape_id                       0.000000  3.434538e-03    TRUE  TRUE
## last_scraped                    1.202014  1.030361e-02   FALSE FALSE
## name                            1.166667  9.872235e+01   FALSE FALSE
## summary                        33.423077  9.437423e+01   FALSE FALSE
## space                         330.954545  7.293584e+01   FALSE FALSE
## description                     1.090909  9.885286e+01   FALSE FALSE
## experiences_offered             0.000000  3.434538e-03    TRUE  TRUE
## neighborhood_overview         557.000000  6.053716e+01   FALSE FALSE
## notes                        1271.692308  4.041077e+01   FALSE FALSE
## transit                       472.368421  6.452123e+01   FALSE FALSE
## access                        189.259259  5.875807e+01   FALSE FALSE
## interaction                   513.523810  5.803682e+01   FALSE FALSE
## house_rules                   131.946667  6.123781e+01   FALSE FALSE
## thumbnail_url                   0.000000  0.000000e+00    TRUE  TRUE
## medium_url                      0.000000  0.000000e+00    TRUE  TRUE
## picture_url                     1.000000  9.997252e+01   FALSE FALSE
## xl_picture_url                  0.000000  0.000000e+00    TRUE  TRUE
## host_id                         1.111111  8.489834e+01   FALSE FALSE
## host_url                        1.111111  8.489834e+01   FALSE FALSE
## host_name                       1.034483  2.887759e+01   FALSE FALSE
## host_since                      1.083333  1.045817e+01   FALSE FALSE
## host_location                   8.523937  3.334936e+00   FALSE FALSE
## host_about                    311.000000  5.447864e+01   FALSE FALSE
## host_response_time              1.593922  1.717269e-02   FALSE FALSE
## host_response_rate              1.995248  2.610249e-01   FALSE FALSE
## host_acceptance_rate            0.000000  3.434538e-03    TRUE  TRUE
## host_is_superhost               4.794229  6.869075e-03   FALSE FALSE
## host_thumbnail_url              3.350000  8.470257e+01   FALSE FALSE
## host_picture_url                3.350000  8.470257e+01   FALSE FALSE
## host_neighbourhood              1.652364  1.147136e+00   FALSE FALSE
## host_listings_count             3.773698  1.579887e-01   FALSE FALSE
## host_total_listings_count       3.773698  1.579887e-01   FALSE FALSE
## host_verifications              1.219233  1.676054e+00   FALSE FALSE
## host_has_profile_pic          433.567164  6.869075e-03   FALSE  TRUE
## host_identity_verified          1.671193  6.869075e-03   FALSE FALSE
## street                          1.161946  8.174200e-01   FALSE FALSE
## neighbourhood                   1.251451  6.800385e-01   FALSE FALSE
## neighbourhood_cleansed          1.151584  7.281220e-01   FALSE FALSE
## neighbourhood_group_cleansed    1.073397  1.717269e-02   FALSE FALSE
## city                            1.162033  8.139854e-01   FALSE FALSE
```

```
## state                         14555.000000  1.717269e-02   FALSE  TRUE
## zipcode                           1.269027  6.525622e-01   FALSE FALSE
## market                          345.261905  6.182168e-02   FALSE  TRUE
## smart_location                    1.161946  8.174200e-01   FALSE FALSE
## country_code                      0.000000  3.434538e-03    TRUE  TRUE
## country                           0.000000  3.434538e-03    TRUE  TRUE
## latitude                          1.000000  5.165545e+01   FALSE FALSE
## longitude                         1.000000  3.944223e+01   FALSE FALSE
## is_location_exact                 5.025662  6.869075e-03   FALSE FALSE
## property_type                    10.102917  1.202088e-01   FALSE FALSE
## room_type                         1.116117  1.030361e-02   FALSE FALSE
## accommodates                      3.110556  5.495260e-02   FALSE FALSE
## bathrooms                        11.757250  6.182168e-02   FALSE FALSE
## bedrooms                          5.557845  3.777991e-02   FALSE FALSE
## beds                              2.982168  6.182168e-02   FALSE FALSE
## bed_type                         98.124567  1.717269e-02   FALSE  TRUE
## amenities                         3.235294  9.416472e+01   FALSE FALSE
## square_feet                       1.240000  3.125429e-01   FALSE FALSE
## weekly_price                      1.103774  1.246737e+00   FALSE FALSE
## monthly_price                     1.258065  2.095068e-01   FALSE FALSE
## security_deposit                  1.727813  4.602281e-01   FALSE FALSE
## cleaning_fee                      1.417006  5.529606e-01   FALSE FALSE
## guests_included                   3.232361  5.495260e-02   FALSE FALSE
## extra_people                      4.800708  3.262811e-01   FALSE FALSE
## minimum_nights                    1.046667  2.095068e-01   FALSE FALSE
## maximum_nights                    7.808937  7.796401e-01   FALSE FALSE
## calendar_updated                  1.712230  2.163759e-01   FALSE FALSE
## has_availability                  0.000000  3.434538e-03    TRUE  TRUE
## availability_30                   6.462671  1.064707e-01   FALSE FALSE
## availability_60                   6.395797  2.095068e-01   FALSE FALSE
## availability_90                   6.296758  3.125429e-01   FALSE FALSE
## availability_365                  8.204082  1.257041e+00   FALSE FALSE
## calendar_last_scraped             1.017633  1.030361e-02   FALSE FALSE
## number_of_reviews                 1.355268  1.030361e+00   FALSE FALSE
## first_review                      1.373737  8.222283e+00   FALSE FALSE
## last_review                       1.401990  4.461464e+00   FALSE FALSE
## review_scores_rating              4.461300  1.751614e-01   FALSE FALSE
## review_scores_accuracy            2.947806  3.091084e-02   FALSE FALSE
## review_scores_cleanliness         1.708784  3.091084e-02   FALSE FALSE
## review_scores_checkin             5.402209  3.091084e-02   FALSE FALSE
## review_scores_communication       5.882832  3.091084e-02   FALSE FALSE
## review_scores_location            2.038813  3.091084e-02   FALSE FALSE
## review_scores_value               1.570521  3.091084e-02   FALSE FALSE
## requires_license                  0.000000  3.434538e-03    TRUE  TRUE
## license                           0.000000  0.000000e+00    TRUE  TRUE
## jurisdiction_names            29114.000000  1.030361e-02   FALSE  TRUE
## instant_bookable                  2.074227  6.869075e-03   FALSE FALSE
```

```
## is_business_travel_ready           13.697627  6.869075e-03   FALSE FALSE
## cancellation_policy                 1.819914  1.717269e-02   FALSE FALSE
## require_guest_profile_picture      29.584034  6.869075e-03   FALSE  TRUE
## require_guest_phone_verification   26.493862  6.869075e-03   FALSE  TRUE
## calculated_host_listings_count      4.558072  8.242891e-02   FALSE FALSE
## reviews_per_month                   1.012766  3.012090e+00   FALSE FALSE
```

```
# Remove variables with zero variance:
variables_to_romove1 <- c("scrape_id", "experiences_offered", "thumbnail_url", " medi
um_url", "xl_picture_url", "host_acceptance_rate", "has_availability", "requires_lice
nse", "license","country_code","country")
fulldata <- fulldata[,!colnames(fulldata) %in% variables_to_romove1, drop=F]
```

2. Categorical variables about location: Remove categorical variables about location with too many different levels or typos, and leave *"neighbourhood_group_cleansed"* which is clean.

```
# Remove categorical variables about location and leave "neighbourhood_group_cleansed
":
variables_to_romove2 <- c("neighbourhood","neighbourhood_cleansed","jurisdiction_name
s","street","city","state","market","smart_location","longitude","is_location_exact")
fulldata <- fulldata[,!colnames(fulldata) %in% variables_to_romove2, drop=F]
```

3. Host information: Remove variables about host basic information which is not related to the room directly.

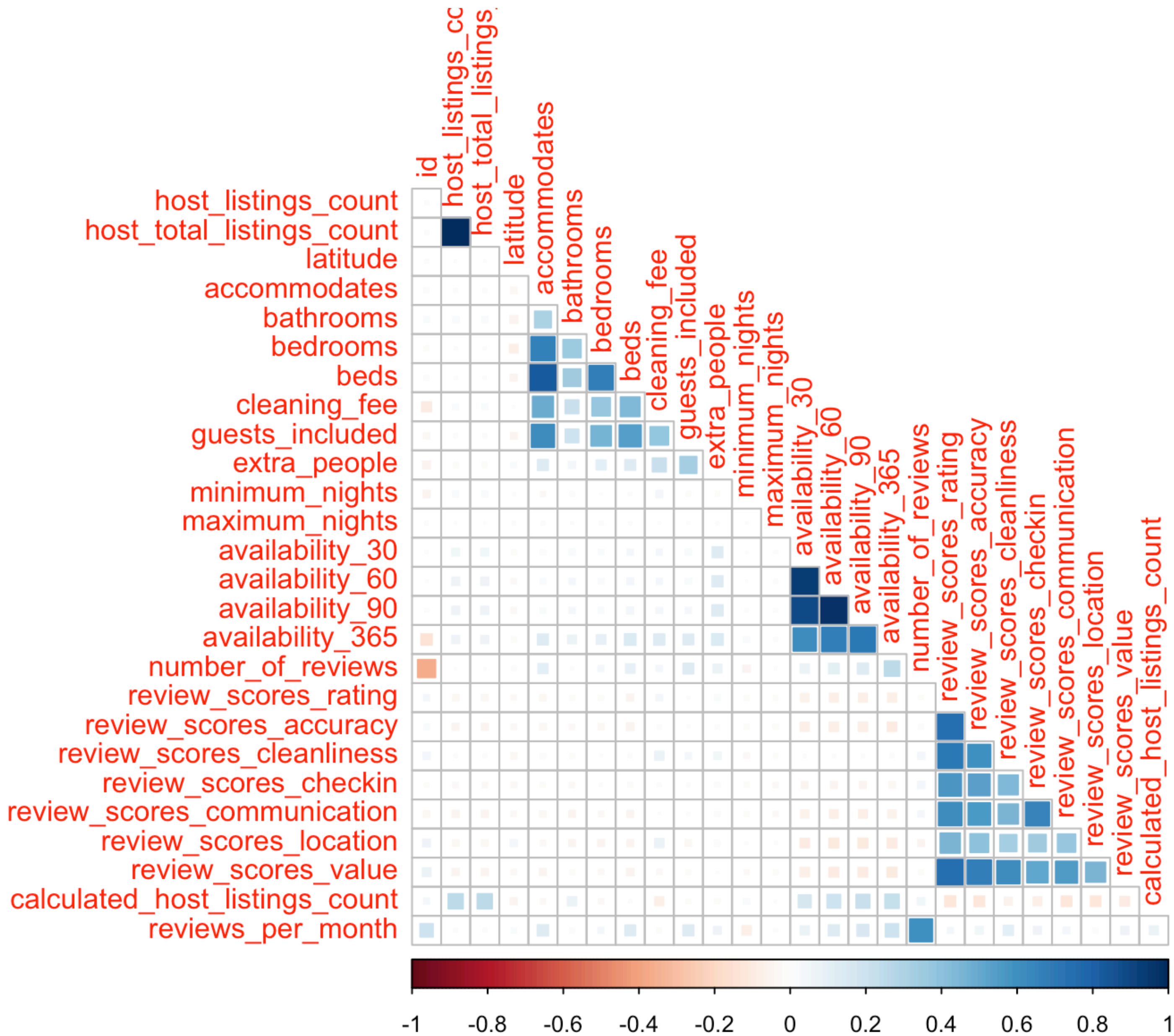
```
variables_to_romove3 <- c("host_location","host_id","host_url","host_name","host_thum
bnail_url","host_picture_url","host_verifications","host_neighbourhood")
fulldata <- fulldata[,!colnames(fulldata) %in% variables_to_romove3, drop=F]
```

4. URL variables: Remove useless url variables which have no contribution to prediction of price.

```
variables_to_romove4 <- c("listing_url","picture_url")
fulldata <- fulldata[,!colnames(fulldata) %in% variables_to_romove4, drop=F]
```

5. Self-related variables: Determine whether any of the numeric variables are highly correlated with one another. A cutoff that I usually use is +-0.7, but since we will be using a tree-based model, we deal with variables with correlations above 0.9 only.

```
# Correlations of numeric data to check for bivariate correlations
nums <- sapply(fulldata, is.numeric)
corrplot(cor(fulldata[,nums]),method = 'square',type = 'lower',diag = F)
```




```
# Remove "availability_30","availability_60","availability_90" because they related t
o each other
variables_to_romove5 <- c("availability_30","availability_60","availability_90")
fulldata <- fulldata[,!colnames(fulldata) %in% variables_to_romove5, drop=F]
```

6. Date-related variables: Remove date-related variables with too many levels

```
variables_to_romove6 <- c("last_scraped","host_since","calendar_last_scraped")
fulldata <- fulldata[,!colnames(fulldata) %in% variables_to_romove6, drop=F]
```

7. first_review & last_review: see if the difference between last review and first review is greater than one year

```
f <- as.Date(fulldata$first_review)
l <- as.Date(fulldata$last_review)
fulldata <- fulldata %>%
  mutate(review_diff = ifelse(l-f < 365, T, F))
fulldata$first_review <- NULL
fulldata$last_review <- NULL
```

Data Preparation

0. Function preparation

```
# Function: Looking for Keyword
keyword.look <- function(x){
  keywords <- data_frame(x)
  keywords %>%
    unnest_tokens(words, x)
}
# Function: Extract keyword
keyword.detect <- function(x,key){
  fulldata <- mutate(fulldata, x = grepl(paste(pattern = key, collapse = "|"),  x = x
))
}
# Function: Derive the length
length.impact <- function(x){
  fulldata <- fulldata %>%
    mutate(x_impact = case_when(str_length(x) == 0 |
                              str_length(x) < mean(str_length(x) ) ~ "short/no de
scription",
                            TRUE ~ "Good description"))
}
```

1. Calendar_updated

```
fulldata <- fulldata %>%
  mutate(calendar_updated_daysago = case_when(
    grepl("today", calendar_updated) ~ 0,
    grepl("yesterday", calendar_updated) ~ 1,
    grepl("a week ago", calendar_updated) ~ 7,
    grepl("never", calendar_updated) ~ 52 * (365 / 12),
    grepl("day", calendar_updated) ~ as.numeric(str_extract(calendar_updated, "^[0-9]
*")),
    grepl("week", calendar_updated) ~ as.numeric(str_extract(calendar_updated, "^[0-9
]*")) * 7,
    grepl("month", calendar_updated) ~ as.numeric(str_extract(calendar_updated, "^[0-
9]*")) * (365 / 12)
  ))
fulldata$calendar_updated <- NULL
```

2. Name

```
# Detect the key words for Name
#keyword.look(fulldata$name) #COZY, SPACIOUS,EASY ACCESS
# Extract keyword_name from name
keyword_name <- c('cozy','spacious','easy access')
fulldata <- fulldata %>%
  mutate(name_keyword = grepl(paste(pattern = keyword_name, collapse = "|"), ignore.c
ase = T, x = fulldata$name))
# See the impact if there's penalty of not putting any names or short (less than avg)
fulldata <- fulldata %>%
  mutate(name_impact = case_when(
    str_length(fulldata$name) == 0 |
      str_length(fulldata$name) < mean(str_length(fulldata$name) ) ~ "short/no Name",
    TRUE ~ "Good Name"
  ))
fulldata$name <- NULL
```

3. Access

```
# lf there's penalty of not putting any access or short (less than avg)
fulldata <- fulldata %>%
  mutate(access_impact = case_when(
    str_length(fulldata$access) == 0 |
      str_length(fulldata$access) <  mean(str_length(fulldata$access)) ~ 'short/no ac
cess',
    TRUE ~ 'Long access'))
fulldata$access <- NULL
```

4. Summary

```
#attraction sights 20
attraction <-c('SoHo','Chelsea',"Statue of Liberty","Central Park","Rockefeller Cente
r","Metropolitan Museum","Broadway","Theater","Museum","Bride","Empire State","9/11",
"High Line","Time Square","Fifth Ave","Grand Central Terminal","One World Observatory
","Frick Collection","New York Public Library","Wall Street"," Radio City Music Hall"
," St Patrick's Cathedral","Carnegie Hall","Bryant Park")
#convenenece
convenience <-c("train", "walking", 'walk', 'shopping', 'mall', 'min', 'minutes', 'sh
op','shops','restaurant','subway')
#MYwords
mywords <- c("safe","train","subways","subway","transportation", 'bar','bars','convin
enient','parking','available','cool','awesome','clean', 'new', 'best','plaza','close'
,'lovely','quiet','food')
#keyword.look(fulldata$summary)#perfect,spot, intern, student
#Attractions
keyword.detect(fulldata$summary, attraction)
fulldata <- fulldata %>%
  mutate(summary_attraction = grepl(paste(pattern = attraction, collapse = "|"),ignor
e.case = T,  x = fulldata$summary))
#Keywords
keyword_summary <- c('student','intern','perfect')
fulldata <- fulldata %>%
  mutate(summary_keyword = grepl(paste(pattern = keyword_summary, collapse = "|"),  i
gnore.case = T,x = fulldata$summary))
#Mywords
fulldata <- fulldata %>%
  mutate(summary_myword =  grepl(paste(pattern = mywords, collapse = "|"),ignore.case
= T,  x = fulldata$summary))
#convenience
fulldata <- fulldata %>%
  mutate(summary_convenience =  grepl(paste(pattern = convenience, collapse = "|"),
ignore.case = T,x = fulldata$summary))
#See the impact of detailed description.
fulldata <- fulldata %>%
  mutate(summary_impact = case_when(
    str_length(fulldata$summary) == 0 |
      str_length(fulldata$summary) <mean(str_length(fulldata$summary)) ~ 'short/no Su
mmary',
    TRUE ~ 'Long Summary'))
fulldata$summary <- NULL
```

5. Space

```
#Look for Key words
#keyword.look(fulldata$space) #private
#"PRIVATE"
fulldata <- fulldata %>%
  mutate(space_keyword = grepl('private',  x = fulldata$space))
#See the impact of detailed description.
fulldata <- fulldata %>%
  mutate(space_impact = case_when(
    str_length(fulldata$space) == 0 |
      str_length(fulldata$space) <mean(str_length(fulldata$space)) ~ 'short/no Summar
y',
    TRUE ~ 'Long Summary'))
fulldata$space <- NULL
```

6. Description

```
#Keywords
#keyword.look(fulldata$description) #student, intern, perfect
#Attractions
#keyword.detect(fulldata$summary, attraction)
fulldata <- fulldata %>%
  mutate(description_attraction = grepl(paste(pattern = attraction, collapse = "|"),
ignore.case = T,x = fulldata$description))
#Keywords
keyword_summary <- c('student','intern','perfect')
fulldata <- fulldata %>%
  mutate(description_keyword = grepl(paste(pattern = keyword_summary, collapse = "|")
, ignore.case = T, x = fulldata$description))
#Mywords
fulldata <- fulldata %>%
  mutate(description_myword =  grepl(paste(pattern = mywords, collapse = "|"), ignore
.case = T, x = fulldata$description))
#convenience
fulldata <- fulldata %>%
  mutate(description_convenience =  grepl(paste(pattern = convenience, collapse = "|"
), ignore.case = T, x = fulldata$description))
#See the impact of detailed description.
fulldata <- fulldata %>%
  mutate(description_impact = case_when(
    str_length(fulldata$description) == 0 |
      str_length(fulldata$description) <mean(str_length(fulldata$description)) ~ 'sho
rt/no Summary',
    TRUE ~ 'Long Summary'))
fulldata$description <- NULL
```

7. Neighborhood

```
#Keywords
#keyword.look(fulldata$neighborhood_overview)#indian, food
#Attraction
fulldata <- fulldata %>%
  mutate(neighborhood_overview_attraction = grepl(paste(pattern = attraction, collaps
e = "|"),  ignore.case = T,x = fulldata$neighborhood_overview))
#Keywords
keyword_neighborhood_overview<- c('indian', 'food')
fulldata <- fulldata %>%
  mutate(neighborhood_overview_keyword = grepl(paste(pattern = keyword_neighborhood_o
verview, collapse = "|"),ignore.case = T,  x = fulldata$neighborhood_overview))
#Mywords
fulldata <- fulldata %>%
  mutate(neighborhood_overview_myword =  grepl(paste(pattern = mywords, collapse = "|
"),  ignore.case = T, x = fulldata$neighborhood_overview))
#convenience
fulldata <- fulldata %>%
  mutate(neighborhood_overview_convenience =  grepl(paste(pattern = convenience, coll
apse = "|"), ignore.case = T, x = fulldata$neighborhood_overview))
#See the impact of detailed description.
fulldata <- fulldata %>%
  mutate(neighborhood_overview_impact = case_when(
    str_length(fulldata$neighborhood_overview) == 0 |
      str_length(fulldata$neighborhood_overview) <mean(str_length(fulldata$neighborho
od_overview)) ~ 'short/no Summary',
    TRUE ~ 'Long Summary'))
fulldata$neighborhood_overview <- NULL
```

8. Interaction

```
# See the impact of detailed Interaction
fulldata <- fulldata %>%
  mutate(interaction_impact = case_when(
    str_length(fulldata$interaction) == 0 |
      str_length(fulldata$interaction) < mean(str_length(fulldata$interaction)) ~ 'sh
ort/no description',
    TRUE ~ 'Good description'))
fulldata$interaction <- NULL
```

9. Transit

```
#Keywords
#keyword.look(fulldata$transit) #parking, limited, near'
transit_keyword <- c('parking','limited')
fulldata <- fulldata %>%
  mutate(transit_keyword = grepl('limited', ignore.case = T, x = fulldata$transit))
#convenience
fulldata <- fulldata %>%
  mutate(transit_convenience =  grepl(paste(pattern = convenience, collapse = "|"), i
gnore.case = T,  x = fulldata$transit))
#See the impact of detailed description.
fulldata <- fulldata %>%
  mutate(neighborhood_overview_impact = case_when(
    str_length(fulldata$transit) == 0 |
      str_length(fulldata$transit) <mean(str_length(fulldata$transit)) ~ 'short/no Su
mmary',
    TRUE ~ 'Long Summary'))
fulldata$transit <- NULL
```

10. Notes

```
#Keywords
#keyword.look(fulldata$notes) #extremely, border,
#Mywords
#Attraction
fulldata <- fulldata %>%
  mutate(notes_attraction = grepl(paste(pattern = attraction, collapse = "|"),  ignor
e.case = T,x = fulldata$notes))
#Mywords
fulldata <- fulldata %>%
  mutate(notes_myword =  grepl(paste(pattern = mywords, collapse = "|"),  ignore.case
= T, x = fulldata$notes))
#convenience
fulldata <- fulldata %>%
  mutate(notes_convenience =  grepl(paste(pattern = convenience, collapse = "|"), ign
ore.case = T, x = fulldata$notes))
#See the impact of detailed description.
fulldata <- fulldata %>%
  mutate(notes_impact = case_when(
    str_length(fulldata$notes) == 0 |
      str_length(fulldata$notes) <mean(str_length(fulldata$notes)) ~ 'short/no Summar
y',
    TRUE ~ 'Long Summary'))
fulldata$notes <- NULL
```

11. House Rules

```
#House Rules
fulldata <- fulldata %>%
  mutate(rules_keyword = grepl('no',  ignore.case = T,x = fulldata$house_rules))
#Good description
fulldata <- fulldata %>%
  mutate(rules_impact = case_when(str_length(fulldata$house_rules) == 0 |
                                   str_length(fulldata$house_rules) < mean(str_lengt
h(fulldata$house_rules)) ~ "Short/No description",
                                  TRUE ~ "Good description"))
fulldata$house_rules <- NULL
```

12. Amenities

```
fulldata <- mutate(fulldata, amenities_impact = str_count(amenities, ','))
# Decorate amenities
## amenities_impact_new <- c("Advanced","Based")
fulldata <-fulldata %>%
  mutate(amenities_impact_new = case_when(amenities_impact > 65 ~"Advanced",
                                          TRUE ~ "Basic"))
fulldata$amenities <- NULL
fulldata$amenities_impact <- NULL
```

13. Property_type

```
## new property_type <- c("Apartment", "House", "Condo", "Loft")
fulldata <-fulldata %>%
  mutate(property_type_new = case_when(property_type == "Apartment" ~"Apartment",
                                       property_type == "House" ~ "House",
                                       property_type == "Condominium" ~ "Condo",
                                       property_type == "Loft"  ~ "Loft",
                                       TRUE ~ "Rest"))
fulldata$property_type <- NULL
```

14. Host_about

```
#Having description on host_about
fulldata <- fulldata %>%
  mutate(selfintro_impact = case_when(str_length(fulldata$host_about) == 0  ~ 0,
                                      TRUE ~ 1))
fulldata$host_about <- NULL
```

15. Rating

```
fulldata$review_scores_checkin <- NULL
fulldata$review_scores_cleanliness <- NULL
fulldata$review_scores_communication <- NULL
fulldata$review_scores_location <- NULL
fulldata$review_scores_accuracy <- NULL
fulldata$review_scores_value <- NULL
```

16. host_response_rate

```
fulldata <-fulldata %>%
  mutate(host_response_rate_new = case_when(host_response_rate %in% c("100%","99%","9
8%","97%","96%","95%","94%","93%","92%","91%","90%","89%","88%","87%","86%","95%","84
%","83%","82%","81%","80%") ~"Excellent",
                                            host_response_rate == "N/A" ~ "Not Applic
apable",
                                            host_response_rate %in% c("0%","10%","11%
","12%","13%","14%","15%","16%","17%","18%","19%","20%") ~ "Bad",
                                            TRUE ~ "Rest"))
fulldata$host_response_rate <- NULL
```

17. host_response_time

```
fulldata <-fulldata %>%
  mutate(host_response_time_new = case_when(host_response_time == "N/A" ~ "Not Applic
apable",
                                            TRUE ~ "Rest"))
fulldata$host_response_time <- NULL
```

Transform those values with high skewness-

```
classes <- lapply(fulldata,function(x) class(x))
numeric_feats <- names(classes[classes=="integer" | classes=="numeric"])
skewed_feats <- sapply(numeric_feats, function(x) skewness(fulldata[[x]]))
skewed_feats <- skewed_feats[abs(skewed_feats) > .75]; skewed_feats
```

```
##            host_listings_count      host_total_listings_count
##                      64.066005                      64.066005
##                   accommodates                      bathrooms
##                       2.329335                       5.677214
##                       bedrooms                           beds
##                       1.935363                       3.250189
##                   cleaning_fee                guests_included
##                       1.488706                       3.624817
##                   extra_people                 minimum_nights
##                       4.148126                      46.755581
##                 maximum_nights              number_of_reviews
##                     178.709444                       3.198972
##           review_scores_rating calculated_host_listings_count
##                      -3.186702                       6.708563
##              reviews_per_month        calendar_updated_daysago
##                       3.150608                       2.476425
```

```
## We can take log transformation of features or other approaches for which skewness
more than 0.75 but here I just skipped this step because normalization is not neccess
ary for advanced trees
```

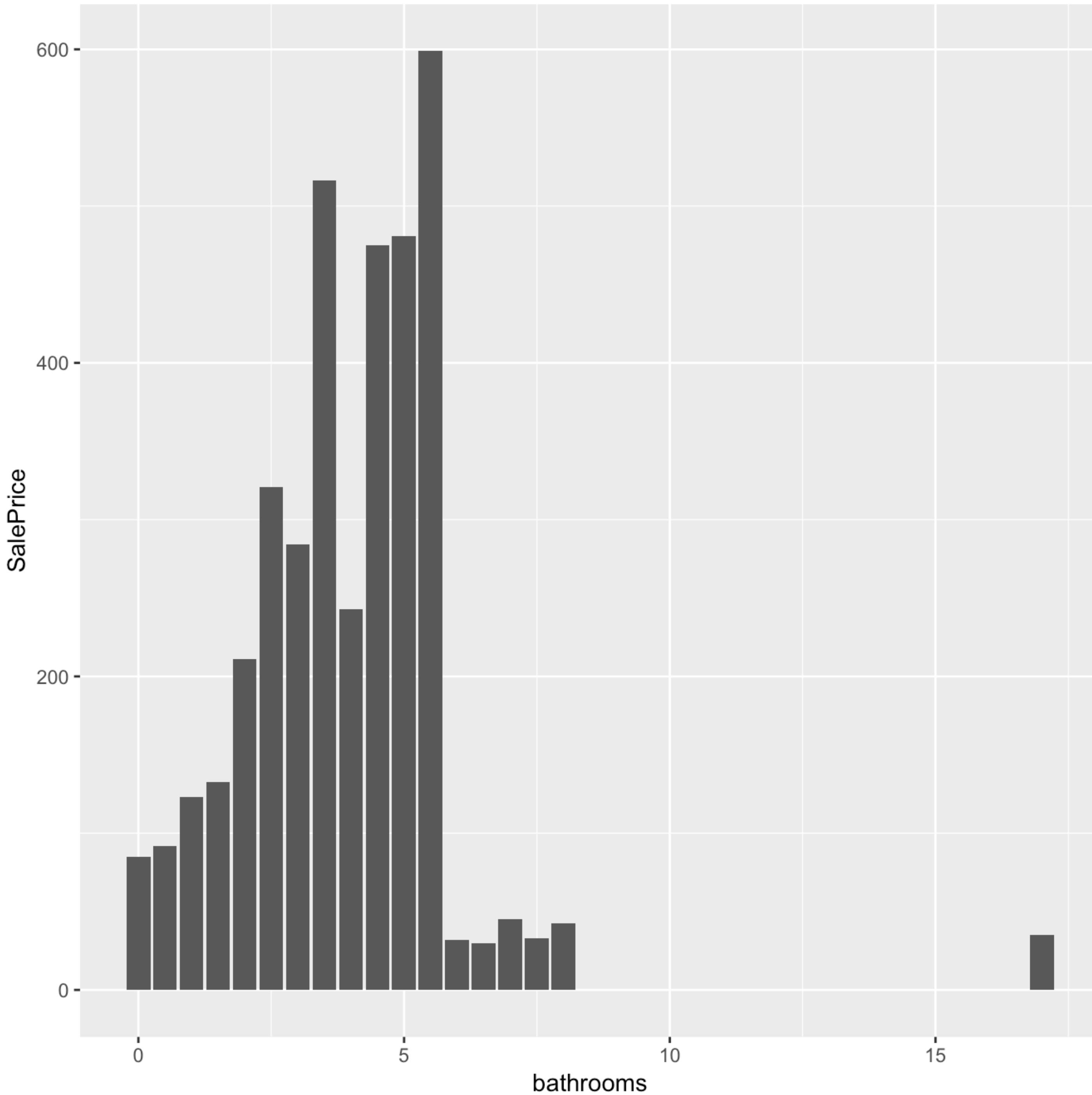
Split combined full dataset into train and test datasets

```
# Convert character columns to factor
for (col in colnames(fulldata)){
  if (typeof(fulldata[,col]) == "character"){
    fulldata[col] = as.factor(fulldata[,col])
  }
}
# Split combined full dataset into train and test datasets
train <- fulldata[1 : nrow(data),]
test <- fulldata[nrow(data) + 1 : nrow(testing),]
# Add SalePrice back to train dataset
train <- cbind(train, SalePrice)
train$id = NULL
```

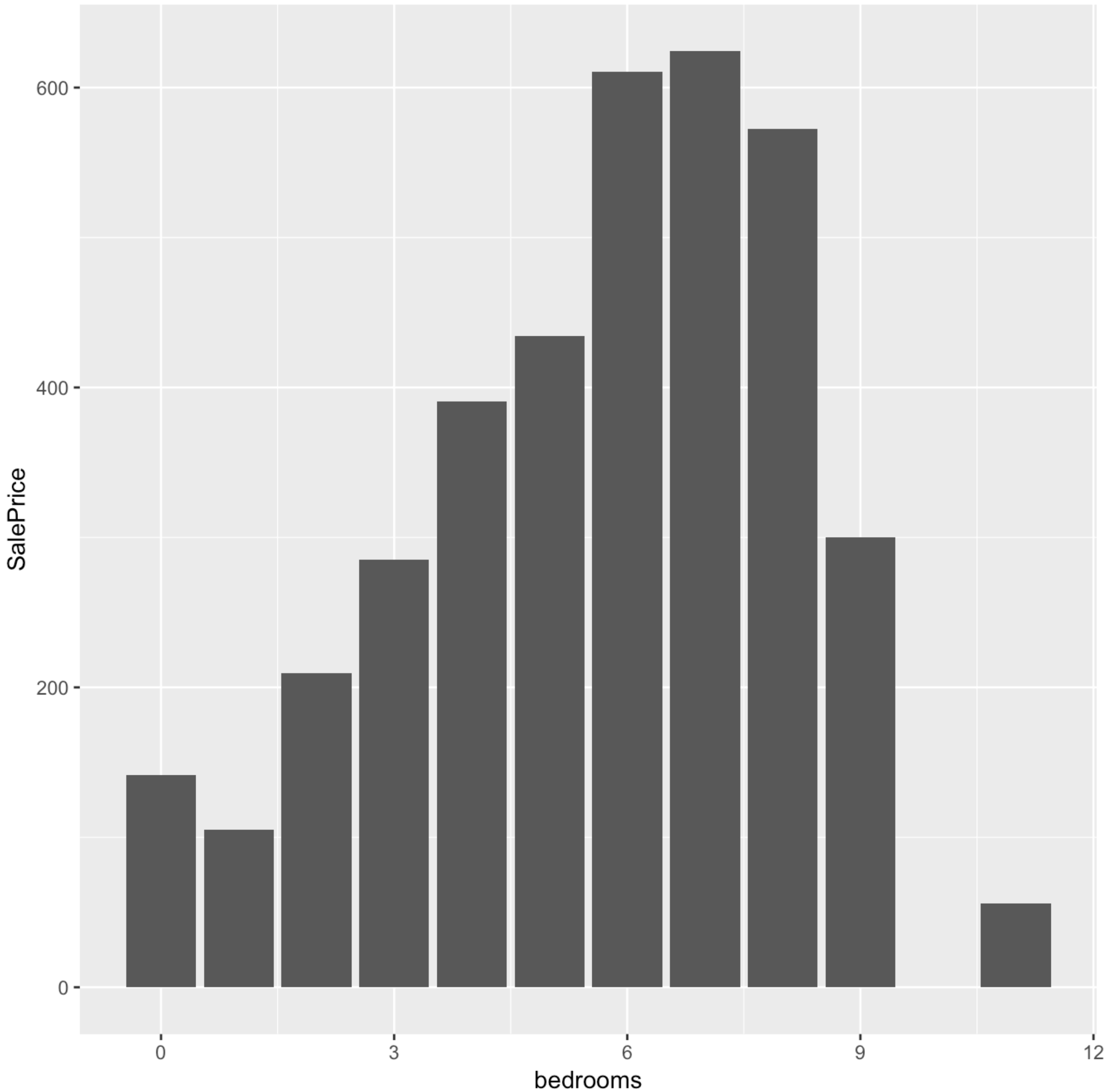
Further operation on variables left now

1. Check which variables left, and track outliers.

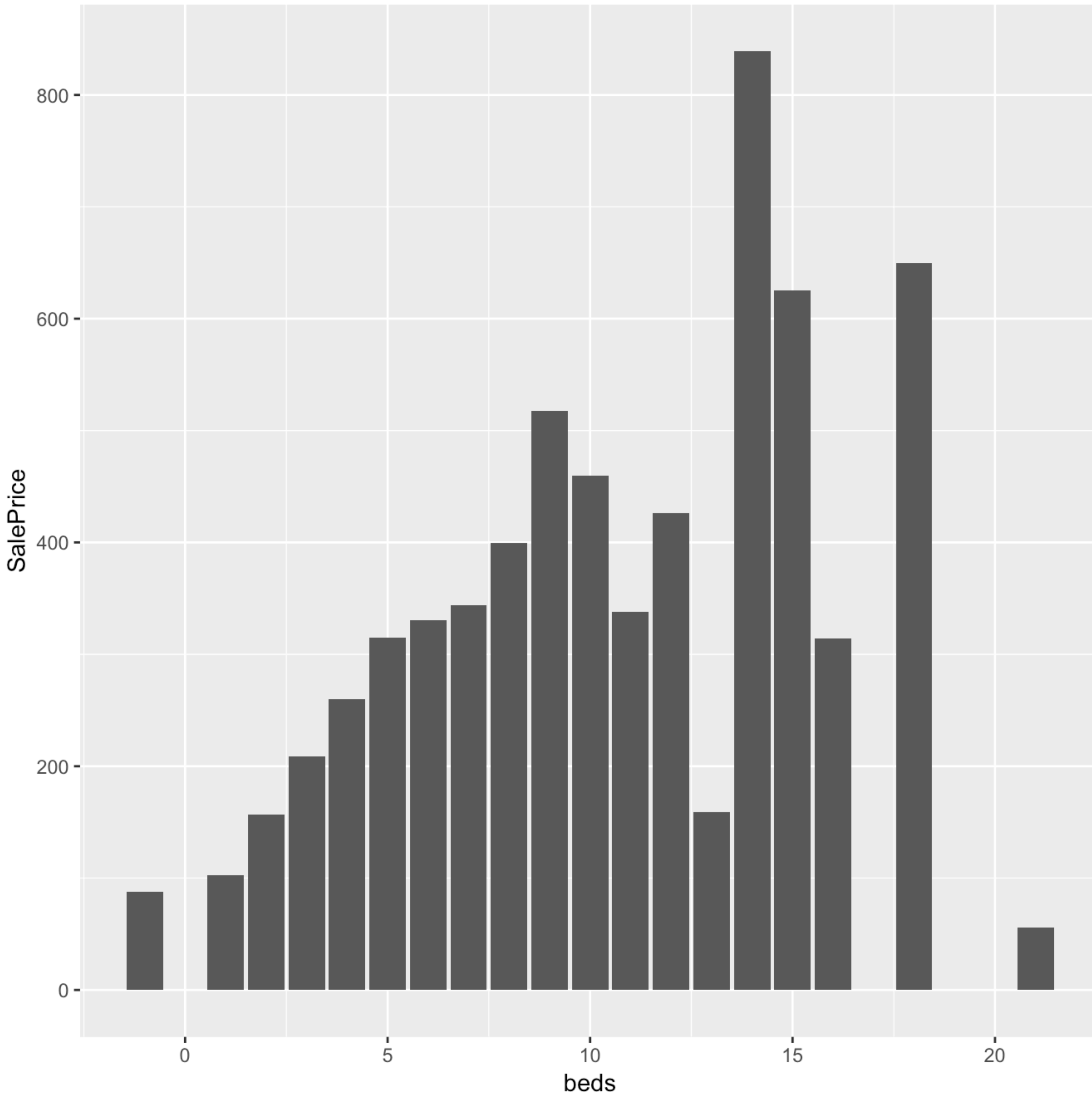
```
# Track outliers
ggplot(train,aes(x=bathrooms,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar')
```



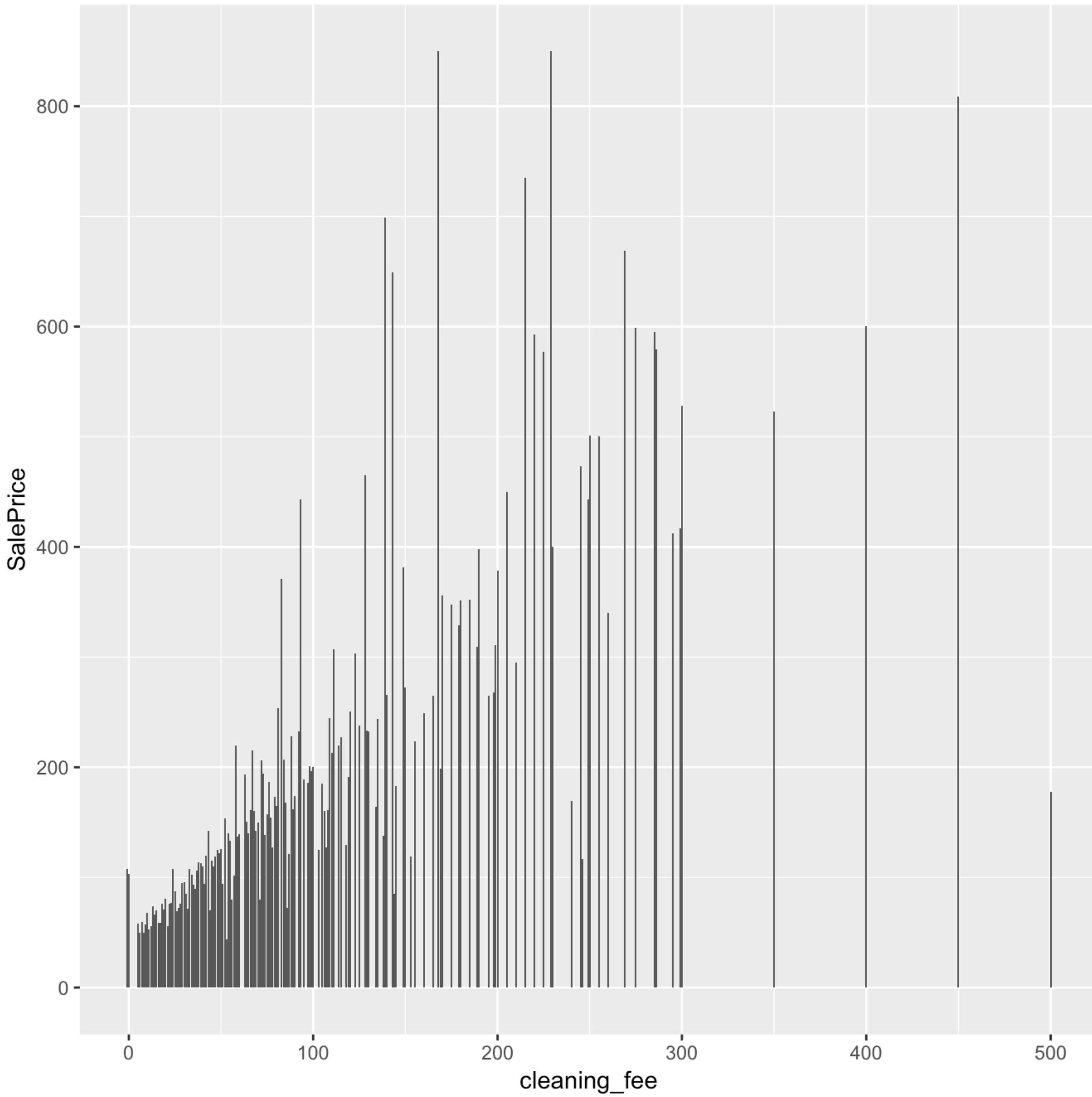

```
train[train$bathrooms>6,]$bathrooms <- mean(train$bathrooms)%>%as.numeric
ggplot(train,aes(x=bedrooms,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar')
```




```
train[train$bedrooms>10,]$bedrooms <- mean(train$bedrooms)%>%as.numeric
ggplot(train,aes(x=beds,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar')
```




```
train[train$beds>20,]$beds <- mean(train$beds)%>%as.numeric
ggplot(train,aes(x=cleaning_fee,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar')
```




```
train[train$cleaning_fee>420,]$cleaning_fee <- mean(train$cleaning_fee)%>%as.numeric
```

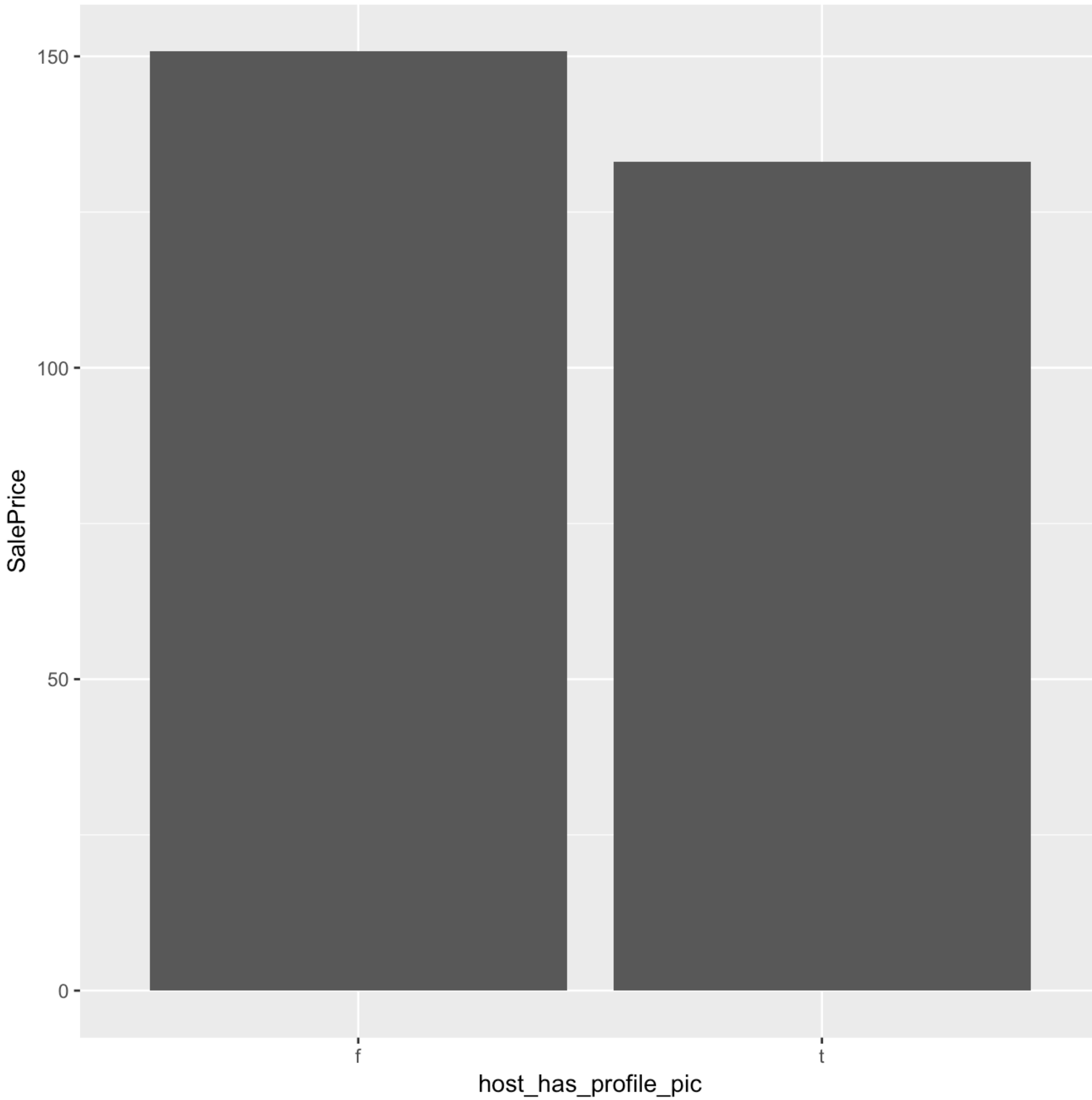
2. Variance of each variable: If any did have zero variance, then we would consider removing that feature. For those variables with near zero variance, we can take a look at their impact on SalePrice one by one from visualized bar chart.

```
# Take a look at the variance of each variable
nearZeroVar(train, saveMetrics = TRUE)
```

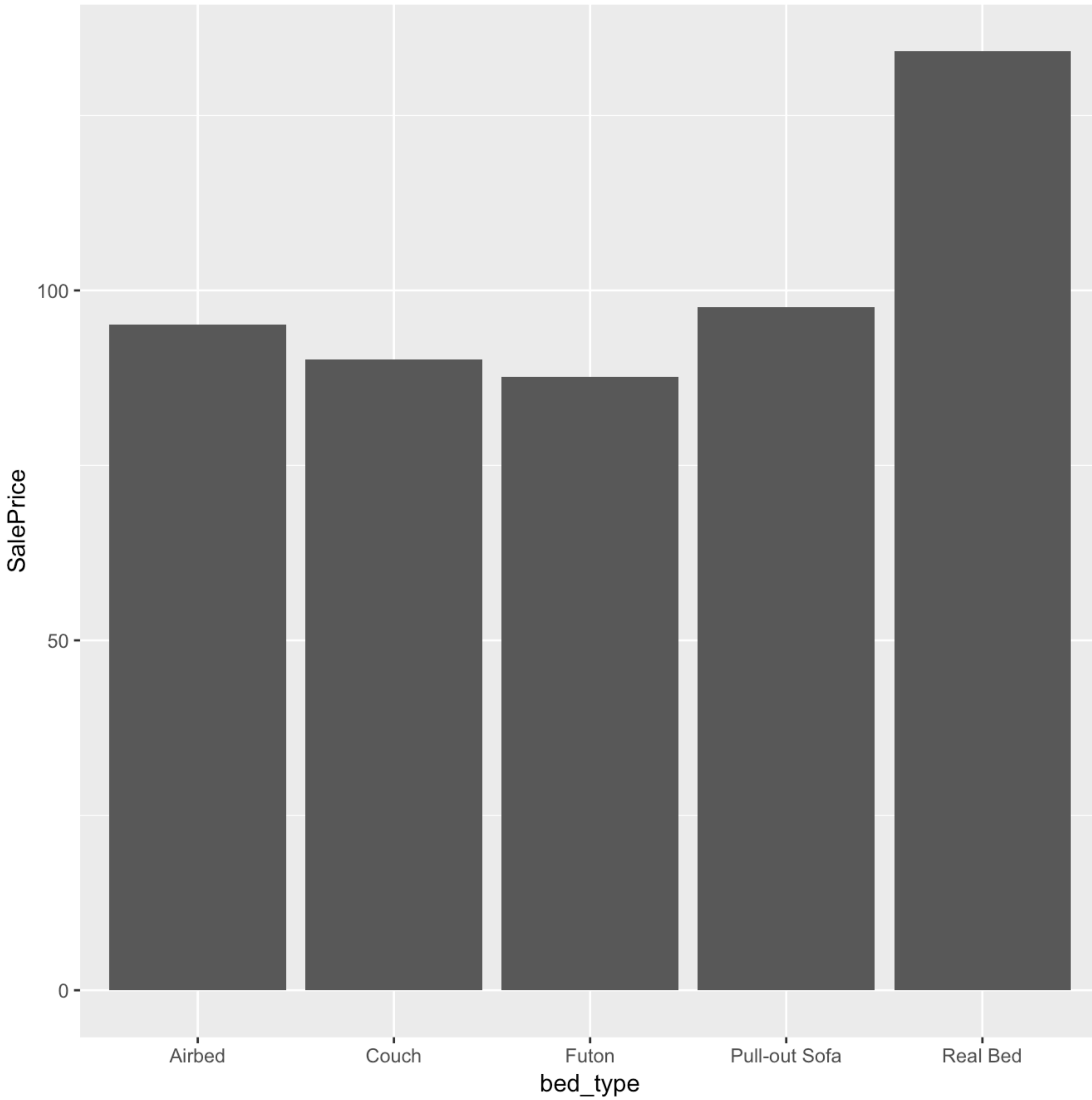
```
##                                    freqRatio percentUnique zeroVar   nzv
## host_is_superhost                   4.794229   0.006869075   FALSE FALSE
## host_listings_count                 3.773698   0.157988735   FALSE FALSE
## host_total_listings_count           3.773698   0.157988735   FALSE FALSE
## host_has_profile_pic              433.567164   0.006869075   FALSE  TRUE
## host_identity_verified              1.671193   0.006869075   FALSE FALSE
## neighbourhood_group_cleansed        1.073397   0.017172689   FALSE FALSE
## latitude                            1.000000  51.655447177   FALSE FALSE
## room_type                           1.116117   0.010303613   FALSE FALSE
## accommodates                        3.110556   0.054952603   FALSE FALSE
## bathrooms                          11.757250   0.048083528   FALSE FALSE
## bedrooms                            5.557845   0.037779915   FALSE FALSE
## beds                                2.982168   0.065256217   FALSE FALSE
## bed_type                           98.124567   0.017172689   FALSE  TRUE
## cleaning_fee                        2.048509   0.552960572   FALSE FALSE
## guests_included                     3.232361   0.054952603   FALSE FALSE
## extra_people                        4.800708   0.326281083   FALSE FALSE
## minimum_nights                      1.046667   0.209506800   FALSE FALSE
## maximum_nights                      7.808937   0.779640060   FALSE FALSE
## availability_365                    8.204082   1.257040802   FALSE FALSE
## number_of_reviews                   1.355268   1.030361313   FALSE FALSE
## review_scores_rating                4.461300   0.175161423   FALSE FALSE
## instant_bookable                    2.074227   0.006869075   FALSE FALSE
## is_business_travel_ready           13.697627   0.006869075   FALSE FALSE
## cancellation_policy                 1.819914   0.017172689   FALSE FALSE
## require_guest_profile_picture      29.584034   0.006869075   FALSE  TRUE
## require_guest_phone_verification   26.493862   0.006869075   FALSE  TRUE
## calculated_host_listings_count      4.558072   0.082428905   FALSE FALSE
## reviews_per_month                   1.012766   3.012089573   FALSE FALSE
## monthly_price_new                 119.813278   0.006869075   FALSE  TRUE
## square_feet_new                    92.922581   0.006869075   FALSE  TRUE
## weekly_price_new                    6.592177   0.006869075   FALSE FALSE
## security_deposit_new                1.358144   0.006869075   FALSE FALSE
## review_diff                         1.324445   0.006869075   FALSE FALSE
## calendar_updated_daysago            1.712230   0.212941338   FALSE FALSE
## name_keyword                        4.220728   0.006869075   FALSE FALSE
## name_impact                         1.327604   0.006869075   FALSE FALSE
## access_impact                       1.985950   0.006869075   FALSE FALSE
## summary_attraction                  3.107208   0.006869075   FALSE FALSE
## summary_keyword                     6.206931   0.006869075   FALSE FALSE
## summary_myword                      4.892734   0.006869075   FALSE FALSE
## summary_convenience                 2.744823   0.006869075   FALSE FALSE
## summary_impact                      1.243316   0.006869075   FALSE FALSE
```

```
## space_keyword                      5.862126   0.006869075   FALSE FALSE
## space_impact                       1.570041   0.006869075   FALSE FALSE
## description_attraction             1.900578   0.006869075   FALSE FALSE
## description_keyword                2.827527   0.006869075   FALSE FALSE
## description_myword                26.390405   0.006869075   FALSE  TRUE
## description_convenience            9.692618   0.006869075   FALSE FALSE
## description_impact                 2.083016   0.006869075   FALSE FALSE
## neighborhood_overview_attraction   3.781738   0.006869075   FALSE FALSE
## neighborhood_overview_keyword      7.434531   0.006869075   FALSE FALSE
## neighborhood_overview_myword       1.232137   0.006869075   FALSE FALSE
## neighborhood_overview_convenience  1.165886   0.006869075   FALSE FALSE
## neighborhood_overview_impact       1.731845   0.006869075   FALSE FALSE
## interaction_impact                 1.634694   0.006869075   FALSE FALSE
## transit_keyword                  178.728395   0.006869075   FALSE  TRUE
## transit_convenience                1.929470   0.006869075   FALSE FALSE
## notes_attraction                  77.058981   0.006869075   FALSE  TRUE
## notes_myword                       4.211384   0.006869075   FALSE FALSE
## notes_convenience                  6.896935   0.006869075   FALSE FALSE
## notes_impact                       2.388734   0.006869075   FALSE FALSE
## rules_keyword                      1.044950   0.006869075   FALSE FALSE
## rules_impact                       2.274036   0.006869075   FALSE FALSE
## amenities_impact_new            2425.333333   0.006869075   FALSE  TRUE
## property_type_new                 10.102917   0.017172689   FALSE FALSE
## selfintro_impact                   1.925643   0.006869075   FALSE FALSE
## host_response_rate_new             2.378439   0.013738151   FALSE FALSE
## host_response_time_new             2.641321   0.006869075   FALSE FALSE
## SalePrice                          1.012422   1.706965242   FALSE FALSE
```

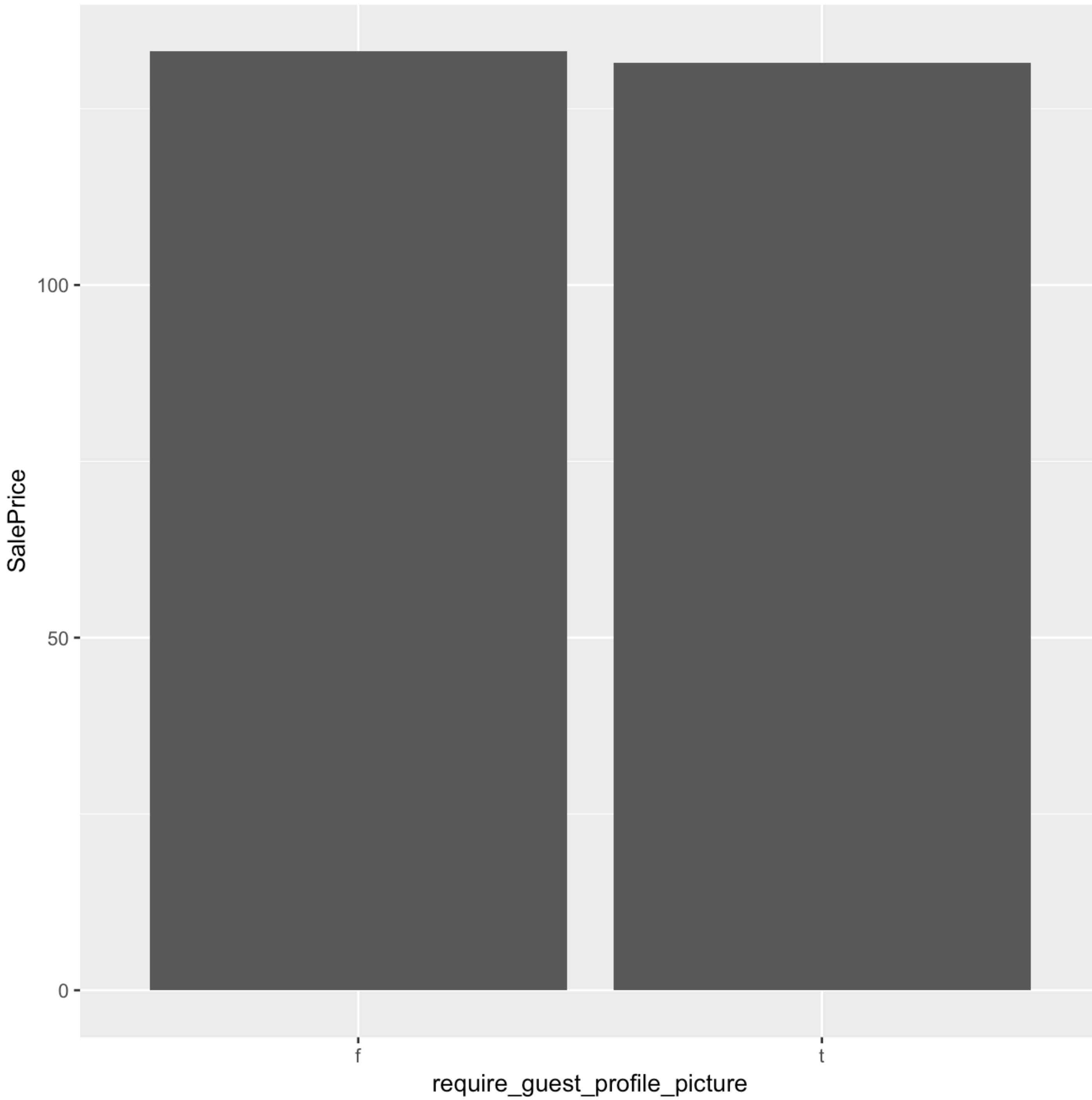
```
# Visualized bar chart
ggplot(train,aes(x=host_has_profile_pic,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



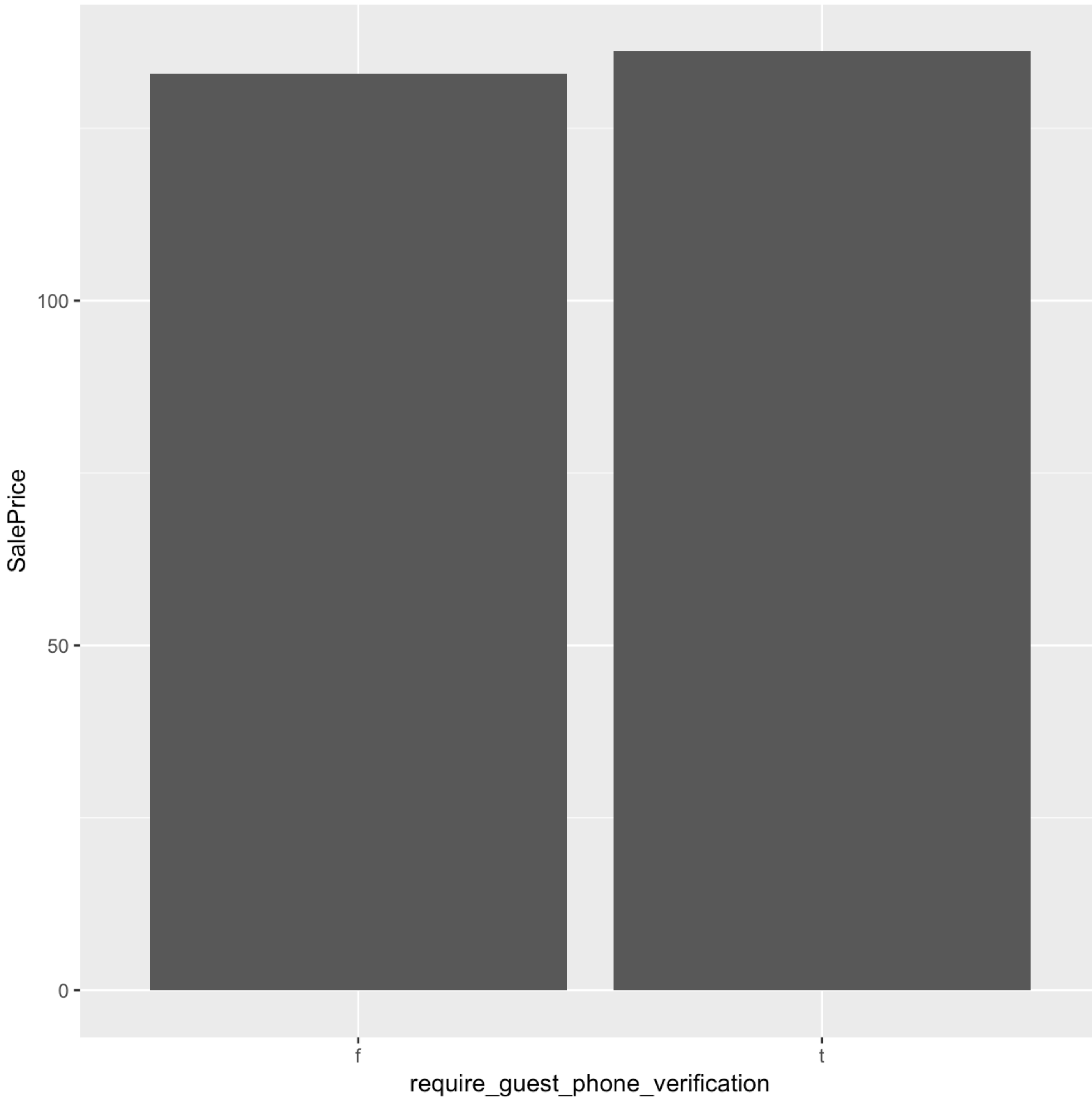

```
ggplot(train,aes(x=bed_type,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



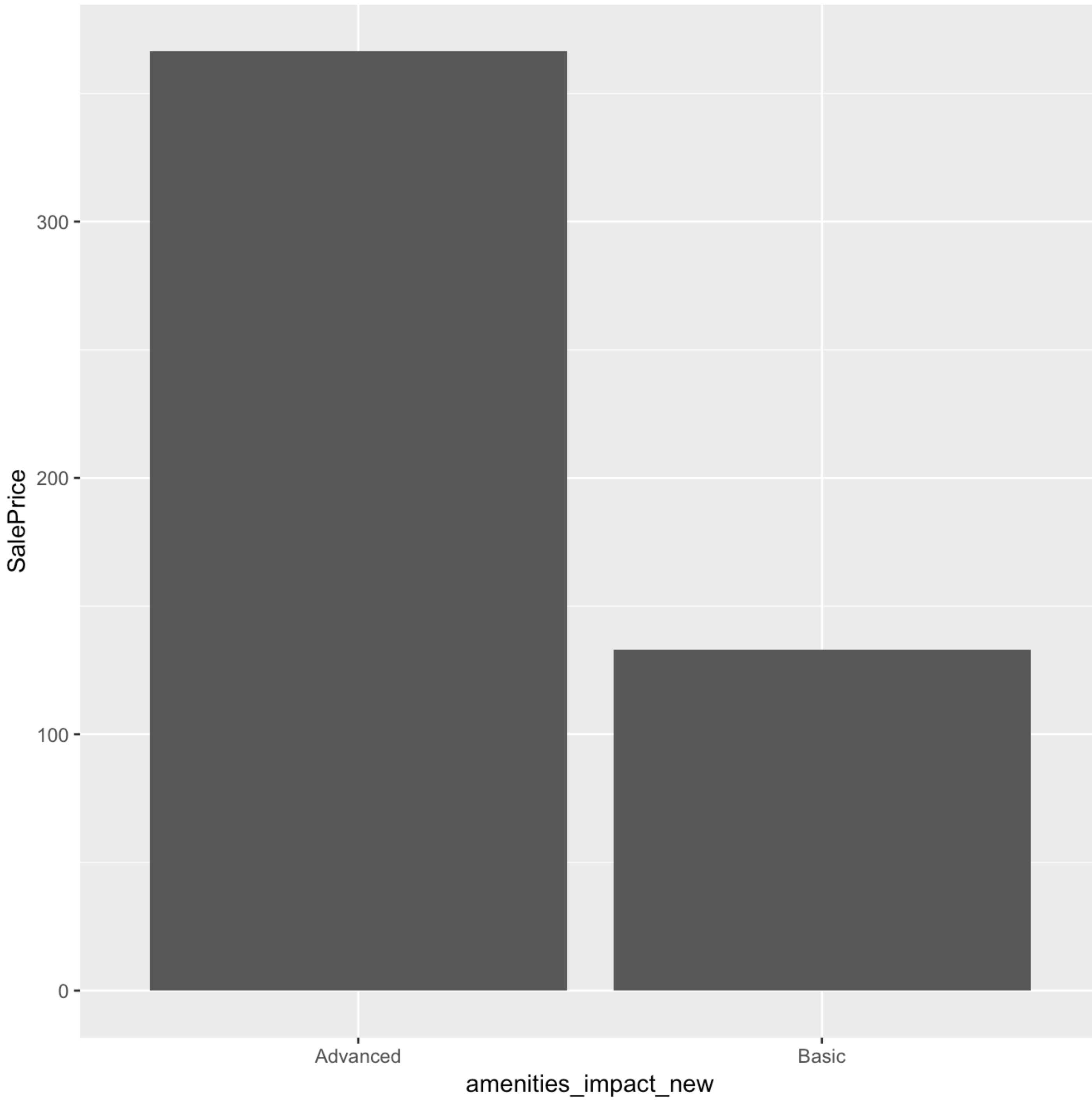

```
ggplot(train,aes(x=require_guest_profile_picture,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #F
```



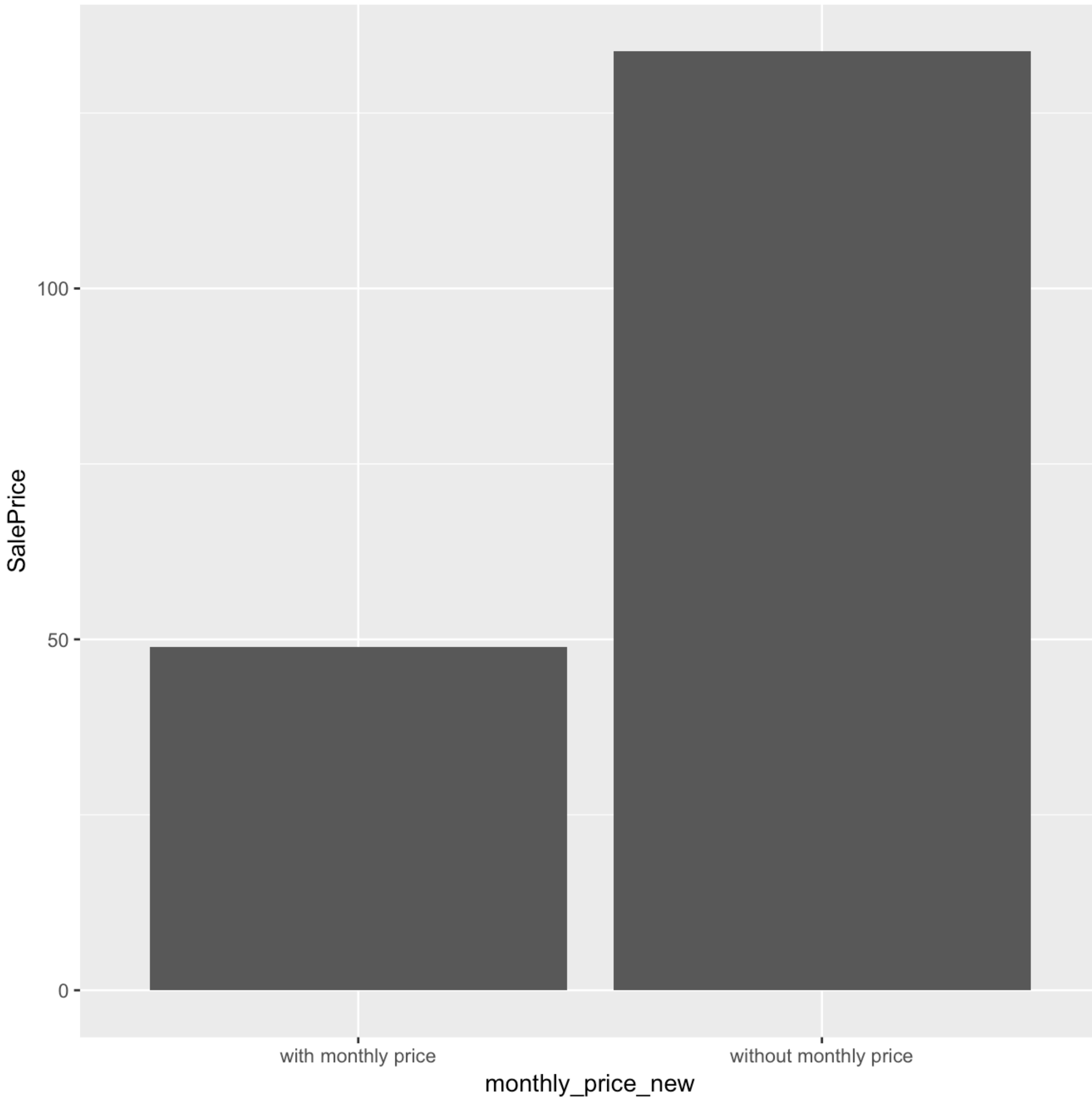

```
ggplot(train,aes(x=require_guest_phone_verification,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



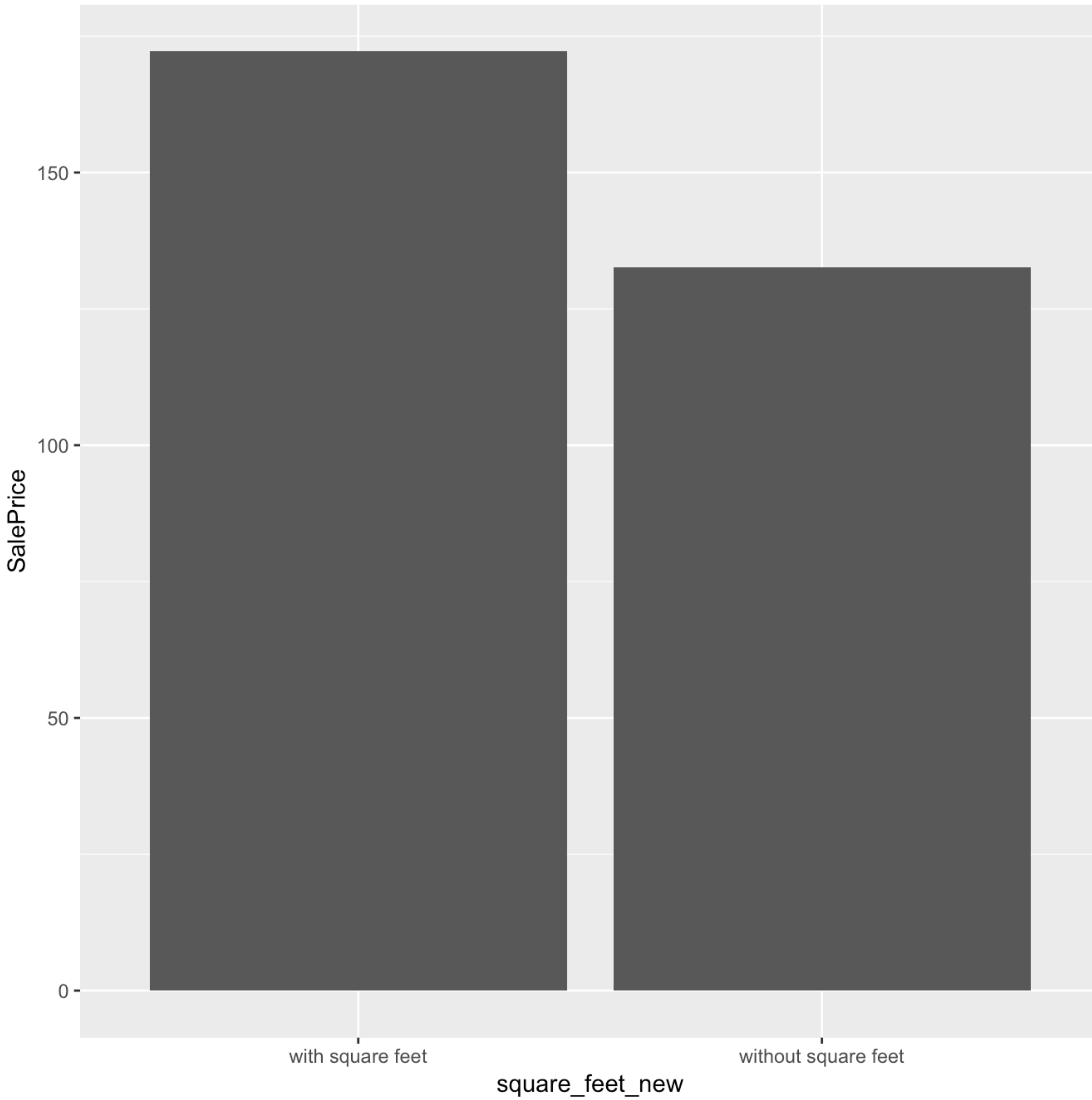

```
ggplot(train,aes(x=amenities_impact_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



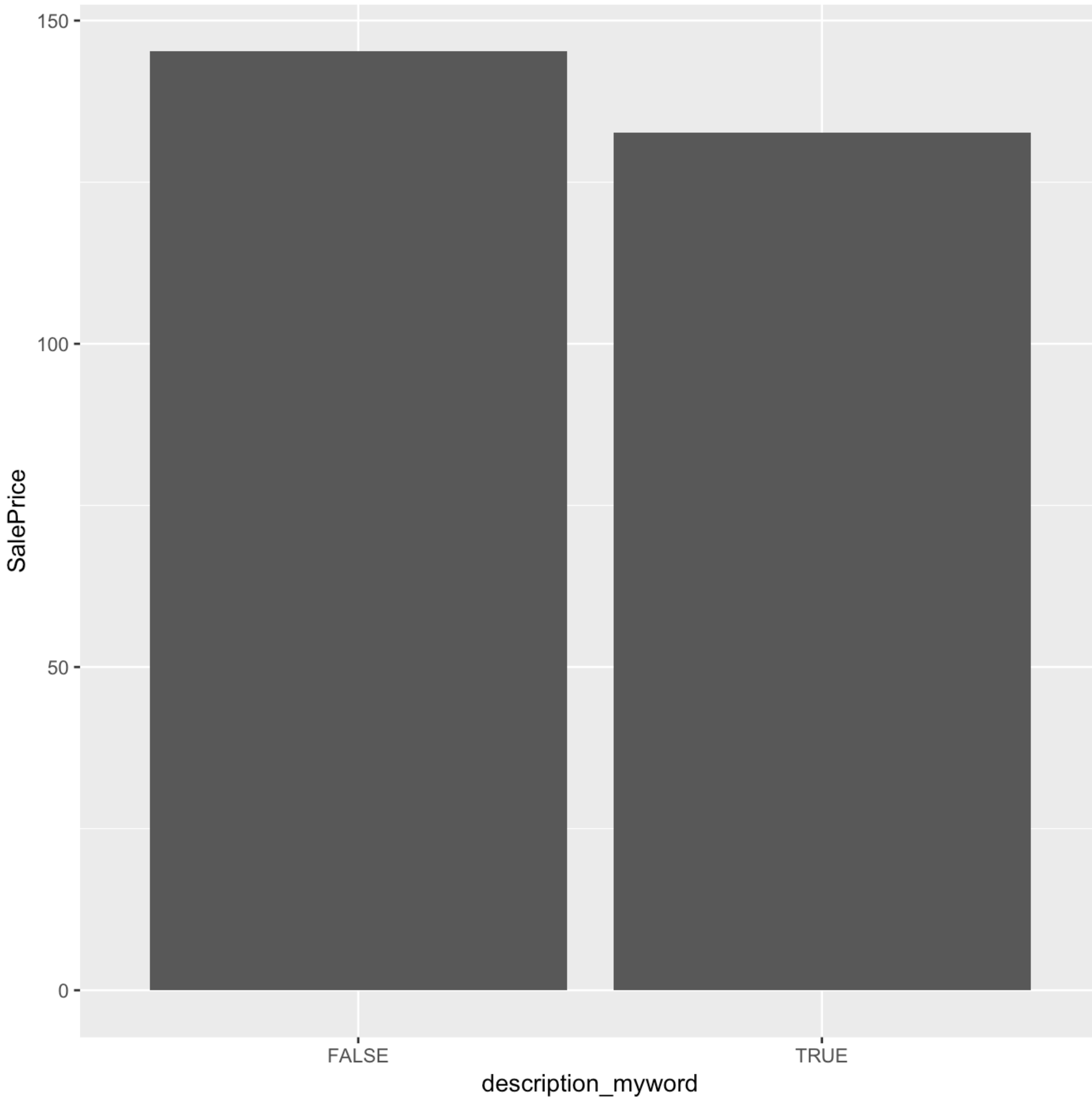

```
ggplot(train,aes(x=monthly_price_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



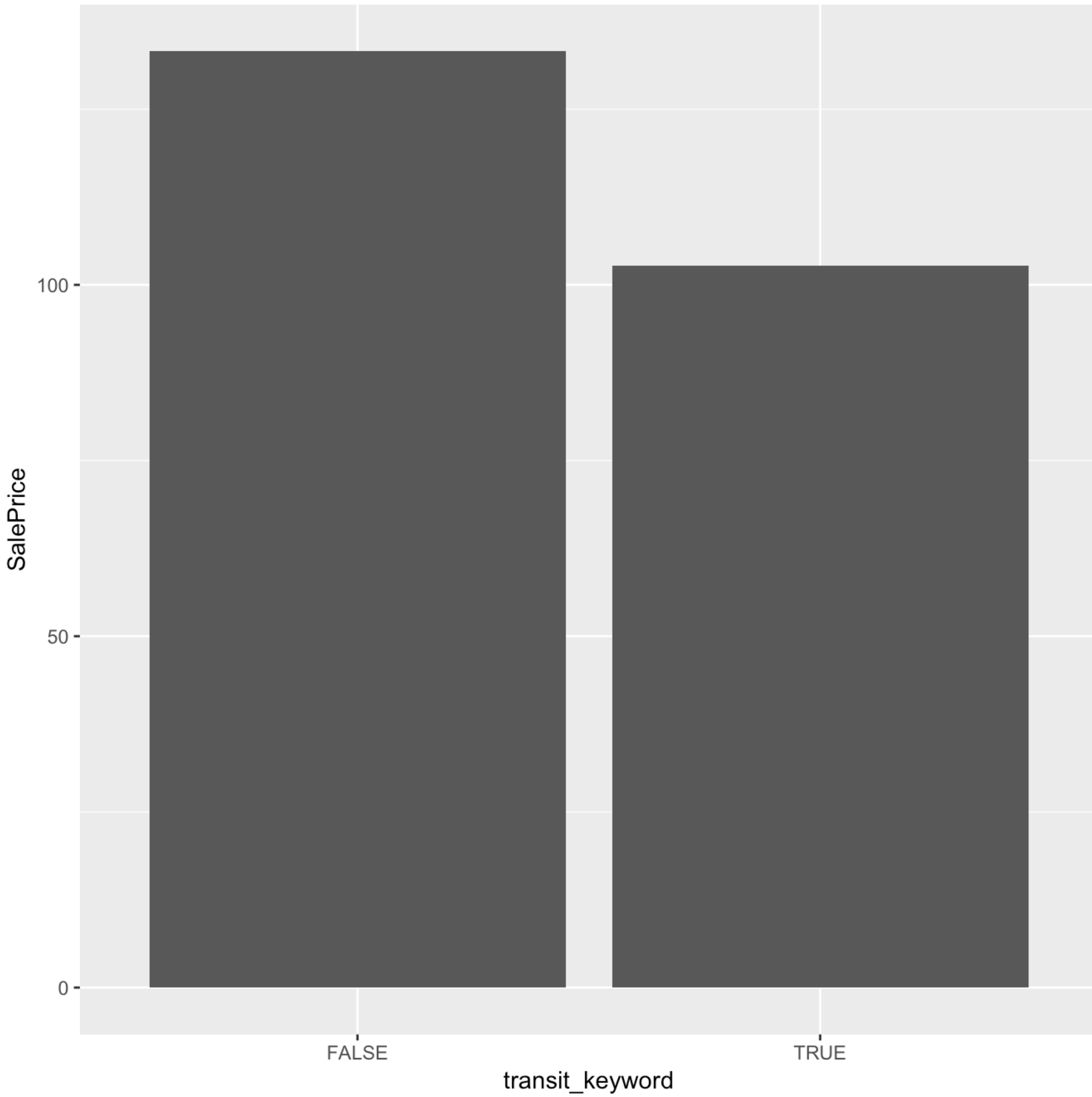

```
ggplot(train,aes(x=square_feet_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



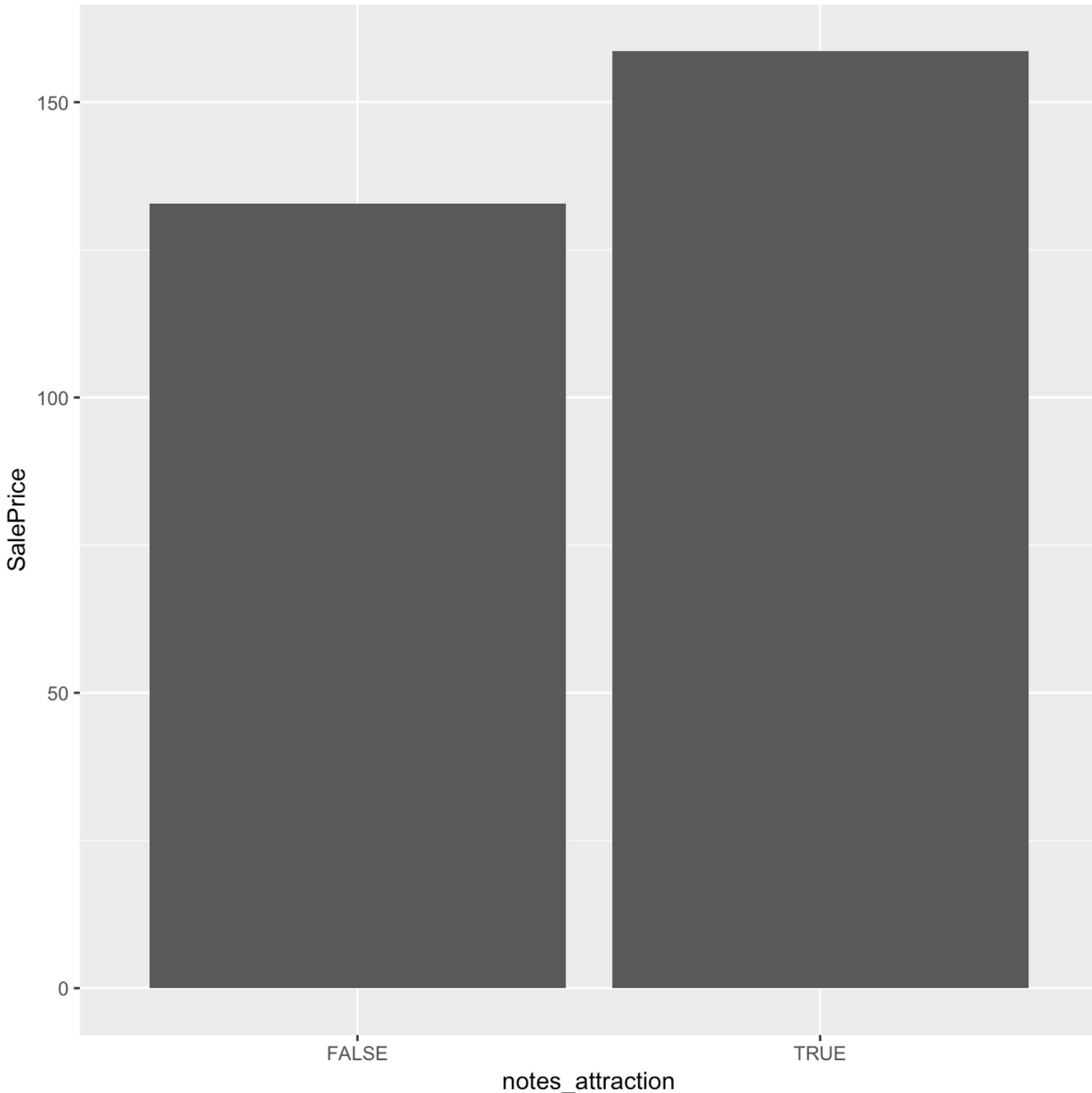

```
ggplot(train,aes(x=description_myword,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```




```
ggplot(train,aes(x=transit_keyword,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```




```
ggplot(train,aes(x=notes_attraction,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```




```
# Remove some variables without contribution to price prediction
train$require_guest_profile_picture <- NULL
```

3. Remove numeric variables with low correration with SalePrice

```r
# Check the correlation of numeric variables with price
for (col in colnames(train)){
  if(is.numeric(train[,col])){
    if( abs(cor(train[,col],train$SalePrice)) < 0.1){
      print(col)
      print( cor(train[,col],train$SalePrice) )
    }
  }
}
```

```
## [1] "host_listings_count"
## [1] 0.01998228
## [1] "host_total_listings_count"
## [1] 0.01998228
## [1] "latitude"
## [1] 0.05461239
## [1] "minimum_nights"
## [1] -0.002865612
## [1] "maximum_nights"
## [1] -0.001012488
## [1] "availability_365"
## [1] 0.0747253
## [1] "number_of_reviews"
## [1] 0.003559374
## [1] "review_scores_rating"
## [1] 0.05107005
## [1] "calculated_host_listings_count"
## [1] -0.09682842
## [1] "reviews_per_month"
## [1] -0.03222919
## [1] "calendar_updated_daysago"
## [1] -0.03656037
## [1] "selfintro_impact"
## [1] 0.01545654
```

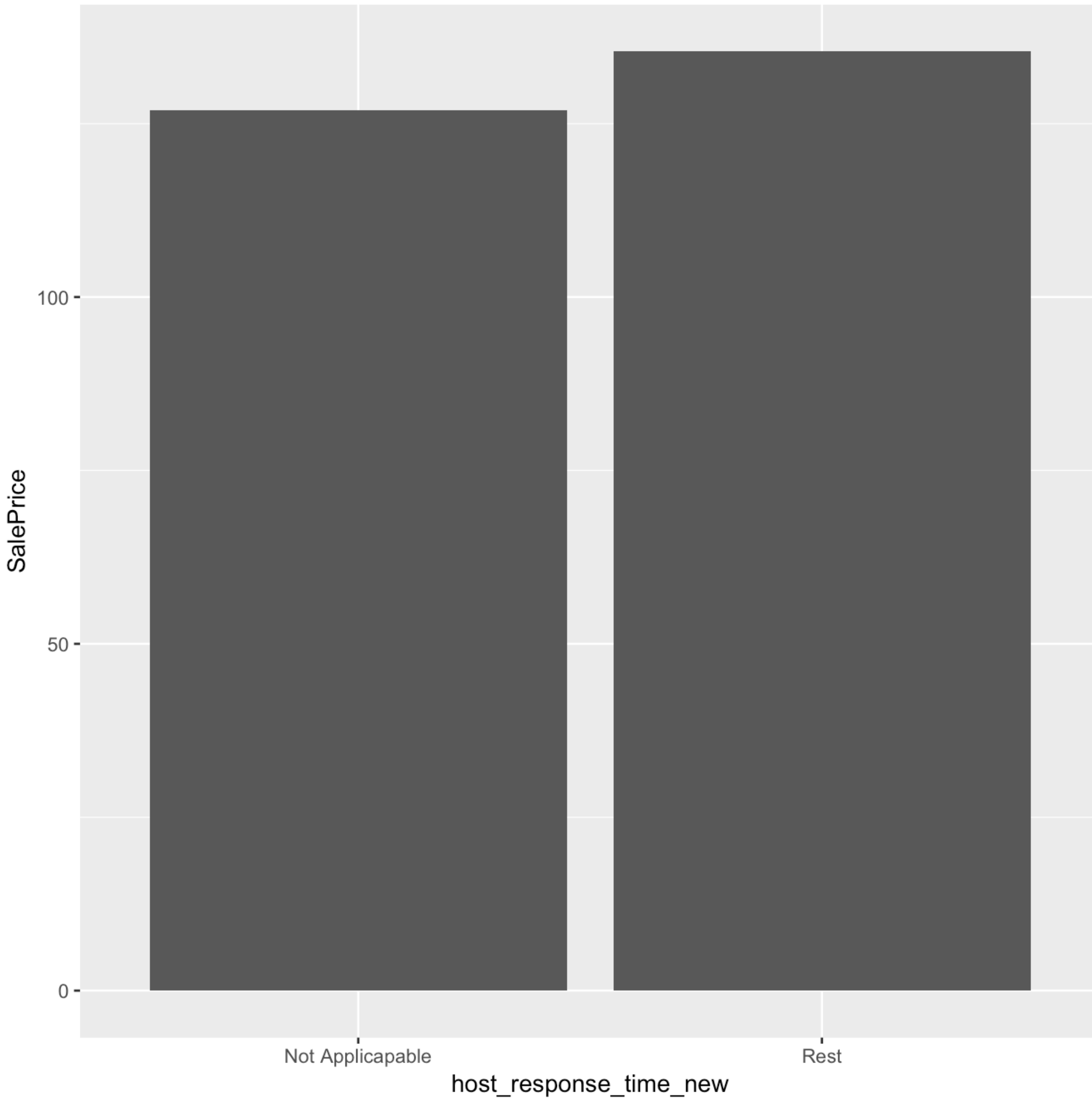
```
# Remove some numeric variables with low correration with SalePrice
train$host_listings_count <- NULL
train$host_total_listings_count <- NULL
train$latitude <- NULL
train$minimum_nights <- NULL
train$maximum_nights <- NULL
train$availability_365 <- NULL
train$number_of_reviews <- NULL
train$review_scores_rating <- NULL
train$calculated_host_listings_count <- NULL
train$reviews_per_month <- NULL
train$calendar_updated_daysago <- NULL
train$selfintro_impact <- NULL
```

4. Check which variables left, and visualize the relationship between each independent variable and dependent variable. Since we have already dealt with numeric variables, we should pay more attention to categorical variables this time.

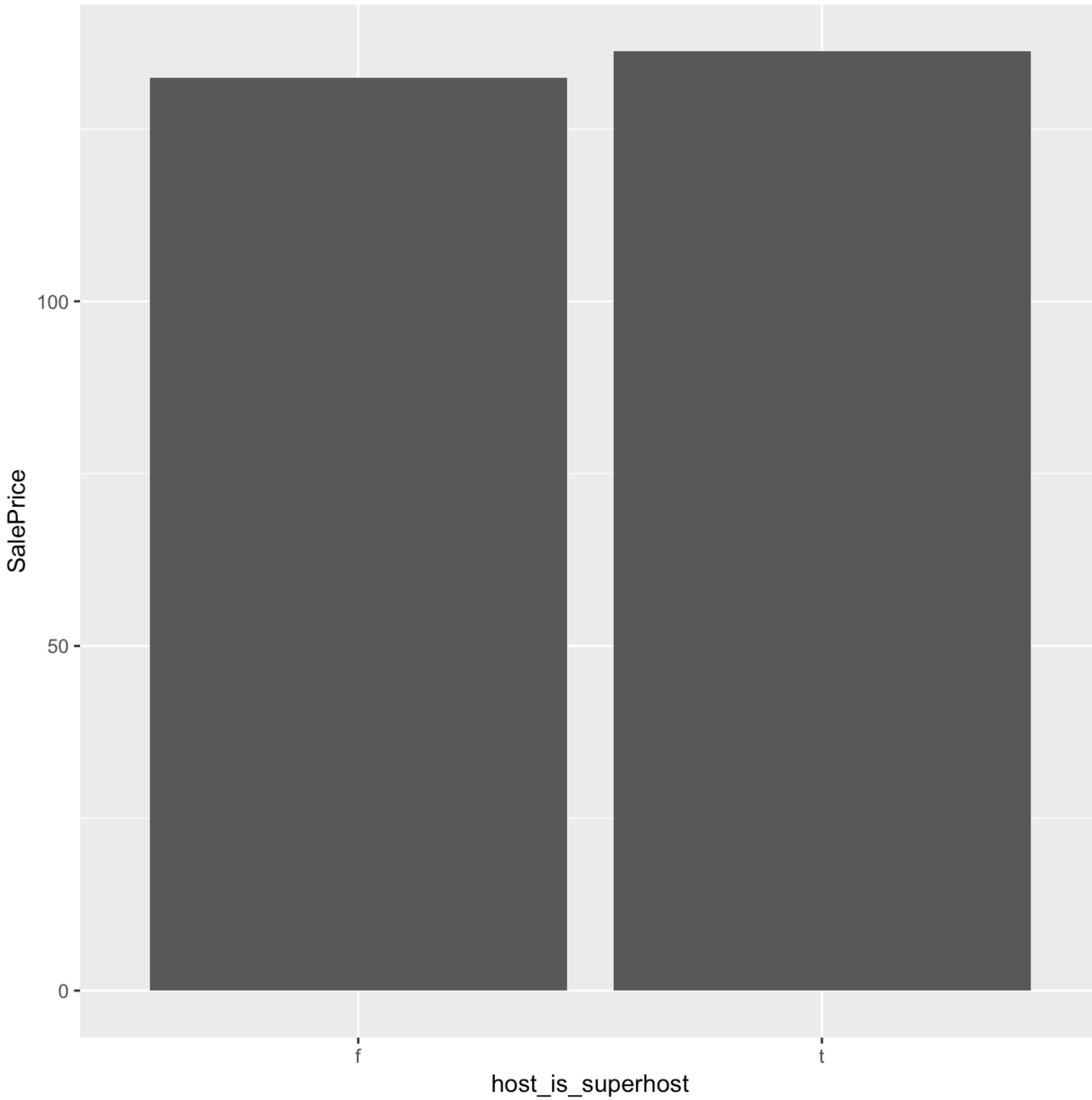
```
# Check which factor variables left
for (col in colnames(train)){
  if(is.factor(train[,col])){
    print(col)
  }
}
```

```
## [1] "host_is_superhost"
## [1] "host_has_profile_pic"
## [1] "host_identity_verified"
## [1] "neighbourhood_group_cleansed"
## [1] "room_type"
## [1] "bed_type"
## [1] "instant_bookable"
## [1] "is_business_travel_ready"
## [1] "cancellation_policy"
## [1] "require_guest_phone_verification"
## [1] "monthly_price_new"
## [1] "square_feet_new"
## [1] "weekly_price_new"
## [1] "security_deposit_new"
## [1] "name_impact"
## [1] "access_impact"
## [1] "summary_impact"
## [1] "space_impact"
## [1] "description_impact"
## [1] "neighborhood_overview_impact"
## [1] "interaction_impact"
## [1] "notes_impact"
## [1] "rules_impact"
## [1] "amenities_impact_new"
## [1] "property_type_new"
## [1] "host_response_rate_new"
## [1] "host_response_time_new"
```

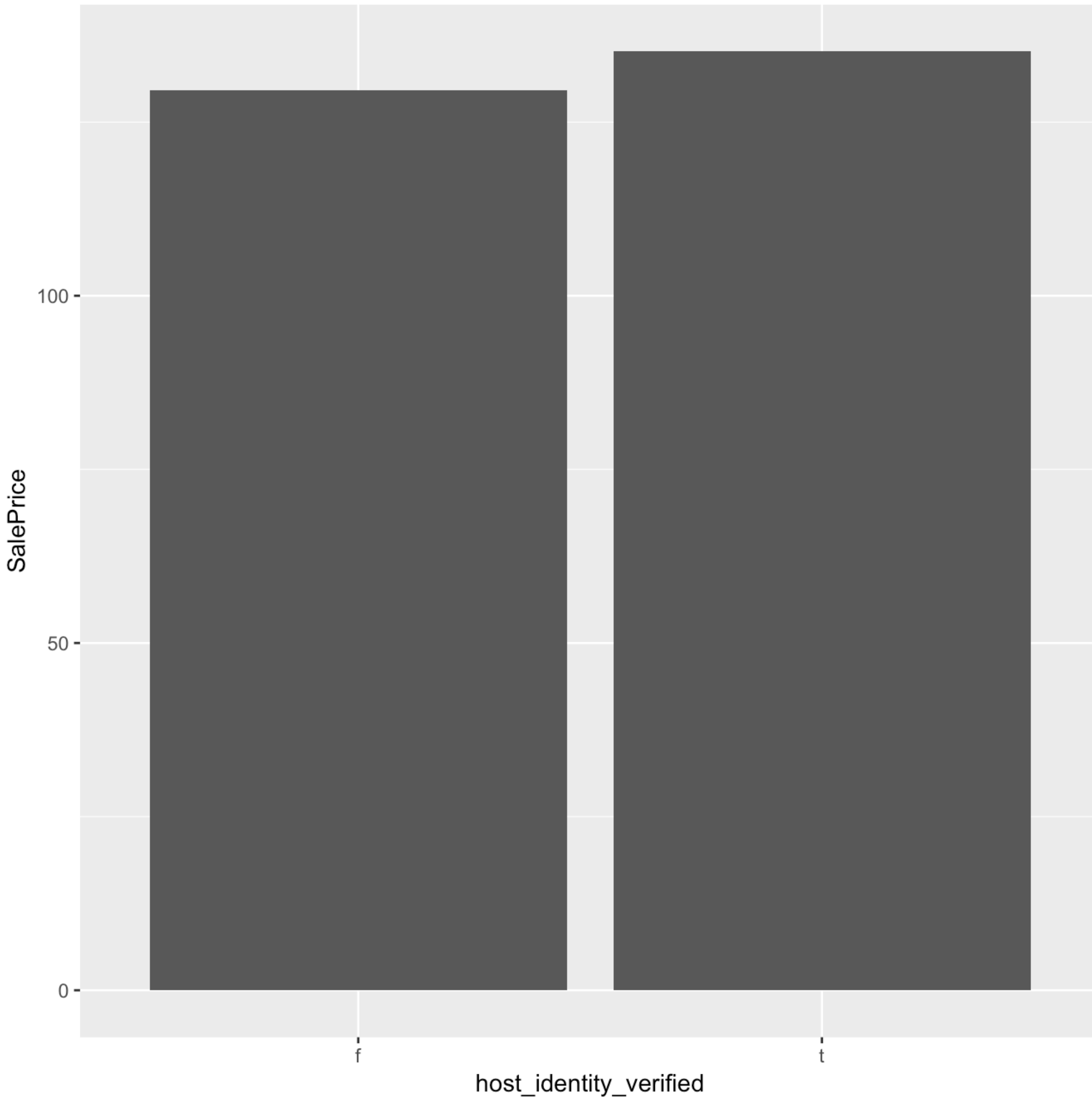
```
# Visualized bar chart
ggplot(train,aes(x=host_response_time_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



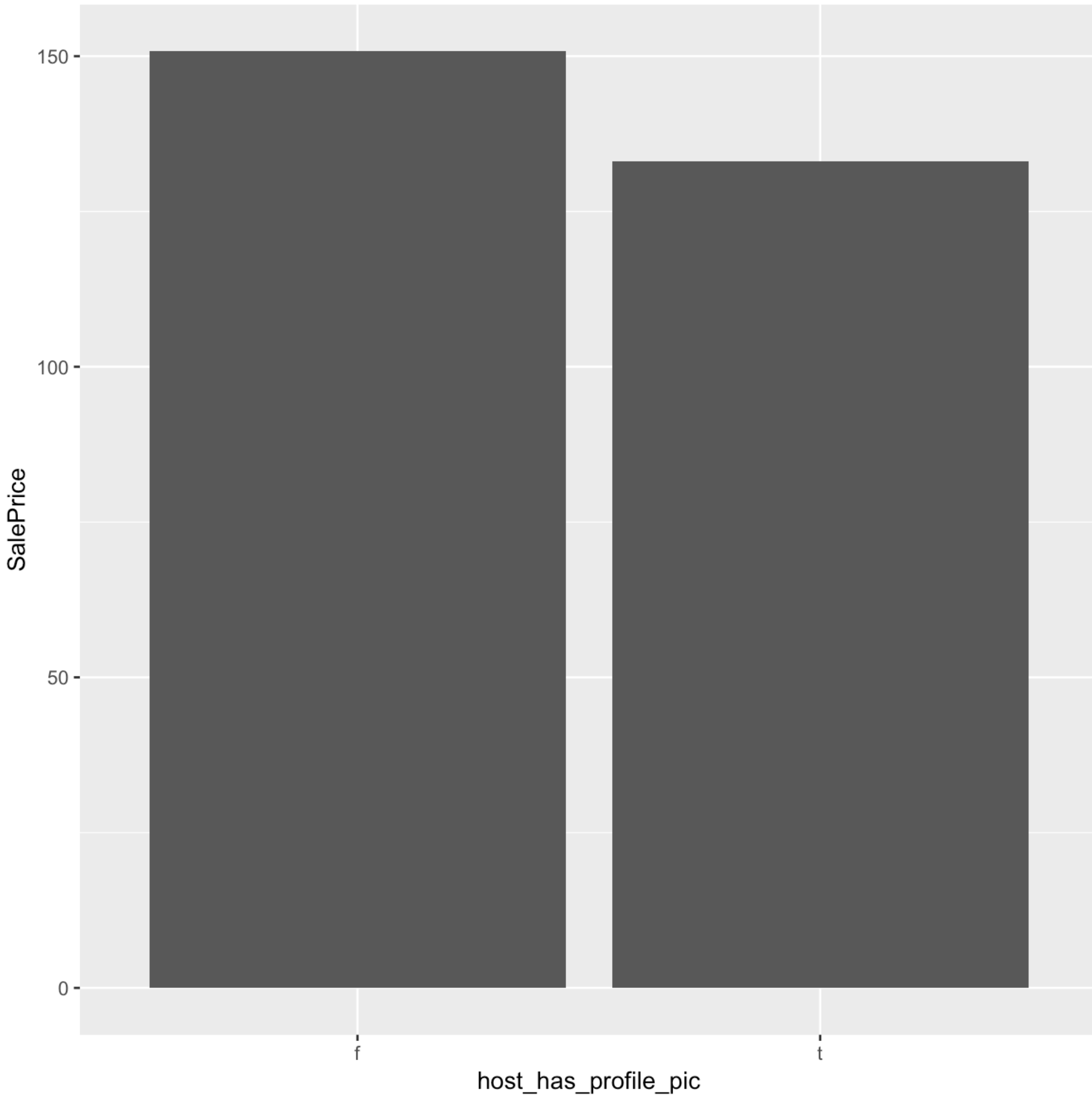

```
ggplot(train,aes(x=host_is_superhost,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



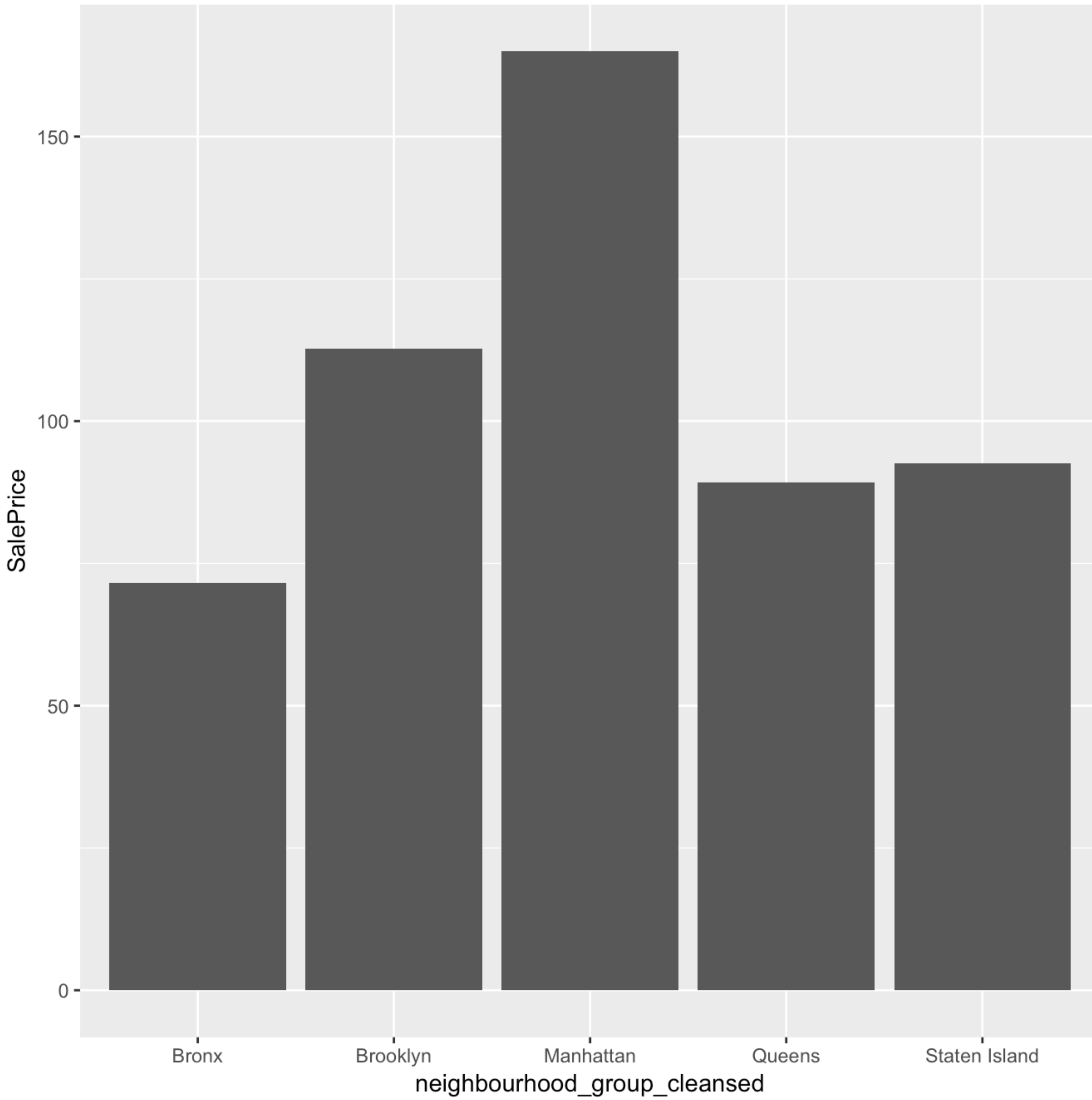

```
ggplot(train,aes(x=host_identity_verified,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



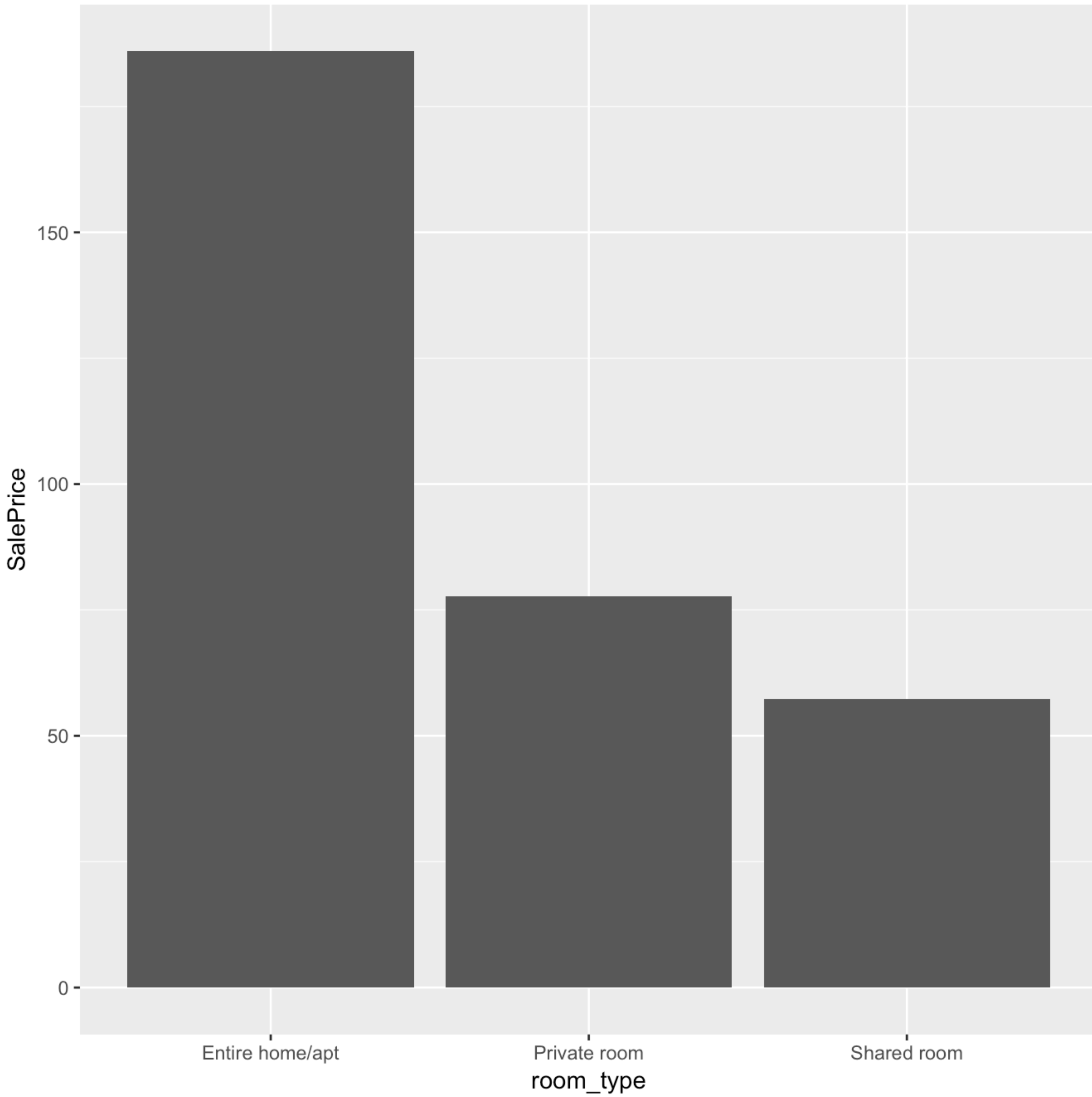

```
ggplot(train,aes(x=host_has_profile_pic,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



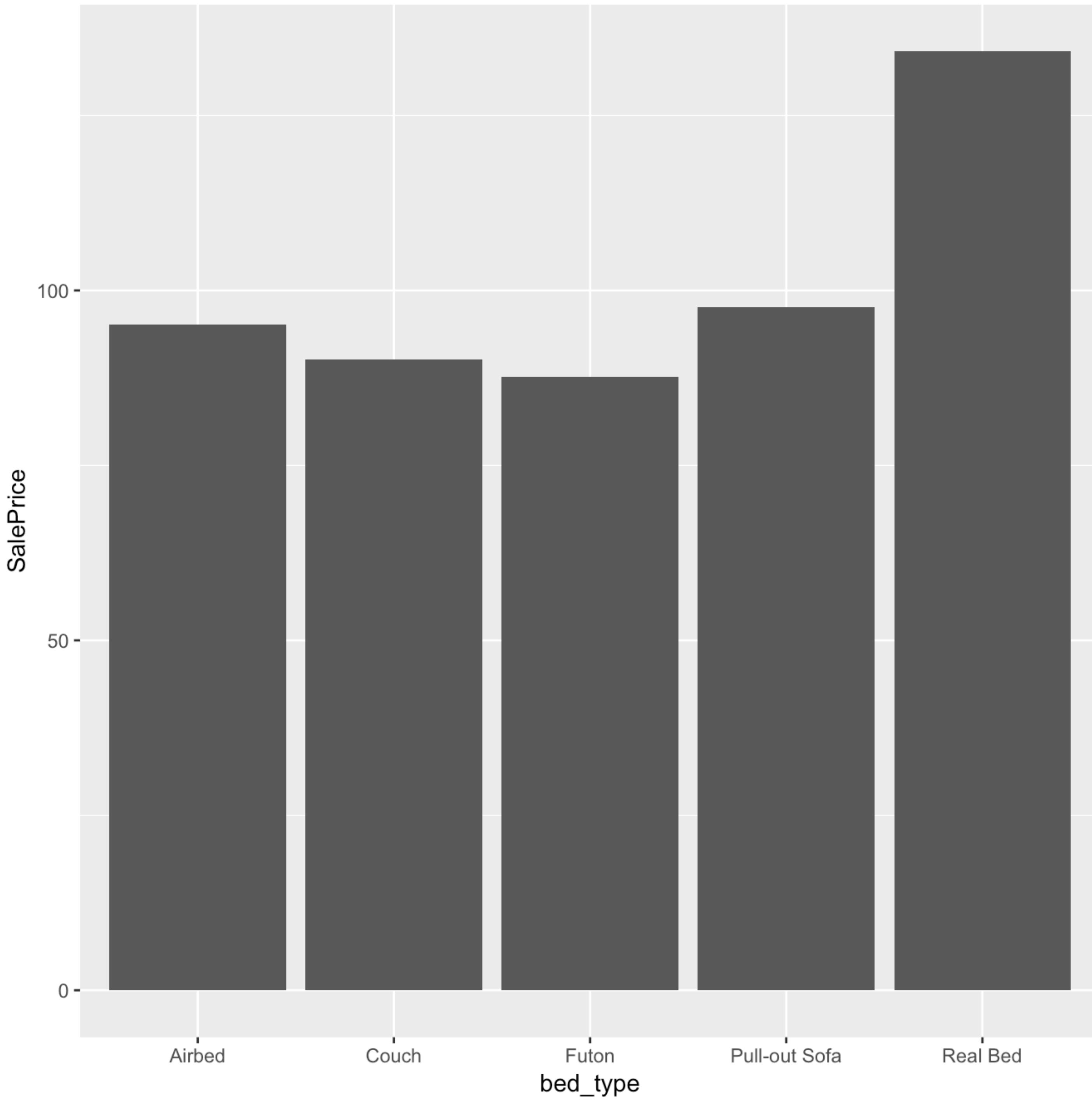

```
ggplot(train,aes(x=neighbourhood_group_cleansed,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



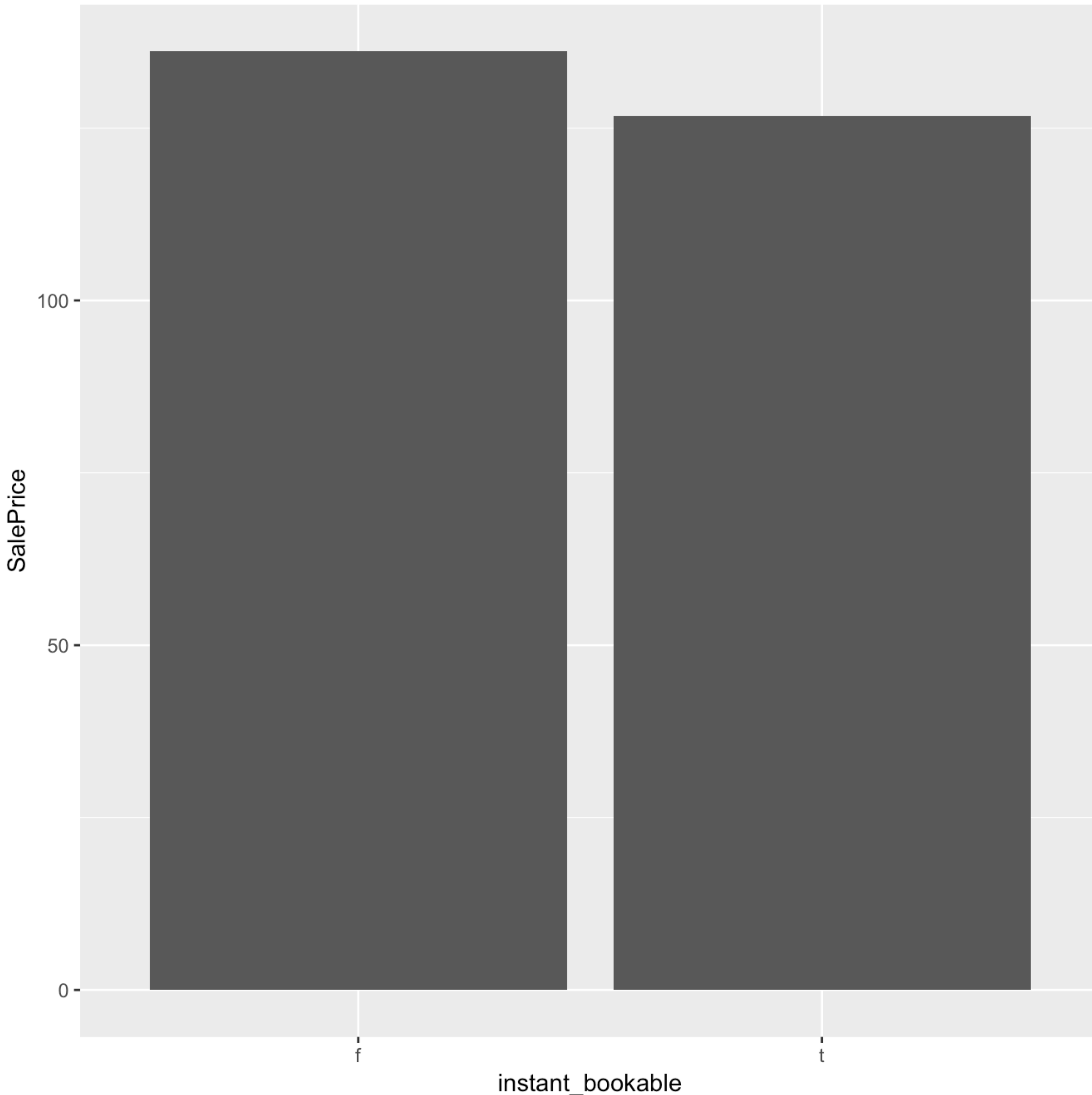

```
ggplot(train,aes(x=room_type,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



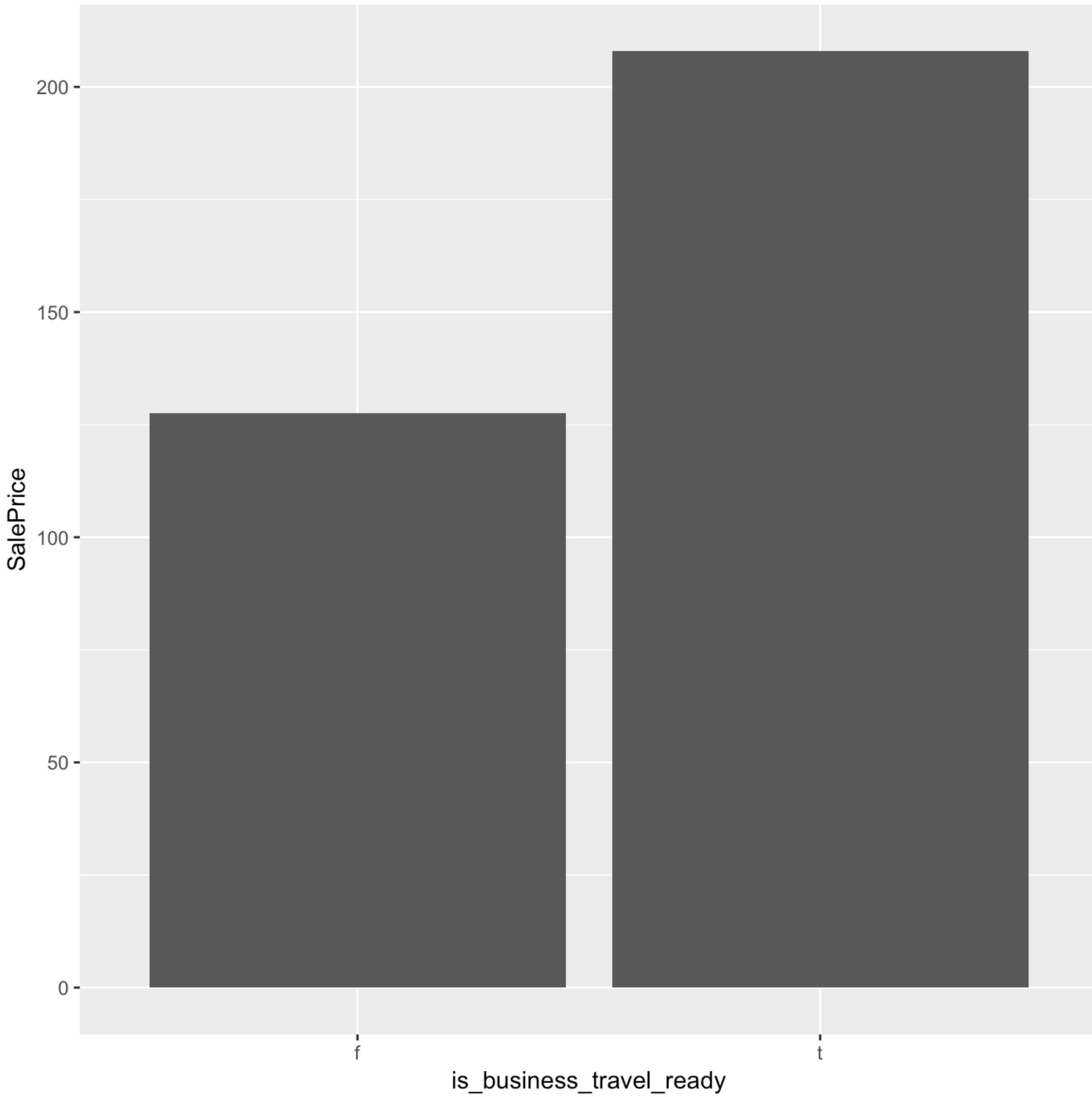

```
ggplot(train,aes(x=bed_type,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



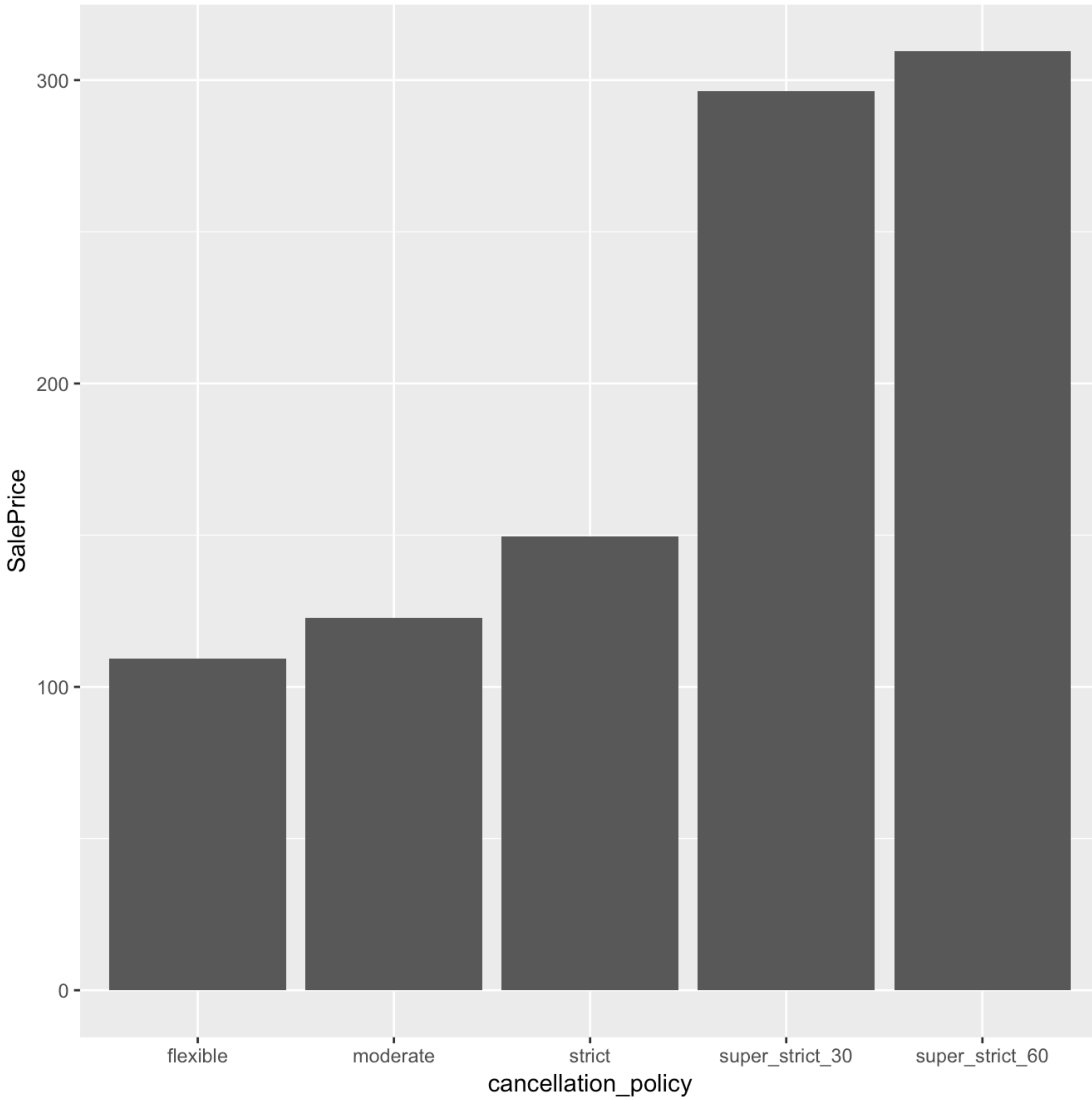

```
ggplot(train,aes(x=instant_bookable,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



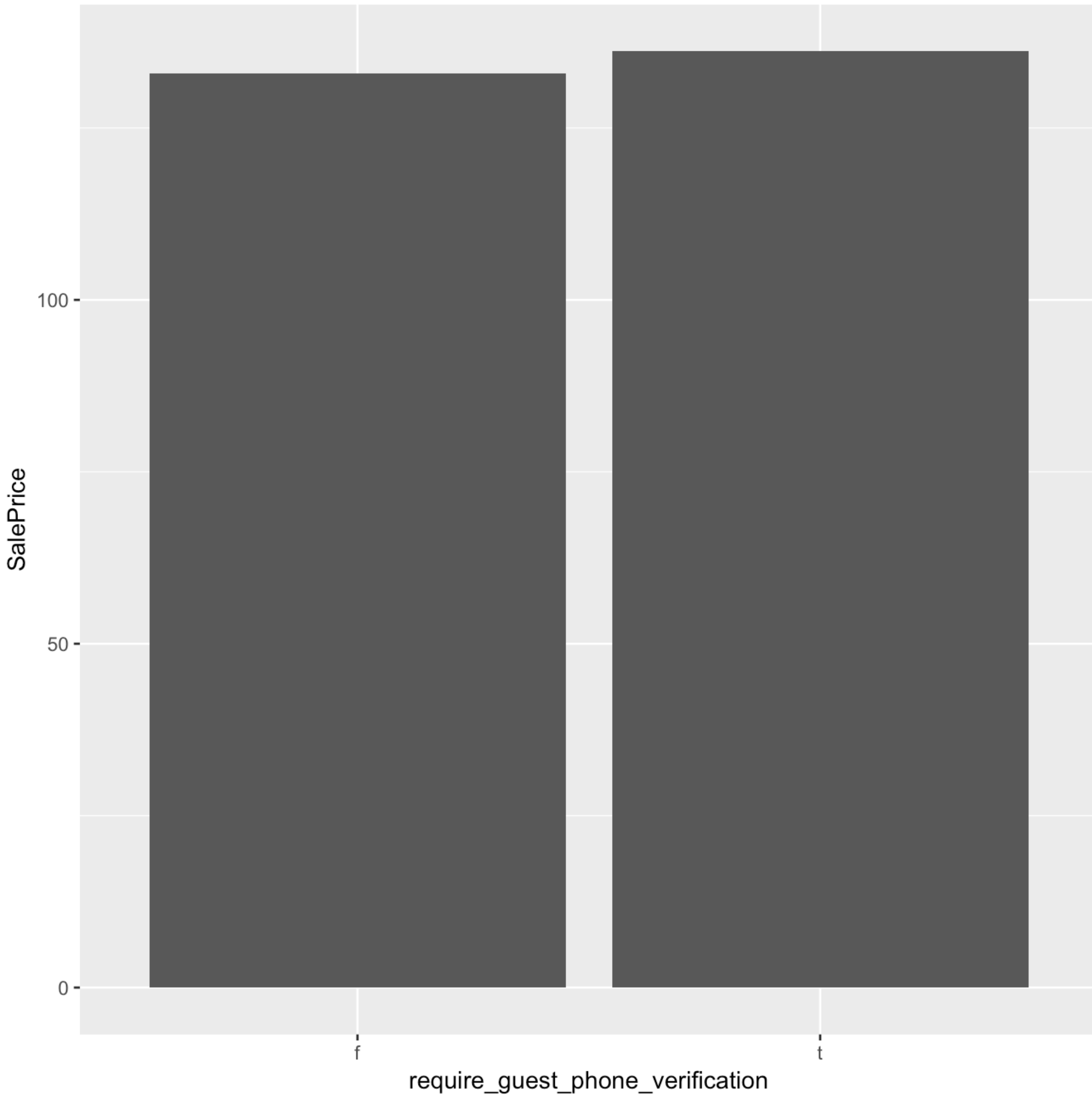

```
ggplot(train,aes(x=is_business_travel_ready,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



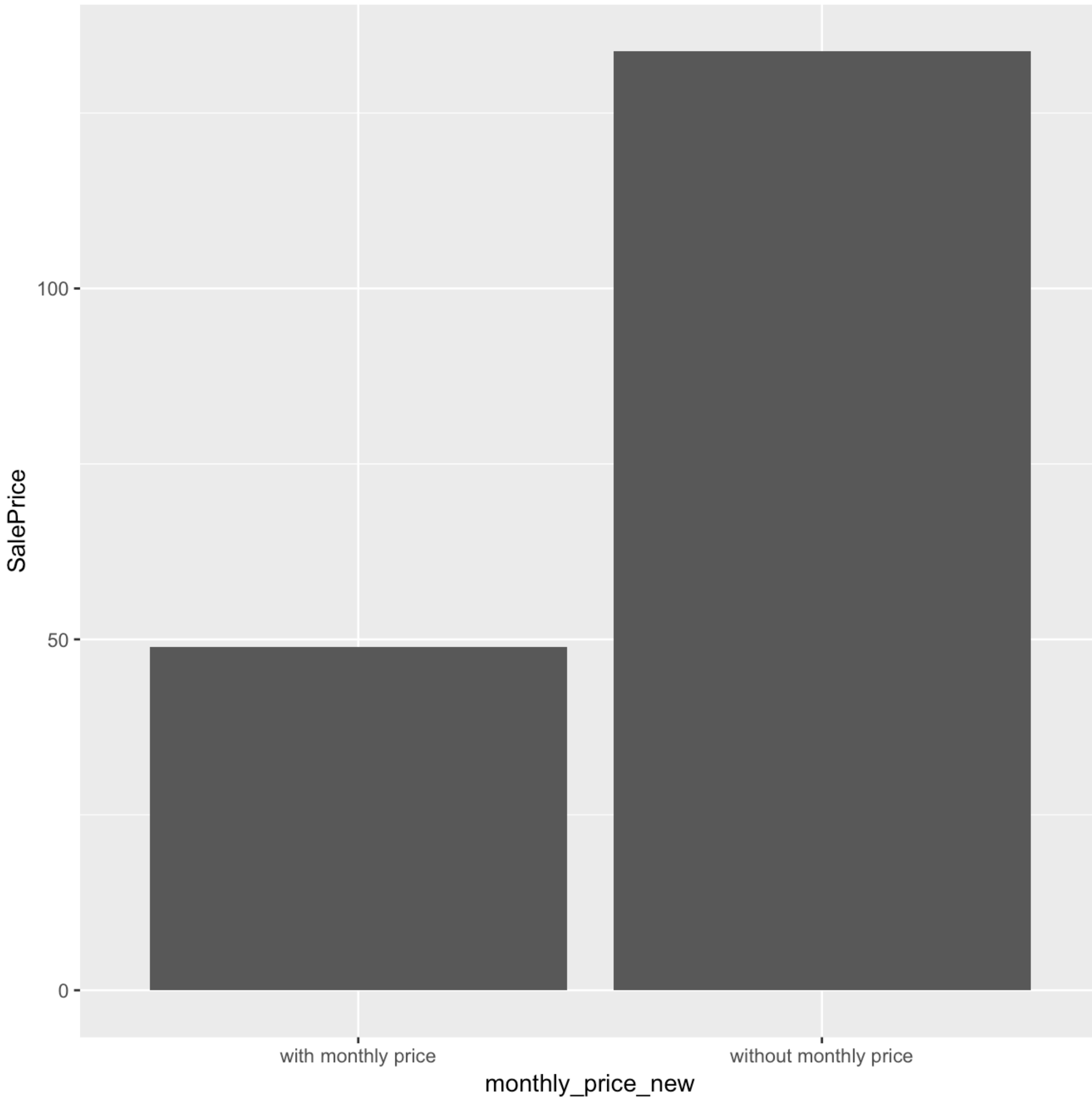

```
ggplot(train,aes(x=cancellation_policy,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



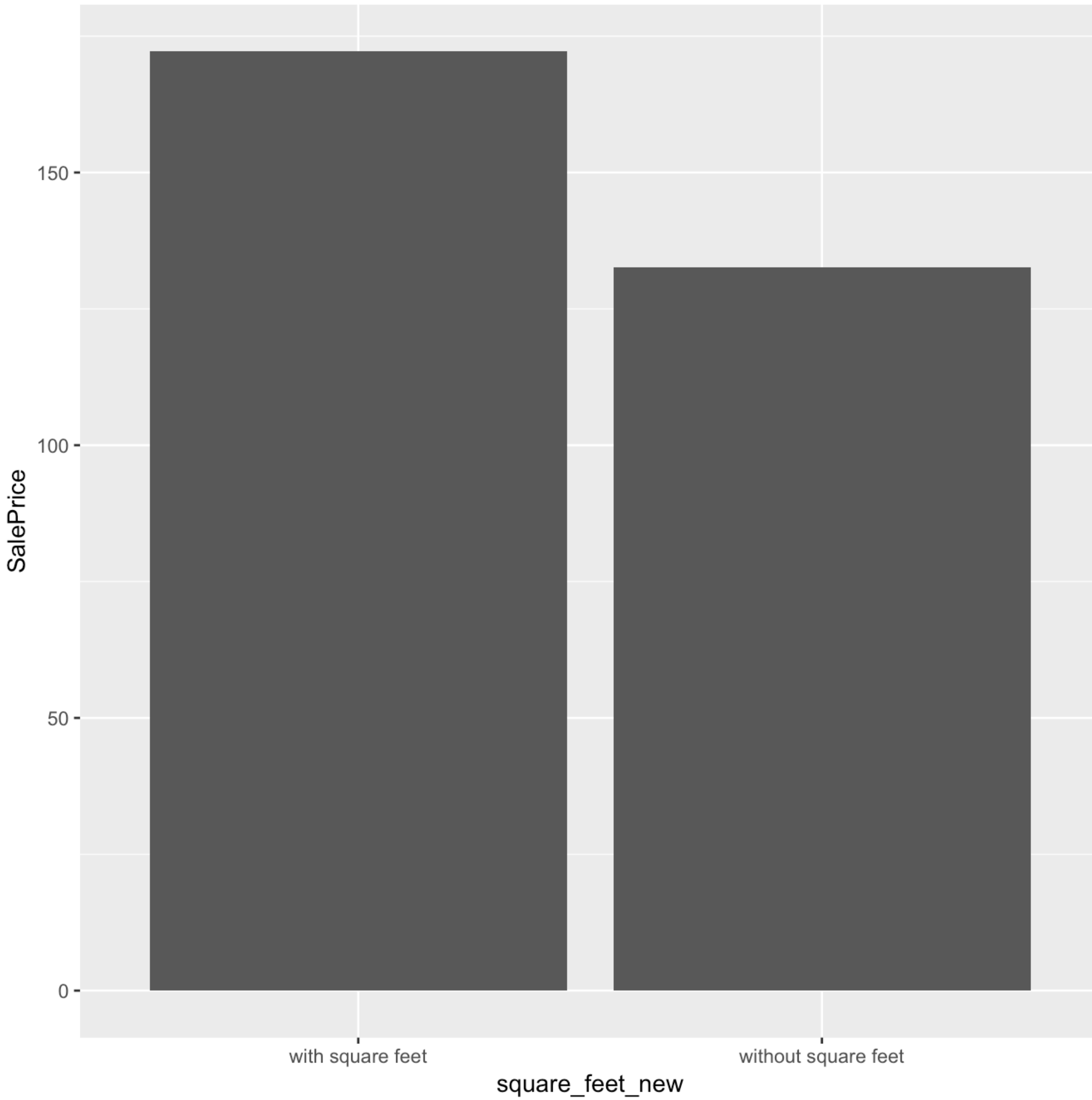

```
ggplot(train,aes(x=require_guest_phone_verification,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



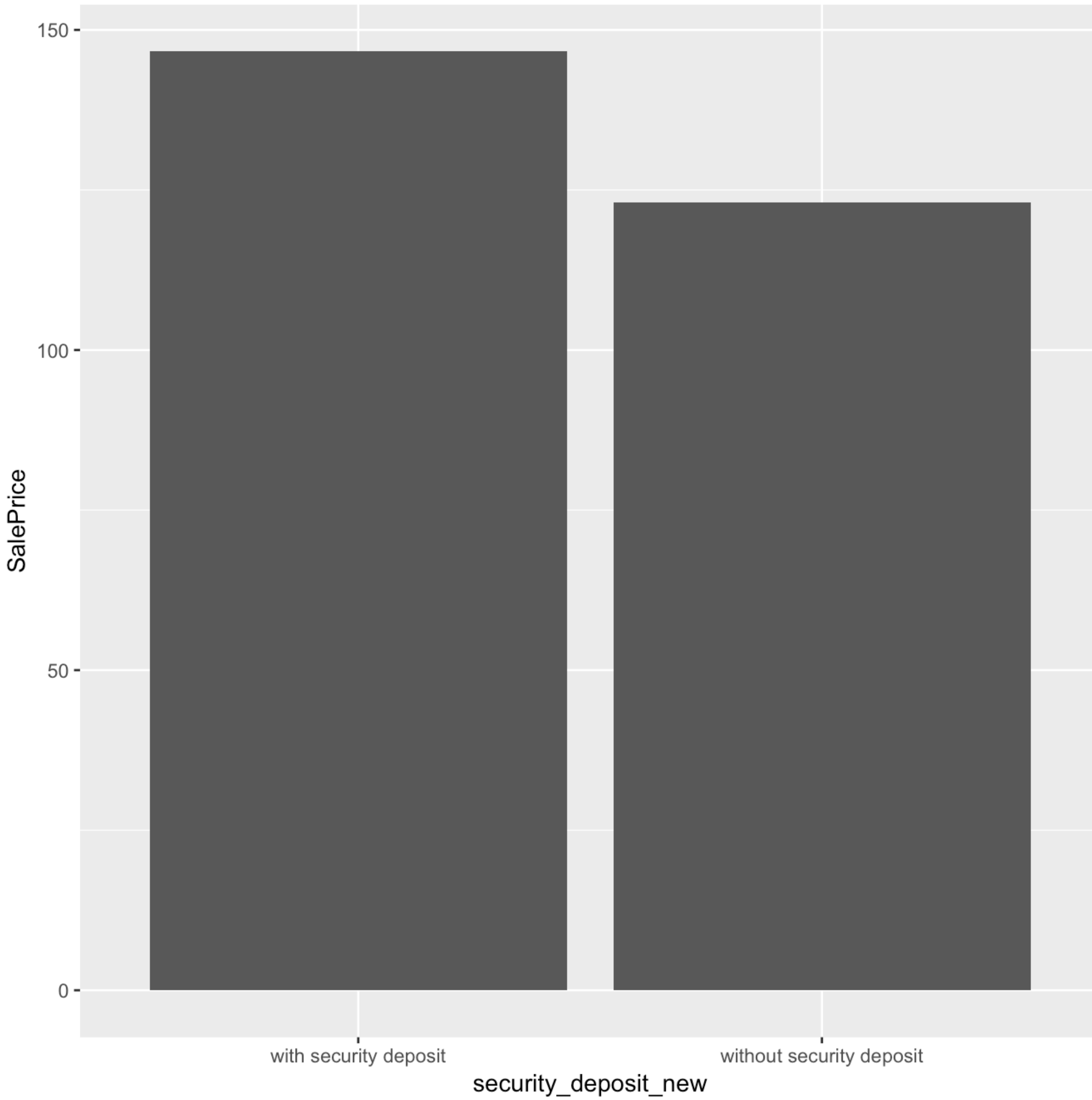

```
ggplot(train,aes(x=monthly_price_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



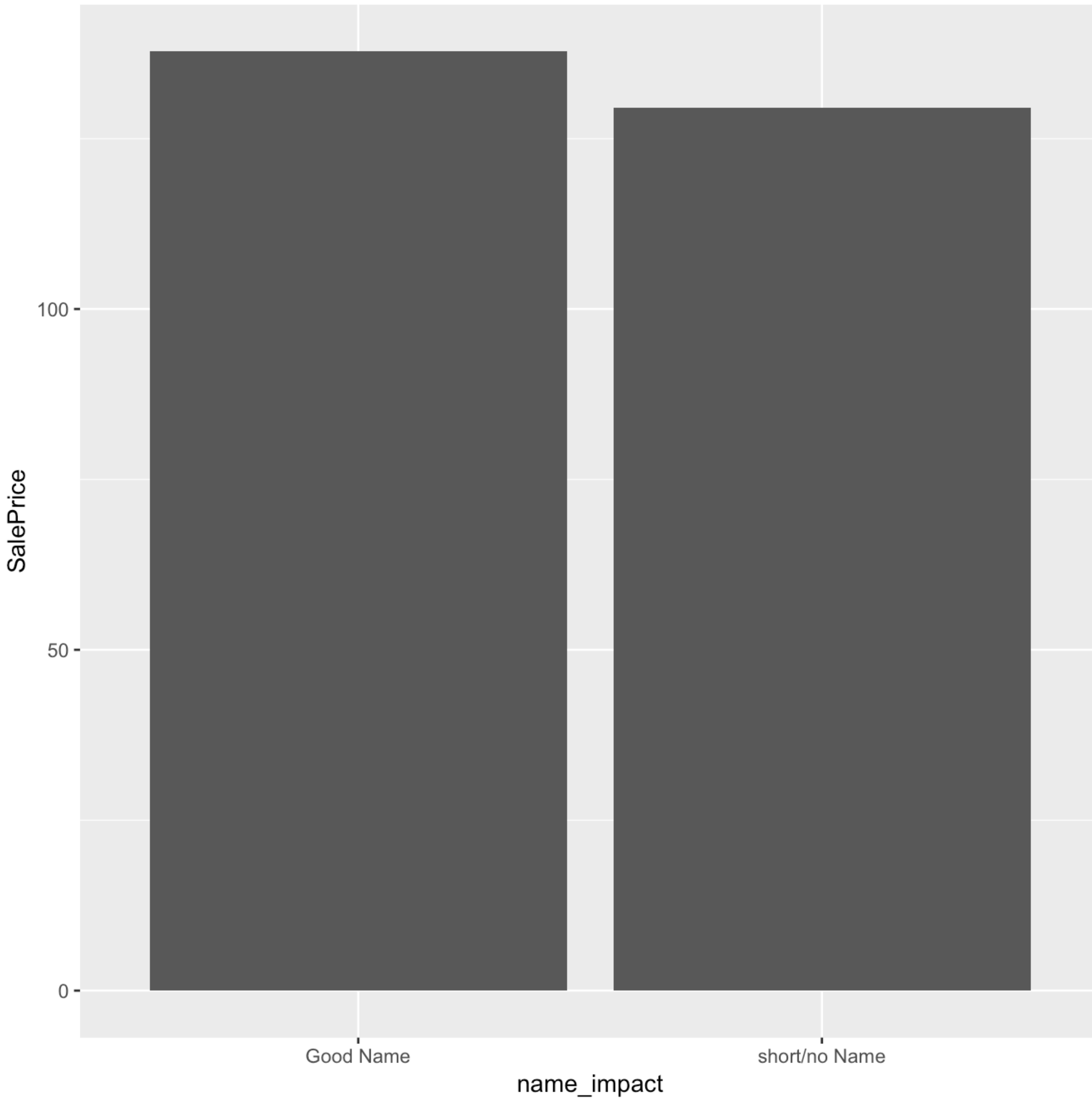

```
ggplot(train,aes(x=square_feet_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



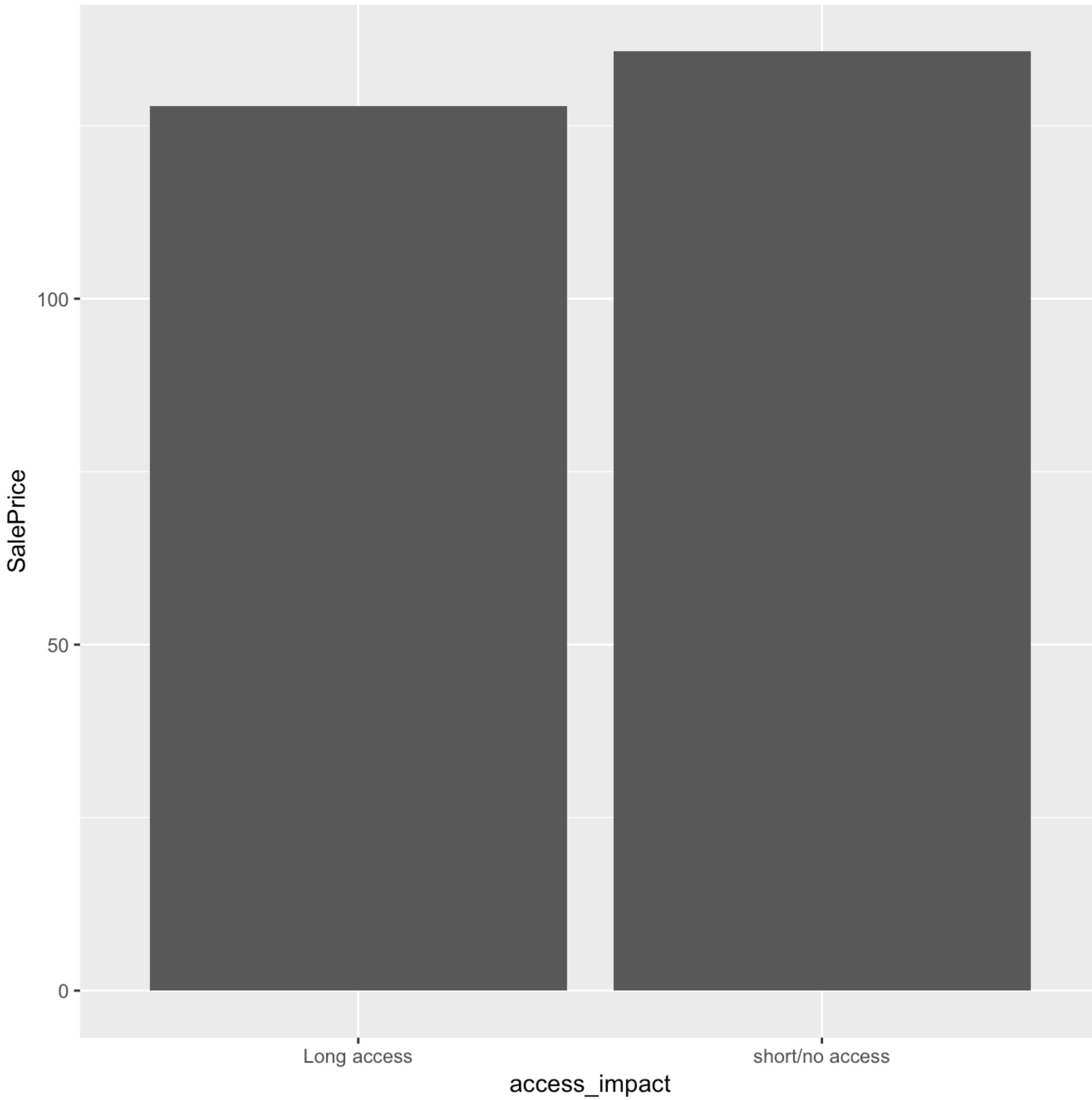

```
ggplot(train,aes(x=security_deposit_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



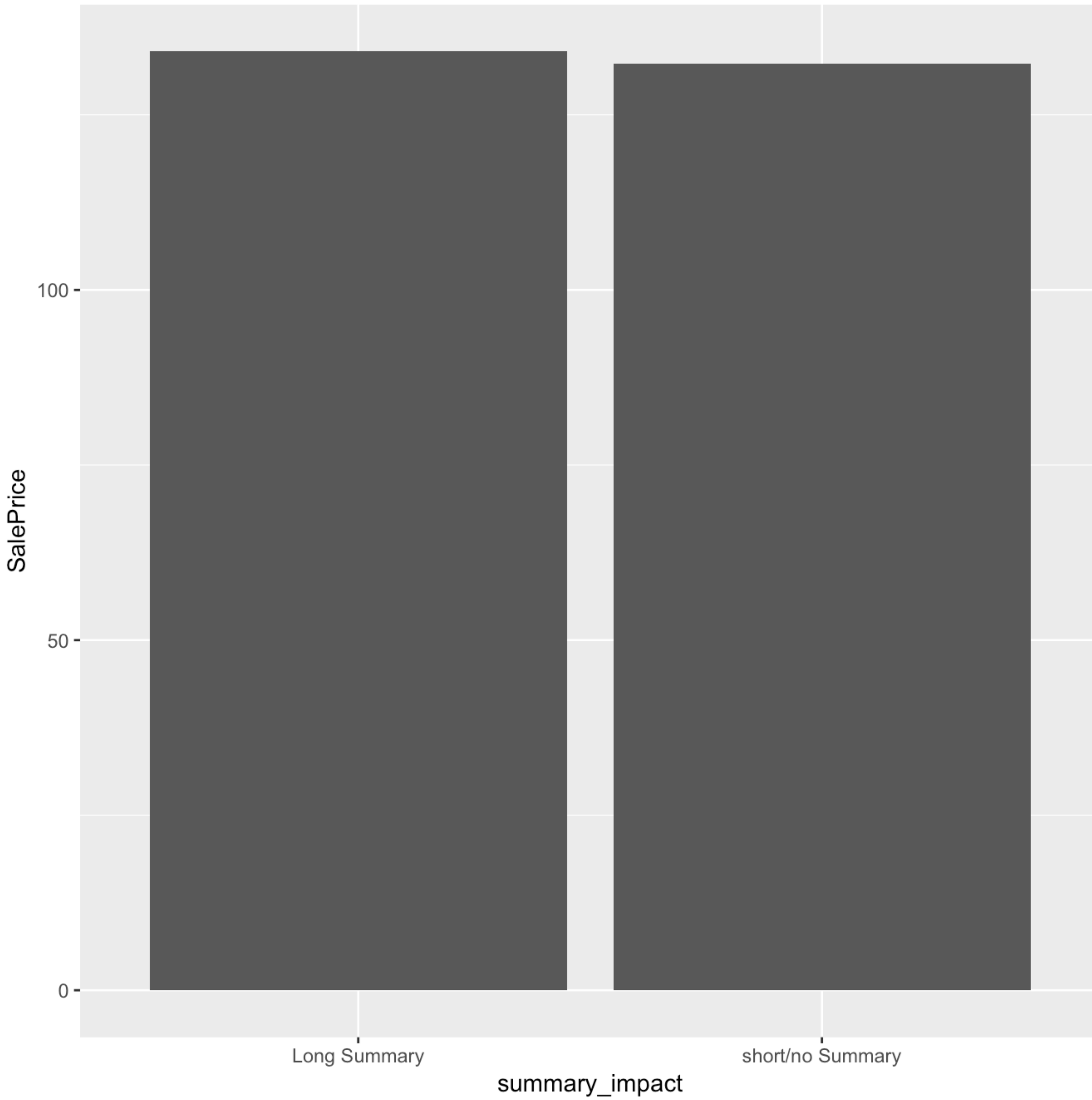

```
ggplot(train,aes(x=name_impact,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



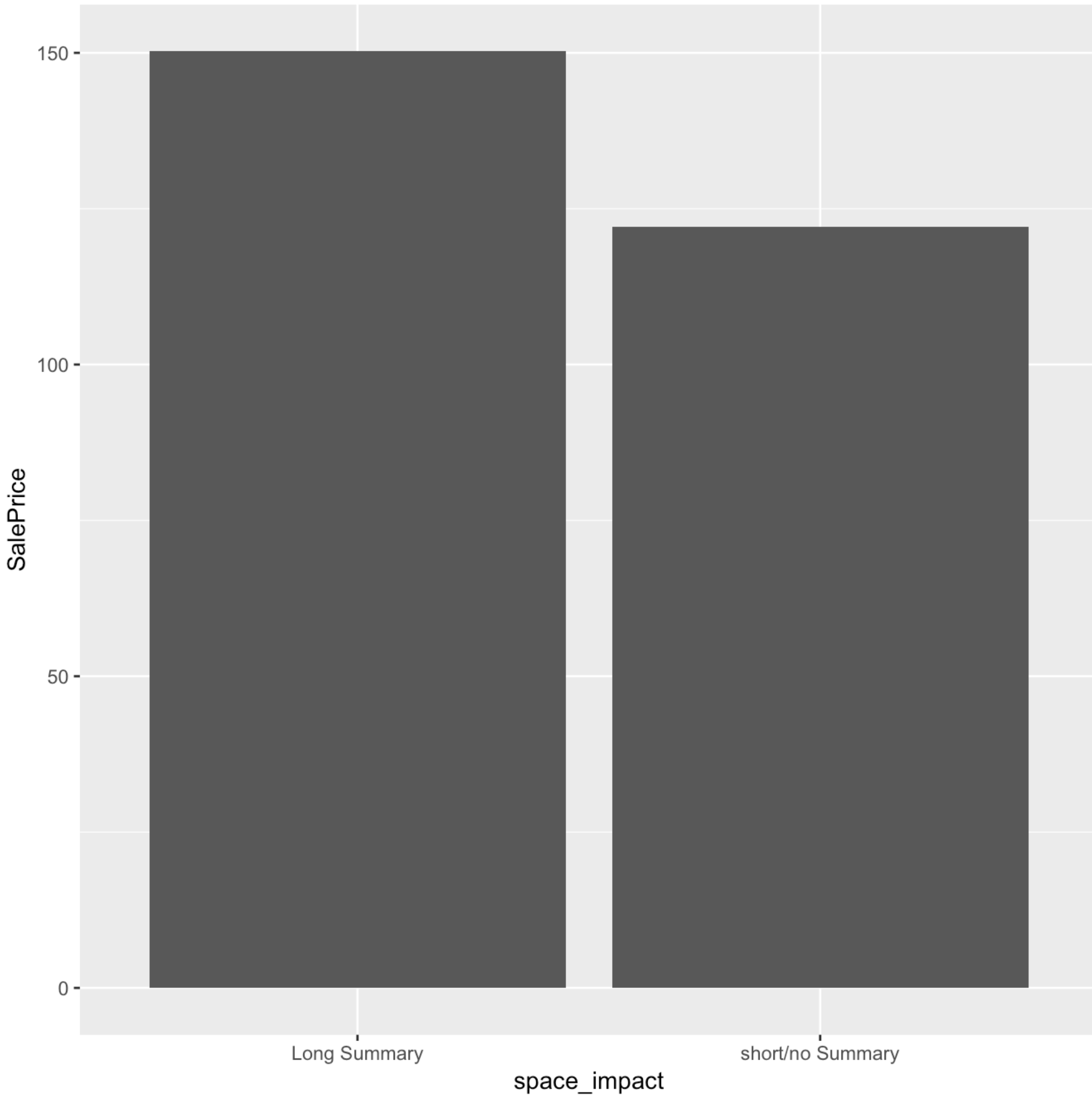

```
ggplot(train,aes(x=access_impact,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



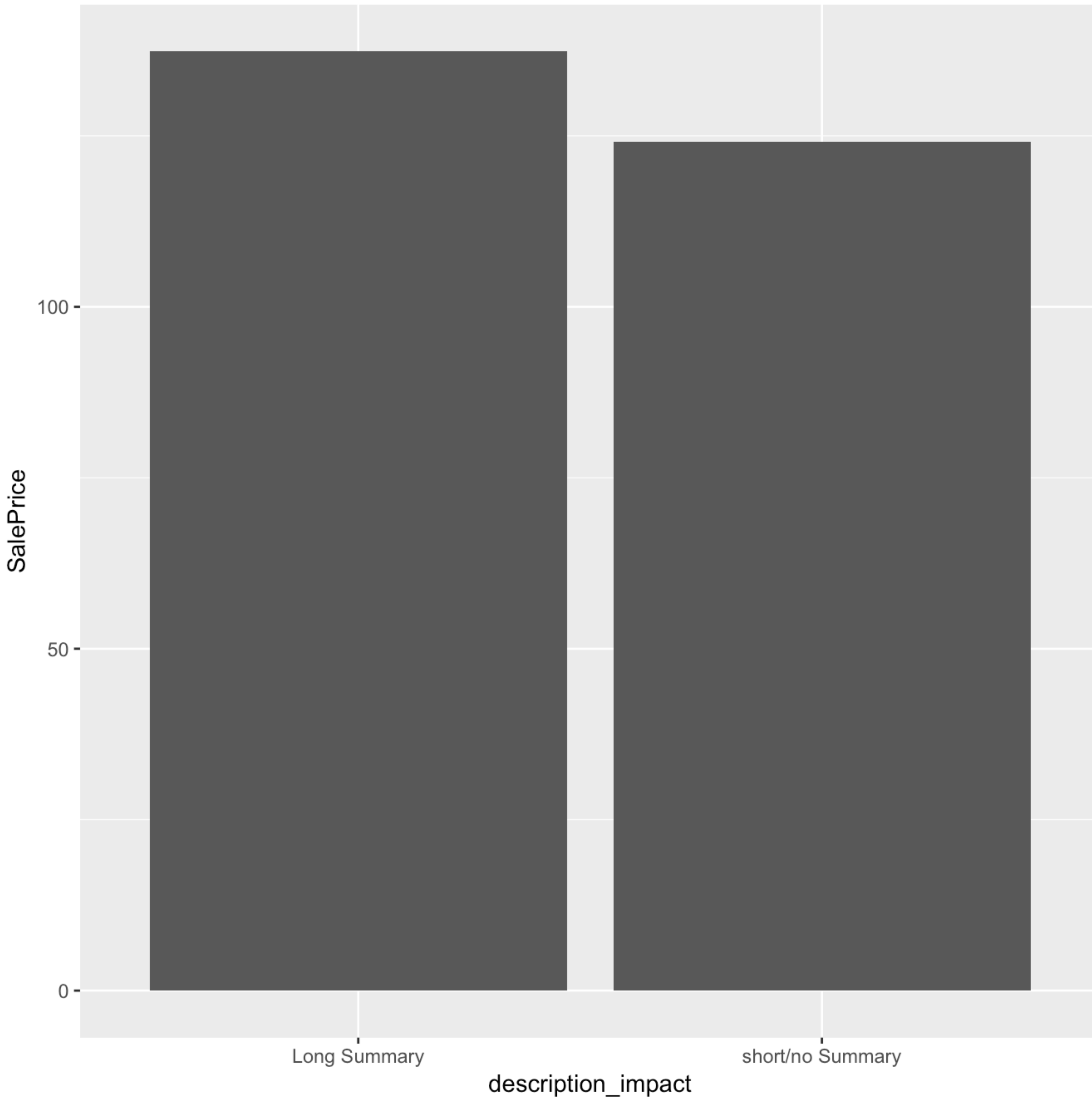

```
ggplot(train,aes(x=summary_impact,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #F
```



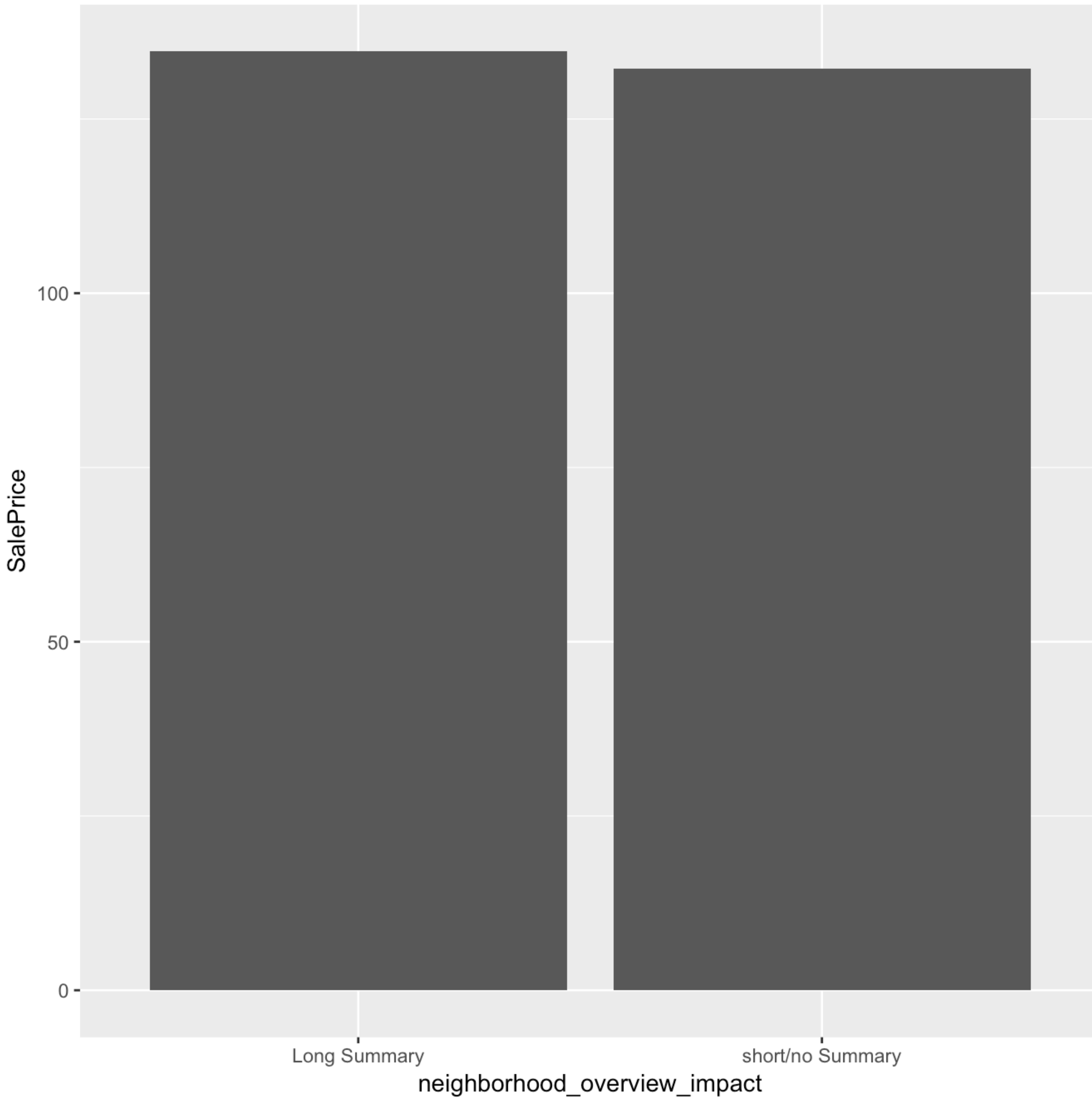

```
ggplot(train,aes(x=space_impact,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



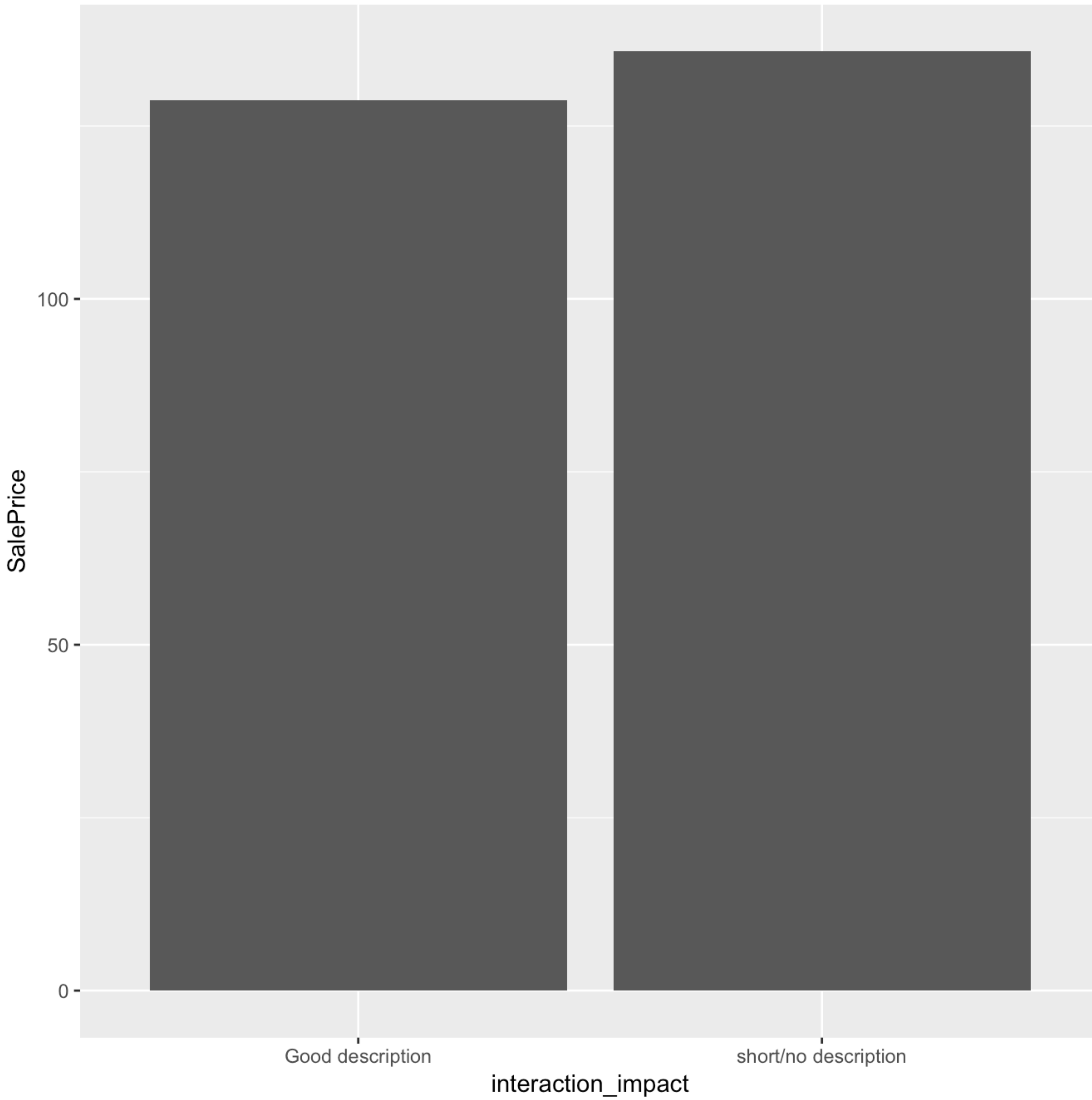

```
ggplot(train,aes(x=description_impact,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



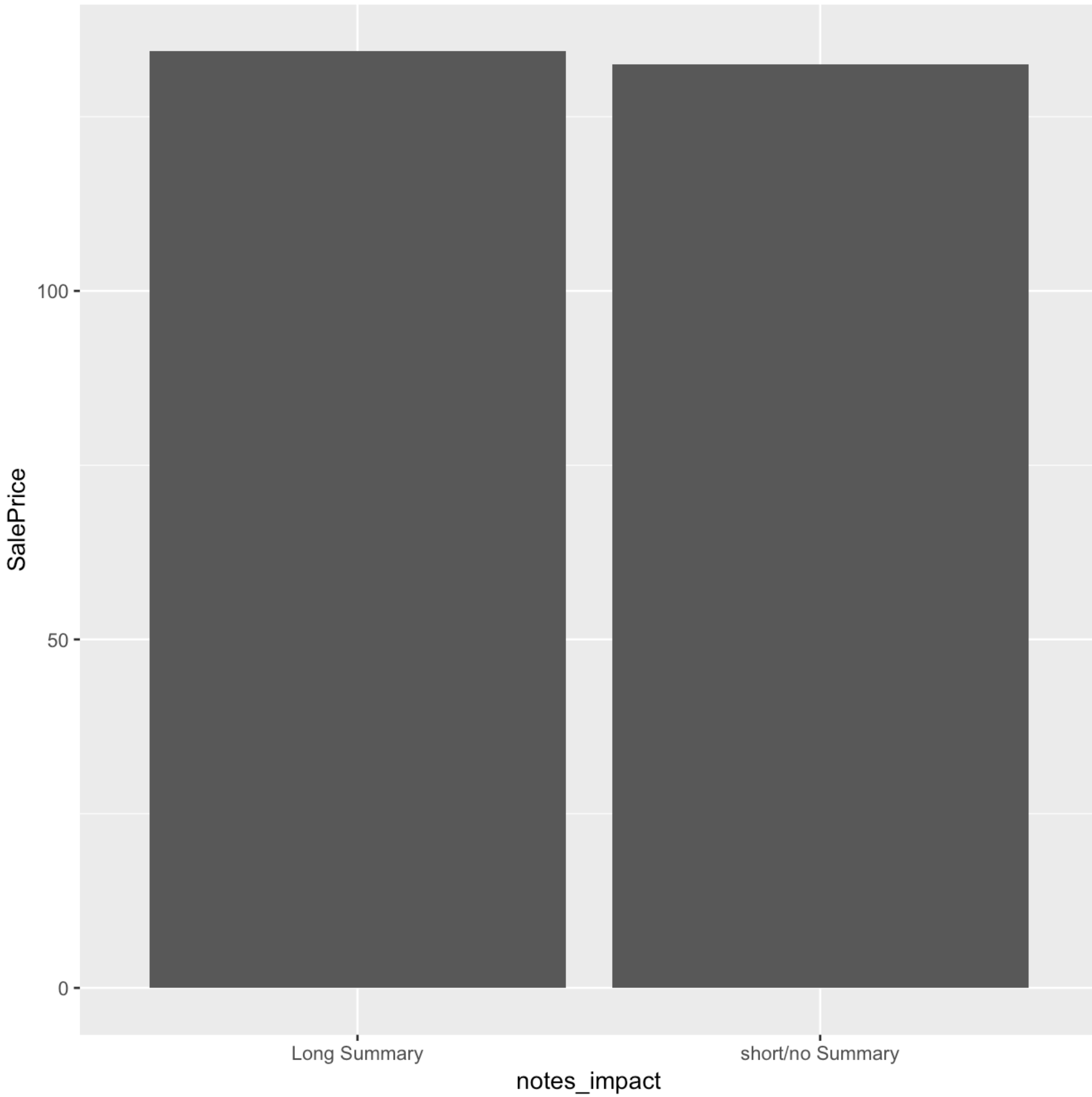

```
ggplot(train,aes(x=neighborhood_overview_impact,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #F
```



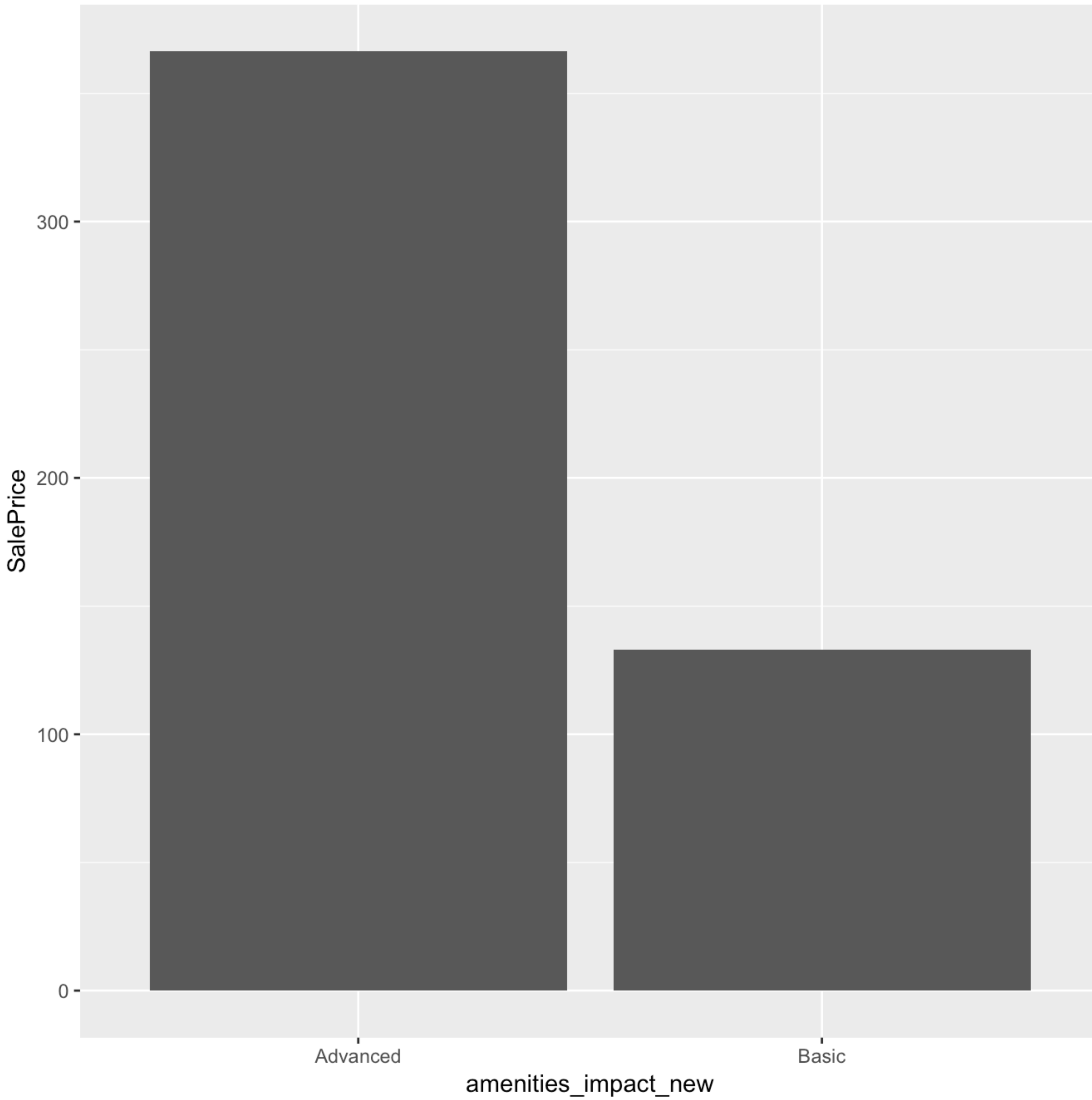

```
ggplot(train,aes(x=interaction_impact,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```



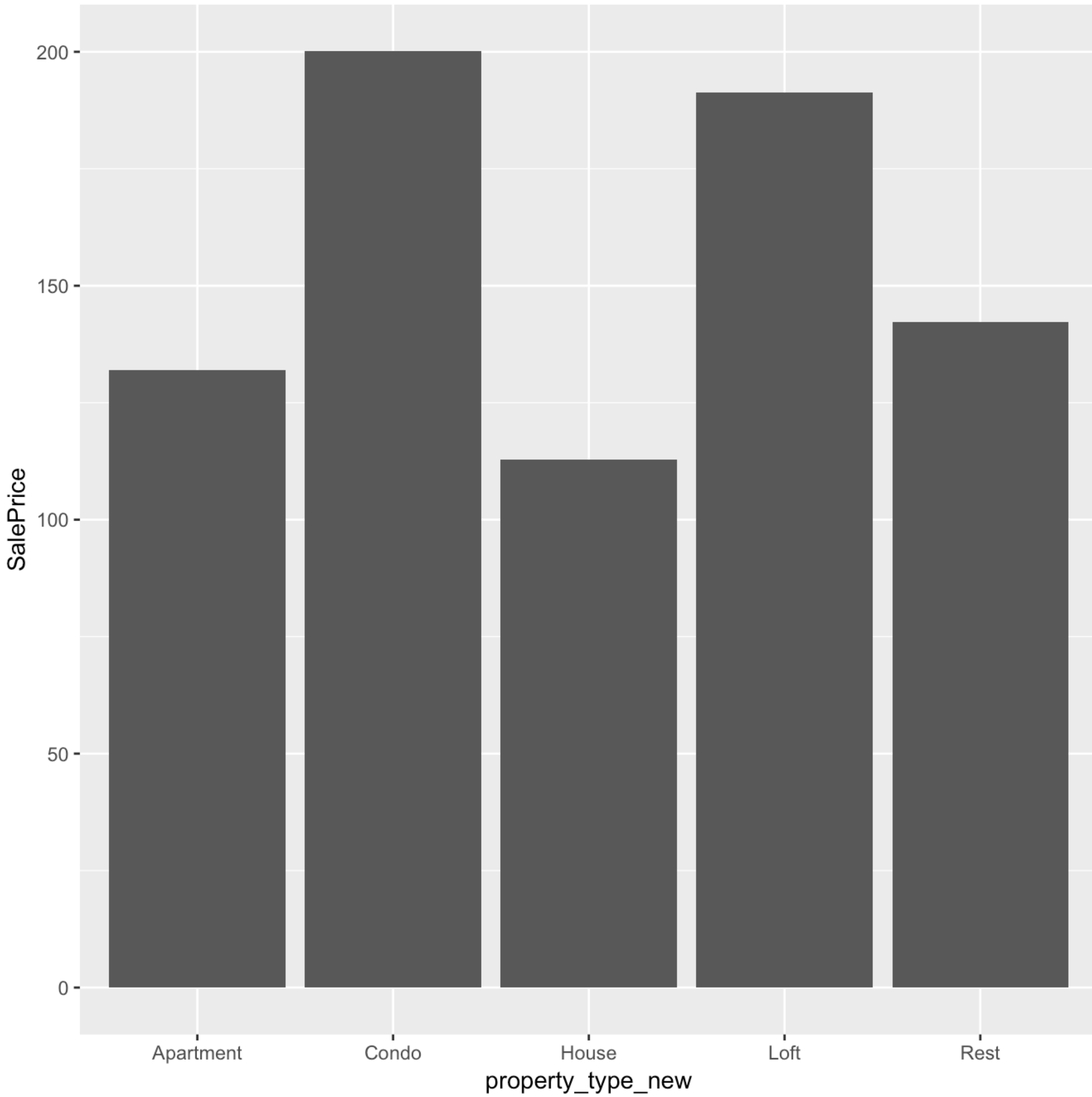

```
ggplot(train,aes(x=notes_impact,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #F
```



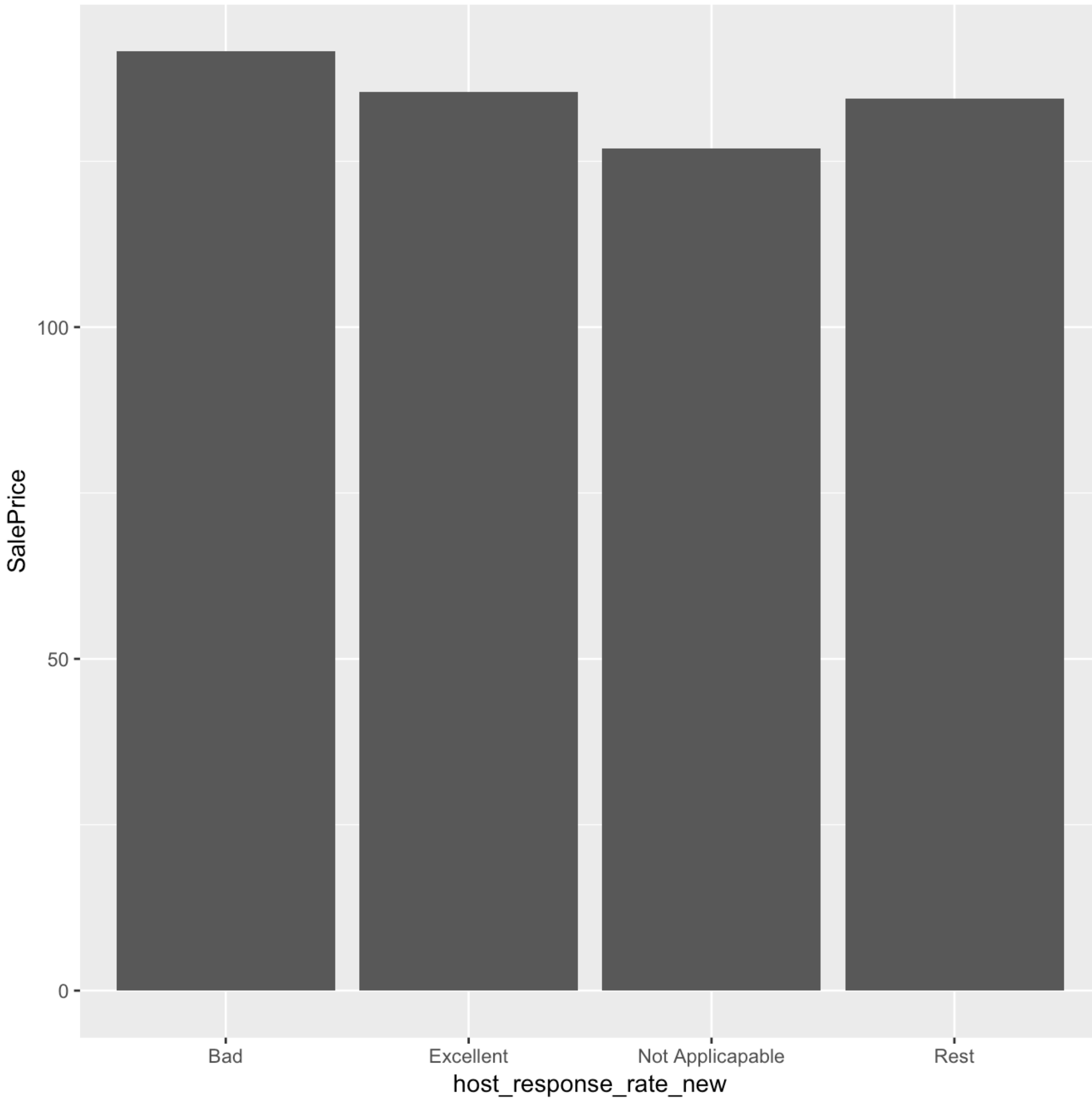

```
ggplot(train,aes(x=amenities_impact_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```




```
ggplot(train,aes(x=property_type_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```




```
ggplot(train,aes(x=host_response_rate_new,y=SalePrice)) +
  stat_summary(fun.y=mean, geom='bar') #T
```




```
# Remove some variables with no obvious difference between two different levels
train$summary_impact <- NULL
train$notes_impact <- NULL
```

Model building and evaluation

```
# Split train dataset into train_inner and test_inner
set.seed(123)
in_train <- createDataPartition(train$SalePrice, p=0.7, list=F)
train_inner <- train[in_train,]
test_inner <- train[-in_train,]
```

With the constrains of time, below just list several different model and their codes. The corresponding results have been omitted.

(a). MLR

```
# Create model: First include all predictor variables to see what will happen
mlr <-lm(formula = SalePrice~.,data = train_inner)
# Evaluation of model
getOption("max.print")
options(max.print = 2000)
summary(mlr)
train_prediction_1<- predict(mlr,test_inner)
```

```
## Warning in predict.lm(mlr, test_inner): prediction from a rank-deficient
## fit may be misleading
```

```
rmse_a = sqrt(mean((train_prediction_1 - test_inner$SalePrice)^2))
rmse_a
# Read in scoring data and apply model to generate predictions scoringData
test_predictions_1 = predict(mlr, newdata=test)
```

```
## Warning in predict.lm(mlr, newdata = test): prediction from a rank-
## deficient fit may be misleading
```

```
# Construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_1)
write.csv(submissionFile, 'submission_MLR.csv',row.names = F)
```

(b). Decision Trees

```
# Create model
modfit <- train(SalePrice ~.,method="rpart",data=train_inner)
# Evaluation of model
train_prediction_2 <- predict(modfit,data =test_inner)
rmse_b = sqrt(mean((train_prediction_2 - test_inner$SalePrice)^2))
rmse_b
# Read in scoring data and apply model to generate predictions scoringData
test_predictions_2 = predict(modfit, newdata=test)
# Construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_2)
write.csv(submissionFile, 'submission_DecisionTree.csv',row.names = F)
```

(c). Random Forest Regression Model

```
# Create model
rf <- randomForest(SalePrice~.,data = train_inner,ntree=1000, proximity=TRUE)
# Verify accuracy
rf $results
# Take a look at contribution of each variable to make prediction
varImp(rf)
# Evaluation of model
train_prediction_3 <- predict(rf,data =test_inner)
rmse_c <- sqrt(mean((train_prediction_3 - test_inner$SalePrice)^2))
rmse_c
# Read in scoring data and apply model to generate predictions scoringData
test_predictions_3 = predict(rf, newdata=test)
# Construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_3)
write.csv(submissionFile, 'submission_RandomForest.csv',row.names = F)
```

(d). Random Forest Regression Model with Cross-validation

```
# Use 10-fold cv to find out optimal value of mtry
trControl_d=trainControl(method="cv",number=10)
tuneGrid_d = expand.grid(mtry=1:5)
# Create model
set.seed(100)
cvForest = train(SalePrice~.,data = train_inner, method="rf",ntree=1000,trControl=trC
ontrol_d,tuneGrid=tuneGrid_d )
# Evaluation of model
train_prediction_4 <- predict(cvForest,test_inner)
rmse_d=sqrt(mean((train_prediction_4-test_inner$SalePrice)^2))
rmse_d
# Read in scoring data and apply model to generate predictions scoringData
test_predictions_4 = predict(cvForest, newdata=test)
# Construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_4)
write.csv(submissionFile, 'submission_RandomForest.csv',row.names = F)
```

(e). Regularized Regression(Lasso)

```
## Create model
tr.control_e <- trainControl(method="repeatedcv", number = 10,repeats = 10)
lambdas_d <- seq(1,0,-.001)
set.seed(123)
lasso_model <- train(SalePrice~., data=train,method="glmnet",metric="RMSE",
                     maximize=FALSE,trControl=tr.control_e,
                     tuneGrid=expand.grid(alpha=1,lambda=c(1,0.1,0.05,0.01,seq(0.009,
0.001,-0.001), 0.00075,0.0005,0.0001)))
## Verify accuracy
lasso_model$results
## Take a look at contribution of each variable to make prediction
varImp(lasso_model)
# Evaluation of model
train_prediction_5 <- predict(lasso_model,data =train)
rmse_e <- sqrt(mean((train_prediction_5 - train$SalePrice)^2))
rmse_e
## Read in scoring data and apply model to generate predictions scoringData
test_predictions_5 = predict(lasso_model, newdata=test)
## Construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_5)
write.csv(submissionFile, 'submission_lasso.csv',row.names = F)
```

(f). Gradient Boosting model(GBM)

```
set.seed(1)
cv.ctrl_gbm <- trainControl(method="repeatedcv",number=5,repeats = 5)
gbm<- train(SalePrice~., method = "gbm", metric = "RMSE", maximize = FALSE,
            trControl =cv.ctrl_gbm, tuneGrid = expand.grid(n.trees = 700,
                                                            interaction.depth = 5, shr
inkage = 0.05,
                                                            n.minobsinnode = 10), data
= train,verbose = FALSE)
varImp(gbm)
prediction_6 <- predict(gbm,newdata = train)
rmse(train$SalePrice,prediction_6)
rmse
## Read in scoring data and apply model to generate predictions scoringData
test_predictions_6 = predict(gbm, newdata=test)
## Construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_6)
write.csv(submissionFile, 'sample_submission.csv',row.names = F)
```

(g). XGBOOST(Extreme Gradient Boosting)

```
#preparing matrix
dtrain <- xgb.DMatrix(data = as.matrix(train[,-241]),label = as.matrix(train$SalePric
e))
#Building model
set.seed(111)
xgb <-  xgboost(booster="gbtree",data = dtrain, nfold = 5,nrounds = 2500, verbose = F
ALSE,
                objective = "reg:linear", eval_metric = "rmse", nthread = 8, eta = 0.
01,
                gamma = 0.0468, max_depth = 6, min_child_weight = 1.41, subsample = 0
.769, colsample_bytree =0.283)
mat <- xgb.importance (feature_names = colnames(dtrain),model = xgb)
xgb.plot.importance (importance_matrix = mat[1:20])
prediction_7 <- predict(xgb,newdata = dtrain)
rmse(train$SalePrice,prediction_7)
rmse
## Read in scoring data and apply model to generate predictions scoringData
test_predictions_7 = predict(xgb, newdata=test)
## Construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_7)
write.csv(submissionFile, 'submission_xgb.csv',row.names = F)
```

(h). cvBoost with Cross-validation

```
set.seed(100)
trControl_h = trainControl(method="cv",number=10)
tuneGrid_h =  expand.grid(n.trees = 1000, interaction.depth = c(1,2),
                          shrinkage = (1:100)*0.001,n.minobsinnode=5)
cvBoost = train(SalePrice~.,data = train_inner,method="gbm", trControl=trControl_h, t
uneGrid=tuneGrid_h)
# Check the results of training and which tuning parameters were selected
cvBoost$results
cvBoost$bestTune
# Check which variables ended up being most important to the model
varImp(cvBoost)
# Evaluation of model
train_prediction_8 <- predict(cvBoost,data = train)
rmse(train$SalePrice,train_prediction_8)
rmse
# read in scoring data and apply model to generate predictions scoringData
test_predictions_8 = predict(cvBoost, newdata=test)
# construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_8)
write.csv(submissionFile, 'submission_cvBoost.csv',row.names = F)
```

(i). Ridge Regression Model

```
# Create model
tr.control_i <- trainControl(method="repeatedcv", number = 10,repeats = 10)
lambdas_i <- seq(1,0,-.001)
set.seed(123)
ridge_model <- train(SalePrice~., data=train, method="glmnet", metric="RMSE",
                     maximize=FALSE, trControl=tr.control_i,
                     tuneGrid=expand.grid(alpha=0,lambda=lambdas_i))
# Verify accuracy
ridge_model$results
## Take a look at contribution of each variable to make prediction
varImp(ridge_model)
## Evaluation of model
train_prediction_9 <- predict(ridge_model,data =train)
rmse(train$SalePrice,train_prediction_9)
rmse
## Read in scoring data and apply model to generate predictions scoringData
test_predictions_9 = predict(ridge_model, newdata=test)
## Construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_9)
write.csv(submissionFile, 'submission_ridge.csv',row.names = F)
```

(j). Bag

```
set.seed(100)
bag = randomForest (SalePrice~., data = train)
## Evaluation of model
train_prediction_10 <- predict(bag,data =train)
rmse(train$SalePrice,train_prediction_10)
## Read in scoring data and apply model to generate predictions scoringData
test_predictions_10 = predict(bag, newdata=test)
## Construct submision from predictions
submissionFile = data.frame(id = test$id, price = test_predictions_10)
write.csv(submissionFile, 'submission_ridge.csv',row.names = F)
```