tutorial9-script.R

garethmoen

2021-11-23

```
# Packages
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(stargazer)
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

```
#### Recap on categorical variables using lm()

# Load in salary dataset
salary <- read.csv("https://raw.githubusercontent.com/ASDS-TCD/StatsI_Fall2021/main/datasets/salary.csv

# A basic bivariate regression of salary on grants
salary_reg1 <- lm(Salary_9_mo~Avg_Cont_Grants, data = salary)

# Hand-coding a dummy variable
salary$rank_d1 <- rep(0, nrow(salary))
salary$rank_d2 <- rep(0, nrow(salary))
salary[which(salary$Rank_Code == 3), "rank_d1"] <- as.factor("1")
salary[which(salary$Rank_Code == 2), "rank_d2"] <- as.factor("1")

salary_reg2 <- lm(Salary_9_mo~Avg_Cont_Grants + rank_d1 + rank_d2, data = salary)

# Don't do this!
salary_reg3 <- lm(Salary_9_mo~Avg_Cont_Grants + Rank_Code, data = salary)

# What's wrong with model 3?
stargazer(salary_reg1, salary_reg2, salary_reg3, type = "text")
```

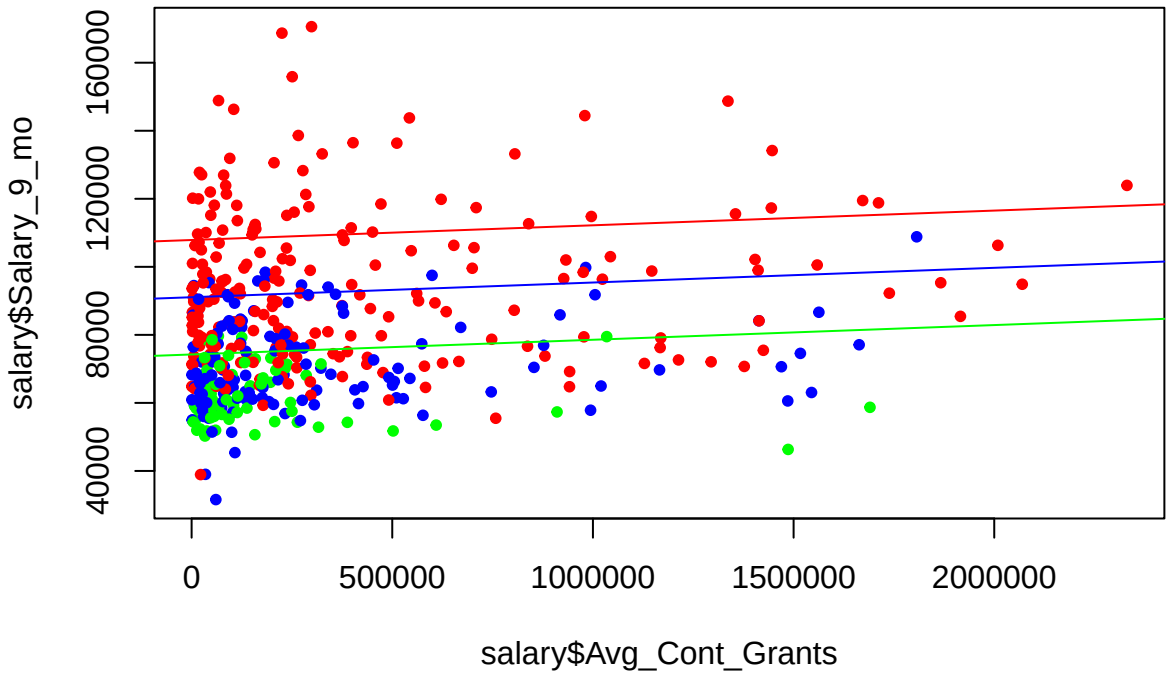
```
##
```

```
## ================================================================================================
##                                          Dependent variable:
##                      ------------------------------------------------------------------------
##                                              Salary_9_mo
##                                (1)                (2)                (3)
## ------------------------------------------------------------------------------------------------
## Avg_Cont_Grants            0.010***           0.004**            0.004**
##                             (0.002)            (0.002)            (0.002)
##
## rank_d1                                     -30,686.170***
##                                               (2,516.577)
##
## rank_d2                                     -22,529.780***
##                                               (1,966.337)
##
## Rank_Code                                                      -16,823.940***
##                                                                 (1,208.234)
##
## Constant                  78,394.080***      92,404.040***      107,886.000***
##                            (1,346.572)        (1,465.852)        (2,393.823)
##
## ------------------------------------------------------------------------------------------------
## Observations                  424                424                424
## R2                           0.035              0.360              0.339
## Adjusted R2                  0.033              0.355              0.336
## Residual Std. Error  21,966.650 (df = 422)  17,937.800 (df = 420)  18,197.920 (df = 421)
## F Statistic         15.274*** (df = 1; 422) 78.586*** (df = 3; 420) 108.072*** (df = 2; 421)
## ================================================================================================
## Note:                                                             *p<0.1; **p<0.05; ***p<0.01
```

```
# read.csv and read_csv will cast factors as integers
class(salary$Rank_Code)
```

```
## [1] "integer"
```

```
# lm() will treat Rank_Code as a continous variable!
plot(salary$Avg_Cont_Grants, salary$Salary_9_mo, type = "p", pch = 20, col = c("red", "blue", "green")[s
abline(salary_reg3$coefficients[1], salary_reg3$coefficients[2], col = "red")
abline(salary_reg3$coefficients[1] + salary_reg3$coefficients[3], salary_reg3$coefficients[2], col = "bl
abline(salary_reg3$coefficients[1] + salary_reg3$coefficients[3]*2, salary_reg3$coefficients[2], col = "
```



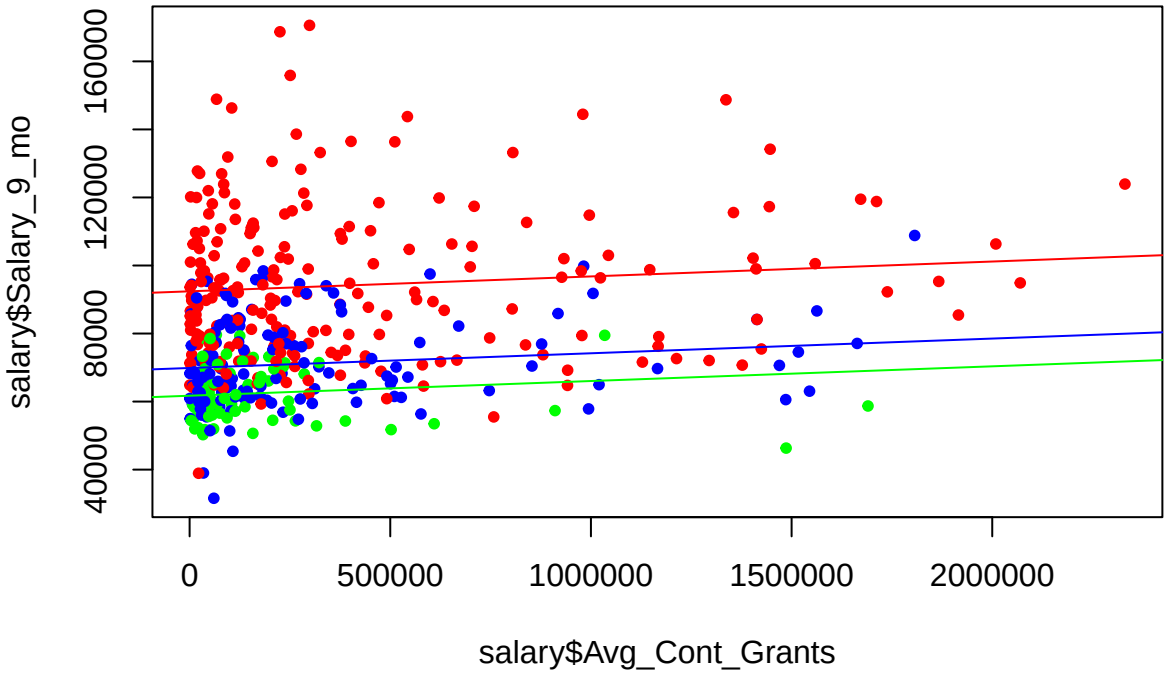

```
# Cast Rank_Code as a factor first
salary_reg4 <- lm(Salary_9_mo~Avg_Cont_Grants + as.factor(Rank_Code), data = salary)

# The result is now identical with hand-coding as a dummy
stargazer(salary_reg2, salary_reg4, type = "text")
```

```
##
## ============================================================
##                                      Dependent variable:
##                                  ---------------------------
##                                          Salary_9_mo
##                                     (1)            (2)
## ------------------------------------------------------------
## Avg_Cont_Grants                    0.004**        0.004**
##                                    (0.002)        (0.002)
##
## rank_d1                          -30,686.170***
##                                   (2,516.577)
##
## rank_d2                          -22,529.780***
##                                   (1,966.337)
##
## as.factor(Rank_Code)2                          -22,529.780***
##                                                 (1,966.337)
##
## as.factor(Rank_Code)3                          -30,686.170***
##                                                 (2,516.577)
##
## Constant                         92,404.040***  92,404.040***
##                                   (1,465.852)    (1,465.852)
##
## ------------------------------------------------------------
## Observations                          424            424
```

```
## R2                                0.360          0.360
## Adjusted R2                       0.355          0.355
## Residual Std. Error (df = 420)  17,937.800     17,937.800
## F Statistic (df = 3; 420)       78.586***      78.586***
## ============================================================
## Note:                          *p<0.1; **p<0.05; ***p<0.01
```

```
# We can see that we have a proper plot now
plot(salary$Avg_Cont_Grants, salary$Salary_9_mo,
     type = "p", pch = 20, col = c("red", "blue", "green")[salary$Rank_Code])
abline(salary_reg4$coefficients[1], salary_reg4$coefficients[2], col = "red")
abline(salary_reg4$coefficients[1] + salary_reg4$coefficients[3], salary_reg3$coefficients[2], col = "bl
abline(salary_reg4$coefficients[1] + salary_reg4$coefficients[4], salary_reg3$coefficients[2], col = "gr
```




```
# Note: this is only really relevent when you have categories > 2, as a simple binary
# variable (i.e. male/female) only has two levels, and therefore only one intercept
# that differs. Note also that ggplot's geom_smooth lm method knows to cast an integer
# as a factor when you supply the variable as an argument to colour or group, etc.

#### Mini project part 2

# Load in data
houses <- read.csv("https://raw.githubusercontent.com/gedeck/practical-statistics-for-data-scientists/ma

# Last week's model
hmod1 <- lm(AdjSalePrice ~ SqFtLot + BldgGrade, houses)
summary(hmod1)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtLot + BldgGrade, data = houses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -1185975  -134669   -31675    89633  9899473
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.121e+06  1.245e+04 -90.041   <2e-16 ***
## SqFtLot      5.272e-01  6.563e-02   8.032    1e-15 ***
## BldgGrade    2.187e+05  1.613e+03 135.558   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 283800 on 22684 degrees of freedom
## Multiple R-squared:  0.4579, Adjusted R-squared:  0.4579
## F-statistic:  9582 on 2 and 22684 DF,  p-value: < 2.2e-16
```

```
# Where possible, character vectors are automatically cast as factors by lm()
class(houses$PropertyType)
```

```
## [1] "character"
```

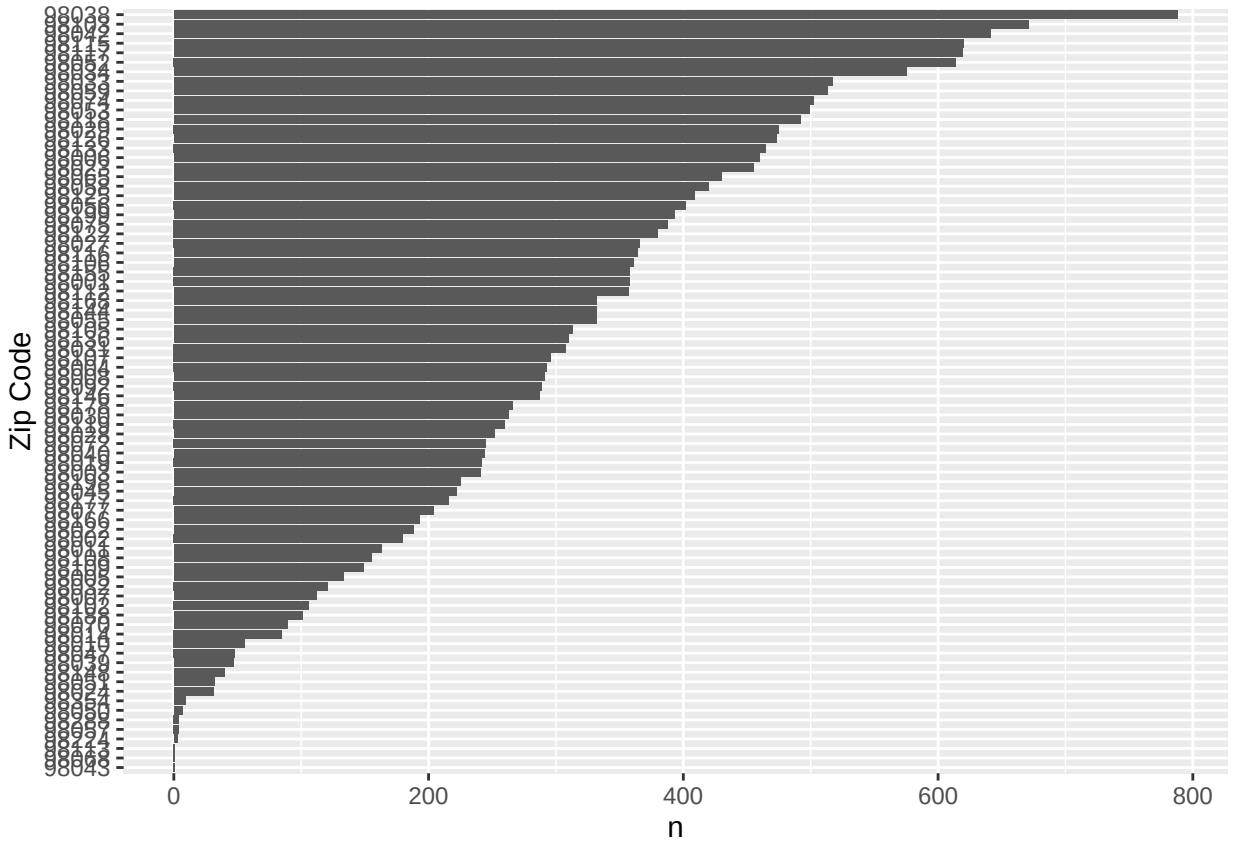
```
typeof(houses$PropertyType)
```

```
## [1] "character"
```

```
# Group by zipcode frequency
houses %>%
  group_by(ZipCode) %>%
  summarise(n = n()) %>%
  arrange(desc(n))
```

```
## # A tibble: 80 x 2
##    ZipCode     n
##      <int> <int>
##  1   98038   788
##  2   98103   671
##  3   98042   641
##  4   98115   620
##  5   98117   619
##  6   98052   614
##  7   98034   575
##  8   98033   517
##  9   98059   513
## 10   98074   502
## # ... with 70 more rows
```

```
# Plot zipcode frequency
houses %>%
  group_by(ZipCode) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  ggplot(aes(as.factor(reorder(ZipCode, n)), n)) +
    geom_col() +
    coord_flip() +
    xlab("Zip Code")
```

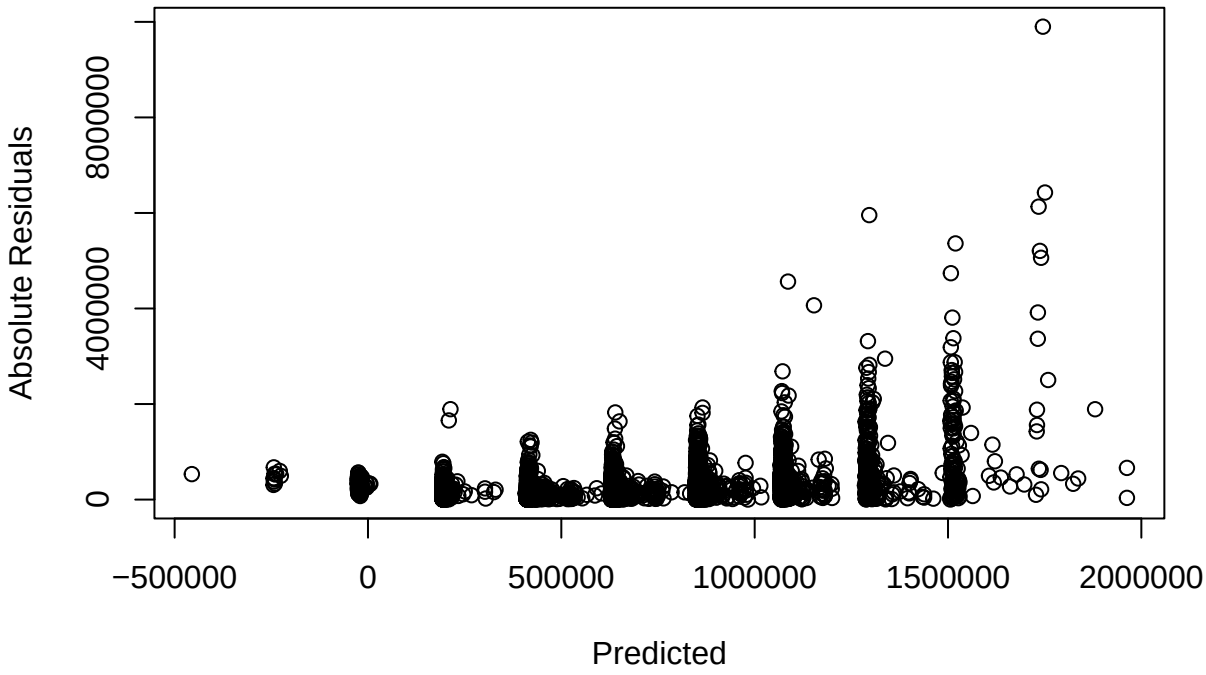



```
# Group by zipcode mean sale price
houses %>%
  group_by(ZipCode) %>%
  summarise(n = n(),
            mean = mean(AdjSalePrice)) %>%
  arrange(desc(mean))
```

```
## # A tibble: 80 x 3
##    ZipCode     n     mean
##      <int> <int>    <dbl>
##  1   98039    47 2154126.
##  2   98004   293 1464104.
##  3   98040   244 1349175.
##  4   98112   357 1094507.
##  5   98109   149  884992.
##  6   98005   133  859939.
##  7   98119   260  856395.
##  8   98102   106  828102.
##  9   98006   460  825382.
## 10   98033   517  822374.
## # ... with 70 more rows
```

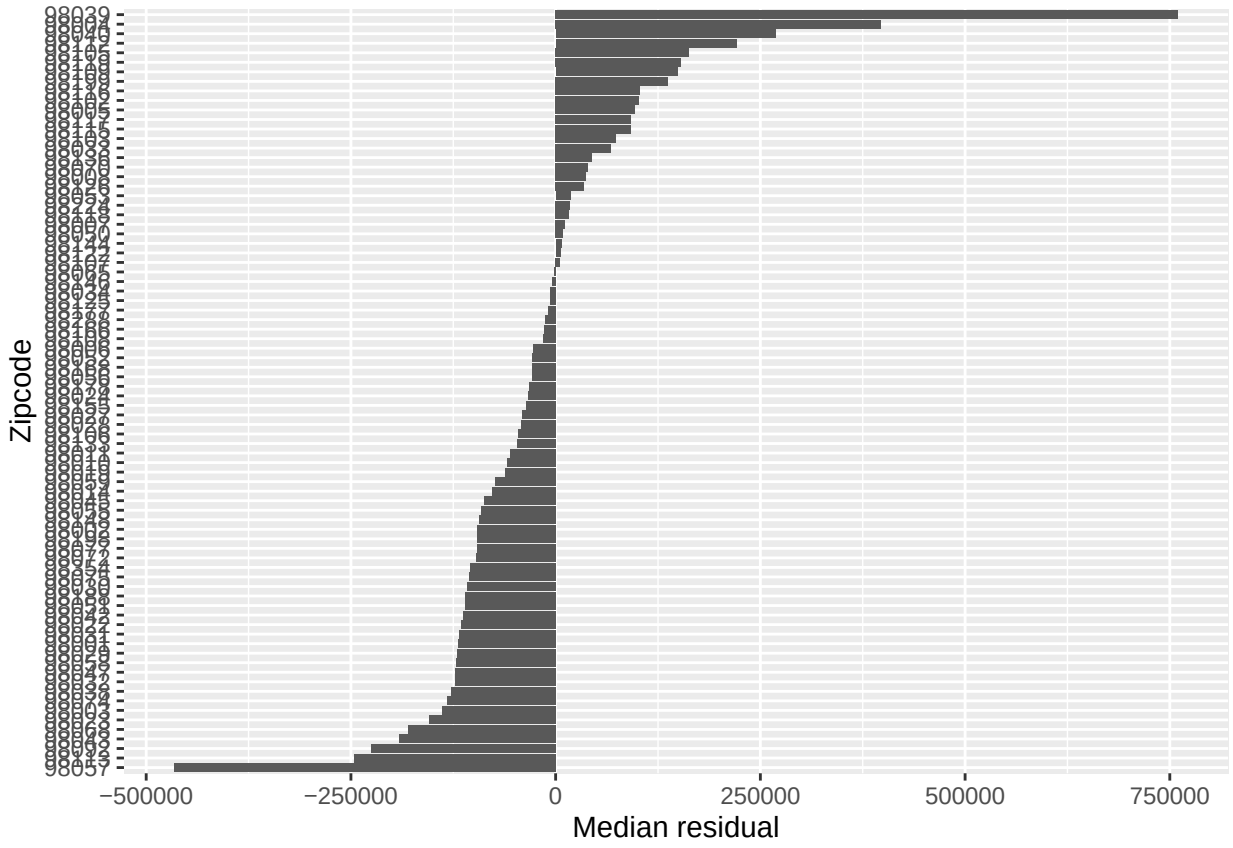
```
# Remove scientific notation from output
options(scipen = 999)

# Plot (absolute) residuals of initial model, hmod1
plot(predict(hmod1), abs(resid(hmod1)), xlab = "Predicted", ylab = "Absolute Residuals")
```

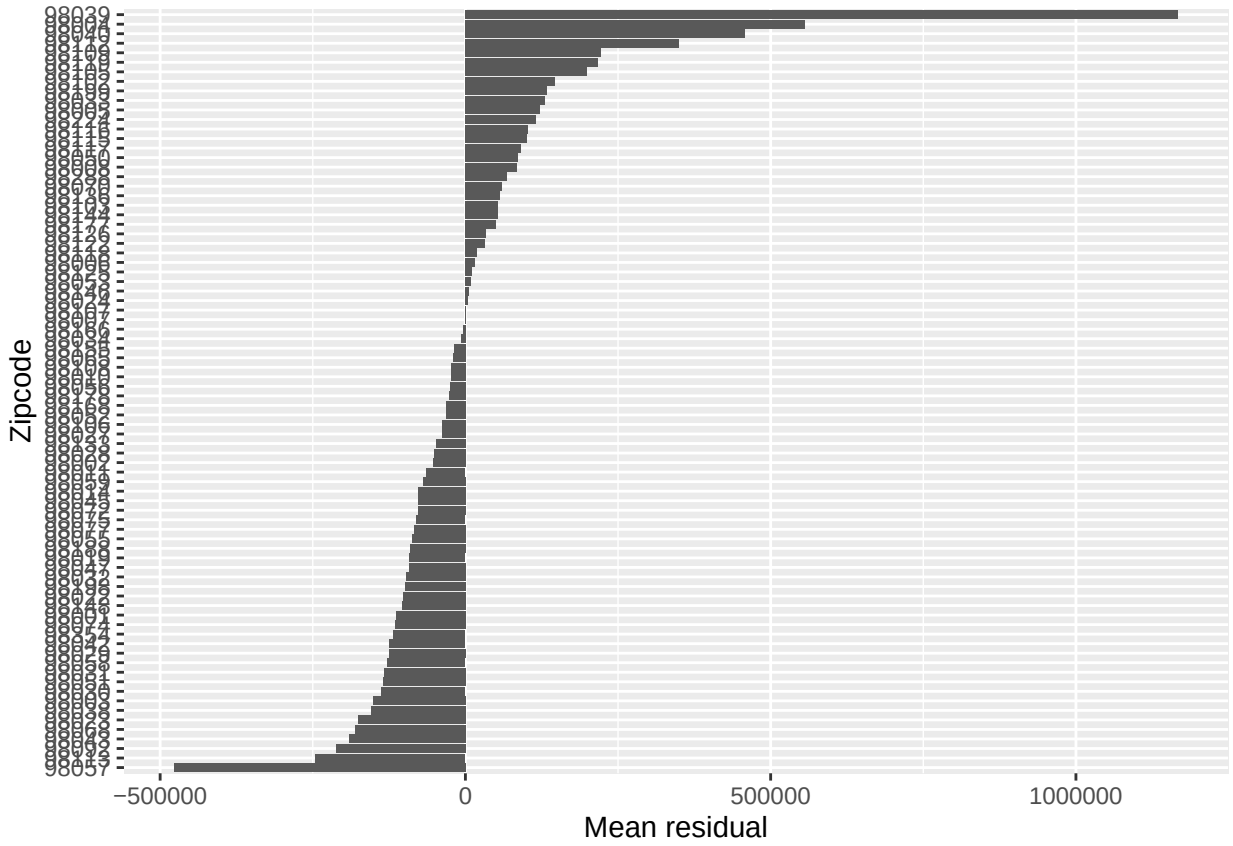



```
# Rather than clustering according to sale price, why not try clustering according
# to residuals (i.e., where our current model is weak at predicting, we see if
# zipcode adds anything).

# Tweaking our initial dplyr code to get the median of the residuals
houses %>%
  mutate(resid = resid(hmod1)) %>%
  group_by(ZipCode) %>%
  summarise(median = median(resid),
            n = n()) %>%
  arrange(desc(median)) %>%
  ggplot(aes(as.factor(reorder(ZipCode, median)), median)) +
    geom_col() +
    coord_flip() +
    ylab("Median residual") +
    xlab("Zipcode")
```




```
# Is mean better?
houses %>%
  mutate(resid = resid(hmod1)) %>%
  group_by(ZipCode) %>%
  summarise(mean = mean(resid),
            n = n()) %>%
  arrange(desc(mean)) %>%
  ggplot(aes(as.factor(reorder(ZipCode, mean)), mean)) +
    geom_col() +
    coord_flip() +
    ylab("Mean residual") +
    xlab("Zipcode")
```




```
# How do we create a new set of groupings?
?ntile

# Use ntile on the cases sorted by median
houses %>%
  mutate(resid = resid(hmod1)) %>%
  group_by(ZipCode) %>%
  summarise(median = median(resid),
            n = n()) %>%
  arrange(desc(median)) %>%
  mutate(ZipGroup = ntile(median, 5)) %>%
  group_by(ZipGroup) %>%
  summarise(n = n(),
            med = median(median))
```

```
## # A tibble: 5 x 3
##   ZipGroup     n      med
##      <int> <int>    <dbl>
## 1        1    16 -129578.
## 2        2    16  -96379.
## 3        3    16  -32959.
## 4        4    16    8255.
## 5        5    16  120177.
```

```
# Another method (Bruce, Bruce & Gedeck)
houses %>%
  mutate(resid = resid(hmod1)) %>%
  group_by(ZipCode) %>%
```

```
  summarise(median = median(resid),
            n = n()) %>%
  arrange(desc(median)) %>%
  mutate(sum_n = cumsum(n),
         ZipGroup = ntile(sum_n, 5)) %>%
  group_by(ZipGroup) %>%
  summarise(n = n(),
            med = median(median))
```

```
## # A tibble: 5 x 3
##   ZipGroup     n      med
##      <int> <int>    <dbl>
## 1        1    16  120177.
## 2        2    16    8255.
## 3        3    16  -32959.
## 4        4    16  -96379.
## 5        5    16 -129578.
```

```
# code for mutating - create a new dataframe "zip_groups"
zip_groups <- houses %>%
  mutate(resid = resid(hmod1)) %>%
  group_by(ZipCode) %>%
  summarise(median = median(resid),
            n = n()) %>%
  arrange(desc(median)) %>%
  mutate(ZipGroup = ntile(median, 5))

head(zip_groups)
```

```
## # A tibble: 6 x 4
##   ZipCode  median     n ZipGroup
##     <int>   <dbl> <int>    <int>
## 1   98039 759928.    47        5
## 2   98004 397460.   293        5
## 3   98040 268615.   244        5
## 4   98112 220772.   357        5
## 5   98105 163129.   313        5
## 6   98119 153424.   260        5
```

```
# join zip_groups to houses using left_join()
?left_join
houses <- houses %>%
  left_join(select(zip_groups, ZipCode, ZipGroup), by = "ZipCode")

# running a regression using ZipGroup
hmod2 <- lm(AdjSalePrice ~ SqFtLot + BldgGrade + as.factor(ZipGroup), houses)
summary(hmod2)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtLot + BldgGrade + as.factor(ZipGroup),
##     data = houses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

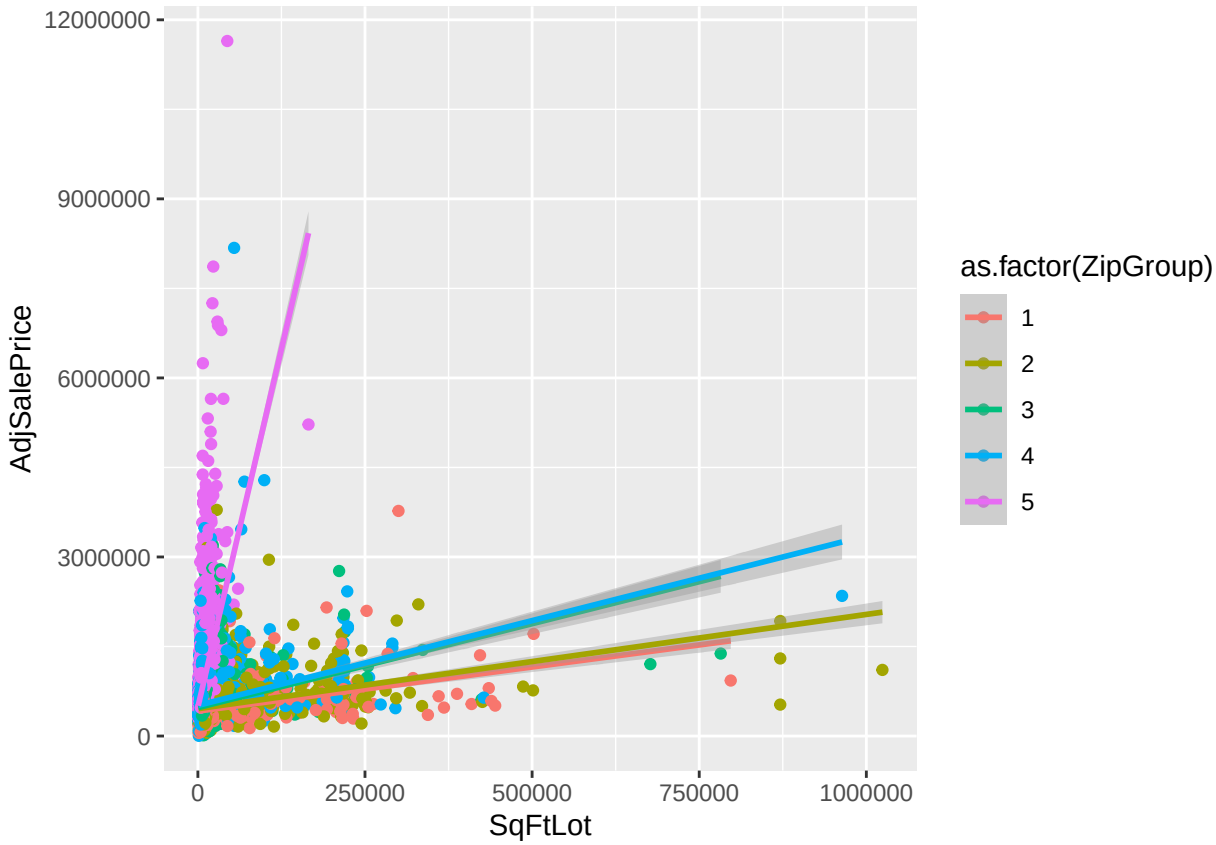
```
## -1123402  -113236   -11890    77551  9733663
##
## Coefficients:
##                            Estimate     Std. Error  t value
## (Intercept)          -1217787.81146    12102.85014 -100.620
## SqFtLot                     1.07515        0.06074   17.701
## BldgGrade              212042.86888     1488.57131  142.447
## as.factor(ZipGroup)2    45275.78031     5850.38366    7.739
## as.factor(ZipGroup)3   113199.47783     5582.99292   20.276
## as.factor(ZipGroup)4   160492.70857     5474.06687   29.319
## as.factor(ZipGroup)5   325115.48565     5368.35765   60.561
##                                Pr(>|t|)
## (Intercept)          < 0.0000000000000002 ***
## SqFtLot              < 0.0000000000000002 ***
## BldgGrade            < 0.0000000000000002 ***
## as.factor(ZipGroup)2   0.0000000000000104 ***
## as.factor(ZipGroup)3 < 0.0000000000000002 ***
## as.factor(ZipGroup)4 < 0.0000000000000002 ***
## as.factor(ZipGroup)5 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 259500 on 22680 degrees of freedom
## Multiple R-squared:  0.5468, Adjusted R-squared:  0.5467
## F-statistic:  4561 on 6 and 22680 DF,  p-value: < 0.00000000000000022
```

```
summary(hmod1)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtLot + BldgGrade, data = houses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1185975  -134669   -31675    89633  9899473
##
## Coefficients:
##                   Estimate     Std. Error t value             Pr(>|t|)
## (Intercept) -1120602.22752    12445.52186 -90.041 < 0.0000000000000002 ***
## SqFtLot            0.52715        0.06563   8.032    0.000000000000001 ***
## BldgGrade     218676.16040     1613.15808 135.558 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 283800 on 22684 degrees of freedom
## Multiple R-squared:  0.4579, Adjusted R-squared:  0.4579
## F-statistic:  9582 on 2 and 22684 DF,  p-value: < 0.00000000000000022
```

```
ggplot(houses, aes(SqFtLot, AdjSalePrice, colour = as.factor(ZipGroup))) +
  geom_point() +
  geom_smooth(method = "lm", aes(colour = as.factor(ZipGroup)))
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# a solution with three groups could be more parsimonious
houses$ZipGroup3 <- recode(houses$ZipGroup, `2` = 1L, `3` = 2L, `4` = 2L, `5` = 3L)
hmod3 <- lm(AdjSalePrice ~ SqFtLot + BldgGrade + as.factor(ZipGroup3), houses)

summary(hmod3)
```

```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtLot + BldgGrade + as.factor(ZipGroup3),
##     data = houses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1100629  -114348   -12427    78117  9734255
##
## Coefficients:
##                             Estimate     Std. Error t value            Pr(>|t|)
## (Intercept)           -1194730.66988    11810.05364 -101.16 <0.0000000000000002
## SqFtLot                     1.09721        0.06083   18.04 <0.0000000000000002
## BldgGrade                211764.56400     1492.82002  141.85 <0.0000000000000002
## as.factor(ZipGroup3)2    116686.71294     3998.38482   29.18 <0.0000000000000002
## as.factor(ZipGroup3)3    304114.34450     4647.02301   65.44 <0.0000000000000002
##
## (Intercept)           ***
## SqFtLot               ***
## BldgGrade             ***
## as.factor(ZipGroup3)2 ***
```

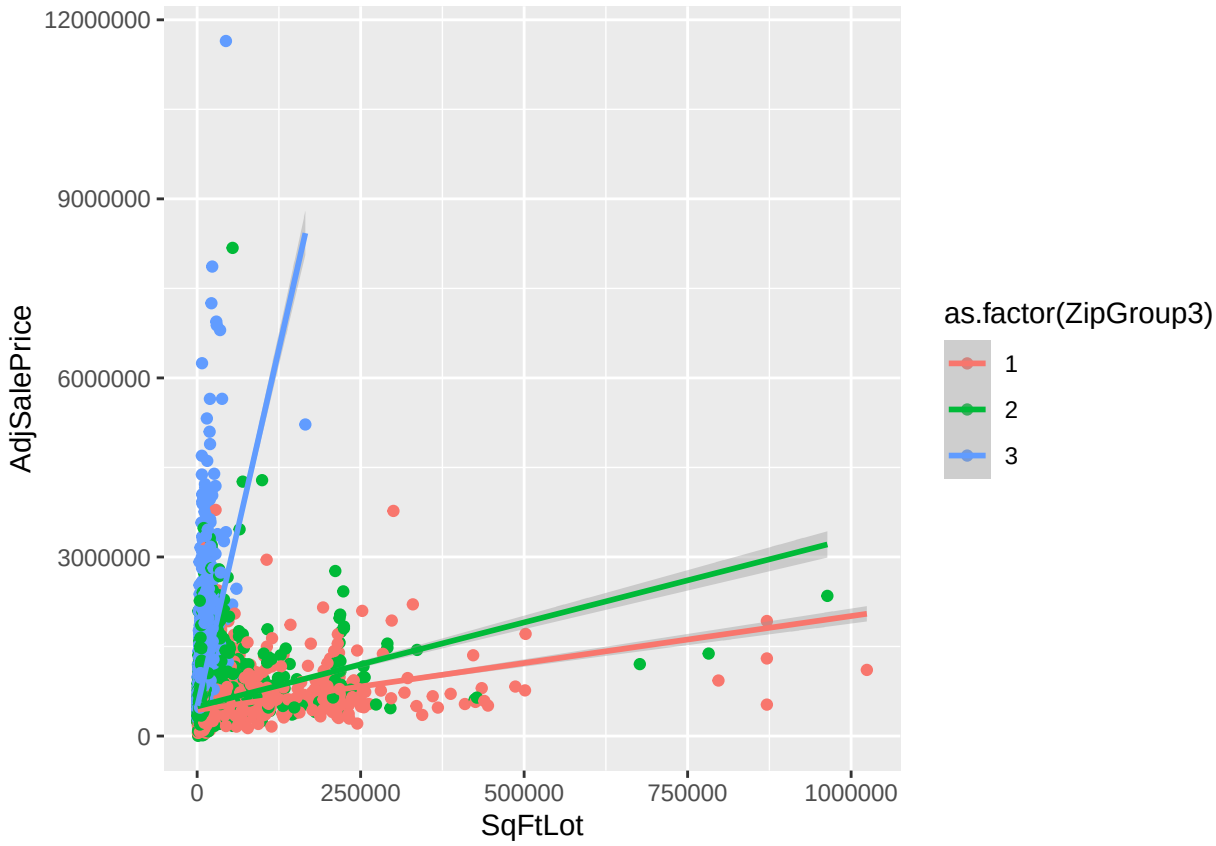
```
## as.factor(ZipGroup3)3 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 260300 on 22682 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.544
## F-statistic:  6767 on 4 and 22682 DF,  p-value: < 0.00000000000000022
```

```
summary(hmod2)
```

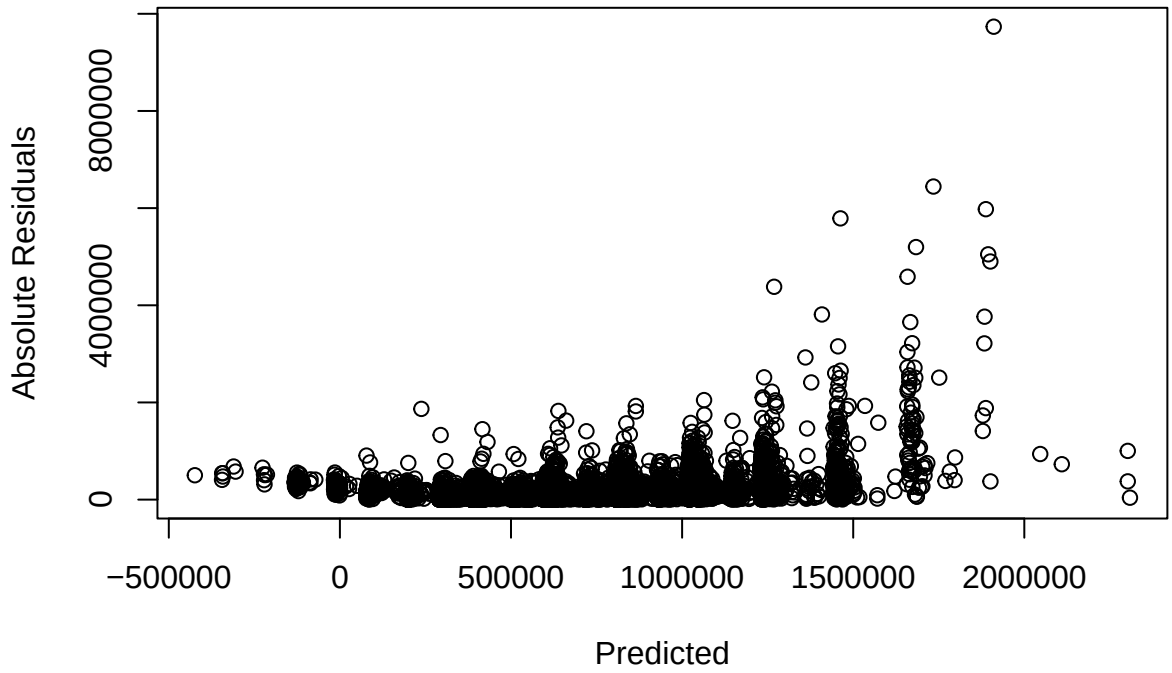
```
##
## Call:
## lm(formula = AdjSalePrice ~ SqFtLot + BldgGrade + as.factor(ZipGroup),
##     data = houses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1123402  -113236   -11890    77551  9733663
##
## Coefficients:
##                             Estimate     Std. Error  t value
## (Intercept)          -1217787.81146    12102.85014 -100.620
## SqFtLot                     1.07515        0.06074   17.701
## BldgGrade              212042.86888     1488.57131  142.447
## as.factor(ZipGroup)2    45275.78031     5850.38366    7.739
## as.factor(ZipGroup)3   113199.47783     5582.99292   20.276
## as.factor(ZipGroup)4   160492.70857     5474.06687   29.319
## as.factor(ZipGroup)5   325115.48565     5368.35765   60.561
##                                Pr(>|t|)
## (Intercept)          < 0.0000000000000002 ***
## SqFtLot              < 0.0000000000000002 ***
## BldgGrade            < 0.0000000000000002 ***
## as.factor(ZipGroup)2   0.0000000000000104 ***
## as.factor(ZipGroup)3 < 0.0000000000000002 ***
## as.factor(ZipGroup)4 < 0.0000000000000002 ***
## as.factor(ZipGroup)5 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 259500 on 22680 degrees of freedom
## Multiple R-squared:  0.5468, Adjusted R-squared:  0.5467
## F-statistic:  4561 on 6 and 22680 DF,  p-value: < 0.00000000000000022
```

```
ggplot(houses, aes(SqFtLot, AdjSalePrice, colour = as.factor(ZipGroup3))) +
  geom_point() +
  geom_smooth(method = "lm", aes(colour = as.factor(ZipGroup3)))
```

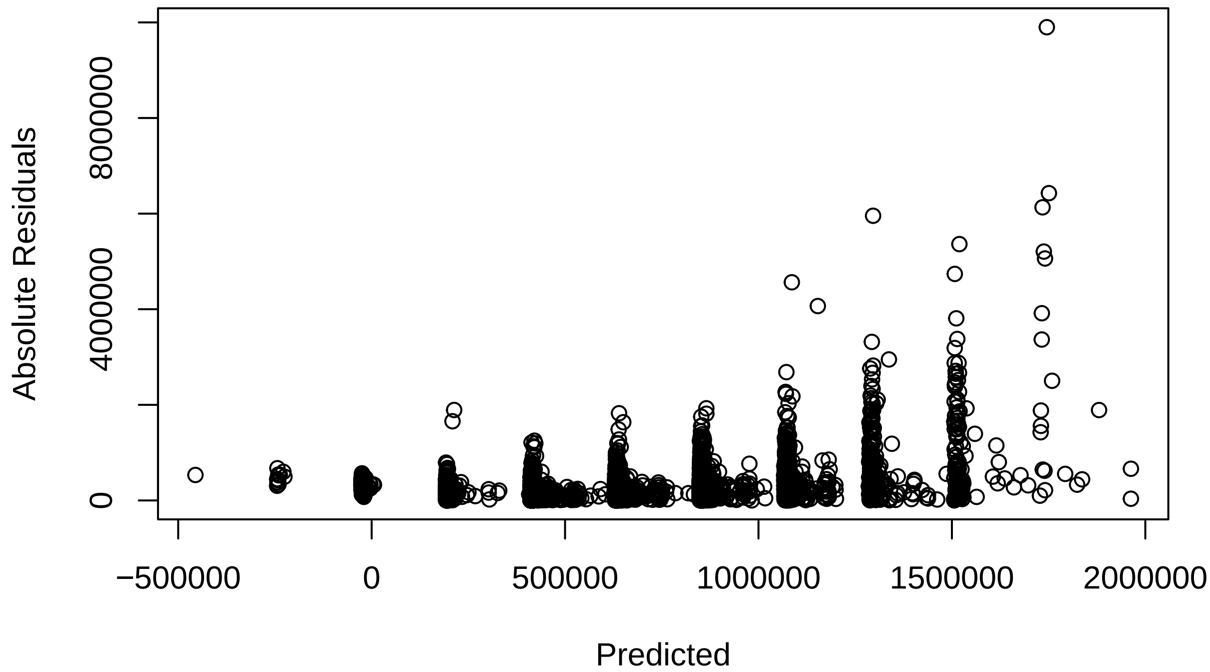
```
## `geom_smooth()` using formula 'y ~ x'
```




```
# let's plot our residuals again and see if our model improved
plot(predict(hmod3), abs(resid(hmod3)), xlab = "Predicted", ylab = "Absolute Residuals")
```




```
plot(predict(hmod1), abs(resid(hmod1)), xlab = "Predicted", ylab = "Absolute Residuals")
```




```
# seems a bit better

# how about our residual standard error?
stargazer(hmod1, hmod2, hmod3, type = "text")
```

```
##
## ==========================================================================================
##                                               Dependent variable:
##                                 ----------------------------------------------------------
##                                                   AdjSalePrice
##                                        (1)                  (2)                  (3)
## ------------------------------------------------------------------------------------------
## SqFtLot                             0.527***             1.075***             1.097***
##                                     (0.066)              (0.061)              (0.061)
##
## BldgGrade                       218,676.200***       212,042.900***       211,764.600**
##                                  (1,613.158)          (1,488.571)          (1,492.820)
##
## as.factor(ZipGroup)2                                  45,275.780***
##                                                        (5,850.384)
##
## as.factor(ZipGroup)3                                 113,199.500***
##                                                        (5,582.993)
##
## as.factor(ZipGroup)4                                 160,492.700***
##                                                        (5,474.067)
##
## as.factor(ZipGroup)5                                 325,115.500***
##                                                        (5,368.358)
##
## as.factor(ZipGroup3)2                                                    116,686.700**
##                                                                           (3,998.385)
##
```

```
## as.factor(ZipGroup3)3                                                                304,114.300***
##                                                                                       (4,647.023)
##
## Constant                    -1,120,602.000***           -1,217,788.000***           -1,194,731.000***
##                                (12,445.520)                (12,102.850)                (11,810.050)
##
## ---------------------------------------------------------------------------------------------------
## Observations                       22,687                      22,687                      22,687
## R2                                 0.458                       0.547                       0.544
## Adjusted R2                        0.458                       0.547                       0.544
## Residual Std. Error     283,766.800 (df = 22684)    259,482.600 (df = 22680)    260,257.000 (df = 2
## F Statistic           9,581.518*** (df = 2; 22684) 4,561.039*** (df = 6; 22680) 6,766.714*** (df = 4
## ===================================================================================================
## Note:                                                                  *p<0.1; **p<0.05; ***
```

```
# seems by adding a location predictor we reduced the RSE by about $25,000. Not bad!
```