STAT 444/844 Winter 2024 Final Project Supplementary Material

Chase(Daolin) An

UW ID: 20885166

Contents

1 Preprocessing

1.1 Loading data and add new variables

```
rm(list=ls())
load("Final.Rdata")
library(mgcv)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

```
library(rfPermute)
```

```
## Warning: package 'rfPermute' was built under R version 4.3.3
```

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.3.3
```

```
set.seed(20885166)
## Check the number of missing values
ayb.missing <- sum(is.na(dat$ayb)); ayb.missing
```

```
## [1] 17
```

```
yr_rmdl.missing <- sum(is.na(dat$yr_rmdl)); yr_rmdl.missing
```

```
## [1] 2410
```

```
stories.missing <- sum(is.na(dat$stories)); stories.missing
```

```
## [1] 4
```

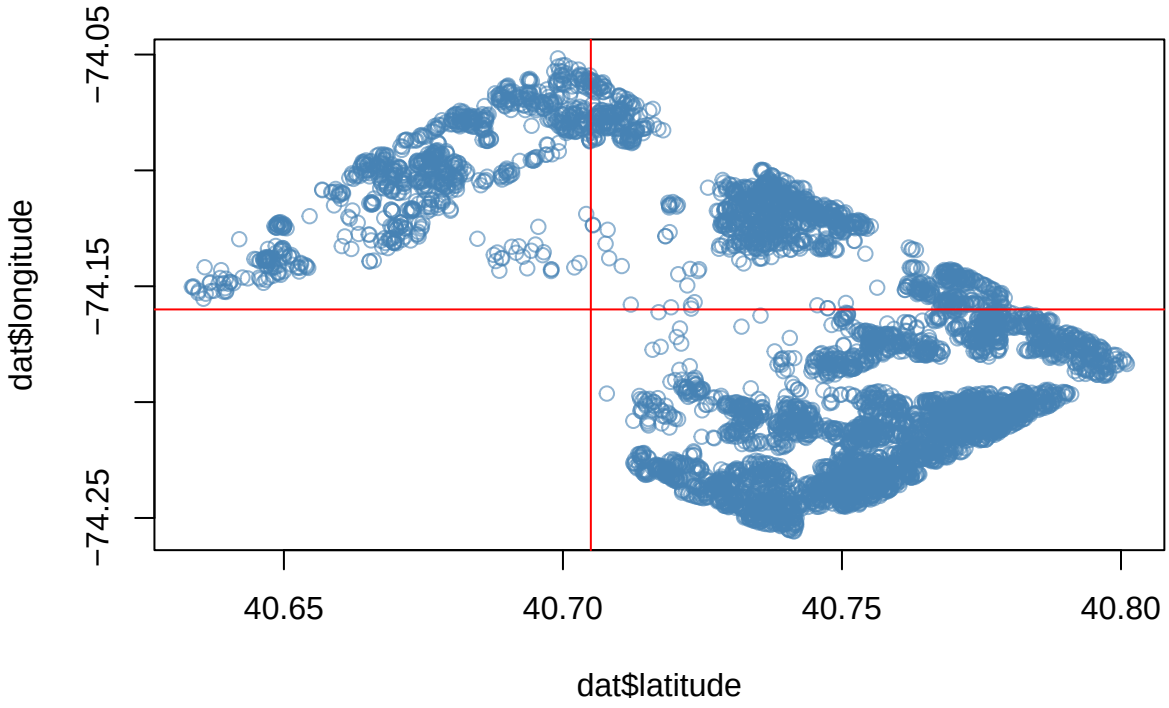
```
kitchens.missing <- sum(is.na(dat$kitchens)); kitchens.missing
```

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

```
quadrant.missing <- sum(is.na(dat$quadrant)); quadrant.missing
```

```
## [1] 32
```

```
## Add new variables
dat$rmdled <- ifelse(is.na(dat$yr_rmdl), "N", "Y")
plot(dat$latitude, dat$longitude, col=adjustcolor("steelblue",0.6))
abline(v=40.705, col="red")
abline(h=-74.16, col="red")
```




```
dat$location[which(dat$latitude >= 40.705 & dat$longitude >= -74.16)] <- "A"
dat$location[which(dat$latitude <= 40.705 & dat$longitude >= -74.16)] <- "S"
dat$location[which(dat$latitude <= 40.705 & dat$longitude <= -74.16)] <- "T"
dat$location[which(dat$latitude >= 40.705 & dat$longitude <= -74.16)] <- "C"
```

1.2 Changing the format of dates(years -> days)

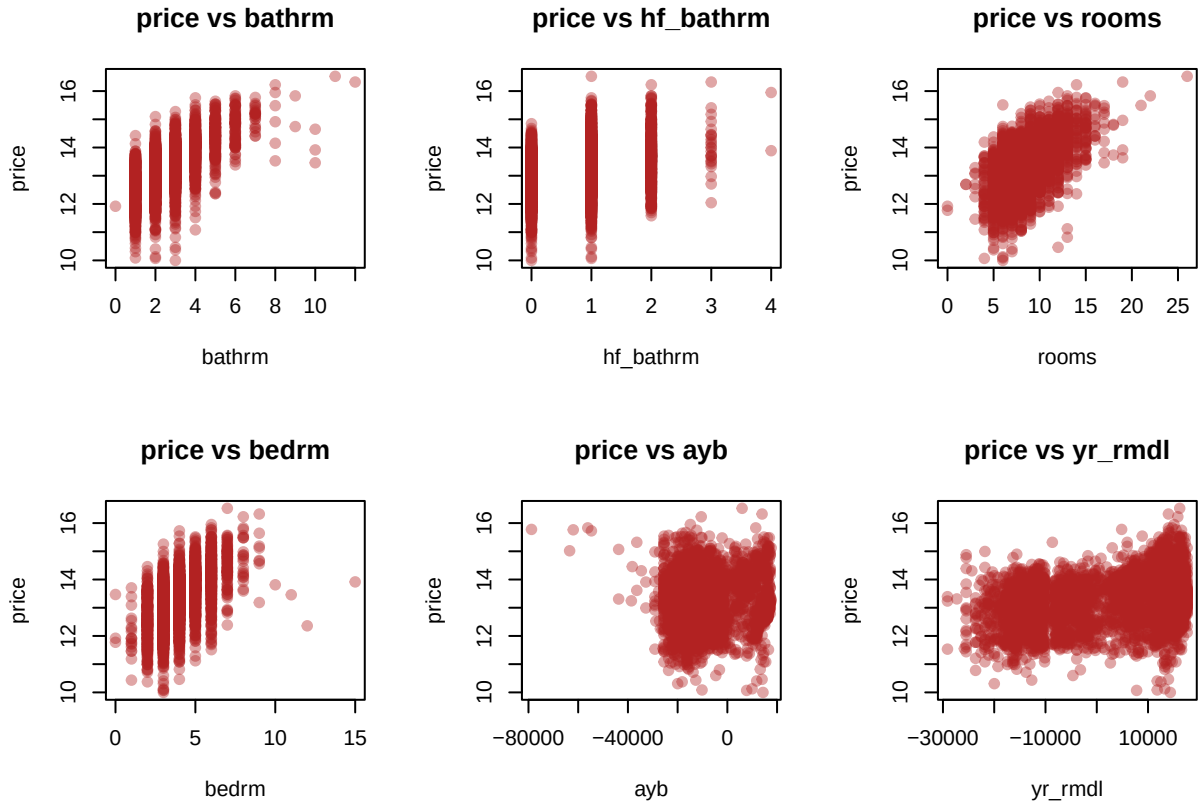
```
dat$ayb <- as.numeric(as.Date(as.character(dat$ayb), format = "%Y"))
dat$yr_rmdl <- as.numeric(as.Date(as.character(dat$yr_rmdl), format = "%Y"))
dat$eyb <- as.numeric(as.Date(as.character(dat$eyb), format = "%Y"))
dat$saledate <- as.numeric(as.Date(dat$saledate))
```

1.3 Missing data handling

```
dat$ayb[which(is.na(dat$ayb))] <- mean(dat$ayb, na.rm = TRUE)
dat$stories[which(is.na(dat$stories))] <- mean(dat$stories, na.rm = TRUE)
dat$kitchens[which(is.na(dat$kitchens))] <- mean(dat$kitchens, na.rm = TRUE)
quadrant.mode <- names(sort(table(dat$quadrant), decreasing = TRUE))[1]
dat$quadrant[which(is.na(dat$quadrant))] <- quadrant.mode
dat$yr_rmdl[which(is.na(dat$yr_rmdl))] <- dat$ayb[is.na(dat$yr_rmdl)]
# Again new variables are added
dat$dsRM <- dat$saledate - dat$yr_rmdl
dat$dsIM <- dat$saledate - dat$eyb
dat$dsBT <- dat$saledate - dat$ayb
```

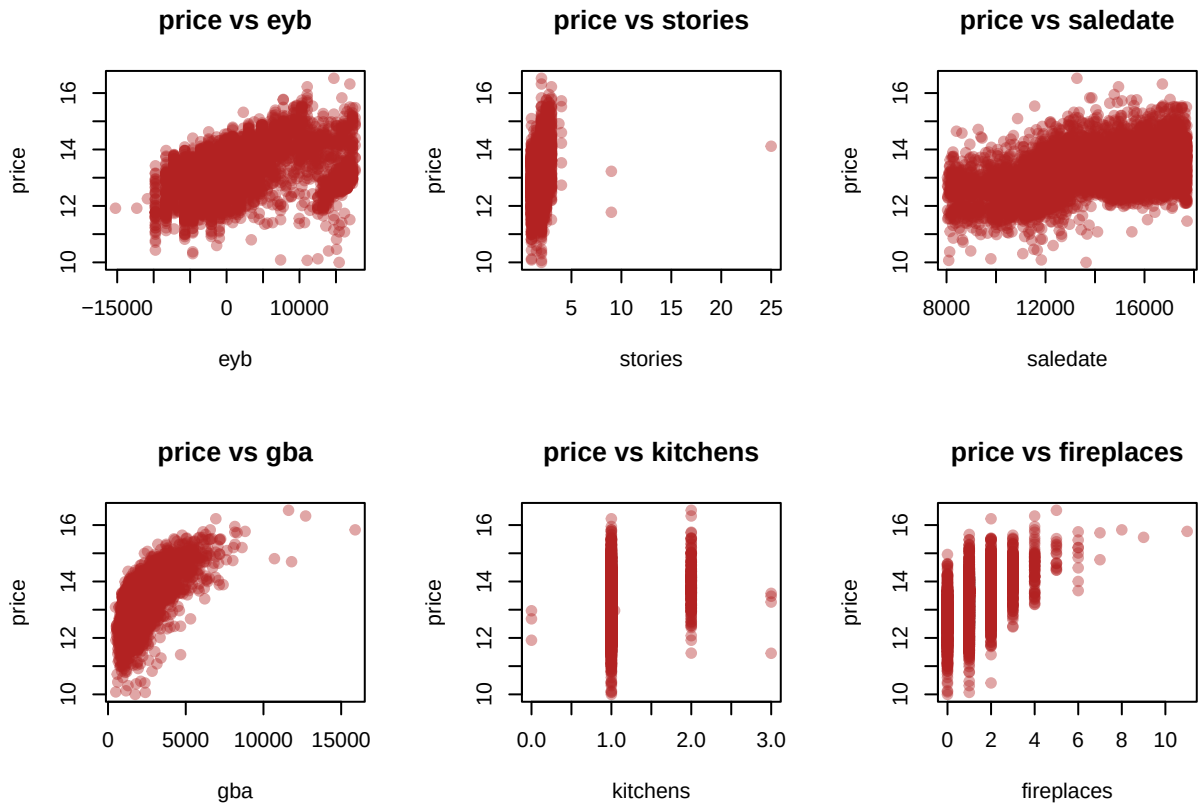
1.4 Transformation and plotting

```
dat$price <- log(dat$price)
dat$landarea <- sqrt(dat$landarea)
par(mfrow = c(2, 3))
plot(dat$bathrm, dat$price, xlab = "bathrm", ylab = "price",
     main = "price vs bathrm", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$hf_bathrm, dat$price, xlab = "hf_bathrm", ylab = "price",
     main = "price vs hf_bathrm", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$rooms, dat$price, xlab = "rooms", ylab = "price",
     main = "price vs rooms", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$bedrm, dat$price, xlab = "bedrm", ylab = "price",
     main = "price vs bedrm", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$ayb, dat$price, xlab = "ayb", ylab = "price",
     main = "price vs ayb", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$yr_rmdl, dat$price, xlab = "yr_rmdl", ylab = "price",
     main = "price vs yr_rmdl", pch=19, col=adjustcolor("firebrick",0.4))
```

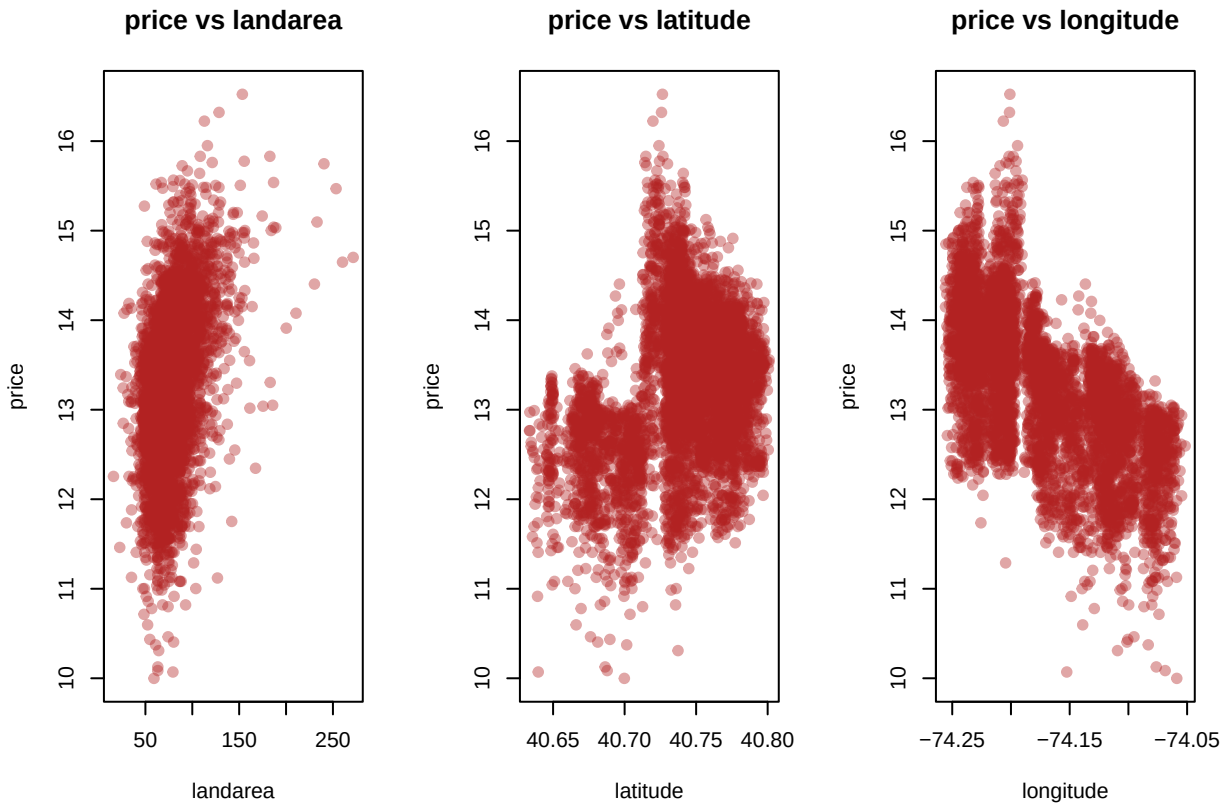



```
par(mfrow = c(2, 3))
plot(dat$eyb, dat$price, xlab = "eyb", ylab = "price",
     main = "price vs eyb", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$stories, dat$price, xlab = "stories", ylab = "price",
     main = "price vs stories", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$saledate, dat$price, xlab = "saledate", ylab = "price",
     main = "price vs saledate", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$gba, dat$price, xlab = "gba", ylab = "price",
```

```
     main = "price vs gba", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$kitchens, dat$price, xlab = "kitchens", ylab = "price",
     main = "price vs kitchens", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$fireplaces, dat$price, xlab = "fireplaces", ylab = "price",
     main = "price vs fireplaces", pch=19, col=adjustcolor("firebrick",0.4))
```




```
par(mfrow = c(1, 3))
plot(dat$landarea, dat$price, xlab = "landarea", ylab = "price",
     main = "price vs landarea", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$latitude, dat$price, xlab = "latitude", ylab = "price",
     main = "price vs latitude", pch=19, col=adjustcolor("firebrick",0.4))
plot(dat$longitude, dat$price, xlab = "longitude", ylab = "price",
     main = "price vs longitude", pch=19, col=adjustcolor("firebrick",0.4))
```



1.5 Extreme value handling

```
## from the plot, notice that there are outliers in bathrm, hf_bathrm, rooms,
##   bedrm, ayb, stories(so unreasonable), gba, kitchens, fireplaces, landarea
table(dat$bathrm, dat$hf_bathrm)
```

```
##
##          0    1    2    3    4
##    0     1    0    0    0    0
##    1   414  494   95    2    0
##    2   713 1134  275    8    0
##    3   524 1120  147    4    0
##    4   152  500   70    4    0
##    5    23  165   35    4    1
##    6     4   49   26    0    0
##    7     0   16    5    2    0
##    8     0    2    3    0    1
##    9     0    0    2    0    0
##    10    1    1    1    0    0
##    11    0    1    0    0    0
##    12    0    0    0    1    0
```

```
dat$bathrm[which(dat$bathrm==11 | dat$bathrm==12)] <- 10
dat$hf_bathrm[which(dat$hf_bathrm==4)] <- 3

table(dat$rooms, dat$bedrm)
```

```
##
##        0   1   2   3   4   5   6   7   8   9  10  11  12  15
##   0    2   0   0   0   0   0   0   0   0   0   0   0   0   0
##   2    0   0   2   0   0   0   0   0   0   0   0   0   0   0
##   3    0   4   0   3   0   0   0   0   0   0   0   0   0   0
##   4    0   9  30   8   4   0   0   0   0   0   0   0   0   0
##   5    0   3 150  85  23   5   0   0   0   0   0   0   0   0
##   6    0   2 116 904 196  23   5   0   0   0   0   0   0   0
##   7    0   1  50 745 433  90   7   1   0   0   0   0   0   0
##   8    0   0   9 403 545 166  30   0   0   0   0   0   0   0
##   9    0   0   5 164 317 161  59  12   1   0   0   0   0   0
##   10   0   0   1  65 194 194  67  14   2   0   0   0   0   0
##   11   0   0   0  17  68  95  51  14   2   2   0   0   0   0
##   12   1   0   0   6  48  78  60  20   2   2   0   0   0   0
##   13   0   0   0   1  10  36  41  12   3   1   0   0   0   0
##   14   0   0   0   0   4  12  17   8   3   0   0   0   1   0
##   15   0   0   0   1   3  12  17   6   4   2   0   0   0   0
##   16   0   0   0   0   0   3   6   1   1   0   0   0   0   0
##   17   0   0   0   0   0   0   3   3   1   0   1   1   0   0
##   18   0   0   0   0   0   1   0   1   1   0   0   0   0   0
##   19   0   0   0   0   0   0   2   0   0   1   0   0   0   1
##   21   0   0   0   0   0   0   0   0   1   0   0   0   0   0
##   22   0   0   0   0   0   0   0   0   1   0   0   0   0   0
##   26   0   0   0   0   0   0   0   1   0   0   0   0   0   0
```

```
dat$rooms[which(dat$rooms > 17)] <- 17
dat$rooms[which(dat$rooms < 3)] <- 3
dat$bedrm[which(dat$bedrm > 9)] <- 9
dat$bedrm[which(dat$bedrm < 1)] <- 1

table(dat$stories, dat$style)
```

```
##
##                    1 Story 1.5 Story Fin 1.5 Story Unfin 2 Story 2.5 Story Fin
##   1                    595            7               0       5             1
##   1.25                  22           50               4       0             0
##   1.5                    0          301              10       7             1
##   1.75                   0           17               0     176             1
##   1.97491661107405       0            0               0       3             1
##   2                      6            5               0    3150            14
##   2.2                    0            0               0       0             1
##   2.25                   0            0               0      65           235
##   2.5                    0            0               0      24           860
##   2.7                    0            0               0       0             1
##   2.75                   0            0               0       0            18
##   3                      0            0               0       7             2
##   3.75                   0            0               0       0             0
##   4                      0            0               0       2             0
##   9                      0            0               0       1             0
##   25                     0            0               0       0             1
##
##                    2.5 Story Unfin 3 Story 4 Story Bi-Level Default Split Foyer
##   1                                0       2       0        1       0          33
##   1.25                             0       0       0        0       0           0
##   1.5                              0       0       0        0       0           0
```

```
##   1.75                          0       0       0       0       0           0
##   1.97491661107405              0       0       0       0       0           0
##   2                             2       4       1       1       0          10
##   2.2                           0       0       0       0       0           0
##   2.25                         58       2       0       0       0           0
##   2.5                          55       2       0       0       1           0
##   2.7                           0       1       0       0       0           0
##   2.75                          1      60       0       0       0           0
##   3                             0     107       0       0       0           0
##   3.75                          0       0       1       0       0           0
##   4                             0       1       3       0       0           0
##   9                             0       0       0       0       1           0
##   25                            0       0       0       0       0           0
##
##                     Split Level
##   1                          28
##   1.25                        0
##   1.5                         4
##   1.75                        0
##   1.97491661107405            0
##   2                          28
##   2.2                         0
##   2.25                        0
##   2.5                         0
##   2.7                         0
##   2.75                        0
##   3                           0
##   3.75                        0
##   4                           0
##   9                           0
##   25                          0
```

```r
dat$stories[which(dat$stories > 3)] <- 3

dat$kitchens[which(dat$kitchens < 1)] <- 1
dat$kitchens[which(dat$kitchens > 2)] <- 2

dat$fireplaces[which(dat$fireplaces > 6)] <- 6
```

2 Levels checking

2.1 Fold1 as testing

```
# Fold1 check
fold1.testing <- dat[which(fold==1), ]
fold1.training <- dat[which(fold!=1), ]
fold1.training$heat <- as.factor(fold1.training$heat)
fold1.training$ac <- as.factor(fold1.training$ac)
fold1.training$style <- as.factor(fold1.training$style)
fold1.training$grade <- as.factor(fold1.training$grade)
fold1.training$cndtn <- as.factor(fold1.training$cndtn)
fold1.training$extwall <- as.factor(fold1.training$extwall)
fold1.training$roof <- as.factor(fold1.training$roof)
fold1.training$intwall <- as.factor(fold1.training$intwall)
fold1.training$nbhd <- as.factor(fold1.training$nbhd)
fold1.training$ward <- as.factor(fold1.training$ward)
fold1.training$quadrant <- as.factor(fold1.training$quadrant)
fold1.training$rmdled <- as.factor(fold1.training$rmdled)
fold1.training$location <- as.factor(fold1.training$location)
fold1.testing$heat <- as.factor(fold1.testing$heat)
fold1.testing$ac <- as.factor(fold1.testing$ac)
fold1.testing$style <- as.factor(fold1.testing$style)
fold1.testing$grade <- as.factor(fold1.testing$grade)
fold1.testing$cndtn <- as.factor(fold1.testing$cndtn)
fold1.testing$extwall <- as.factor(fold1.testing$extwall)
fold1.testing$roof <- as.factor(fold1.testing$roof)
fold1.testing$intwall <- as.factor(fold1.testing$intwall)
fold1.testing$nbhd <- as.factor(fold1.testing$nbhd)
fold1.testing$ward <- as.factor(fold1.testing$ward)
fold1.testing$quadrant <- as.factor(fold1.testing$quadrant)
fold1.testing$rmdled <- as.factor(fold1.testing$rmdled)
fold1.testing$location <- as.factor(fold1.testing$location)
heat.mismatched1 <- setdiff(levels(fold1.testing$heat), levels(fold1.training$heat))
ac.mismatched1 <- setdiff(levels(fold1.testing$ac), levels(fold1.training$ac))
style.mismatched1 <- setdiff(levels(fold1.testing$style), levels(fold1.training$style))
grade.mismatched1 <- setdiff(levels(fold1.testing$grade), levels(fold1.training$grade))
cndtn.mismatched1 <- setdiff(levels(fold1.testing$cndtn), levels(fold1.training$cndtn))
extwall.mismatched1 <- setdiff(levels(fold1.testing$extwall), levels(fold1.training$extwall))
roof.mismatched1 <- setdiff(levels(fold1.testing$roof), levels(fold1.training$roof))
intwall.mismatched1 <- setdiff(levels(fold1.testing$intwall), levels(fold1.training$intwall))
nbhd.mismatched1 <- setdiff(levels(fold1.testing$nbhd), levels(fold1.training$nbhd))
ward.mismatched1 <- setdiff(levels(fold1.testing$ward), levels(fold1.training$ward))
quadrant.mismatched1 <- setdiff(levels(fold1.testing$quadrant), levels(fold1.training$quadrant))
rmdled.mismatched1 <- setdiff(levels(fold1.testing$rmdled), levels(fold1.training$rmdled))
location.mismatched1 <- setdiff(levels(fold1.testing$location), levels(fold1.training$location))
## Only roof and intwall have mismatches in fold1
roof.mismatched1; intwall.mismatched1
```

```
## [1] "Composition Ro"
```

```
## [1] "Terrazo"     "Vinyl Sheet"
```

```
avgprice.Composition1 <- mean(fold1.testing$price[fold1.testing$roof == "Composition Ro"])
roof.levels1 <- levels(fold1.training$roof)
```

```r
avgprices.roof1 <- numeric(length(roof.levels1))
for (i in 1:length(roof.levels1)) {
  avgprices.roof1[i] <- mean(fold1.training$price[fold1.training$roof == roof.levels1[i]])
}; roof.cloestmatch1 <- roof.levels1[which.min(abs(avgprices.roof1 - avgprice.Composition1))]

avgprice.Terrazo1 <- mean(fold1.testing$price[fold1.testing$intwall == "Terrazo"])
avgprice.Vinyl1 <- mean(fold1.testing$price[fold1.testing$intwall == "Vinyl Sheet"])
intwall.levels1 <- levels(fold1.training$intwall)
avgprices.intwall1 <- numeric(length(intwall.levels1))
for (i in 1:length(intwall.levels1)) {
  avgprices.intwall1[i] <- mean(fold1.training$price[fold1.training$intwall == intwall.levels1[i]])
}
Terrazo_intwall.cloestmatch1 <- intwall.levels1[which.min(abs(avgprices.intwall1 - avgprice.Terrazo1))]
Vinyl_intwall.cloestmatch1 <- intwall.levels1[which.min(abs(avgprices.intwall1 - avgprice.Vinyl1))]
## Note they match to the same level, so I assign a single variable to hold the value
Terrazo_intwall.cloestmatch1; Vinyl_intwall.cloestmatch1
```

```
## [1] "Parquet"
```

```
## [1] "Parquet"
```

```r
intwall.cloestmatch1 <- Terrazo_intwall.cloestmatch1
```

2.2 Fold2 as testing

```r
# Fold2 check
fold2.testing <- dat[which(fold==2), ]
fold2.training <- dat[which(fold!=2), ]
fold2.training$heat <- as.factor(fold2.training$heat)
fold2.training$ac <- as.factor(fold2.training$ac)
fold2.training$style <- as.factor(fold2.training$style)
fold2.training$grade <- as.factor(fold2.training$grade)
fold2.training$cndtn <- as.factor(fold2.training$cndtn)
fold2.training$extwall <- as.factor(fold2.training$extwall)
fold2.training$roof <- as.factor(fold2.training$roof)
fold2.training$intwall <- as.factor(fold2.training$intwall)
fold2.training$nbhd <- as.factor(fold2.training$nbhd)
fold2.training$ward <- as.factor(fold2.training$ward)
fold2.training$quadrant <- as.factor(fold2.training$quadrant)
fold2.training$rmdled <- as.factor(fold2.training$rmdled)
fold2.training$location <- as.factor(fold2.training$location)
fold2.testing$heat <- as.factor(fold2.testing$heat)
fold2.testing$ac <- as.factor(fold2.testing$ac)
fold2.testing$style <- as.factor(fold2.testing$style)
fold2.testing$grade <- as.factor(fold2.testing$grade)
fold2.testing$cndtn <- as.factor(fold2.testing$cndtn)
fold2.testing$extwall <- as.factor(fold2.testing$extwall)
fold2.testing$roof <- as.factor(fold2.testing$roof)
fold2.testing$intwall <- as.factor(fold2.testing$intwall)
fold2.testing$nbhd <- as.factor(fold2.testing$nbhd)
fold2.testing$ward <- as.factor(fold2.testing$ward)
fold2.testing$quadrant <- as.factor(fold2.testing$quadrant)
fold2.testing$rmdled <- as.factor(fold2.testing$rmdled)
fold2.testing$location <- as.factor(fold2.testing$location)
```

```
heat.mismatched2 <- setdiff(levels(fold2.testing$heat), levels(fold2.training$heat))
ac.mismatched2 <- setdiff(levels(fold2.testing$ac), levels(fold2.training$ac))
style.mismatched2 <- setdiff(levels(fold2.testing$style), levels(fold2.training$style))
grade.mismatched2 <- setdiff(levels(fold2.testing$grade), levels(fold2.training$grade))
cndtn.mismatched2 <- setdiff(levels(fold2.testing$cndtn), levels(fold2.training$cndtn))
extwall.mismatched2 <- setdiff(levels(fold2.testing$extwall), levels(fold2.training$extwall))
roof.mismatched2 <- setdiff(levels(fold2.testing$roof), levels(fold2.training$roof))
intwall.mismatched2 <- setdiff(levels(fold2.testing$intwall), levels(fold2.training$intwall))
nbhd.mismatched2 <- setdiff(levels(fold2.testing$nbhd), levels(fold2.training$nbhd))
ward.mismatched2 <- setdiff(levels(fold2.testing$ward), levels(fold2.training$ward))
quadrant.mismatched2 <- setdiff(levels(fold2.testing$quadrant), levels(fold2.training$quadrant))
rmdled.mismatched2 <- setdiff(levels(fold2.testing$rmdled), levels(fold2.training$rmdled))
location.mismatched2 <- setdiff(levels(fold2.testing$location), levels(fold2.training$location))
## Only heat has mismatches in fold2
heat.mismatched2
```

```
## [1] "Air Exchng"
```

```
avgprice.AirEx2 <- mean(fold2.testing$price[fold2.testing$heat == "Air Exchng"])
heat.levels2 <- levels(fold2.training$heat)
avgprices.heat2 <- numeric(length(heat.levels2))
for (i in 1:length(heat.levels2)) {
  avgprices.heat2[i] <- mean(fold2.training$price[fold2.training$heat == heat.levels2[i]])
}; heat.cloestmatch2 <- heat.levels2[which.min(abs(avgprices.heat2 - avgprice.AirEx2))]
heat.cloestmatch2
```

```
## [1] "Ht Pump"
```

2.3 Fold3 as testing

```
# Fold3 check
fold3.testing <- dat[which(fold==3), ]
fold3.training <- dat[which(fold!=3), ]
fold3.training$heat <- as.factor(fold3.training$heat)
fold3.training$ac <- as.factor(fold3.training$ac)
fold3.training$style <- as.factor(fold3.training$style)
fold3.training$grade <- as.factor(fold3.training$grade)
fold3.training$cndtn <- as.factor(fold3.training$cndtn)
fold3.training$extwall <- as.factor(fold3.training$extwall)
fold3.training$roof <- as.factor(fold3.training$roof)
fold3.training$intwall <- as.factor(fold3.training$intwall)
fold3.training$nbhd <- as.factor(fold3.training$nbhd)
fold3.training$ward <- as.factor(fold3.training$ward)
fold3.training$quadrant <- as.factor(fold3.training$quadrant)
fold3.training$rmdled <- as.factor(fold3.training$rmdled)
fold3.training$location <- as.factor(fold3.training$location)
fold3.testing$heat <- as.factor(fold3.testing$heat)
fold3.testing$ac <- as.factor(fold3.testing$ac)
fold3.testing$style <- as.factor(fold3.testing$style)
fold3.testing$grade <- as.factor(fold3.testing$grade)
fold3.testing$cndtn <- as.factor(fold3.testing$cndtn)
fold3.testing$extwall <- as.factor(fold3.testing$extwall)
fold3.testing$roof <- as.factor(fold3.testing$roof)
fold3.testing$intwall <- as.factor(fold3.testing$intwall)
```

```
fold3.testing$nbhd <- as.factor(fold3.testing$nbhd)
fold3.testing$ward <- as.factor(fold3.testing$ward)
fold3.testing$quadrant <- as.factor(fold3.testing$quadrant)
fold3.testing$rmdled <- as.factor(fold3.testing$rmdled)
fold3.testing$location <- as.factor(fold3.testing$location)
heat.mismatched3 <- setdiff(levels(fold3.testing$heat), levels(fold3.training$heat))
ac.mismatched3 <- setdiff(levels(fold3.testing$ac), levels(fold3.training$ac))
style.mismatched3 <- setdiff(levels(fold3.testing$style), levels(fold3.training$style))
grade.mismatched3 <- setdiff(levels(fold3.testing$grade), levels(fold3.training$grade))
cndtn.mismatched3 <- setdiff(levels(fold3.testing$cndtn), levels(fold3.training$cndtn))
extwall.mismatched3 <- setdiff(levels(fold3.testing$extwall), levels(fold3.training$extwall))
roof.mismatched3 <- setdiff(levels(fold3.testing$roof), levels(fold3.training$roof))
intwall.mismatched3 <- setdiff(levels(fold3.testing$intwall), levels(fold3.training$intwall))
nbhd.mismatched3 <- setdiff(levels(fold3.testing$nbhd), levels(fold3.training$nbhd))
ward.mismatched3 <- setdiff(levels(fold3.testing$ward), levels(fold3.training$ward))
quadrant.mismatched3 <- setdiff(levels(fold3.testing$quadrant), levels(fold3.training$quadrant))
rmdled.mismatched3 <- setdiff(levels(fold3.testing$rmdled), levels(fold3.training$rmdled))
location.mismatched3 <- setdiff(levels(fold3.testing$location), levels(fold3.training$location))
## Only style and roof have mismatches in fold3
style.mismatched3; roof.mismatched3
```

```
## [1] "Default"
```

```
## [1] "Metal- Pre"
```

```
avgprice.Default3 <- mean(fold3.testing$price[fold3.testing$style == "Default"])
style.levels3 <- levels(fold3.training$style)
avgprices.style3 <- numeric(length(style.levels3))
for (i in 1:length(style.levels3)) {
  avgprices.style3[i] <- mean(fold3.training$price[fold3.training$style == style.levels3[i]])
}; style.cloestmatch3 <- style.levels3[which.min(abs(avgprices.style3 - avgprice.Default3))]
style.cloestmatch3
```

```
## [1] "2 Story"
```

```
avgprice.Metal3 <- mean(fold3.testing$price[fold3.testing$roof == "Metal- Pre"])
roof.levels3 <- levels(fold3.training$roof)
avgprices.roof3 <- numeric(length(roof.levels3))
for (i in 1:length(roof.levels3)) {
  avgprices.roof3[i] <- mean(fold3.training$price[fold3.training$roof == roof.levels3[i]])
}; roof.cloestmatch3 <- roof.levels3[which.min(abs(avgprices.roof3 - avgprice.Metal3))]
roof.cloestmatch3
```

```
## [1] "Composition Ro"
```

2.4 Fold4 as testing

```
# Fold4 check
fold4.testing <- dat[which(fold==4), ]
fold4.training <- dat[which(fold!=4), ]
fold4.training$heat <- as.factor(fold4.training$heat)
fold4.training$ac <- as.factor(fold4.training$ac)
fold4.training$style <- as.factor(fold4.training$style)
fold4.training$grade <- as.factor(fold4.training$grade)
fold4.training$cndtn <- as.factor(fold4.training$cndtn)
```

```r
fold4.training$extwall <- as.factor(fold4.training$extwall)
fold4.training$roof <- as.factor(fold4.training$roof)
fold4.training$intwall <- as.factor(fold4.training$intwall)
fold4.training$nbhd <- as.factor(fold4.training$nbhd)
fold4.training$ward <- as.factor(fold4.training$ward)
fold4.training$quadrant <- as.factor(fold4.training$quadrant)
fold4.training$rmdled <- as.factor(fold4.training$rmdled)
fold4.training$location <- as.factor(fold4.training$location)
fold4.testing$heat <- as.factor(fold4.testing$heat)
fold4.testing$ac <- as.factor(fold4.testing$ac)
fold4.testing$style <- as.factor(fold4.testing$style)
fold4.testing$grade <- as.factor(fold4.testing$grade)
fold4.testing$cndtn <- as.factor(fold4.testing$cndtn)
fold4.testing$extwall <- as.factor(fold4.testing$extwall)
fold4.testing$roof <- as.factor(fold4.testing$roof)
fold4.testing$intwall <- as.factor(fold4.testing$intwall)
fold4.testing$nbhd <- as.factor(fold4.testing$nbhd)
fold4.testing$ward <- as.factor(fold4.testing$ward)
fold4.testing$quadrant <- as.factor(fold4.testing$quadrant)
fold4.testing$rmdled <- as.factor(fold4.testing$rmdled)
fold4.testing$location <- as.factor(fold4.testing$location)
heat.mismatched4 <- setdiff(levels(fold4.testing$heat), levels(fold4.training$heat))
ac.mismatched4 <- setdiff(levels(fold4.testing$ac), levels(fold4.training$ac))
style.mismatched4 <- setdiff(levels(fold4.testing$style), levels(fold4.training$style))
grade.mismatched4 <- setdiff(levels(fold4.testing$grade), levels(fold4.training$grade))
cndtn.mismatched4 <- setdiff(levels(fold4.testing$cndtn), levels(fold4.training$cndtn))
extwall.mismatched4 <- setdiff(levels(fold4.testing$extwall), levels(fold4.training$extwall))
roof.mismatched4 <- setdiff(levels(fold4.testing$roof), levels(fold4.training$roof))
intwall.mismatched4 <- setdiff(levels(fold4.testing$intwall), levels(fold4.training$intwall))
nbhd.mismatched4 <- setdiff(levels(fold4.testing$nbhd), levels(fold4.training$nbhd))
ward.mismatched4 <- setdiff(levels(fold4.testing$ward), levels(fold4.training$ward))
quadrant.mismatched4 <- setdiff(levels(fold4.testing$quadrant), levels(fold4.training$quadrant))
rmdled.mismatched4 <- setdiff(levels(fold4.testing$rmdled), levels(fold4.training$rmdled))
location.mismatched4 <- setdiff(levels(fold4.testing$location), levels(fold4.training$location))
## Only extwall and intwall have mismatches in fold4
extwall.mismatched4; intwall.mismatched4
```

```
## [1] "Stucco Block"
```

```
## [1] "Parquet"
```

```r
avgprice.StuBl4 <- mean(fold4.testing$price[fold4.testing$extwall == "Stucco Block"])
extwall.levels4 <- levels(fold4.training$extwall)
avgprices.extwall4 <- numeric(length(extwall.levels4))
for (i in 1:length(extwall.levels4)) {
  avgprices.extwall4[i] <- mean(fold4.training$price[fold4.training$extwall == extwall.levels4[i]])
}; extwall.cloestmatch4 <- extwall.levels4[which.min(abs(avgprices.extwall4 - avgprice.StuBl4))]
extwall.cloestmatch4
```

```
## [1] "Aluminum"
```

```r
avgprice.Parquet4 <- mean(fold4.testing$price[fold4.testing$intwall == "Parquet"])
intwall.levels4 <- levels(fold4.training$intwall)
avgprices.intwall4 <- numeric(length(intwall.levels4))
for (i in 1:length(intwall.levels4)) {
  avgprices.intwall4[i] <- mean(fold4.training$price[fold4.training$intwall == intwall.levels4[i]])
```

```
}; intwall.cloestmatch4 <- intwall.levels4[which.min(abs(avgprices.intwall4 - avgprice.Parquet4))]
intwall.cloestmatch4
```

```
## [1] "Default"
```

2.5 Fold5 as testing

```
# Fold5 check
fold5.testing <- dat[which(fold==5), ]
fold5.training <- dat[which(fold!=5), ]
fold5.training$heat <- as.factor(fold5.training$heat)
fold5.training$ac <- as.factor(fold5.training$ac)
fold5.training$style <- as.factor(fold5.training$style)
fold5.training$grade <- as.factor(fold5.training$grade)
fold5.training$cndtn <- as.factor(fold5.training$cndtn)
fold5.training$extwall <- as.factor(fold5.training$extwall)
fold5.training$roof <- as.factor(fold5.training$roof)
fold5.training$intwall <- as.factor(fold5.training$intwall)
fold5.training$nbhd <- as.factor(fold5.training$nbhd)
fold5.training$ward <- as.factor(fold5.training$ward)
fold5.training$quadrant <- as.factor(fold5.training$quadrant)
fold5.training$rmdled <- as.factor(fold5.training$rmdled)
fold5.training$location <- as.factor(fold5.training$location)
fold5.testing$heat <- as.factor(fold5.testing$heat)
fold5.testing$ac <- as.factor(fold5.testing$ac)
fold5.testing$style <- as.factor(fold5.testing$style)
fold5.testing$grade <- as.factor(fold5.testing$grade)
fold5.testing$cndtn <- as.factor(fold5.testing$cndtn)
fold5.testing$extwall <- as.factor(fold5.testing$extwall)
fold5.testing$roof <- as.factor(fold5.testing$roof)
fold5.testing$intwall <- as.factor(fold5.testing$intwall)
fold5.testing$nbhd <- as.factor(fold5.testing$nbhd)
fold5.testing$ward <- as.factor(fold5.testing$ward)
fold5.testing$quadrant <- as.factor(fold5.testing$quadrant)
fold5.testing$rmdled <- as.factor(fold5.testing$rmdled)
fold5.testing$location <- as.factor(fold5.testing$location)
heat.mismatched5 <- setdiff(levels(fold5.testing$heat), levels(fold5.training$heat))
ac.mismatched5 <- setdiff(levels(fold5.testing$ac), levels(fold5.training$ac))
style.mismatched5 <- setdiff(levels(fold5.testing$style), levels(fold5.training$style))
grade.mismatched5 <- setdiff(levels(fold5.testing$grade), levels(fold5.training$grade))
cndtn.mismatched5 <- setdiff(levels(fold5.testing$cndtn), levels(fold5.training$cndtn))
extwall.mismatched5 <- setdiff(levels(fold5.testing$extwall), levels(fold5.training$extwall))
roof.mismatched5 <- setdiff(levels(fold5.testing$roof), levels(fold5.training$roof))
intwall.mismatched5 <- setdiff(levels(fold5.testing$intwall), levels(fold5.training$intwall))
nbhd.mismatched5 <- setdiff(levels(fold5.testing$nbhd), levels(fold5.training$nbhd))
ward.mismatched5 <- setdiff(levels(fold5.testing$ward), levels(fold5.training$ward))
quadrant.mismatched5 <- setdiff(levels(fold5.testing$quadrant), levels(fold5.training$quadrant))
rmdled.mismatched5 <- setdiff(levels(fold5.testing$rmdled), levels(fold5.training$rmdled))
location.mismatched5 <- setdiff(levels(fold5.testing$location), levels(fold5.training$location))
## Only heat has mismatches in fold5
heat.mismatched5
```

```
## [1] "Evp Cool"
```

```r
avgprice.EvpC5 <- mean(fold5.testing$price[fold5.testing$heat == "Evp Cool"])
heat.levels5 <- levels(fold5.training$heat)
avgprices.heat5 <- numeric(length(heat.levels5))
for (i in 1:length(heat.levels5)) {
  avgprices.heat5[i] <- mean(fold5.training$price[fold5.training$heat == heat.levels5[i]])
}; heat.cloestmatch5 <- heat.levels5[which.min(abs(avgprices.heat5 - avgprice.EvpC5))]
heat.cloestmatch5
```

```
## [1] "Elec Base Brd"
```

3 Five-Fold CV functions

3.1 For Smoothing

```
CV.SM <- function(formula) {
  RMLSEs <- numeric(5)
  for(i in 1:5) {
    testing <- dat[which(fold==i), ]
    training <- dat[which(fold!=i), ]
    if (i == 1) {
      testing$roof[testing$roof=="Composition Ro"] <- "Comp Shingle"
      testing$intwall[testing$intwall=="Terrazo"] <- "Parquet"
      testing$intwall[testing$intwall=="Vinyl Sheet"] <- "Parquet"
    } else if (i == 2) {
      testing$heat[testing$heat=="Air Exchng"] <- "Ht Pump"
      testing$nbhd[testing$nbhd=="C2"] <- "C3"
    } else if (i == 3) {
      testing$style[testing$style=="Default"] <- "2 Story"
      testing$roof[testing$roof=="Metal- Pre"] <- "Composition Ro"
    } else if (i == 4) {
      testing$extwall[testing$extwall=="Stucco Block"] <- "Aluminum"
      testing$intwall[testing$intwall=="Parquet"] <- "Default"
      testing$nbhd[testing$nbhd=="C2"] <- "C3"
      testing$nbhd[testing$nbhd=="F4"] <- "F5"
    } else {
      testing$heat[testing$heat=="Evp Cool"] <- "Elec Base Brd"
    }
    model <- mgcv::gam(formula, data=training)
    pred <- predict(model, newdata=testing)
    RMLSEs[i] <- sqrt(mean((testing$price - pred)^2))
  }
  mean(RMLSEs)
}
```

3.2 For Random Forest

```
CV.RF <- function(formula, mtry=10, nodesize=1, ntree=500) {
  RMLSEs <- numeric(5)
  for (i in 1:5) {
    testing <- dat[which(fold==i), ]
    training <- dat[which(fold!=i), ]
    model <- randomForest::randomForest(formula, mtry=mtry, ntree=ntree,
                                        nodesize=nodesize, data=training)
    pred <- predict(model, newdata=testing)
    RMLSEs[i] <- sqrt(mean((testing$price - pred)^2))
  }
  mean(RMLSEs)
}
```

3.3 For Boosting

```
CV.BT <- function(formula, shrinkage=0.1, n.trees=100, interaction.depth=1) {
  RMLSEs <- numeric(5)
```

```
  for(i in 1:5) {
    testing <- dat[which(fold==i), ]
    training <- dat[which(fold!=i), ]
    training$heat <- as.factor(training$heat)
    training$ac <- as.factor(training$ac)
    training$style <- as.factor(training$style)
    training$grade <- as.factor(training$grade)
    training$cndtn <- as.factor(training$cndtn)
    training$extwall <- as.factor(training$extwall)
    training$roof <- as.factor(training$roof)
    training$intwall <- as.factor(training$intwall)
    training$nbhd <- as.factor(training$nbhd)
    training$ward <- as.factor(training$ward)
    training$quadrant <- as.factor(training$quadrant)
    training$rmdled <- as.factor(training$rmdled)
    training$location <- as.factor(training$location)
    testing$heat <- as.factor(testing$heat)
    testing$ac <- as.factor(testing$ac)
    testing$style <- as.factor(testing$style)
    testing$grade <- as.factor(testing$grade)
    testing$cndtn <- as.factor(testing$cndtn)
    testing$extwall <- as.factor(testing$extwall)
    testing$roof <- as.factor(testing$roof)
    testing$intwall <- as.factor(testing$intwall)
    testing$nbhd <- as.factor(testing$nbhd)
    testing$ward <- as.factor(testing$ward)
    testing$quadrant <- as.factor(testing$quadrant)
    testing$rmdled <- as.factor(testing$rmdled)
    testing$location <- as.factor(testing$location)
    model <- gbm::gbm(formula, data=training, distribution="gaussian",
                      shrinkage=shrinkage, n.trees=n.trees,
                      interaction.depth=interaction.depth,
                      bag.fraction=1)
    pred <- predict(model, newdata=testing)
    RMLSEs[i] <- sqrt(mean((testing$price - pred)^2))
  }
  mean(RMLSEs)
}
```

4 Model building

4.1 Smoothing Models

4.1.1 Fit a full model without smoothing terms

```
smoothing1.formula <- formula(price ~ bathrm + rooms + ayb + yr_rmdl + eyb +
                                      stories + saledate + gba + landarea +
                                      latitude + longitude + dsRM + dsIM + dsBT
                                      + hf_bathrm + bedrm + kitchens +
                                      fireplaces +
                                      heat + ac + style + grade + cndtn +
                                      extwall + roof + intwall + nbhd + ward +
                                      quadrant + rmdled + location)
SCV.result1 <- CV.SM(smoothing1.formula); SCV.result1
```

```
## [1] 0.250923
```

4.1.2 Next fit a full model with smoothing terms

```
smoothing2.formula <- formula(price ~ s(bathrm) + s(rooms) + s(ayb) + s(yr_rmdl)
                                      + s(eyb) + stories + s(saledate) +
                                      s(gba) + s(landarea) + s(latitude) +
                                      s(longitude) + s(dsRM) + s(dsIM) + s(dsBT)
                                      + hf_bathrm + bedrm + kitchens +
                                      fireplaces +
                                      heat + ac + style + grade +
                                      cndtn + extwall + roof + intwall + nbhd +
                                      ward + quadrant + rmdled + location)
SCV.result2 <- CV.SM(smoothing2.formula); SCV.result2
```

```
## [1] 0.1982023
```

```
smoothing2 <- mgcv::gam(smoothing2.formula, data=dat)
summary(smoothing2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## price ~ s(bathrm) + s(rooms) + s(ayb) + s(yr_rmdl) + s(eyb) +
##     stories + s(saledate) + s(gba) + s(landarea) + s(latitude) +
##     s(longitude) + s(dsRM) + s(dsIM) + s(dsBT) + hf_bathrm +
##     bedrm + kitchens + fireplaces + heat + ac + style + grade +
##     cndtn + extwall + roof + intwall + nbhd + ward + quadrant +
##     rmdled + location
##
## Parametric coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           12.780055   0.299431  42.681  < 2e-16 ***
## stories               -0.003894   0.018712  -0.208 0.835149
## hf_bathrm              0.028971   0.004694   6.171 7.24e-10 ***
## bedrm                  0.016740   0.003650   4.586 4.62e-06 ***
## kitchens               0.006756   0.013125   0.515 0.606777
```

```
## fireplaces             0.026439   0.004084   6.474 1.03e-10 ***
## heatAir Exchng        -0.159235   0.248391  -0.641 0.521505
## heatElec Base Brd      0.071698   0.161830   0.443 0.657749
## heatEvp Cool          -0.787581   0.235256  -3.348 0.000820 ***
## heatForced Air         0.178377   0.135622   1.315 0.188478
## heatGravity Furnac     0.340781   0.168644   2.021 0.043356 *
## heatHot Water Rad      0.165023   0.135656   1.216 0.223849
## heatHt Pump            0.198565   0.137810   1.441 0.149680
## heatNo Data            0.511694   0.211324   2.421 0.015493 *
## heatWall Furnace       0.287635   0.169325   1.699 0.089426 .
## heatWarm Cool          0.172988   0.135628   1.275 0.202198
## heatWater Base Brd     0.173499   0.153772   1.128 0.259246
## acY                    0.035842   0.009581   3.741 0.000185 ***
## style1.5 Story Fin     0.013685   0.015884   0.862 0.388963
## style1.5 Story Unfin  -0.053460   0.052495  -1.018 0.308534
## style2 Story          -0.004376   0.020659  -0.212 0.832249
## style2.5 Story Fin     0.023265   0.028723   0.810 0.417986
## style2.5 Story Unfin   0.013369   0.032481   0.412 0.680645
## style3 Story           0.017563   0.038463   0.457 0.647950
## style4 Story          -0.129948   0.096158  -1.351 0.176621
## styleBi-Level          0.095846   0.135334   0.708 0.478839
## styleDefault           0.110842   0.139390   0.795 0.426536
## styleSplit Foyer       0.124234   0.032297   3.847 0.000121 ***
## styleSplit Level       0.020878   0.027771   0.752 0.452199
## gradeAverage          -0.050128   0.011065  -4.530 6.01e-06 ***
## gradeExcellent         0.138265   0.018417   7.508 6.94e-14 ***
## gradeExceptional-A     0.251767   0.031932   7.885 3.75e-15 ***
## gradeExceptional-B     0.480200   0.039320  12.213  < 2e-16 ***
## gradeExceptional-C     0.713828   0.086451   8.257  < 2e-16 ***
## gradeExceptional-D     0.839929   0.076186  11.025  < 2e-16 ***
## gradeFair Quality     -0.073313   0.054851  -1.337 0.181410
## gradeGood Quality      0.034706   0.011590   2.994 0.002762 **
## gradeLow Quality      -0.760539   0.193777  -3.925 8.78e-05 ***
## gradeSuperior          0.234324   0.022639  10.350  < 2e-16 ***
## gradeVery Good         0.063921   0.013886   4.603 4.25e-06 ***
## cndtnExcellent         0.236256   0.027290   8.657  < 2e-16 ***
## cndtnFair             -0.127440   0.037417  -3.406 0.000664 ***
## cndtnGood              0.078679   0.006651  11.829  < 2e-16 ***
## cndtnPoor             -0.232371   0.080109  -2.901 0.003737 **
## cndtnVery Good         0.200260   0.010886  18.397  < 2e-16 ***
## extwallBrick Veneer    0.015429   0.031609   0.488 0.625471
## extwallBrick/Siding   -0.009014   0.021555  -0.418 0.675819
## extwallBrick/Stone    -0.006543   0.031737  -0.206 0.836663
## extwallBrick/Stucco   -0.007067   0.028759  -0.246 0.805894
## extwallCommon Brick    0.008502   0.020330   0.418 0.675807
## extwallConcrete       -0.028876   0.080174  -0.360 0.718739
## extwallConcrete Block -0.123184   0.098859  -1.246 0.212791
## extwallDefault        -0.019025   0.135977  -0.140 0.888732
## extwallFace Brick      0.022274   0.059609   0.374 0.708665
## extwallHardboard      -0.121107   0.053332  -2.271 0.023196 *
## extwallMetal Siding    0.085202   0.087894   0.969 0.332401
## extwallShingle        -0.001246   0.024207  -0.051 0.958936
## extwallStone           0.020499   0.029369   0.698 0.485220
## extwallStone Veneer    0.021080   0.054638   0.386 0.699643
```

```
## extwallStone/Siding    0.007398   0.030509   0.242 0.808413
## extwallStone/Stucco   -0.024864   0.037785  -0.658 0.510534
## extwallStucco          0.023201   0.021665   1.071 0.284246
## extwallStucco Block    0.032244   0.112687   0.286 0.774782
## extwallVinyl Siding   -0.021861   0.020575  -1.063 0.288046
## extwallWood Siding     0.006502   0.020792   0.313 0.754516
## roofClay Tile          0.068438   0.026708   2.562 0.010419 *
## roofComp Shingle       0.020115   0.014554   1.382 0.167009
## roofComposition Ro     0.118092   0.195854   0.603 0.546559
## roofConcrete Tile      0.099252   0.157947   0.628 0.529774
## roofMetal- Cpr         0.249634   0.116544   2.142 0.032237 *
## roofMetal- Pre        -0.066664   0.192413  -0.346 0.729008
## roofMetal- Sms        -0.004929   0.023290  -0.212 0.832403
## roofNeopren            0.147448   0.083718   1.761 0.078250 .
## roofShake              0.005941   0.023555   0.252 0.800887
## roofShingle            0.058054   0.028120   2.065 0.039014 *
## roofSlate              0.026003   0.015401   1.688 0.091377 .
## roofTypical           -0.102028   0.079592  -1.282 0.199933
## intwallCeramic Tile   -0.035375   0.124752  -0.284 0.776756
## intwallDefault         0.061700   0.120905   0.510 0.609847
## intwallHardwood        0.060064   0.018286   3.285 0.001027 **
## intwallHardwood/Carp   0.033255   0.019124   1.739 0.082102 .
## intwallLt Concrete     0.312374   0.085242   3.665 0.000250 ***
## intwallParquet         0.032476   0.191069   0.170 0.865041
## intwallTerrazo         0.013119   0.194327   0.068 0.946179
## intwallVinyl Sheet    -0.004059   0.191221  -0.021 0.983065
## intwallWood Floor      0.011949   0.022873   0.522 0.601404
## nbhdA2                 0.082982   0.069089   1.201 0.229769
## nbhdA3                -0.083471   0.062685  -1.332 0.183047
## nbhdA4                -0.384361   0.074966  -5.127 3.04e-07 ***
## nbhdA5                 0.170897   0.079443   2.151 0.031502 *
## nbhdA6                -0.200914   0.129819  -1.548 0.121761
## nbhdA7                -0.064136   0.032902  -1.949 0.051304 .
## nbhdA8                -0.110110   0.106457  -1.034 0.301035
## nbhdA9                 0.197824   0.257472   0.768 0.442321
## nbhdB1                 0.212578   0.270569   0.786 0.432092
## nbhdB2                 0.079898   0.064050   1.247 0.212292
## nbhdB3                -0.090391   0.059022  -1.531 0.125706
## nbhdB4                 0.312071   0.072541   4.302 1.72e-05 ***
## nbhdB5                 0.057936   0.051284   1.130 0.258640
## nbhdB6                -0.273726   0.267491  -1.023 0.306203
## nbhdB7                -0.127467   0.068662  -1.856 0.063442 .
## nbhdB8                 0.082708   0.034243   2.415 0.015751 *
## nbhdB9                -0.127335   0.055877  -2.279 0.022713 *
## nbhdC1                -0.074399   0.137089  -0.543 0.587353
## nbhdC2                 0.220720   0.324479   0.680 0.496388
## nbhdC3                 0.147797   0.067059   2.204 0.027564 *
## nbhdC4                -0.096562   0.054788  -1.762 0.078042 .
## nbhdC5                -0.424345   0.136999  -3.097 0.001961 **
## nbhdC6                 0.182363   0.098446   1.852 0.064017 .
## nbhdC7                 0.345235   0.104809   3.294 0.000994 ***
## nbhdC8                 0.356857   0.243298   1.467 0.142499
## nbhdC9                 0.240948   0.132715   1.816 0.069496 .
## nbhdD1                -0.024126   0.066922  -0.361 0.718478
```

```
## nbhdD2                0.043823   0.064488   0.680 0.496819
## nbhdD3                0.290987   0.239194   1.217 0.223832
## nbhdD4                0.075991   0.079737   0.953 0.340618
## nbhdD5                0.225918   0.283374   0.797 0.425343
## nbhdD6               -0.001178   0.059231  -0.020 0.984135
## nbhdD7               -0.256227   0.057608  -4.448 8.84e-06 ***
## nbhdD8                0.243672   0.080090   3.042 0.002357 **
## nbhdD9               -0.209072   0.107228  -1.950 0.051250 .
## nbhdE1                0.172690   0.254722   0.678 0.497828
## nbhdE2                0.176050   0.070095   2.512 0.012045 *
## nbhdE3                0.257739   0.077806   3.313 0.000930 ***
## nbhdE4               -0.167206   0.268336  -0.623 0.533229
## nbhdE5                0.246742   0.217458   1.135 0.256564
## nbhdE6                0.127893   0.081415   1.571 0.116267
## nbhdE7               -0.072735   0.043995  -1.653 0.098334 .
## nbhdE8                0.019461   0.062599   0.311 0.755906
## nbhdE9               -0.117669   0.077107  -1.526 0.127055
## nbhdF1                0.029329   0.037669   0.779 0.436237
## nbhdF2                0.132322   0.075997   1.741 0.081711 .
## nbhdF3               -0.065226   0.068331  -0.955 0.339843
## nbhdF4                0.206461   0.223856   0.922 0.356412
## nbhdF5                0.116520   0.069429   1.678 0.093348 .
## nbhdF6                0.186140   0.076868   2.422 0.015485 *
## nbhdF7                0.328009   0.078431   4.182 2.93e-05 ***
## nbhdF8               -0.244076   0.109147  -2.236 0.025376 *
## wardWard 2            0.324613   0.133391   2.434 0.014982 *
## wardWard 3            0.003220   0.246002   0.013 0.989557
## wardWard 4           -0.030394   0.245483  -0.124 0.901468
## wardWard 5           -0.061911   0.229751  -0.269 0.787577
## wardWard 6            0.275379   0.184608   1.492 0.135834
## wardWard 7           -0.437478   0.226640  -1.930 0.053622 .
## wardWard 8           -0.575839   0.219609  -2.622 0.008762 **
## quadrantNW            0.057471   0.030589   1.879 0.060321 .
## quadrantSE            0.063120   0.043110   1.464 0.143206
## quadrantSW            0.010621   0.084812   0.125 0.900348
## rmdledY              -0.017172   0.016788  -1.023 0.306383
## locationC             0.047738   0.048205   0.990 0.322061
## locationS            -0.035580   0.033202  -1.072 0.283943
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df       F  p-value
## s(bathrm)    2.600  3.356  31.405  < 2e-16 ***
## s(rooms)     5.755  6.846   3.169  0.00274 **
## s(ayb)       3.994  5.148   8.003  < 2e-16 ***
## s(yr_rmdl)   5.689  6.924   4.385 6.02e-05 ***
## s(eyb)       5.879  6.922   1.858  0.05865 .
## s(saledate)  8.561  8.830 355.969  < 2e-16 ***
## s(gba)       8.316  8.869  40.675  < 2e-16 ***
## s(landarea)  1.000  1.000 305.198  < 2e-16 ***
## s(latitude)  7.987  8.776   4.958 5.58e-06 ***
## s(longitude) 8.326  8.882   4.474 1.39e-05 ***
## s(dsRM)      8.463  8.522  31.755  < 2e-16 ***
```

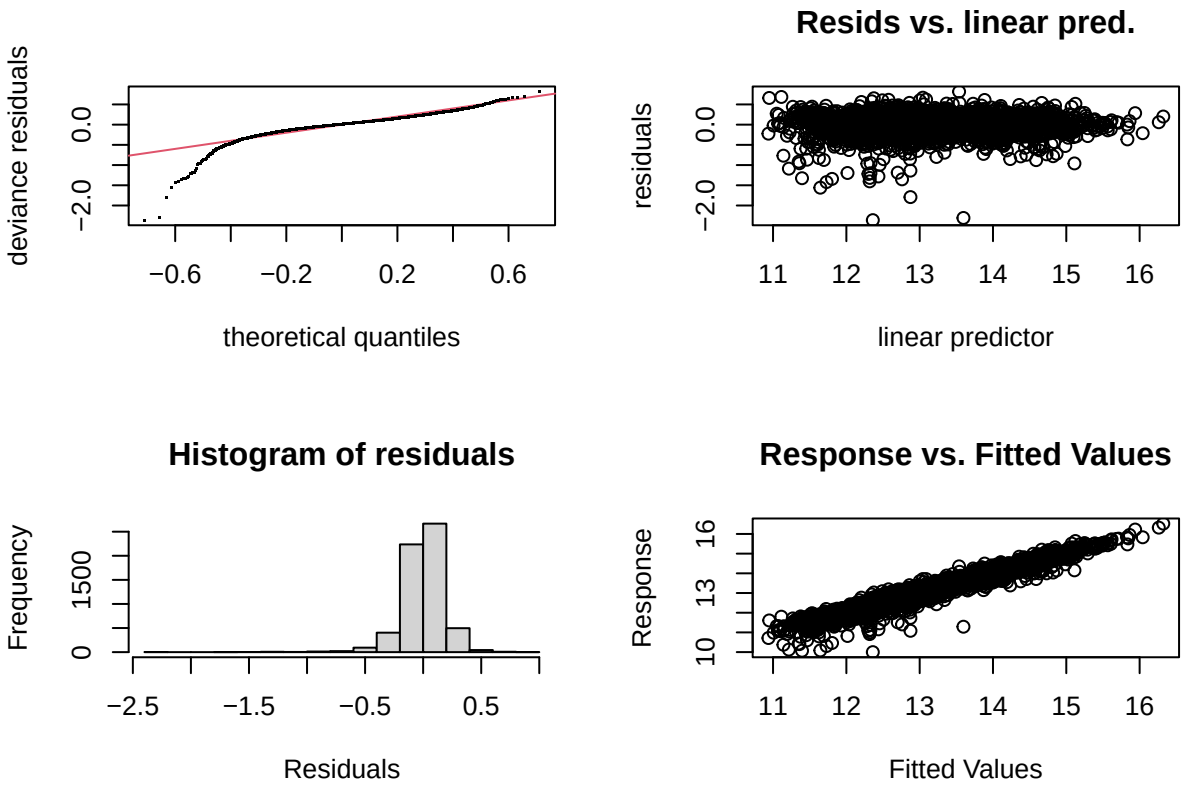
```
## s(dsIM)      8.342  8.521   7.510  < 2e-16 ***
## s(dsBT)      8.024  8.349   8.660  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 263/268
## R-sq.(adj) =  0.944   Deviance explained = 94.7%
## GCV = 0.037043  Scale est. = 0.035611  n = 6000
```

4.1.3 Then fit a reduced model

```
## stories, kitchens, extwall, and rmdled are removed
smoothing3.formula <- formula(price ~ s(bathrm) + s(rooms) + s(ayb) + s(yr_rmdl)
                                    + s(eyb) + s(saledate) + s(gba) +
                                    s(landarea) + s(latitude) + s(longitude) +
                                    s(dsRM) + s(dsIM) + s(dsBT) + hf_bathrm +
                                    bedrm + fireplaces +
                                    heat + ac + style + grade +
                                    cndtn + roof + intwall + nbhd +
                                    ward + quadrant + location)
SCV.result3 <- CV.SM(smoothing3.formula); SCV.result3
```

```
## [1] 0.1974723
```

```
smoothing3 <- mgcv::gam(smoothing3.formula, data=dat)
gam.check(smoothing3)
```



```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 28 iterations.
## The RMS GCV score gradient at convergence was 2.125314e-08 .
## The Hessian was positive definite.
## Model rank =  240 / 245
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                k'  edf k-index p-value
## s(bathrm)    9.00 2.58    1.02   0.935
## s(rooms)     9.00 5.78    1.01   0.715
## s(ayb)       9.00 3.99    0.98   0.055 .
## s(yr_rmdl)   9.00 5.77    0.98   0.030 *
## s(eyb)       9.00 5.71    0.99   0.225
## s(saledate)  9.00 8.56    1.00   0.455
## s(gba)       9.00 8.33    1.00   0.450
## s(landarea)  9.00 1.00    0.97   0.015 *
## s(latitude)  9.00 7.95    0.98   0.075 .
## s(longitude) 9.00 8.31    1.00   0.405
## s(dsRM)      9.00 8.46    0.96  <2e-16 ***
## s(dsIM)      9.00 8.35    0.98   0.105
## s(dsBT)      9.00 8.02    0.87  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.1.4 Then grid-searching on k of bathrm for the reduced model

```
bathrm.ks <- c(6, 7, 8, 9, 10)
bathrm.CVs <- numeric(length(bathrm.ks))
for (i in 1:length(bathrm.ks)) {
  formula <- formula(price ~ s(bathrm, k=bathrm.ks[i]) + s(rooms) + s(ayb) +
                             s(yr_rmdl) + s(eyb) + s(saledate) + s(gba) +
                             s(landarea) + s(latitude) + s(longitude) +
                             s(dsRM) + s(dsIM) + s(dsBT) +
                             hf_bathrm + bedrm + fireplaces +
                             heat + ac + style + grade + cndtn + roof +
                             intwall + nbhd + ward + quadrant + location)
  bathrm.CVs[i] <- CV.SM(formula)
}; bathrm.ks[which.min(bathrm.CVs)]; min(bathrm.CVs)
```

```
## [1] 10
```

```
## [1] 0.1974723
```

4.1.5 Then grid-searching on k of rooms for the reduced model

```
rooms.ks <- c(7, 8, 9, 10, 11)
rooms.CVs <- numeric(length(rooms.ks))
for (i in 1:length(rooms.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=rooms.ks[i]) +
                             s(ayb) + s(yr_rmdl) + s(eyb) + s(saledate) +
                             s(gba) + s(landarea) + s(latitude) +
```

```
                            s(longitude) + s(dsRM) + s(dsIM) + s(dsBT) +
                            hf_bathrm + bedrm + fireplaces +
                            heat + ac + style + grade + cndtn + roof +
                            intwall + nbhd + ward + quadrant + location)
  rooms.CVs[i] <- CV.SM(formula)
}; rooms.ks[which.min(rooms.CVs)]; min(rooms.CVs)
```

```
## [1] 10
```

```
## [1] 0.1974723
```

4.1.6 Then grid-searching on k of ayb for the reduced model

```
ayb.ks <- c(5, 10, 15, 20, 25)
ayb.CVs <- numeric(length(ayb.ks))
for (i in 1:length(ayb.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) +
                            s(ayb, k=ayb.ks[i]) + s(yr_rmdl) + s(eyb) +
                            s(saledate) + s(gba) + s(landarea) + s(latitude) +
                            s(longitude) + s(dsRM) + s(dsIM) + s(dsBT) +
                            hf_bathrm + bedrm + fireplaces +
                            heat + ac + style + grade + cndtn + roof +
                            intwall + nbhd + ward + quadrant + location)
  ayb.CVs[i] <- CV.SM(formula)
}; ayb.ks[which.min(ayb.CVs)]; min(ayb.CVs)
```

```
## [1] 20
```

```
## [1] 0.1974098
```

4.1.7 Then grid-searching on k of yr_rmdl for the reduced model

```
yr_rmdl.ks <- c(15, 20, 25, 30, 35)
yr_rmdl.CVs <- numeric(length(yr_rmdl.ks))
for (i in 1:length(yr_rmdl.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
                            s(yr_rmdl, k=yr_rmdl.ks[i]) + s(eyb) + s(saledate) + s(gba) +
                            s(landarea) + s(latitude) + s(longitude) +
                            s(dsRM) + s(dsIM) + s(dsBT) +
                            hf_bathrm + bedrm + fireplaces +
                            heat + ac + style + grade + cndtn + roof +
                            intwall + nbhd + ward + quadrant + location)
  yr_rmdl.CVs[i] <- CV.SM(formula)
}; yr_rmdl.ks[which.min(yr_rmdl.CVs)]; min(yr_rmdl.CVs)
```

```
## [1] 20
```

```
## [1] 0.1973706
```

4.1.8 Then grid-searching on k of eyb for the reduced model

```
eyb.ks <- c(45, 55, 60, 65, 70)
eyb.CVs <- numeric(length(eyb.ks))
for (i in 1:length(eyb.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
```

```
                              s(yr_rmdl, k=20) + s(eyb, k=eyb.ks[i]) +
                              s(saledate) + s(gba) + s(landarea) + s(latitude) +
                              s(longitude) + s(dsRM) + s(dsIM) + s(dsBT) +
                              hf_bathrm + bedrm + fireplaces +
                              heat + ac + style + grade + cndtn + roof +
                              intwall + nbhd + ward + quadrant + location)
  eyb.CVs[i] <- CV.SM(formula)
}; eyb.ks[which.min(eyb.CVs)]; min(eyb.CVs)
```

```
## [1] 65
```

```
## [1] 0.1975519
```

4.1.9 Then grid-searching on k of saledate for the reduced model

```
saledate.ks <- c(20, 21, 22, 23, 24, 25)
saledate.CVs <- numeric(length(saledate.ks))
for (i in 1:length(saledate.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
                              s(yr_rmdl, k=20) + s(eyb, k=65) +
                              s(saledate, k=saledate.ks[i]) + s(gba) +
                              s(landarea) + s(latitude) + s(longitude) +
                              s(dsRM) + s(dsIM) + s(dsBT) +
                              hf_bathrm + bedrm + fireplaces +
                              heat + ac + style + grade + cndtn + roof +
                              intwall + nbhd + ward + quadrant + location)
  saledate.CVs[i] <- CV.SM(formula)
}; saledate.ks[which.min(saledate.CVs)]; min(saledate.CVs)
```

```
## [1] 24
```

```
## [1] 0.196936
```

4.1.10 Then grid-searching on k of gba for the reduced model

```
gba.ks <- c(10, 11, 12, 13, 14)
gba.CVs <- numeric(length(gba.ks))
for (i in 1:length(gba.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
                              s(yr_rmdl, k=20) + s(eyb, k=65) +
                              s(saledate, k=24) + s(gba, k=gba.ks[i]) +
                              s(landarea) + s(latitude) + s(longitude) +
                              s(dsRM) + s(dsIM) + s(dsBT) +
                              hf_bathrm + bedrm + fireplaces +
                              heat + ac + style + grade + cndtn + roof +
                              intwall + nbhd + ward + quadrant + location)
  gba.CVs[i] <- CV.SM(formula)
}; gba.ks[which.min(gba.CVs)]; min(gba.CVs)
```

```
## [1] 10
```

```
## [1] 0.196936
```

4.1.11 Then grid-searching on k of longitude for the reduced model

```
longitude.ks <- c(10, 11, 12, 13, 14)
longitude.CVs <- numeric(length(longitude.ks))
for (i in 1:length(longitude.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
                             s(yr_rmdl, k=20) + s(eyb, k=65) +
                             s(saledate, k=24) + s(gba, k=10) + s(landarea) +
                             s(latitude) + s(longitude, k=longitude.ks[i]) +
                             s(dsRM) + s(dsIM) + s(dsBT) +
                             hf_bathrm + bedrm + fireplaces +
                             heat + ac + style + grade + cndtn + roof +
                             intwall + nbhd + ward + quadrant + location)
  longitude.CVs[i] <- CV.SM(formula)
}; longitude.ks[which.min(longitude.CVs)]; min(longitude.CVs)
```

```
## [1] 10
```

```
## [1] 0.196936
```

4.1.12 Then adding interaction terms1

```
smoothing4.formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) +
                                      s(ayb, k=20) + s(yr_rmdl, k=20) +
                                      s(eyb, k=65) + s(saledate, k=24) +
                                      s(gba, k=10) + s(landarea) + s(latitude) +
                                      s(longitude, k=10) + s(dsRM) + s(dsIM) +
                                      s(dsBT) +
                                      hf_bathrm + bedrm + fireplaces +
                                      heat + ac + style + grade + cndtn +
                                      roof + intwall + nbhd + ward + quadrant +
                                      location +
                                      te(saledate, ayb))
SCV.result4 <- CV.SM(smoothing4.formula); SCV.result4
```

```
## [1] 0.1948838
```

4.1.13 Then adding interaction terms2

```
smoothing5.formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) +
                                      s(ayb, k=20) + s(yr_rmdl, k=20) +
                                      s(eyb, k=65) + s(saledate, k=24) +
                                      s(gba, k=10) + s(landarea) + s(latitude) +
                                      s(longitude, k=10) + s(dsRM) + s(dsIM) +
                                      s(dsBT) +
                                      hf_bathrm + bedrm + fireplaces +
                                      heat + ac + style + grade + cndtn +
                                      roof + intwall + nbhd + ward + quadrant +
                                      location +
                                      te(saledate, ayb) + te(saledate, eyb))
SCV.result5 <- CV.SM(smoothing5.formula); SCV.result5
```

```
## [1] 0.1948298
```

4.1.14 Then adding interaction terms3

```
smoothing6.formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) +
                                      s(ayb, k=20) + s(yr_rmdl, k=20) +
                                      s(eyb, k=65) + s(saledate, k=24) +
                                      s(gba, k=10) + s(landarea) + s(latitude) +
                                      s(longitude, k=10) + s(dsRM) + s(dsIM) +
                                      s(dsBT) +
                                      hf_bathrm + bedrm + fireplaces +
                                      heat + ac + style + grade + cndtn +
                                      roof + intwall + nbhd + ward + quadrant +
                                      location +
                                      te(saledate, ayb) + te(saledate, eyb) +
                                      te(saledate, yr_rmdl))
SCV.result6 <- CV.SM(smoothing6.formula); SCV.result6
```

```
## [1] 0.1943961
```

4.1.15 Then adding interaction terms4

```
smoothing7.formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) +
                                      s(ayb, k=20) + s(yr_rmdl, k=20) +
                                      s(eyb, k=65) + s(saledate, k=24) +
                                      s(gba, k=10) + s(landarea) + s(latitude)
                                      + s(longitude, k=10) + s(dsRM) +
                                      s(dsIM) + s(dsBT) +
                                      hf_bathrm + bedrm + fireplaces +
                                      heat + ac + style + grade + cndtn +
                                      roof + intwall + nbhd + ward + quadrant +
                                      location +
                                      te(saledate, ayb) + te(saledate, eyb) +
                                      te(saledate, yr_rmdl) +
                                      te(landarea, longitude))
SCV.result7 <- CV.SM(smoothing7.formula); SCV.result7
```

```
## [1] 0.1941798
```

4.1.16 Then adding interaction terms5

```
smoothing8.formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) +
                                      s(ayb, k=20) + s(yr_rmdl, k=20) +
                                      s(eyb, k=65) + s(saledate, k=24) +
                                      s(gba, k=10) + s(landarea) + s(latitude)
                                      + s(longitude, k=10) + s(dsRM) +
                                      s(dsIM) + s(dsBT) +
                                      hf_bathrm + bedrm + fireplaces +
                                      heat + ac + style + grade + cndtn +
                                      roof + intwall + nbhd + ward + quadrant +
                                      location +
                                      te(saledate, ayb) + te(saledate, eyb) +
                                      te(saledate, yr_rmdl) +
                                      te(landarea, longitude) +
                                      te(landarea, latitude))
SCV.result8 <- CV.SM(smoothing8.formula); SCV.result8
```

```
## [1] 0.193775
```

4.1.17 Then grid-searching on k of interaction term1

```
int1.ks <- c(8, 9, 11)
int1.CVs <- numeric(length(int1.ks))
for (i in 1:length(int1.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
                             s(yr_rmdl, k=20) + s(eyb, k=65) +
                             s(saledate, k=24) + s(gba, k=10) + s(landarea) +
                             s(latitude) + s(longitude, k=10) + s(dsRM) +
                             s(dsIM) + s(dsBT) +
                             hf_bathrm + bedrm + fireplaces +
                             heat + ac + style + grade + cndtn + roof +
                             intwall + nbhd + ward + quadrant + location +
                             te(saledate, ayb, k=int1.ks[i]) +
                             te(saledate, eyb) + te(saledate, yr_rmdl) +
                             te(landarea, longitude) + te(landarea, latitude))
  int1.CVs[i] <- CV.SM(formula)
}; int1.ks[which.min(int1.CVs)]; min(int1.CVs)
```

```
## [1] 9
```

```
## [1] 0.1936082
```

4.1.18 Then grid-searching on k of interaction term2

```
int2.ks <- c(8, 9, 11)
int2.CVs <- numeric(length(int2.ks))
for (i in 1:length(int2.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
                             s(yr_rmdl, k=20) + s(eyb, k=65) +
                             s(saledate, k=24) + s(gba, k=10) + s(landarea) +
                             s(latitude) + s(longitude, k=10) + s(dsRM) +
                             s(dsIM) + s(dsBT) +
                             hf_bathrm + bedrm + fireplaces +
                             heat + ac + style + grade + cndtn + roof +
                             intwall + nbhd + ward + quadrant + location +
                             te(saledate, ayb, k=9) +
                             te(saledate, eyb, k=int2.ks[i]) +
                             te(saledate, yr_rmdl) + te(landarea, longitude) +
                             te(landarea, latitude))
  int2.CVs[i] <- CV.SM(formula)
}; int2.ks[which.min(int2.CVs)]; min(int2.CVs)
```

```
## [1] 8
```

```
## [1] 0.1929718
```

4.1.19 Then grid-searching on k of interaction term3

```
int3.ks <- c(8, 9, 11)
int3.CVs <- numeric(length(int3.ks))
for (i in 1:length(int3.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
```

```
                            s(yr_rmdl, k=20) + s(eyb, k=65) +
                            s(saledate, k=24) + s(gba, k=10) + s(landarea) +
                            s(latitude) + s(longitude, k=10) + s(dsRM) +
                            s(dsIM) + s(dsBT) +
                            hf_bathrm + bedrm + fireplaces +
                            heat + ac + style + grade + cndtn + roof +
                            intwall + nbhd + ward + quadrant + location +
                            te(saledate, ayb, k=9) + te(saledate, eyb, k=8) +
                            te(saledate, yr_rmdl, k=int3.ks[i]) +
                            te(landarea, longitude) + te(landarea, latitude))
  int3.CVs[i] <- CV.SM(formula)
}; int3.ks[which.min(int3.CVs)]; min(int3.CVs)
```

```
## [1] 9
```

```
## [1] 0.1921476
```

4.1.20 Then grid-searching on k of interaction term4

```
int4.ks <- c(8, 9, 11)
int4.CVs <- numeric(length(int4.ks))
for (i in 1:length(int4.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
                            s(yr_rmdl, k=20) + s(eyb, k=65) +
                            s(saledate, k=24) + s(gba, k=10) + s(landarea) +
                            s(latitude) + s(longitude, k=10) + s(dsRM) +
                            s(dsIM) + s(dsBT) +
                            hf_bathrm + bedrm + fireplaces +
                            heat + ac + style + grade + cndtn + roof +
                            intwall + nbhd + ward + quadrant + location +
                            te(saledate, ayb, k=9) +
                            te(saledate, eyb, k=8) +
                            te(saledate, yr_rmdl, k=9) +
                            te(landarea, longitude, k=int4.ks[i]) +
                            te(landarea, latitude))
  int4.CVs[i] <- CV.SM(formula)
}; int4.ks[which.min(int4.CVs)]; min(int4.CVs)
```

```
## [1] 9
```

```
## [1] 0.1925376
```

The above k values for this interaction term do not decrease the CV score, so default k is used.

4.1.21 Then grid-searching on k of interaction term5

```
int5.ks <- c(8, 9, 11)
int5.CVs <- numeric(length(int5.ks))
for (i in 1:length(int5.ks)) {
  formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) + s(ayb, k=20) +
                            s(yr_rmdl, k=20) + s(eyb, k=65) +
                            s(saledate, k=24) + s(gba, k=10) + s(landarea) +
                            s(latitude) + s(longitude, k=10) + s(dsRM) +
                            s(dsIM) + s(dsBT) +
                            hf_bathrm + bedrm + fireplaces +
```

```
                              heat + ac + style + grade + cndtn + roof +
                              intwall + nbhd + ward + quadrant + location +
                              te(saledate, ayb, k=9) +
                              te(saledate, eyb, k=10) +
                              te(saledate, yr_rmdl) +
                              te(landarea, longitude) +
                              te(landarea, latitude, k=int5.ks[i]))
  int5.CVs[i] <- CV.SM(formula)
}; int5.ks[which.min(int5.CVs)]; min(int5.CVs)
```

```
## [1] 8
```

```
## [1] 0.1926102
```

The above k values for this interaction term do not decrease the CV score, so default k is used.

4.1.22 The final smoothing model

```
finalSM.formula <- formula(price ~ s(bathrm, k=10) + s(rooms, k=10) +
                                   s(ayb, k=20) + s(yr_rmdl, k=20) +
                                   s(eyb, k=65) + s(saledate, k=24) +
                                   s(gba, k=10) + s(landarea) + s(latitude) +
                                   s(longitude, k=10) + s(dsRM) + s(dsIM) +
                                   s(dsBT) + hf_bathrm + bedrm + fireplaces +
                                   heat + ac + style + grade + cndtn + roof +
                                   intwall + nbhd + ward + quadrant +
                                   location + te(saledate, ayb, k=9) +
                                   te(saledate, eyb, k=10) +
                                   te(saledate, yr_rmdl) +
                                   te(landarea, longitude) +
                                   te(landarea, latitude))
finalSM <- mgcv::gam(finalSM.formula, data=dat)
```

4.2 Random Forest Models

4.2.1 First fit a full Random Forest model

```
full.formula <- formula(price ~ heat + ac + style + grade + cndtn + extwall +
                                roof + intwall + nbhd + ward + quadrant +
                                location + rmdled + bathrm + hf_bathrm + rooms +
                                bedrm + ayb + yr_rmdl + eyb + stories + saledate
                                + gba + kitchens + fireplaces + landarea +
                                latitude + longitude + dsRM + dsIM + dsBT)
RFfull.result <- CV.RF(formula=full.formula); RFfull.result
```

```
## [1] 0.2001356
```

```
RFfull <- randomForest::randomForest(full.formula, data=dat)
rfPermute::importance(RFfull, data=dat, target="price")
```

```
##            IncNodePurity
## heat           3.6079208
## ac             4.2803340
## style          5.5757172
## grade        263.7725247
```

```
## cndtn         33.7417154
## extwall        7.9918166
## roof           4.0944941
## intwall        3.6990036
## nbhd         756.5995344
## ward         198.2541090
## quadrant       5.1429102
## location      12.5108250
## rmdled         3.3206894
## bathrm       198.3057561
## hf_bathrm      5.0812874
## rooms         28.0151544
## bedrm         50.1983950
## ayb           21.4194174
## yr_rmdl       67.0989904
## eyb          150.6013355
## stories        8.0997212
## saledate     839.7341007
## gba          314.2359479
## kitchens       0.8815905
## fireplaces    10.9742924
## landarea      34.8822819
## latitude      98.2515192
## longitude    529.1568523
## dsRM          52.8263700
## dsIM          58.3398507
## dsBT          58.2260137
```

4.2.2 Next remove insigificant variables

```
## kitchens removed
RFre.formula <- formula(price ~ heat + ac + style + grade + cndtn + extwall +
                                roof + intwall + nbhd + ward + quadrant +
                                location + rmdled + bathrm + hf_bathrm + rooms +
                                bedrm + ayb + yr_rmdl + eyb + stories +
                                saledate + gba + fireplaces + landarea +
                                latitude + longitude + dsRM + dsIM + dsBT)
RFre.result <- CV.RF(formula=RFre.formula); RFre.result
```

```
## [1] 0.1994111
```

4.2.3 Grid-searching on mtry

```
mtrys <- seq(11, 18, by=1)
mtry.results <- numeric(length(mtrys))
for (i in 1:length(mtrys)) {
  mtry.results[i] <- CV.RF(formula=RFre.formula, mtry=mtrys[i])
}; mtry.results; min(mtry.results)
```

```
## [1] 0.1993706 0.1986801 0.1983188 0.1981627 0.1981691 0.1981082 0.1982353
## [8] 0.1983221
```

```
## [1] 0.1981082
```

```
mtry.best <- mtrys[which.min(mtry.results)]; mtry.best
```

```
## [1] 16
```

4.2.4 Fix mtry and grid-searching on nodesize

```
nodesizes <- seq(1, 6, by=1)
nodesize.results <- numeric(length(nodesizes))
for (i in 1:length(nodesizes)) {
  nodesize.results[i] <- CV.RF(formula=RFre.formula, mtry=mtry.best,
                               nodesize=nodesizes[i])
}; nodesize.results; min(nodesize.results)
```

```
## [1] 0.1978287 0.1979824 0.1980827 0.1981380 0.1983795 0.1984524
```

```
## [1] 0.1978287
```

```
nodesize.best <- nodesizes[which.min(nodesize.results)]; nodesize.best
```

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

4.2.5 Fix mtry, nodesize and then grid-searching on ntree

```
ntrees <- c(350, 400, 450, 525, 550)
ntrees.results <- numeric(length(ntrees))
for (i in 1:length(ntrees)) {
  ntrees.results[i] <- CV.RF(formula=RFre.formula, mtry=mtry.best,
                             nodesize=nodesize.best, ntree=ntrees[i])
}; ntrees.results; min(ntrees.results)
```

```
## [1] 0.1981373 0.1977115 0.1979255 0.1977870 0.1979783
```

```
## [1] 0.1977115
```

```
ntree.best <- ntrees[which.min(ntrees.results)]; ntree.best
```

```
## [1] 400
```

4.2.6 The final Random Forest model

```
finalRF.formula <- formula(price ~ heat + ac + style + grade + cndtn + extwall +
                                   roof + intwall + nbhd + ward + quadrant +
                                   location + rmdled + bathrm + hf_bathrm +
                                   rooms + bedrm + ayb + yr_rmdl + eyb +
                                   stories + saledate + gba + fireplaces +
                                   landarea + latitude + longitude +
                                   dsRM + dsIM + dsBT)
finalRF <- randomForest::randomForest(finalRF.formula, mtry=mtry.best,
                                      nodesize=nodesize.best,
                                      ntree=ntree.best, data=dat)
```

4.3 Boosting Models

```
dat.copy <- dat

dat.copy$heat <- as.factor(dat.copy$heat)
```

```
dat.copy$ac <- as.factor(dat.copy$ac)
dat.copy$style <- as.factor(dat.copy$style)
dat.copy$grade <- as.factor(dat.copy$grade)
dat.copy$cndtn <- as.factor(dat.copy$cndtn)
dat.copy$extwall <- as.factor(dat.copy$extwall)
dat.copy$roof <- as.factor(dat.copy$roof)
dat.copy$intwall <- as.factor(dat.copy$intwall)
dat.copy$nbhd <- as.factor(dat.copy$nbhd)
dat.copy$ward <- as.factor(dat.copy$ward)
dat.copy$quadrant <- as.factor(dat.copy$quadrant)
dat.copy$rmdled <- as.factor(dat.copy$rmdled)
dat.copy$location <- as.factor(dat.copy$location)
```

4.3.1 First fit a full Boosting model

```
gbm.formula <- formula(price ~ heat + ac + style + grade + cndtn + extwall +
                               roof + intwall + nbhd + ward + quadrant +
                               location + rmdled + bathrm + hf_bathrm + rooms +
                               bedrm + ayb + yr_rmdl + eyb + stories + saledate
                               + gba + kitchens + fireplaces + landarea +
                               latitude + longitude + dsRM + dsIM + dsBT)
gbm.CV <- CV.BT(gbm.formula); gbm.CV
```

```
## [1] 0.2371541
```

```
gbm <- gbm::gbm(formula=gbm.formula, distribution="gaussian",
                bag.fraction=1, data=dat.copy)
relative.influence(gbm)
```

```
## n.trees not given. Using 100 trees.
```

```
##       heat         ac      style      grade      cndtn    extwall       roof
##    0.00000    0.00000    0.00000 1470.10636  212.95096    0.00000    0.00000
##    intwall       nbhd       ward   quadrant   location     rmdled     bathrm
##    0.00000 5617.44541   55.65163    0.00000    0.00000    0.00000 1121.68809
##  hf_bathrm      rooms      bedrm        ayb    yr_rmdl        eyb    stories
##   12.53287    0.00000   15.46291    0.00000    0.00000   83.84397    0.00000
##   saledate        gba   kitchens fireplaces   landarea   latitude  longitude
## 5347.58561 2014.56593    0.00000   21.22023   12.33627    0.00000 2467.42747
##       dsRM       dsIM       dsBT
##    0.00000    0.00000  121.21853
```

4.3.2 Next remove some of the insignificant terms

```
reduced.formula <- formula(price ~ grade + cndtn + nbhd + ward + bathrm +
                                   hf_bathrm + bedrm + eyb + saledate + gba +
                                   landarea + longitude + dsRM + dsIM + dsBT)
CV.BT(formula=reduced.formula)
```

```
## [1] 0.2371165
```

The CV score of the reduced model hardly changes, so the full Boosting model is kept.

4.3.3 Then grid-searching on interaction.depth

```
interactions <- c(2, 3, 4, 5, 6, 7, 8, 9, 10)
interactions.results <- numeric(length(interactions))
for (i in 1:length(interactions)) {
  interactions.results[i] <- CV.BT(formula=gbm.formula,
                                   interaction.depth=interactions[i])
}; interactions.results; min(interactions.results)
```

```
## [1] 0.2041957 0.1930313 0.1900203 0.1877799 0.1851756 0.1844722 0.1820787
## [8] 0.1811064 0.1801997
```

```
## [1] 0.1801997
```

```
interactions.best <- interactions[which.min(interactions.results)]
interactions.best
```

```
## [1] 10
```

4.3.4 Then grid-searching on n.treeses

```
n.treeses <- c(100, 150, 200, 250, 300, 325, 350, 375,
               400, 450, 500, 550, 600, 650, 700)
n.treeses.results <- numeric(length(n.treeses))
for (i in 1:length(n.treeses)) {
  n.treeses.results[i] <- CV.BT(formula=gbm.formula,
                                interaction.depth=interactions.best,
                                n.trees=n.treeses[i])
}; n.treeses.results; min(n.treeses.results)
```

```
##  [1] 0.1801997 0.1781235 0.1774188 0.1772149 0.1772432 0.1772575 0.1772200
##  [8] 0.1771813 0.1772125 0.1771816 0.1769938 0.1771319 0.1772839 0.1773412
## [15] 0.1775201
```

```
## [1] 0.1769938
```

```
n.treeses.best <- n.treeses[which.min(n.treeses.results)]; n.treeses.best
```

```
## [1] 500
```

4.3.5 Then grid-searching on shrinkages

```
shrinkages <- c(0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15,
                0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
shrinkages.results <- numeric(length(shrinkages))
for (i in 1:length(shrinkages)) {
  shrinkages.results[i] <- CV.BT(formula=gbm.formula,
                                 interaction.depth=interactions.best,
                                 n.trees=n.treeses.best,
                                 shrinkage=shrinkages[i])
}; shrinkages.results; min(shrinkages.results)
```

```
##  [1] 0.1763152 0.1765363 0.1752821 0.1779232 0.1769938 0.1787892 0.1800139
##  [8] 0.1800006 0.1803145 0.1822741 0.1840444 0.1900498 0.2028388 0.2066324
## [15] 0.2172494 0.2307998 0.2423098 0.2614415
```

```
## [1] 0.1752821
```

```
shrinkages.best <- shrinkages[which.min(shrinkages.results)]; shrinkages.best
```

```
## [1] 0.08
```

4.3.6 The final Boosting model

```
finalGBM.formula <- formula(price ~ heat + ac + style + grade + cndtn +
                                    extwall + roof + intwall + nbhd + ward +
                                    quadrant + location + rmdled + bathrm +
                                    hf_bathrm + rooms + bedrm + ayb +
                                    yr_rmdl + eyb + stories + saledate + gba +
                                    kitchens + fireplaces + landarea +
                                    latitude + longitude + dsRM + dsIM + dsBT)
finalGBM <- gbm::gbm(formula=finalGBM.formula, distribution="gaussian",
                     bag.fraction=1, shrinkage=shrinkages.best,
                     interaction.depth=interactions.best,
                     n.trees=n.treeses.best,
                     data=dat.copy)
```