Smoothing Model Building

A Trial-and-Error Approach

Bruce Liu

- UW ID: 20843361
- Kaggle public score: 0.15326
- Kaggle private score: 0.15962
- Kaggle submission count/times: 29

Important: follow the structure of the document and summarize important handlings in the summary section. The structure here is kept at a minimum, feel free if you have more subsections and details to add.

Summary

Preprocessing

Transformation (if any, delete if none)

- price: Applied a power transform of 1/10 on the response variate price.
- saledate: Converted it to a numeric value of days since January 1st, 1970 (the beginning of Unix time)
- cndtn, heat, ward, grade: Converted to a numeric variable with the order increasing in price
- roof, extwall, intwall, nbhd: Applied factor() for better use in the model.
- ac: Applied factor() and ordered by increase in price
- style: Applied factor() and ordered by increase in price, unused

New Variables (if any, delete if none)

- proximity: This variable is defined as the distance from the most expensive property in the dataset. The measurement is in degrees.

Missing data handling

- ayb: replace the missing values in ayb with the mean year based on what kind of heating the housing unit has.
- intwall: replace the values of 'Terrazo' and 'Vinyl Sheet' in dtest with their closest counterparts
- style: replace the values of 'Default' in dtest with the styles of housing units most common in their wards, unused

Model Building

A trial-and-error approach was used with `bam` for efficiency. The main model metric used was REML. Adding, removing, and applying different transformations to the model and using REML where lower is better to determine the best model to run.

Final Model

- The final model is $price^{1/10} \sim$ proximity by nbhd + saledate,ayb,eyb + gba $\otimes$ landarea + grade + bathrm + fireplaces + roof + extwall + intwall + heat by ac + hf_bathrm + cndtn + rooms

1.Preprocessing

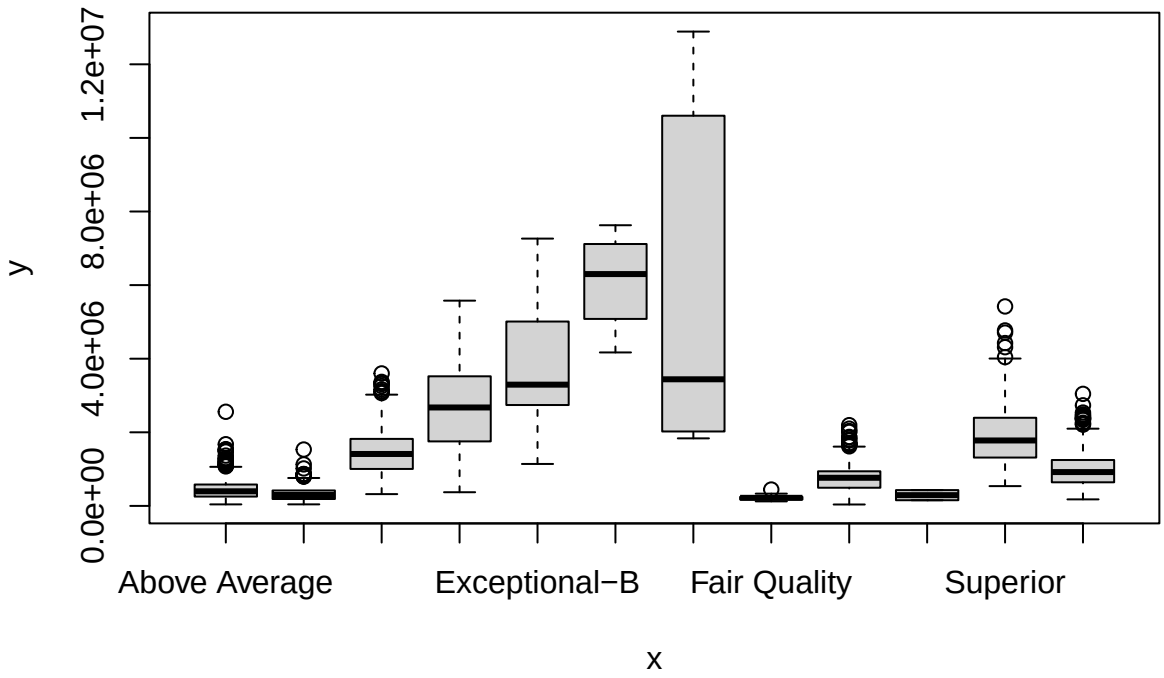
1.1 Loading data

```
load("smooth.Rdata")
```

For categorical variates, it may be useful to see whether or not there are any clear trends.

We'll take the example of the `grade` variate.

```
plot(factor(dtrain$grade), dtrain$price)
```



We can quite clearly see in the plot that there is a trend that the better the grade, the higher the price. We reorder the grade variable from lowest price to highest price. We do the same with the variates `cndtn`, `ward`, `ac`, and `heat`. In order for use to use the smoothing function `s()` for these variates, we convert them to numerics using `as.numeric()`.

```
# Preprocessing the dtrain data

dtrain$grade <- as.numeric(factor(dtrain$grade, levels = c("Low Quality",
                                                           "Fair Quality", "Average",
                                                           "Above Average", "Good Quality",
                                                           "Very Good", "Excellent",
                                                           "Superior","Exceptional-A",
                                                           "Exceptional-B", "Exceptional-C",
                                                           "Exceptional-D")))
maxpricelat <- dtrain$latitude[which.max(dtrain$price)]
maxpricelon <- dtrain$longitude[which.max(dtrain$price)]
```

```
proximity <- sqrt((dtrain$latitude - maxpricelat)^2 + (dtrain$longitude - maxpricelon)^2)
dtrain$proximity <- proximity
dtrain$saledate <- as.numeric(as.Date(dtrain$saledate))
dtrain$cndtn <- as.numeric(factor(dtrain$cndtn, levels = c("Poor","Fair","Average",
                                                           "Good", "Very Good", "Excellent")))
dtrain$style <- as.numeric(factor(dtrain$style,levels = c("Split Foyer","1 Story",
                                                          "1.5 Story Fin", "1.5 Story Unfin",
                                                          "2.5 Story Unfin", "2 Story",
                                                          "Bi-Level","Split Level",
                                                          "2.5 Story Fin", "3 Story", "4 Story")))
dtrain$ac <- factor(dtrain$ac, levels = c("N", "Y"))
dtrain$ward <- as.numeric(factor(dtrain$ward, levels = c("Ward 7", "Ward 8", "Ward 5",
                                                         "Ward 6", "Ward 4", "Ward 1",
                                                         "Ward 3", "Ward 2")))
dtrain$nbhd <- factor(dtrain$nbhd)
dtrain$heat <- factor(dtrain$heat, levels = c("Wall Furnace", "Air-Oil", "Elec Base Brd",
                                                         "Gravity Furnace", "No Data","Evp Cool",
                                                         "Hot Water Rad", "Forced Air", "Water Base Brd",
                                                         "Ht Pump","Warm Cool","Air Exchng"))
dtrain$roof <- factor(dtrain$roof)
dtrain$extwall <- factor(dtrain$extwall)
dtrain$intwall <- factor(dtrain$intwall)

# Preprocessing the dtest data

proximity <- sqrt((dtest$latitude - maxpricelat)^2 + (dtest$longitude - maxpricelon)^2)
dtest$proximity <- proximity
dtest$saledate <- as.numeric(as.Date(dtest$saledate))
dtest$grade <- as.numeric(factor(dtest$grade, levels = c("Low Quality",
                                                                 "Fair Quality", "Average",
                                                                 "Above Average", "Good Quality",
                                                                 "Very Good", "Excellent",
                                                                 "Superior","Exceptional-A",
                                                                 "Exceptional-B", "Exceptional-C",
                                                                 "Exceptional-D")))
dtest$ward <- as.numeric(factor(dtest$ward, levels = c("Ward 7", "Ward 8", "Ward 5",
                                                       "Ward 6", "Ward 4", "Ward 1",
                                                       "Ward 3", "Ward 2")))
dtest$heat <- factor(dtest$heat, levels = c("Wall Furnace", "Air-Oil", "Elec Base Brd",
                                                       "Gravity Furnace", "No Data","Evp Cool",
                                                       "Hot Water Rad", "Forced Air", "Water Base Brd",
                                                       "Ht Pump","Warm Cool","Air Exchng"))
```

1.2 Missing data handling

We notice that in the `dtrain` dataset, there are variables that are N/A and variables that were not originally in the `dtrain` dataset. How do we impute them? We ended up using `ayb` as a variable of note for the model. However, a couple of the data points have N/A for the `ayb` column. We naively choose consider the type of heating used by the housing unit under the assumption that the type of heating may be indicative of when the house would have been built. After finding the heating of this unit, we impute the average age of all housing units with the same type of heating as its value for `ayb`. What do we do with the `intwall` variable? We have two different variables in "Terrazo" and "Vinyl Sheet". For this, we look at the closest analogue of the interior wall material. For "Terrazo", we learn that it is most similar to concrete. So, we impute the

`dtrain` value of "Lt Concrete" to the `dtest` data. Since "Vinyl Sheet" does not particularly match well with any of the descriptors in `dtrain`, we impute "Default" instead. Finally for `style`, despite not using it in the model, we impute the most common type of housing unit in their ward. It happens to be "2 Story" and that's what we impute onto the `dtest` data.

```
for (i in 1:1000) {
  if (is.na(dtest[i,"ayb"])) {
    if (dtest[i, "heat"] == "Forced Air") {
      dtest[i, "ayb"] <- round(
            mean(dtrain[which(dtrain$heat == "Forced Air"),"ayb"],na.rm = TRUE))
    }
  }
  if (dtest[i, "intwall"] == "Terrazo") {
    dtest[i, "intwall"] <- "Lt Concrete"
  }
  if (dtest[i, "intwall"] == "Vinyl Sheet") {
    dtest[i, "intwall"] <- "Default"
  }
  if (dtest[i, "style"] == "Default") {
    dtest[i, "style"] <- "2 Story"
  }
}
```

We finish preprocessing the rest of the test data here.

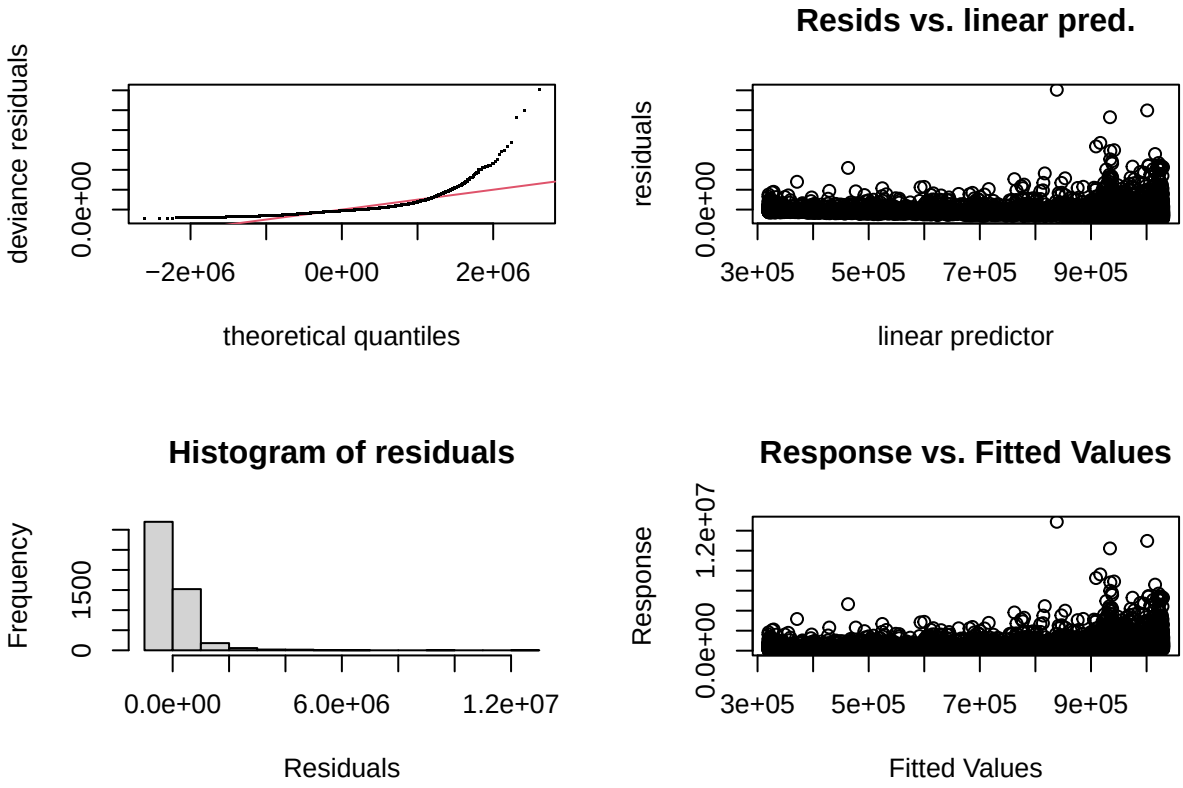
```
dtest$cndtn <- as.numeric(factor(dtest$cndtn, levels = c("Poor","Fair","Average",
                                                         "Good", "Very Good", "Excellent")))
dtest$nbhd <- factor(dtest$nbhd)
dtest$style <- factor(dtest$style)
dtest$ac <- factor(dtest$ac, levels = c("N", "Y"))
dtest$roof <- factor(dtest$roof)
dtest$extwall <- factor(dtest$extwall)
dtest$intwall <- factor(dtest$intwall)
dtest$style <- as.numeric(factor(dtest$style,levels = c("Split Foyer","1 Story",
                                                        "1.5 Story Fin", "1.5 Story Unfin",
                                                        "2.5 Story Unfin", "2 Story",
                                                        "Bi-Level","Split Level",
                                                        "2.5 Story Fin", "3 Story", "4 Story")))
```

2. Model building

For most of the model building, while we began with using `locfit`, it seemed to misbehave after adding approximately eight variates to the formula (i.e. R would produce a fatal error forcing the environment to restart.). For this reason, we switch to `gam` in the `mgcv` library for our model building.

Before we start adding terms using `s()`, `ti()`, and `te()`, perhaps it may be useful to do a check on the residuals of the response. We'll use the converted saledate variate and check for the residual fit.

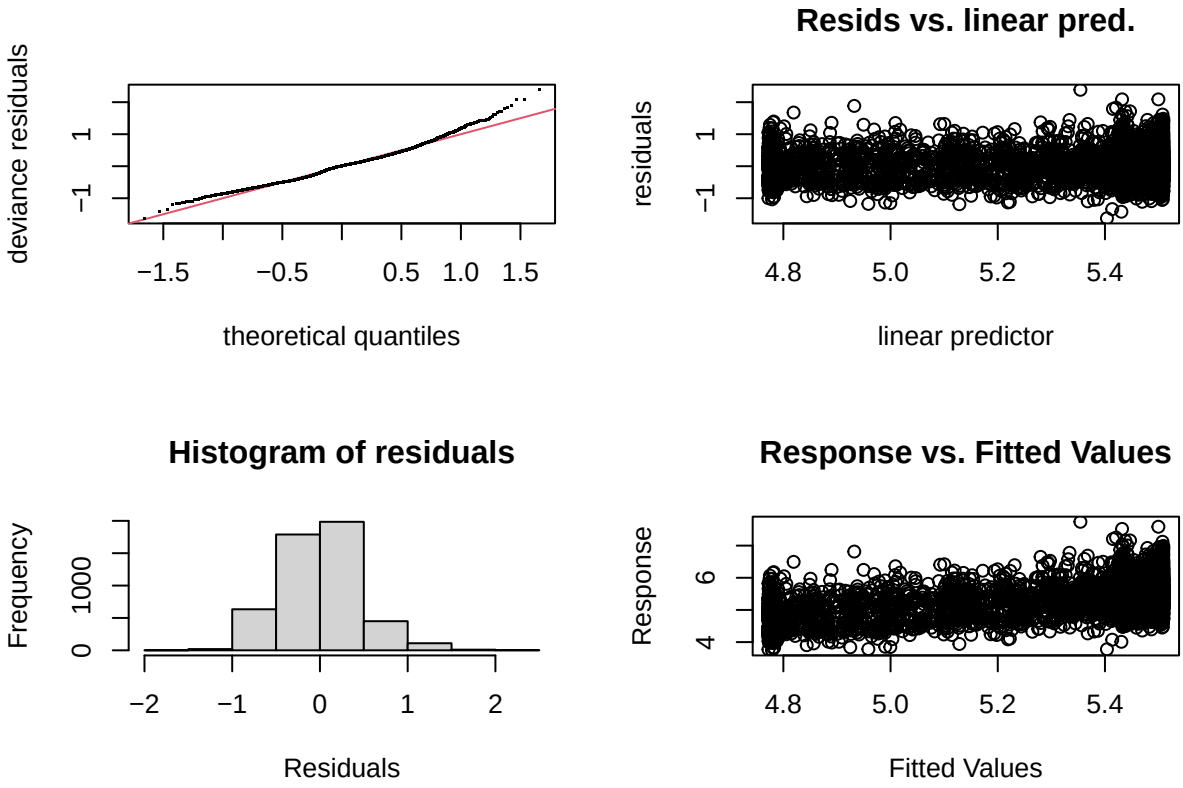
```
library(mgcv)
fit <- gam(price ~ s(saledate), data = dtrain)
gam.check(fit)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 7 iterations.
## The RMS GCV score gradient at convergence was 35888.54 .
## The Hessian was positive definite.
## Model rank =  10 / 10
##
## 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(saledate) 9.00 5.92    0.97  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Hmm... the residuals seem to not be fit quite well. What is not shown clearly in these plots is that the residuals go up to 1.2×10^7. It might be useful to scale down the price variate instead. We recall from A1 that the saledate residual fit using BIC has degree 8. Let's scale price by a power of 1/8.

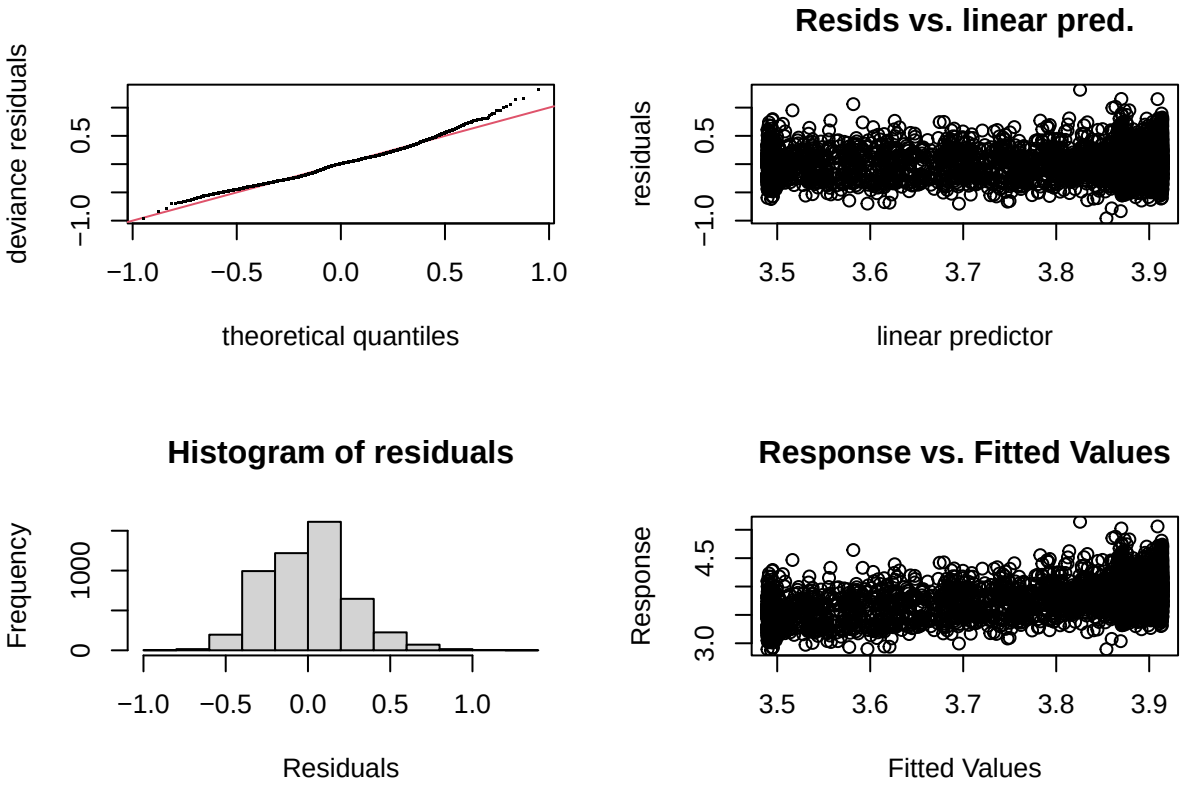
```
fit <- gam(price^(1/8) ~ s(saledate), data = dtrain)
gam.check(fit)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 5 iterations.
## The RMS GCV score gradient at convergence was 2.60828e-06 .
## The Hessian was positive definite.
## Model rank =  10 / 10
##
## 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(saledate) 9.00 6.76    0.97    0.03 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The residuals are much better behaved. I decided in the end to scale price by a power of 1/10 because I wanted 99% of all the residuals to lie between $[-1, 1]$.

```
fit <- gam(price^(1/10) ~ s(saledate), data = dtrain)
gam.check(fit)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 5 iterations.
## The RMS GCV score gradient at convergence was 3.863024e-07 .
## The Hessian was positive definite.
## Model rank =  10 / 10
##
## 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(saledate) 9.00 6.78    0.97    0.02 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

With the scaling of the response variate under control, we move on to adding terms to the model. I mentioned during the summary that the approach to model building was trial-and-error. This was simply due to me not knowing of model building software in R for smoothing. Regardless, we used `bam` from the `mgcv` package for model building. It runs faster than `gam` and uses less memory. The catch is the criterion used for `bam`. Looking at the documentation of `bam` and `gam`, we see that the criterion for `gam` is GCV.Cp and the criterion for `bam` is fREML. In general, GCV.Cp is a more accurate measure for the accuracy of a model. However, because of the nature of the generalized cross-validation score, it takes much more time to process than fREML as the model that we want to make increases in size. We note that fREML is fast Restricted Maximum Likelihood.

Our trial and error method is as follows:

1. Begin with a model.
2. Create a new model with an adjustment.

3. Use the fREML criterion to determine whether to keep the new model.

Let's illustrate an example here.

```
m <- bam(price^(1/10) ~ s(saledate), data = dtrain)
mAdj <- bam(price^(1/10) ~ s(saledate) + s(fireplaces), data = dtrain)
summary(m)[26]
```

```
## $sp.criterion
##    fREML
## 279.2239
```

```
summary(mAdj)[26]
```

```
## $sp.criterion
##     fREML
## -1057.294
```

Since our adjusted model has a lower fREML, we use that model as our new starting point. What is of note is one of the quirks of `bam`. If we try to run a thin plate spline on the number of wards, we get an error. It seems that if you do not define the value of k, `s()` requires a k value of at least 9. Setting k to be however many categories we have in our variable was enough but it did make it interesting to note this down.

Listed here is an example.

```
mErr <- bam(price^(1/10) ~ s(cndtn), data = dtrain)
```

```
## Error in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): A term has fewer unique covaria
```

This model will not work. But

```
mFix <- bam(price^(1/10) ~ s(cndtn, k = 6), data = dtrain)
# we recall that cndtn has 6 categories.
```

will.

In choosing which variable to consider interaction terms with, we went with variables that made the most sense. In the context of the variables given, it would be logical for the variables `gba` and `landarea` to have the some sort of interaction with one another. Likewise, determining `proximity` based on what neighbourhood a unit is in makes sense as does determining the heating of a unit based on whether they had AC or not. For `proximity` and `heat`, we used the `by` parameter in `s()`. For `gba` and `landarea`, we ended up seeing that `te()` was the most significant term.

```
mComp <- bam(price^(1/10) ~ s(gba) + s(landarea) + te(gba, landarea),
             data = dtrain)
summary(mComp)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## price^(1/10) ~ s(gba) + s(landarea) + te(gba, landarea)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.788325   0.003051    1242   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Approximate significance of smooth terms:
##                     edf Ref.df      F p-value
## s(gba)            1.224  1.403  0.421   0.591
## s(landarea)       1.000  1.000  4.400   0.036 *
## te(gba,landarea) 7.576  8.645 14.408  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.477   Deviance explained = 47.8%
## fREML = -551.52  Scale est. = 0.046538  n = 5000
```

```
# comparing gba and landarea
```

After continuing this trial and error, we end up with our final model.

```
final <- bam(price^(1/10)~s(proximity, by = nbhd)+s(saledate,ayb,eyb)
                      +te(gba, landarea)+s(grade)+s(bathrm)+s(fireplaces)+roof
                      +extwall+intwall+s(as.numeric(heat),by = ac)+s(hf_bathrm, k = 5)+s(ward, k = 8)
                      +s(cndtn, k = 6)+s(rooms), data=dtrain)
summary(final)[10]
```

```
## $r.sq
## [1] 0.9631094
```

```
summary(final)[26]
```

```
## $sp.criterion
##     fREML
## -6698.476
```

Of course, during evaluation, we want to use `gam` as it is more accurate than `bam`. However the correlation and the fREML are very solid. Thus, our final model for submission is

```
fit <- gam(price^(1/10)~s(proximity, by = nbhd)+s(saledate,ayb,eyb)
                      +te(gba, landarea)+s(grade)+s(bathrm)+s(fireplaces)+roof
                      +extwall+intwall+s(as.numeric(heat),by = ac)+s(hf_bathrm, k = 5)+s(ward, k = 8)
                      +s(cndtn, k = 6)+s(rooms), data=dtrain)
```

Concluding Thoughts

So, the question is, how did it do? All things considered, I think the model did alright for an approach that was not rigourous. I think that with models like this, there is always room for improvement like adjusting the explanatory variables and testing other combinations of tensor product or interaction terms. All in all, I think it went as well as it could.