Data 621 Final Project

Jose Rodriguez, Vyanna Hill, Christian Uriostegui

2023-12-09

Abstract

For app developers, their app performance in the app store accounts for future funding for new projects. This project reviewed linear regression and its accuracy towards its predictions in an app's rating in the Google App Store. In the first path of the project, the team reviewed the Google App Store data set for its viability in linear regression as the response feature is continuous and not discrete for count regression. After a review of the data set's non-normality and model performance, the project's regression scope expanded toward generalized additive modeling (GAM).

The team created four different GAMs models with various splines on the features. The splines help better fit the highly skewed features of rating count and stars, as the increased knots on the features boosted the R^2 20%+. This exploration of the different types of splines resulted in the third model with the closest fit with an R^2 of 96%, a deviance of 669, and an RSME of 15%. The final model's performance highlighted a need for smoothing with a non-normal data set with a continuous response.

Keywords

Generalized Additive Model, Linear Prediction, Kernels, Continuos Data, Validation

Introduction

For app developers, the funding source for the next project depends on the success of their published apps. The app's popularity in continuous active users partially depends on the app store. Consumers review multiple areas of the app's review page than its volume of downloads: whether the app won any awards, the average rating, the download price, etc. Apps with quality ratings receive a bonus with their position in the app store at the highest viewability and attractive banners on their popularity. Data science can inform the project team of their predicted rating in the app store and re-strategize the next project. For this project, the team will assess which regression model best fits app rating predictions.

Literature Review

The literature review navigated the features used in the model creation and the types of regressions that will best fit the dataset. For the project, the team will ensure the price of the app, the date of the latest update, categories are features included based on the journals researched below.

Background on the features The paid status of the app is one of the features highlighted of great importance. In Lee and Raghu (2014), the team noted that free-to-download app was a statistically significant feature in two of the four generalized linear models. This significance of the free status in their regression model saw a 1.7x increase in the rating in the top performance as their project focused on the sustainability of the top-performing apps in the app store (pg.157). In contrast, free status harms favorability, as Wondwesen

(2023) found through an ordinary least squares regression model that free status apps receive a lower rating than paid status apps. This can be from the app experience between free-to-download apps and the paid version, as free-to-downloads may have simpler features and advertisement might lower the app experience (pg.12).

Another feature for inclusion in the project's chosen regression model is the last update date. In Lee and Raghu (2014), their regression model highlighted the importance of the last update in their sustainability model as apps were more likely to maintain their position in the top-performing apps list by 2.9x (pg. 159). The last app update was broken down into different subsets by Wu et al. (2021), as different types of updates provided different increases in the app rating. Their research team noticed updates in functionality on the main features of the app and saw an increase in positive reviews (pg.938). Shifting focus onto the algorithms used in the research surrounding app reviews.

Background on alogrithms The algorithms used in the research journals were mixed as each team had a different measure of app rating. In We et al (2021), the team uses logistic regression for their review of the app store rating. Their model's average accuracy in its prediction was 86%; however, their scope was on the sentiment of the reviews rather than the rating. The team Kapoor and Vij (2020) approach their prediction logistically, as they transformed their discrete response to binary. This transformation gave the team insight into features important between a low/high rating rating, but cannot predict the app rating. The only journal that utilized linear regression was Wondwesen (2023), as their team used ordinary least squares regression in rating prediction. The team will keep in mind that the data set provided response variable is continuous, so the options in regression models are limited.

Methodology

The team created a process for the rating predictions: data pre-processing, exploration, preparation, model building, model selection, experiment and results.For the project's data, the team sourced a scraped pull from the Google App store by Prakash and Koshy (2021) on Kaggle. The data set at first glance need to be processed before the team can explore the data set's statistics.

Data Preprocessing Data Subset

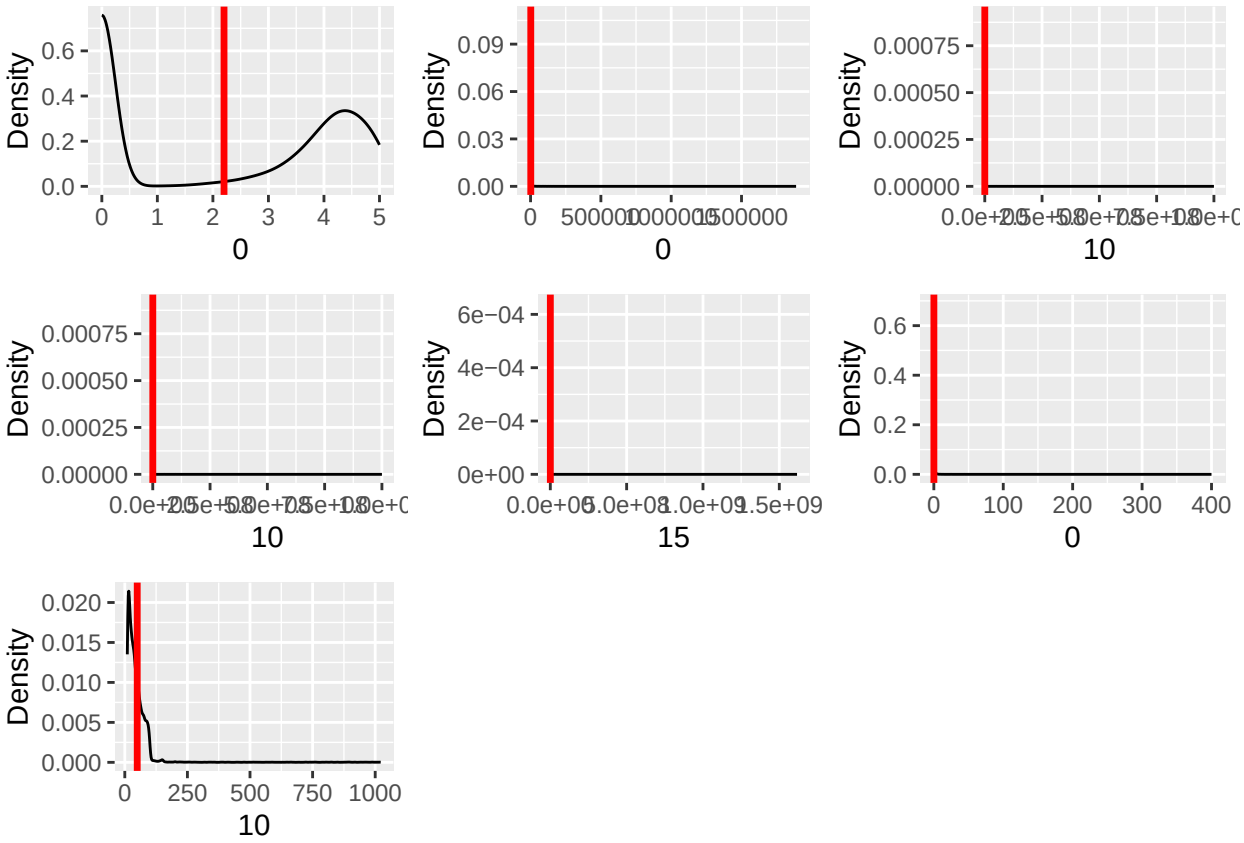
The data was pre-processed using the tidyverse package. First, a column subset was taken which yielded a total of 18 variables: 17 of those being predictors (x) and one target variable (y). The target variable, also known as the response variable, is 'Rating'. 'This column 'Rating' contains the accumulated average score of a given app between 0 and 5 stars.

Missing data

Missing data was handled by removing NA values. Two reasons prompted the omission of missing values: 1) removing missing values only removed about 5% of observations, 2) It is not known if the missing data points are missing completely at random (MCAR), hence imputation could introduce bias to the model.

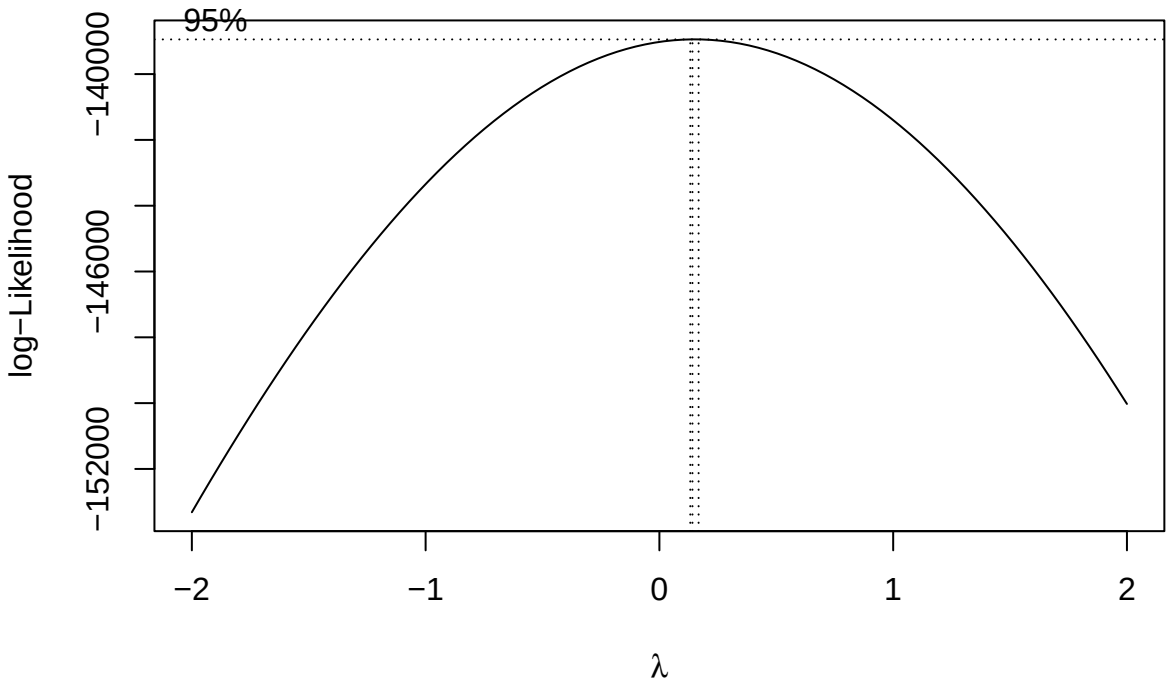
Data Exploration

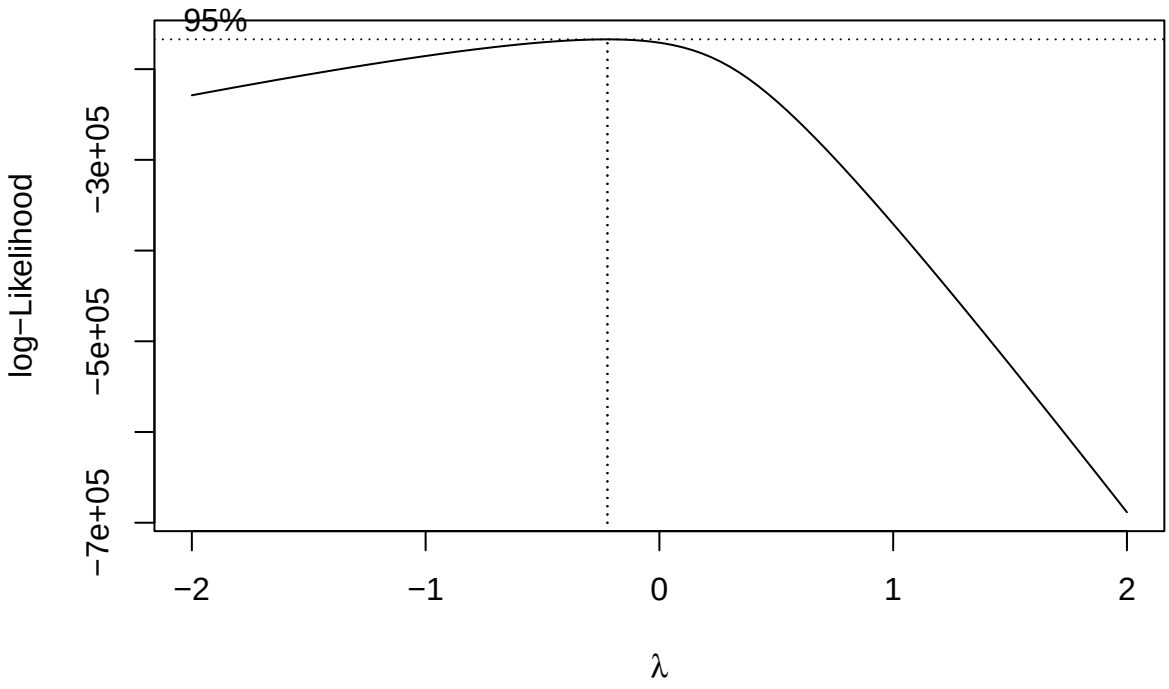
```
##
## Attaching package: 'ggpubr'

## The following object is masked from 'package:forecast':
##
##     gghistogram
```



Distribution of Numeric Variables:

All numerical variables were found to be highly skewed. Transformations such as box-cox, logarithmic and square-root were explored. However, they did not normalize the data. Consequently, several assumptions are violated: 1) Normality, 2) linearity. Ultimately, this leads to assuming that the data exhibits complex relationships where a Nonparametric Regression could be the best choice.


95%
log−Likelihood
−140000
−146000
−152000
−2
−1
0
1
2
λ



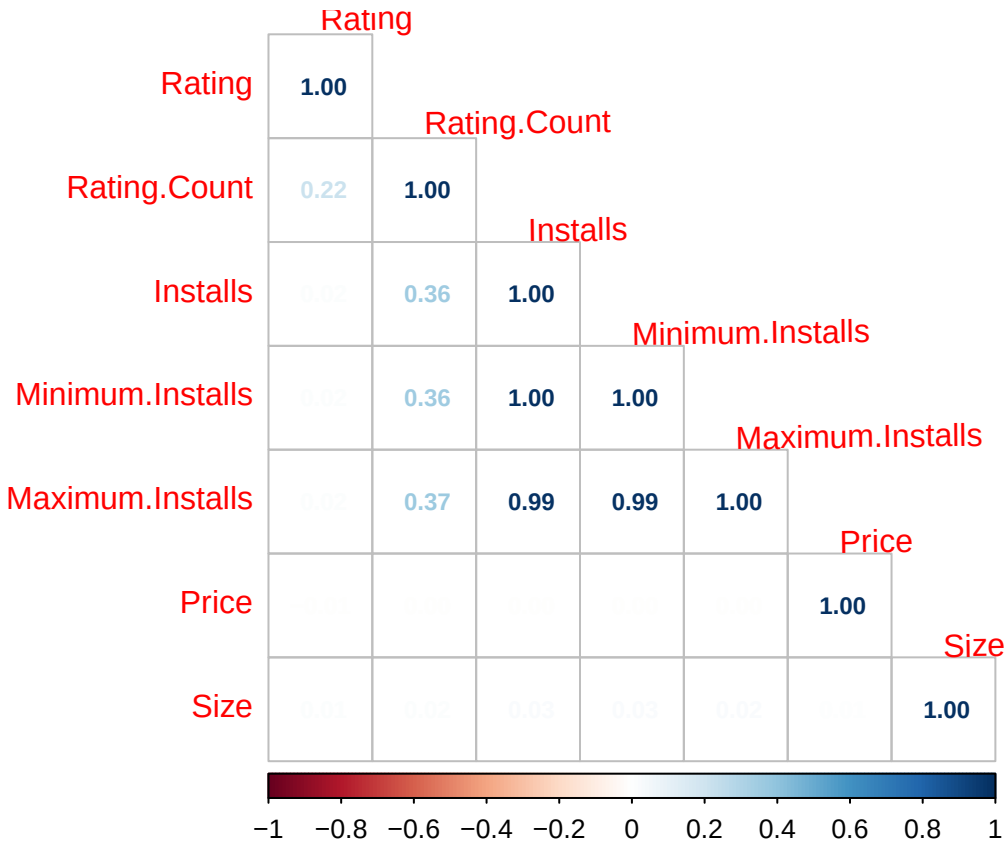
Categorial Variables:

Categorical variables can easily be handled by most R modeling functions. As a caveat, some modeling functions require categorical predictors to be of class factor. During the dplyr data wrangling segment, all categorical variables were converted to factor type using a list.

Checking for Multi-Collinearity:

|'Maximum.Installs' and 'Minimum.Installs' were found to be highly correlated to 'Installs'.

```
## corrplot 0.92 loaded
```



Model Building

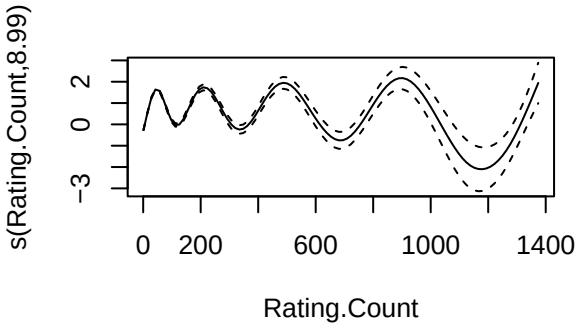
Testing GAM Modeling: The team decided to shift the focus from linear models towards Generalized Additive Models (GAM), as the previous linear model attempts produced poor results. In the first model below, the R^2 value increase to 30% with 30% of the deviance accounted. This improvement in the results shows GAM may be a better regression model for fitting the data provided.

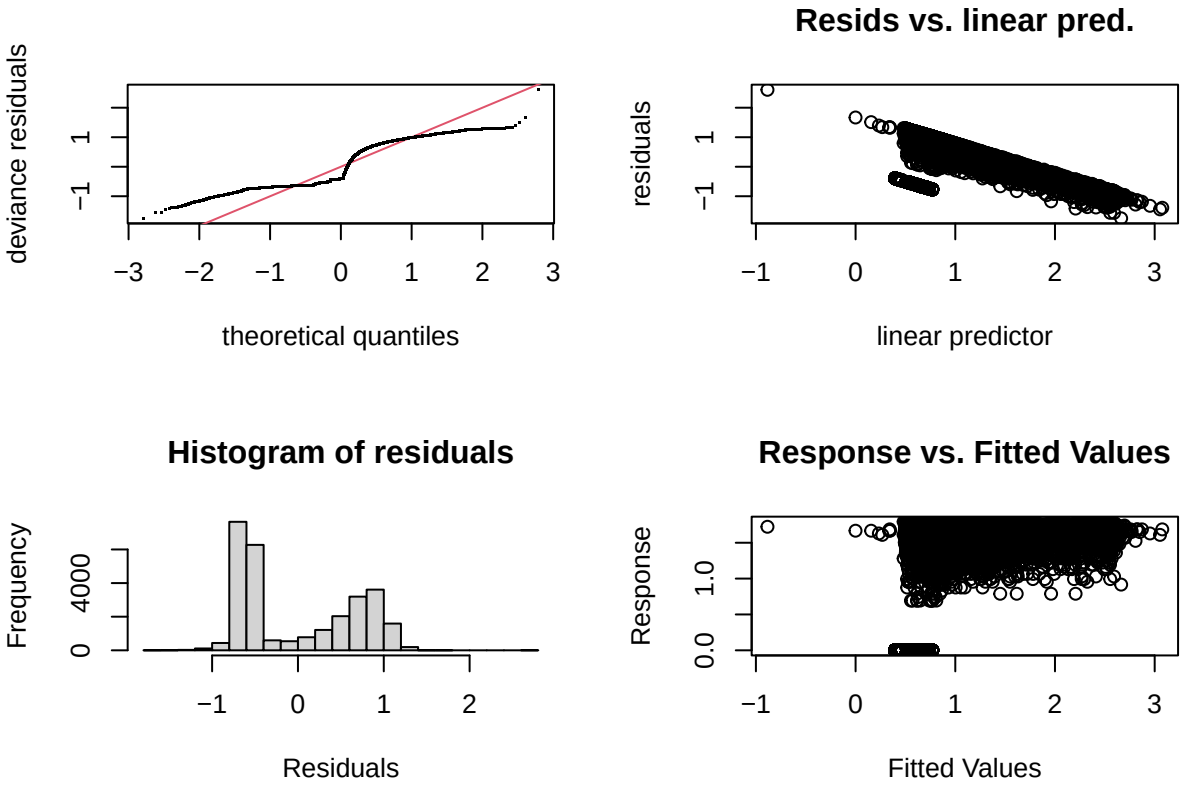
The team did notice the R^2 increase with a smoothing function on Rating count. Rating count distribution is highly skewed,as many cases are below <10. The smoothing function provided additional support in shifting the best fit around the feature to a better angle.

```
## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:forecast':
##
##     getResponse

## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-38. For overview type 'help("mgcv-package")'.

##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rating ~ Category + s(Rating.Count)
##
## Parametric coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.876336   0.037001  23.684  < 2e-16 ***
## CategoryAdventure              0.102716   0.054577   1.882  0.05984 .
## CategoryArcade                 0.108581   0.046060   2.357  0.01841 *
## CategoryArt & Design          -0.071333   0.058887  -1.211  0.22577
## CategoryAuto & Vehicles        0.042156   0.059304   0.711  0.47719
## CategoryBeauty                -0.161486   0.068174  -2.369  0.01786 *
## CategoryBoard                  0.157829   0.068635   2.300  0.02148 *
## CategoryBooks & Reference      0.052939   0.040955   1.293  0.19616
## CategoryBusiness              -0.191480   0.040490  -4.729 2.27e-06 ***
## CategoryCard                  -0.089461   0.081990  -1.091  0.27522
## CategoryCasino                -0.041824   0.098975  -0.423  0.67261
## CategoryCasual                 0.044247   0.045649   0.969  0.33241
## CategoryComics                -0.179650   0.126668  -1.418  0.15612
## CategoryCommunication         -0.027565   0.046808  -0.589  0.55593
## CategoryDating                 0.044089   0.086203   0.511  0.60904
## CategoryEducation              0.035485   0.039054   0.909  0.36357
## CategoryEducational            0.068277   0.056794   1.202  0.22930
## CategoryEntertainment          0.056653   0.040515   1.398  0.16203
## CategoryEvents                -0.214327   0.066579  -3.219  0.00129 **
## CategoryFinance                0.027858   0.043904   0.635  0.52575
## CategoryFood & Drink          -0.176770   0.043287  -4.084 4.45e-05 ***
## CategoryHealth & Fitness      -0.200757   0.042619  -4.710 2.48e-06 ***
## CategoryHouse & Home          -0.166230   0.062567  -2.657  0.00789 **
## CategoryLibraries & Demo      -0.147138   0.090922  -1.618  0.10561
## CategoryLifestyle             -0.080694   0.041125  -1.962  0.04976 *
## CategoryMaps & Navigation     -0.020654   0.051542  -0.401  0.68863
## CategoryMedical               -0.083021   0.050598  -1.641  0.10085
## CategoryMusic                  0.119688   0.089241   1.341  0.17987
## CategoryMusic & Audio          0.030854   0.039963   0.772  0.44008
## CategoryNews & Magazines       0.098161   0.047619   2.061  0.03927 *
## CategoryParenting             -0.024464   0.112881  -0.217  0.82843
## CategoryPersonalization        0.113211   0.042040   2.693  0.00709 **
## CategoryPhotography            0.007807   0.049182   0.159  0.87388
## CategoryProductivity          -0.093525   0.042791  -2.186  0.02885 *
## CategoryPuzzle                 0.086796   0.046154   1.881  0.06004 .
## CategoryRacing                 0.108154   0.069970   1.546  0.12218
## CategoryRole Playing           0.048771   0.071245   0.685  0.49363
## CategoryShopping              -0.076451   0.042964  -1.779  0.07519 .
## CategorySimulation             0.063894   0.054522   1.172  0.24125
## CategorySocial                 0.071417   0.046472   1.537  0.12436
## CategorySports                 0.015942   0.047117   0.338  0.73510
## CategoryStrategy              -0.077917   0.073683  -1.057  0.29031
## CategoryTools                  0.033476   0.040441   0.828  0.40780
```

```
## CategoryTravel & Local          -0.039129   0.043630  -0.897  0.36981
## CategoryTrivia                  -0.047677   0.073454  -0.649  0.51630
## CategoryVideo Players & Editors  0.027881   0.062581   0.446  0.65595
## CategoryWeather                  0.044660   0.086682   0.515  0.60640
## CategoryWord                     0.124663   0.077846   1.601  0.10930
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                    edf Ref.df    F p-value
## s(Rating.Count) 8.989      9 1256  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.318   Deviance explained = 31.9%
## GCV = 0.45585  Scale est. = 0.45493   n = 28246
```






```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 11 iterations.
## The RMS GCV score gradient at convergence was 2.413531e-07 .
## The Hessian was positive definite.
## Model rank =  57 / 57
##
## 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(Rating.Count) 9.00 8.99    0.05  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Introducing shrinking and other smoothing techinques: From the data exploration, the team is aware of the non-linearity of the features in the data set. The current features may not have a linear fit, but its best fit shape can be a polynomial. Using splines, its adjusting the the line with penalties towards mse; which makes the shape more linear. In order to capture the majority of the points in the data set, the best fit line may take the shape of different curves to better fit the response Rating. The team used shrinking smoothed in the features provided as it lessens the penalty.

For the models below, the team use splines with a mix of basis. In addition, the method REML (Restricted Maximum Likelihood) was applied as the model has random effects in the features .For Model two, the cubic spline is applied for the features Installs and Rating.counts. These additional splines boosted the R^2 of the model to 72% with its deviance at 72%. In the plots below, the splines of rating count have a closer fit onto

the feature than the previous iteration. Checking the gam, the model did coverage with the terms provided. However, the p-values of the smoothed items are significant. This means the smoothing knots need to be increased as the number of bases are seeing a pattern in the model's residuals.

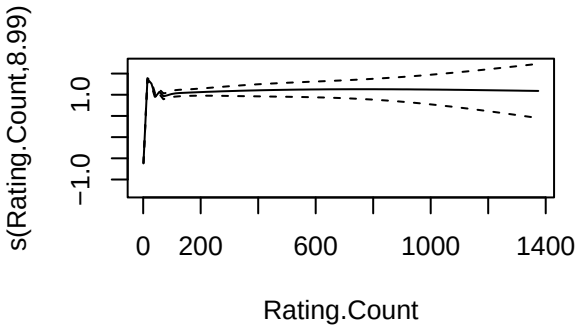
In the third iteration, the team added more bases to the smoothing function. For the feature Installs, the increases bases improved the p-value out of significance. Rating.count p-value still broke the null hypothesis, which points the feature will need more base functions to correct the null.

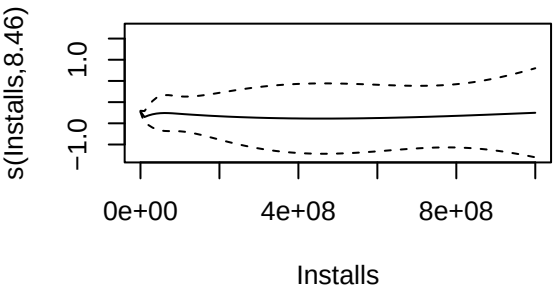
```
#reviewing previous model, apply cubic splines and updating method to reml
gam1<- gam(Rating ~s(Rating.Count,bs ='cs') +
                 s(Installs,bs ='cs')+Category+Ad.Supported+In.App.Purchases
               , data = app.store.train,method = "REML")

summary(gam1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rating ~ s(Rating.Count, bs = "cs") + s(Installs, bs = "cs") +
##     Category + Ad.Supported + In.App.Purchases
##
## Parametric coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.886232   0.024111  36.757  < 2e-16 ***
## CategoryAdventure              0.028653   0.034767   0.824 0.409869
## CategoryArcade                 0.103253   0.029344   3.519 0.000434 ***
## CategoryArt & Design          -0.061032   0.037554  -1.625 0.104134
## CategoryAuto & Vehicles       -0.010386   0.037928  -0.274 0.784210
## CategoryBeauty                -0.045290   0.043490  -1.041 0.297704
## CategoryBoard                  0.048240   0.043719   1.103 0.269854
## CategoryBooks & Reference     -0.031293   0.026159  -1.196 0.231593
## CategoryBusiness              -0.058024   0.026109  -2.222 0.026269 *
## CategoryCard                  -0.037508   0.052230  -0.718 0.472677
## CategoryCasino                -0.053068   0.063143  -0.840 0.400664
## CategoryCasual                 0.069039   0.029065   2.375 0.017538 *
## CategoryComics                -0.095030   0.080680  -1.178 0.238865
## CategoryCommunication         -0.016878   0.029988  -0.563 0.573544
## CategoryDating                -0.049776   0.054928  -0.906 0.364841
## CategoryEducation             -0.003790   0.025003  -0.152 0.879514
## CategoryEducational            0.028430   0.036205   0.785 0.432307
## CategoryEntertainment         -0.017833   0.025887  -0.689 0.490908
## CategoryEvents                -0.081066   0.042550  -1.905 0.056769 .
## CategoryFinance               -0.050544   0.028209  -1.792 0.073178 .
## CategoryFood & Drink          -0.039357   0.027795  -1.416 0.156795
## CategoryHealth & Fitness      -0.086886   0.027339  -3.178 0.001484 **
## CategoryHouse & Home          -0.109224   0.039972  -2.733 0.006289 **
## CategoryLibraries & Demo      -0.087977   0.057968  -1.518 0.129108
## CategoryLifestyle             -0.035756   0.026346  -1.357 0.174739
## CategoryMaps & Navigation     -0.024767   0.033023  -0.750 0.453265
## CategoryMedical               -0.035206   0.032460  -1.085 0.278117
## CategoryMusic                 -0.019978   0.056852  -0.351 0.725285
## CategoryMusic & Audio         -0.014730   0.025526  -0.577 0.563899
```

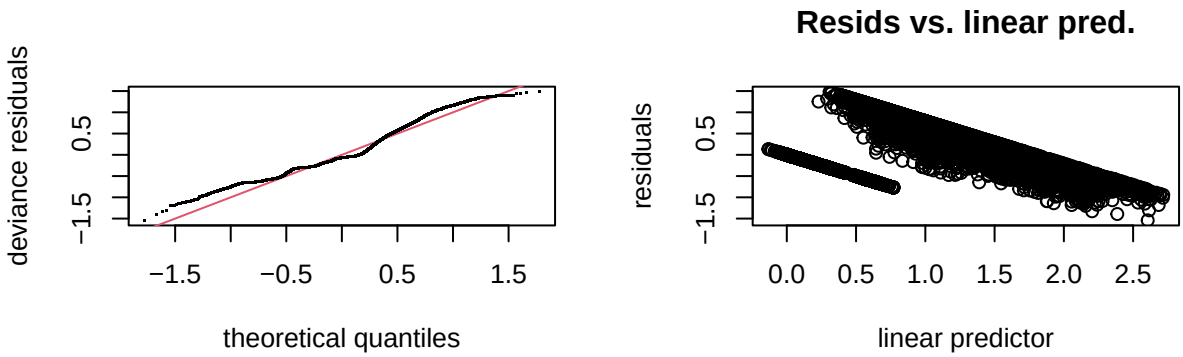
```
## CategoryNews & Magazines        -0.022372   0.030371  -0.737 0.461358
## CategoryParenting                0.060616   0.071926   0.843 0.399373
## CategoryPersonalization         -0.026906   0.026881  -1.001 0.316873
## CategoryPhotography             -0.053137   0.031424  -1.691 0.090850 .
## CategoryProductivity            -0.039925   0.027447  -1.455 0.145777
## CategoryPuzzle                   0.013883   0.029392   0.472 0.636681
## CategoryRacing                  -0.021462   0.044587  -0.481 0.630266
## CategoryRole Playing            -0.058801   0.045420  -1.295 0.195465
## CategoryShopping                -0.001409   0.027658  -0.051 0.959371
## CategorySimulation              -0.021949   0.034756  -0.632 0.527704
## CategorySocial                   0.005198   0.029716   0.175 0.861149
## CategorySports                  -0.048762   0.030079  -1.621 0.105001
## CategoryStrategy                -0.024697   0.046964  -0.526 0.598984
## CategoryTools                   -0.013053   0.025862  -0.505 0.613778
## CategoryTravel & Local          -0.021071   0.027965  -0.753 0.451162
## CategoryTrivia                   0.032660   0.046792   0.698 0.485191
## CategoryVideo Players & Editors -0.077981   0.039891  -1.955 0.050612 .
## CategoryWeather                 -0.088102   0.055232  -1.595 0.110698
## CategoryWord                     0.064489   0.049625   1.300 0.193778
## Ad.SupportedTRUE                 0.005398   0.006007   0.899 0.368868
## In.App.PurchasesTRUE            -0.010627   0.010272  -1.035 0.300864
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df      F p-value
## s(Rating.Count) 8.992      9 2280.9  <2e-16 ***
## s(Installs)     8.455      9  640.6  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.723   Deviance explained = 72.4%
## -REML =  16406  Scale est. = 0.18445   n = 28246
```

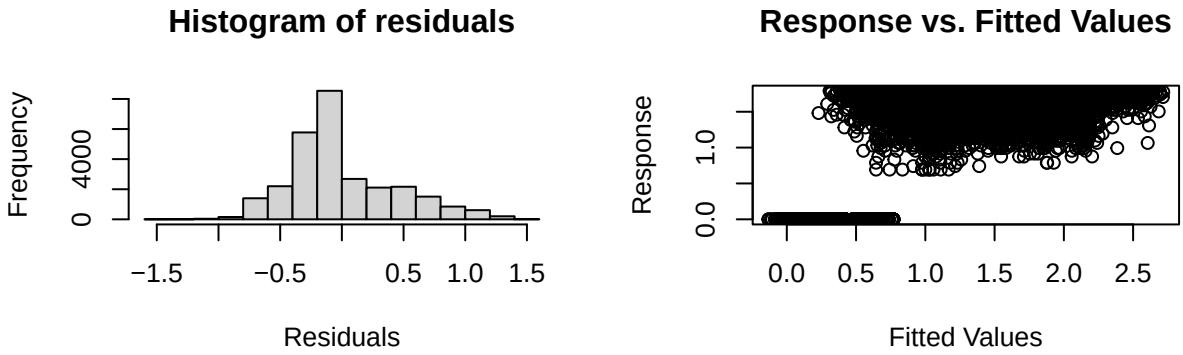
```
par(mfrow = c(2, 2))
plot(gam1)

par(mfrow = c(2, 2))
```






```
gam.check(gam1)
```






```
##
## Method: REML   Optimizer: outer newton
## full convergence after 13 iterations.
## Gradient range [-0.009266724,0.007954757]
## (score 16405.92 & scale 0.1844488).
## Hessian positive definite, eigenvalue range [1.33556,14095.51].
## Model rank =  68 / 68
##
## 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(Rating.Count) 9.00 8.99    0.24  <2e-16 ***
## s(Installs)     9.00 8.46    0.96   0.005 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

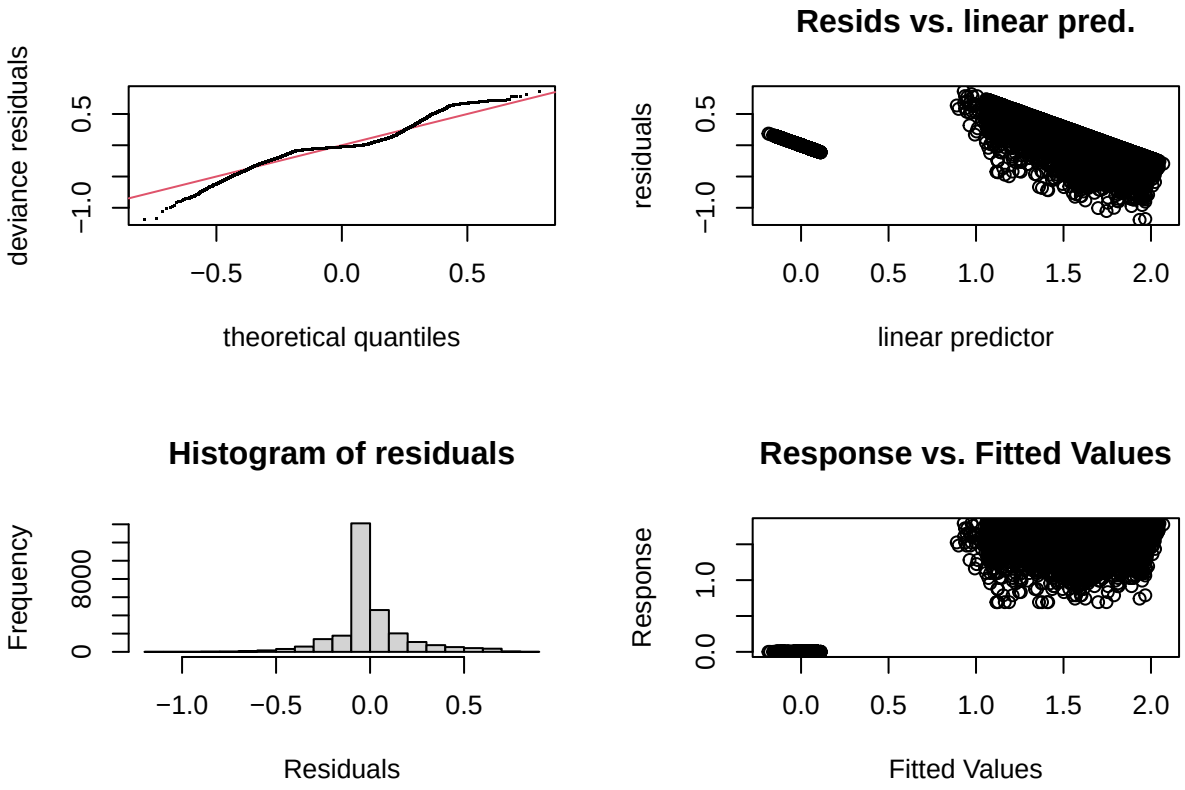
```
#Increasing the basis functions in the smoothing items
gam2<- gam(Rating ~s(Rating.Count,bs="cs",k=100) +
                  s(Installs,bs="cs",k=15)+Category+Ad.Supported+In.App.Purchases, data = app.store.trai
summary(gam2)
```

```
##
## Family: gaussian
## Link function: identity
```

```
##
## Formula:
## Rating ~ s(Rating.Count, bs = "cs", k = 100) + s(Installs, bs = "cs",
##     k = 15) + Category + Ad.Supported + In.App.Purchases
##
## Parametric coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      0.8411833  0.0107114  78.532  < 2e-16 ***
## CategoryAdventure                0.0128011  0.0154388   0.829 0.407026
## CategoryArcade                   0.0358445  0.0130278   2.751 0.005938 **
## CategoryArt & Design             0.0027729  0.0166655   0.166 0.867853
## CategoryAuto & Vehicles          0.0083279  0.0168306   0.495 0.620741
## CategoryBeauty                   0.0305568  0.0192833   1.585 0.113063
## CategoryBoard                   -0.0029274  0.0194065  -0.151 0.880097
## CategoryBooks & Reference        0.0407299  0.0116199   3.505 0.000457 ***
## CategoryBusiness                 0.0212637  0.0115947   1.834 0.066678 .
## CategoryCard                    -0.0192775  0.0232268  -0.830 0.406562
## CategoryCasino                  -0.0305994  0.0280677  -1.090 0.275634
## CategoryCasual                   0.0083190  0.0129161   0.644 0.519526
## CategoryComics                  -0.0059056  0.0357801  -0.165 0.868904
## CategoryCommunication            0.0249516  0.0133102   1.875 0.060855 .
## CategoryDating                  -0.0089680  0.0243723  -0.368 0.712908
## CategoryEducation                0.0325883  0.0111085   2.934 0.003353 **
## CategoryEducational              0.0220572  0.0160718   1.372 0.169945
## CategoryEntertainment            0.0145564  0.0115009   1.266 0.205642
## CategoryEvents                   0.0014345  0.0188656   0.076 0.939389
## CategoryFinance                  0.0001282  0.0125266   0.010 0.991837
## CategoryFood & Drink             0.0212010  0.0123402   1.718 0.085800 .
## CategoryHealth & Fitness         0.0046638  0.0121387   0.384 0.700826
## CategoryHouse & Home            -0.0338125  0.0177301  -1.907 0.056522 .
## CategoryLibraries & Demo        -0.0117397  0.0257142  -0.457 0.648001
## CategoryLifestyle                0.0245952  0.0117016   2.102 0.035574 *
## CategoryMaps & Navigation        0.0035185  0.0146545   0.240 0.810257
## CategoryMedical                  0.0165489  0.0144027   1.149 0.250558
## CategoryMusic                   -0.0189875  0.0252429  -0.752 0.451943
## CategoryMusic & Audio            0.0306534  0.0113397   2.703 0.006872 **
## CategoryNews & Magazines         0.0232081  0.0134840   1.721 0.085234 .
## CategoryParenting                0.0488286  0.0319036   1.531 0.125904
## CategoryPersonalization          0.0477863  0.0119388   4.003 6.28e-05 ***
## CategoryPhotography              0.0029816  0.0139505   0.214 0.830762
## CategoryProductivity             0.0117926  0.0121887   0.968 0.333302
## CategoryPuzzle                   0.0189556  0.0130514   1.452 0.146405
## CategoryRacing                  -0.0172154  0.0198374  -0.868 0.385496
## CategoryRole Playing             0.0068727  0.0202026   0.340 0.733717
## CategoryShopping                 0.0297091  0.0122815   2.419 0.015569 *
## CategorySimulation              -0.0122668  0.0154450  -0.794 0.427070
## CategorySocial                   0.0154719  0.0131963   1.172 0.241032
## CategorySports                   0.0017260  0.0133565   0.129 0.897183
## CategoryStrategy                -0.0124408  0.0208894  -0.596 0.551477
## CategoryTools                    0.0071624  0.0114877   0.623 0.532975
## CategoryTravel & Local           0.0191664  0.0124172   1.544 0.122711
## CategoryTrivia                   0.0289766  0.0207618   1.396 0.162825
## CategoryVideo Players & Editors -0.0232262  0.0177170  -1.311 0.189884
## CategoryWeather                  0.0173610  0.0245362   0.708 0.479220
```

```
## CategoryWord                    0.0650006  0.0220441   2.949 0.003194 **
## Ad.SupportedTRUE                0.0127698  0.0026642   4.793 1.65e-06 ***
## In.App.PurchasesTRUE           -0.0020792  0.0045669  -0.455 0.648907
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                    edf Ref.df    F p-value
## s(Rating.Count) 75.12     99 2207  <2e-16 ***
## s(Installs)     12.04     14   84  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.946   Deviance explained = 94.6%
## -REML = -6387.9  Scale est. = 0.036175  n = 28246
```

```
gam.check(gam2)
```




```
##
## Method: REML   Optimizer: outer newton
## full convergence after 4 iterations.
## Gradient range [-0.0006836017,5.906114e-08]
## (score -6387.869 & scale 0.03617524).
## Hessian positive definite, eigenvalue range [1.365487,14092.6].
## Model rank =  163 / 163
```

```
##
## 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(Rating.Count) 99.0 75.1    0.32  <2e-16 ***
## s(Installs)     14.0 12.0    1.01    0.62
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The fourth version of the gam model adds the penalty to the model. There was not a big change the results from the previous.

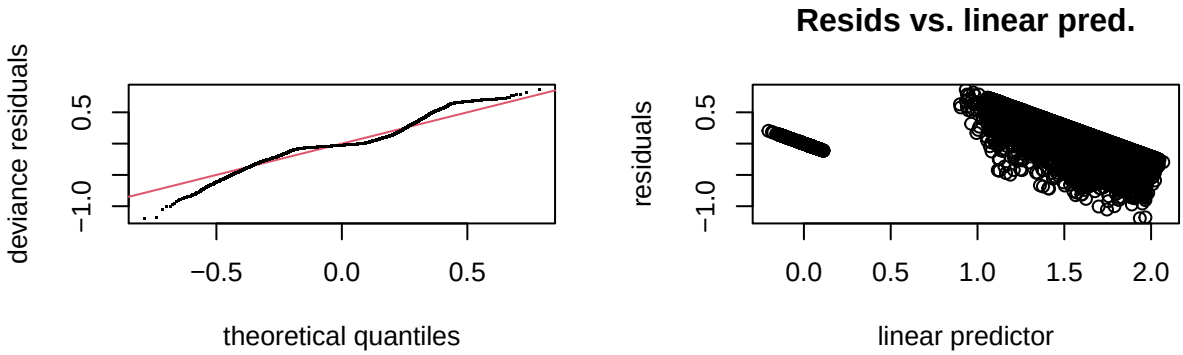
In the next model, the rating count smoothing was removed and editors choice, content rating, and last updated to the model. Although the R^2 dropped to 52%, the fitted vs actual plot appears more normal. Checking the AIC and a few other metrics, it does appear the fourth model has the advantage than the model with the higher log likelihood and lower deviance.

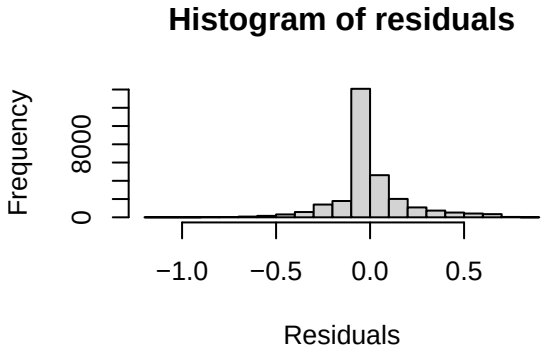
```
#Increasing the basis functions in rating count and add extra penalty
gam3<- gam(Rating ~s(Rating.Count,bs="cs",k=100) +
                 s(Installs,bs="cs",k=15)+Category+Ad.Supported+In.App.Purchases+Price, data = app.stor

summary(gam3)
```

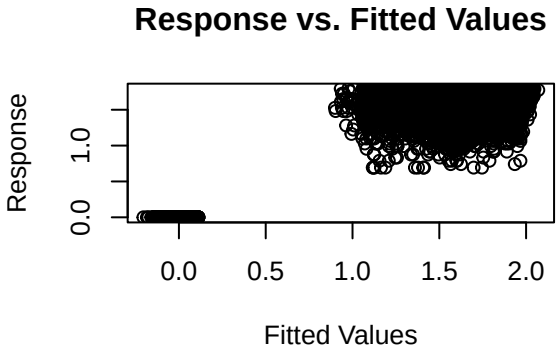
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rating ~ s(Rating.Count, bs = "cs", k = 100) + s(Installs, bs = "cs",
##     k = 15) + Category + Ad.Supported + In.App.Purchases + Price
##
## Parametric coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   0.8410683  0.0107087  78.540  < 2e-16 ***
## CategoryAdventure             0.0131809  0.0154361   0.854 0.393169
## CategoryArcade                0.0362480  0.0130253   2.783 0.005391 **
## CategoryArt & Design          0.0030105  0.0166637   0.181 0.856635
## CategoryAuto & Vehicles       0.0084906  0.0168287   0.505 0.613890
## CategoryBeauty                0.0307668  0.0192815   1.596 0.110575
## CategoryBoard                -0.0021880  0.0194041  -0.113 0.910224
## CategoryBooks & Reference     0.0411506  0.0116168   3.542 0.000397 ***
## CategoryBusiness              0.0215115  0.0115918   1.856 0.063500 .
## CategoryCard                 -0.0186334  0.0232239  -0.802 0.422364
## CategoryCasino               -0.0299304  0.0280654  -1.066 0.286228
## CategoryCasual                0.0086341  0.0129142   0.669 0.503776
## CategoryComics               -0.0055742  0.0357788  -0.156 0.876194
## CategoryCommunication         0.0252039  0.0133071   1.894 0.058232 .
## CategoryDating               -0.0085664  0.0243705  -0.352 0.725210
## CategoryEducation             0.0328999  0.0111053   2.963 0.003054 **
## CategoryEducational           0.0221472  0.0160702   1.378 0.168169
## CategoryEntertainment         0.0148265  0.0114978   1.290 0.197233
## CategoryEvents                0.0016307  0.0188639   0.086 0.931114
## CategoryFinance               0.0005974  0.0125216   0.048 0.961946
```

```
## CategoryFood & Drink             0.0213913  0.0123377   1.734 0.082962 .
## CategoryHealth & Fitness         0.0049303  0.0121358   0.406 0.684555
## CategoryHouse & Home            -0.0336183  0.0177280  -1.896 0.057925 .
## CategoryLibraries & Demo        -0.0116549  0.0257132  -0.453 0.650362
## CategoryLifestyle                0.0248355  0.0116986   2.123 0.033768 *
## CategoryMaps & Navigation        0.0037747  0.0146521   0.258 0.796698
## CategoryMedical                  0.0173394  0.0144055   1.204 0.228729
## CategoryMusic                   -0.0188408  0.0252415  -0.746 0.455420
## CategoryMusic & Audio            0.0310046  0.0113365   2.735 0.006243 **
## CategoryNews & Magazines         0.0236376  0.0134805   1.753 0.079533 .
## CategoryParenting                0.0491284  0.0319023   1.540 0.123579
## CategoryPersonalization          0.0481107  0.0119359   4.031 5.57e-05 ***
## CategoryPhotography              0.0029273  0.0139482   0.210 0.833773
## CategoryProductivity             0.0119770  0.0121855   0.983 0.325668
## CategoryPuzzle                   0.0192615  0.0130493   1.476 0.139938
## CategoryRacing                  -0.0166189  0.0198066  -0.839 0.401443
## CategoryRole Playing             0.0079508  0.0201968   0.394 0.693831
## CategoryShopping                 0.0299245  0.0122789   2.437 0.014813 *
## CategorySimulation              -0.0121535  0.0154431  -0.787 0.431295
## CategorySocial                   0.0158773  0.0131932   1.203 0.228813
## CategorySports                   0.0021511  0.0133533   0.161 0.872025
## CategoryStrategy                -0.0114885  0.0208853  -0.550 0.582271
## CategoryTools                    0.0075453  0.0114853   0.657 0.511215
## CategoryTravel & Local           0.0194019  0.0124147   1.563 0.118108
## CategoryTrivia                   0.0294569  0.0207596   1.419 0.155925
## CategoryVideo Players & Editors -0.0234057  0.0177143  -1.321 0.186416
## CategoryWeather                  0.0176217  0.0245347   0.718 0.472618
## CategoryWord                     0.0657247  0.0220404   2.982 0.002866 **
## Ad.SupportedTRUE                 0.0125068  0.0026650   4.693 2.70e-06 ***
## In.App.PurchasesTRUE            -0.0021654  0.0045664  -0.474 0.635360
## Price                           -0.0005054  0.0004015  -1.259 0.208056
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                    edf Ref.df       F p-value
## s(Rating.Count) 75.12     99 2209.07  <2e-16 ***
## s(Installs)     10.31     14   83.37  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.946   Deviance explained = 94.6%
## -REML = -6379.3  Scale est. = 0.036175  n = 28246
```

```
gam.check(gam3)
```

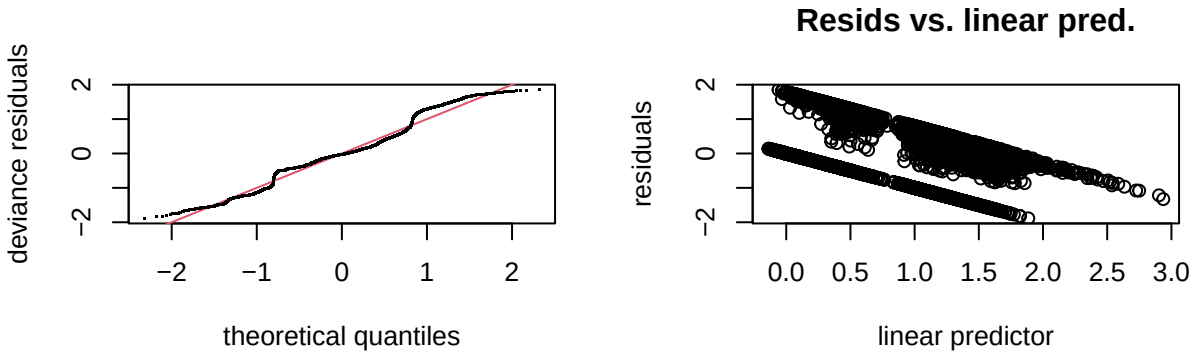


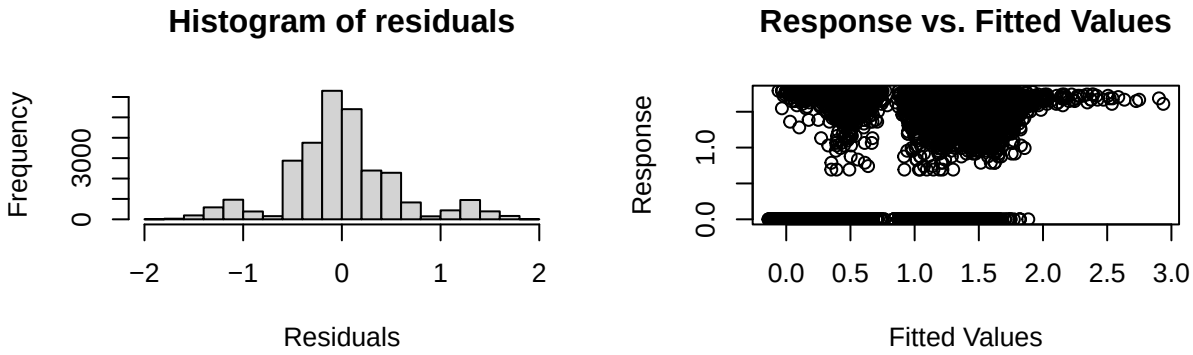





```
##
## Method: REML   Optimizer: outer newton
## full convergence after 13 iterations.
## Gradient range [-6.036449e-06,4.858201e-06]
## (score -6379.309 & scale 0.03617468).
## Hessian positive definite, eigenvalue range [0.9989728,14097.6].
## Model rank =  164 / 164
##
## Basis dimension (k) checking 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(Rating.Count) 99.0 75.1    0.31  <2e-16 ***
## s(Installs)     14.0 10.3    0.99     0.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#remove smoothing on rating count and add in editors choice
gam4<- gam(Rating ~Rating.Count+s(Installs,bs="cs",k=15)+Category+Ad.Supported+In.App.Purchases+Editors
#summary(gam4)
gam.check(gam4)
```






```
##
## Method: REML   Optimizer: outer newton
## full convergence after 7 iterations.
## Gradient range [-0.008567201,0.0117497]
## (score 23908.05 & scale 0.3146651).
## Hessian positive definite, eigenvalue range [0.4637132,14089.51].
## Model rank =  72 / 72
##
## 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(Installs) 14.0 12.3       1    0.52
```

```
#check AIC
AIC(gam3,gam4)
```

```
##             df       AIC
## gam3 137.86165 -13462.02
## gam4  71.52052  47571.99
```

```
library(broom)

glance(gam3)
```

```
## # A tibble: 1 x 7
##      df logLik     AIC     BIC deviance df.residual  nobs
##   <dbl>  <dbl>   <dbl>   <dbl>    <dbl>       <dbl> <int>
## 1  136.  6869. -13462. -12325.    1017.      28110. 28246
```

```
glance(gam4)
```

```
## # A tibble: 1 x 7
##      df  logLik    AIC    BIC deviance df.residual  nobs
##   <dbl>   <dbl>  <dbl>  <dbl>    <dbl>       <dbl> <int>
## 1  70.3 -23714. 47572. 48162.    8866.      28176. 28246
```

Improved model gam 3: For model 3 we decided to increase the k value from 100 to 150 and 15 to 20 in an effort to see if it improved our metrics.

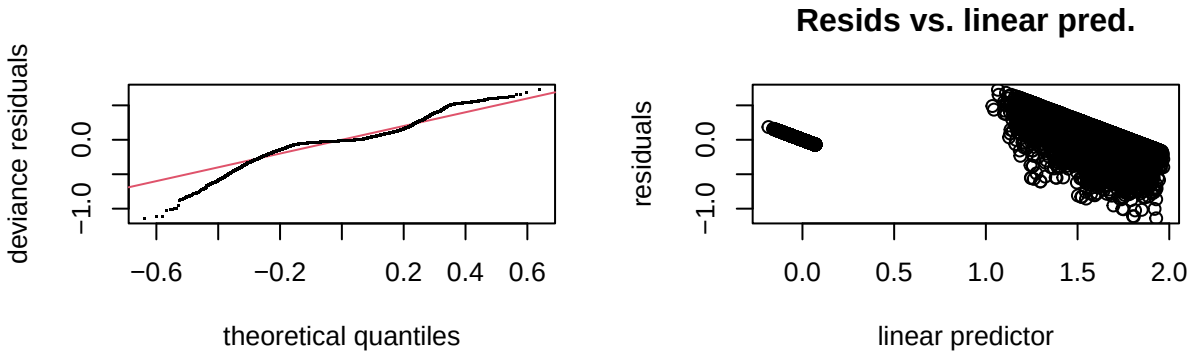
We observed improvements in key areas such as the logLik, AIC, and BIC. In addition, there is a lower deviance in the updated gam3 model.

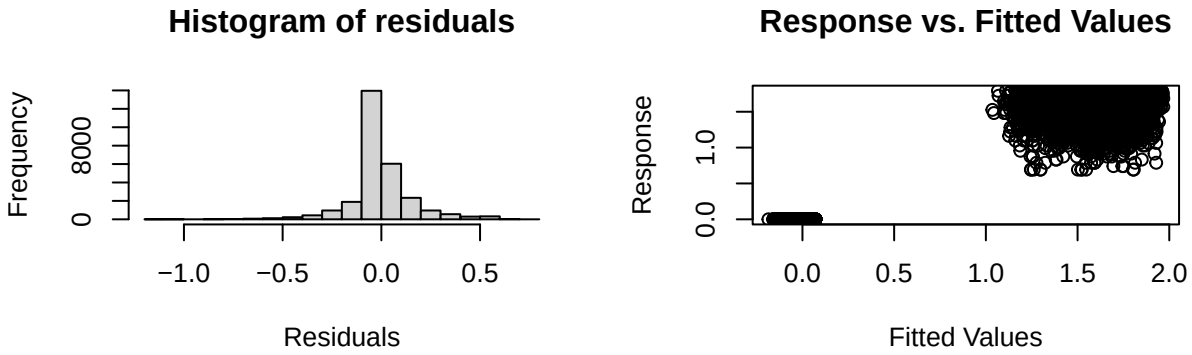
```
#Increased k folds to improve metrics
gam3_update1<- gam(Rating ~s(Rating.Count,bs="cs",k=150) +
                 s(Installs,bs="cs",k=20)+Category+Ad.Supported+In.App.Purchases+Price, data = app.stor

summary(gam3_update1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rating ~ s(Rating.Count, bs = "cs", k = 150) + s(Installs, bs = "cs",
##     k = 20) + Category + Ad.Supported + In.App.Purchases + Price
##
## Parametric coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.8378909  0.0087042  96.263  < 2e-16 ***
## CategoryAdventure              0.0143087  0.0125410   1.141  0.25390
## CategoryArcade                 0.0290436  0.0105846   2.744  0.00607 **
## CategoryArt & Design           0.0038759  0.0135418   0.286  0.77471
## CategoryAuto & Vehicles        0.0094248  0.0136696   0.689  0.49053
## CategoryBeauty                 0.0293035  0.0156618   1.871  0.06135 .
## CategoryBoard                 -0.0069876  0.0157647  -0.443  0.65760
## CategoryBooks & Reference      0.0433708  0.0094400   4.594 4.36e-06 ***
## CategoryBusiness               0.0250091  0.0094191   2.655  0.00793 **
## CategoryCard                  -0.0055891  0.0188815  -0.296  0.76722
## CategoryCasino                -0.0292344  0.0228092  -1.282  0.19996
## CategoryCasual                 0.0067884  0.0104955   0.647  0.51777
## CategoryComics                 0.0035115  0.0290545   0.121  0.90380
## CategoryCommunication          0.0268396  0.0108117   2.482  0.01305 *
## CategoryDating                -0.0171642  0.0198066  -0.867  0.38617
## CategoryEducation              0.0365254  0.0090252   4.047 5.20e-05 ***
```

```
## CategoryEducational              0.0222154  0.0130591   1.701  0.08893 .
## CategoryEntertainment            0.0172096  0.0093426   1.842  0.06548 .
## CategoryEvents                   0.0050359  0.0153192   0.329  0.74236
## CategoryFinance                  0.0029961  0.0101766   0.294  0.76844
## CategoryFood & Drink             0.0241659  0.0100252   2.411  0.01594 *
## CategoryHealth & Fitness         0.0124245  0.0098597   1.260  0.20763
## CategoryHouse & Home            -0.0265805  0.0143989  -1.846  0.06490 .
## CategoryLibraries & Demo        -0.0030949  0.0208859  -0.148  0.88220
## CategoryLifestyle                0.0235936  0.0095072   2.482  0.01308 *
## CategoryMaps & Navigation        0.0034424  0.0119037   0.289  0.77244
## CategoryMedical                  0.0226132  0.0117043   1.932  0.05336 .
## CategoryMusic                   -0.0261506  0.0205105  -1.275  0.20232
## CategoryMusic & Audio            0.0378139  0.0092128   4.105 4.06e-05 ***
## CategoryNews & Magazines         0.0239415  0.0109534   2.186  0.02884 *
## CategoryParenting                0.0499695  0.0259084   1.929  0.05378 .
## CategoryPersonalization          0.0508263  0.0097007   5.239 1.62e-07 ***
## CategoryPhotography              0.0003155  0.0113322   0.028  0.97779
## CategoryProductivity             0.0159239  0.0099022   1.608  0.10782
## CategoryPuzzle                   0.0214401  0.0106079   2.021  0.04327 *
## CategoryRacing                  -0.0288681  0.0161029  -1.793  0.07303 .
## CategoryRole Playing             0.0072576  0.0164215   0.442  0.65852
## CategoryShopping                 0.0288859  0.0099774   2.895  0.00379 **
## CategorySimulation              -0.0107649  0.0125616  -0.857  0.39147
## CategorySocial                   0.0127934  0.0107196   1.193  0.23270
## CategorySports                   0.0028399  0.0108505   0.262  0.79353
## CategoryStrategy                -0.0099616  0.0169991  -0.586  0.55788
## CategoryTools                    0.0052591  0.0093355   0.563  0.57320
## CategoryTravel & Local           0.0232253  0.0100886   2.302  0.02134 *
## CategoryTrivia                   0.0237025  0.0168629   1.406  0.15985
## CategoryVideo Players & Editors -0.0087638  0.0143979  -0.609  0.54274
## CategoryWeather                  0.0200530  0.0199513   1.005  0.31486
## CategoryWord                     0.0348833  0.0179074   1.948  0.05143 .
## Ad.SupportedTRUE                 0.0141255  0.0021654   6.523 6.99e-11 ***
## In.App.PurchasesTRUE             0.0048077  0.0037113   1.295  0.19518
## Price                           -0.0001817  0.0003259  -0.558  0.57714
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                    edf Ref.df       F p-value
## s(Rating.Count) 97.52    149 2305.29  <2e-16 ***
## s(Installs)     13.02     19   68.81  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.964   Deviance explained = 96.4%
## -REML = -12207  Scale est. = 0.02384   n = 28246
```

```
gam.check(gam3_update1)
```






```
##
## Method: REML   Optimizer: outer newton
## full convergence after 15 iterations.
## Gradient range [-0.0005655895,0.0005650734]
## (score -12207.09 & scale 0.02383959).
## Hessian positive definite, eigenvalue range [1.163578,14097.66].
## Model rank =  219 / 219
##
## 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(Rating.Count) 149.0  97.5    0.45  <2e-16 ***
## s(Installs)      19.0  13.0    1.02    0.92
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

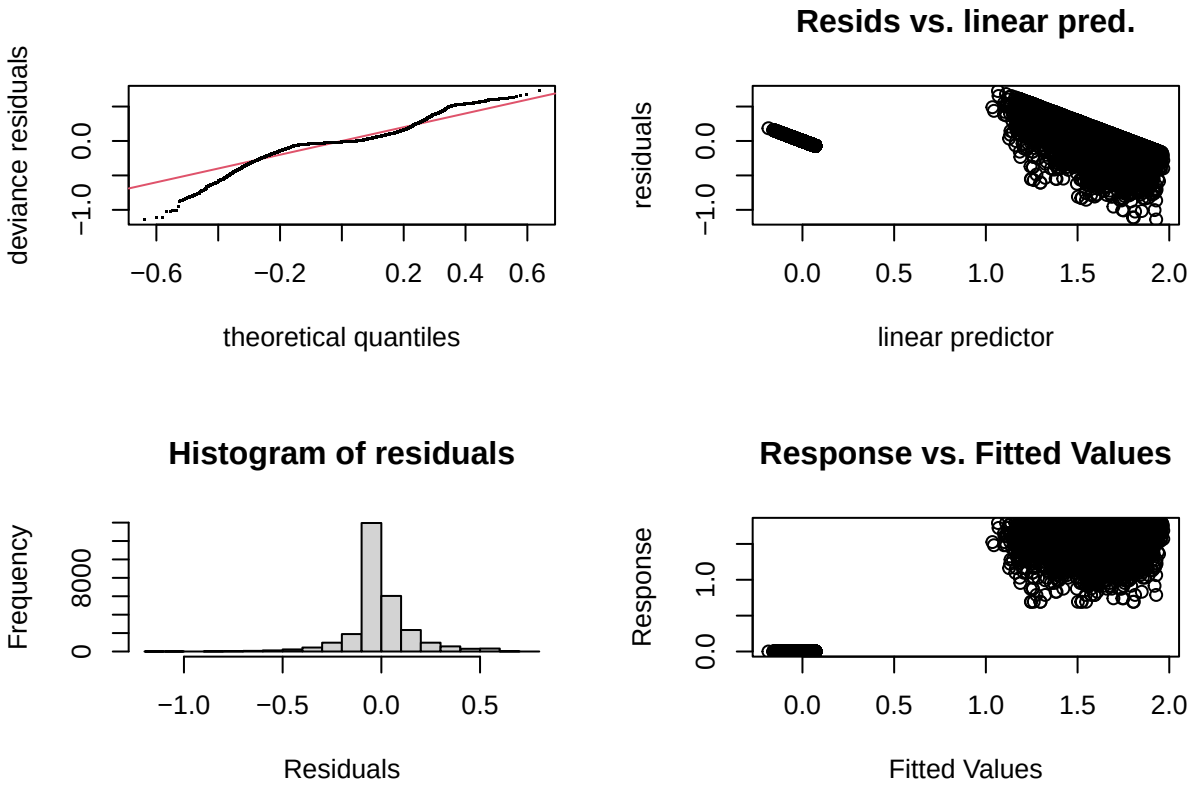
```
glance(gam3)
```

```
## # A tibble: 1 x 7
##      df logLik     AIC     BIC deviance df.residual  nobs
##   <dbl>  <dbl>   <dbl>   <dbl>    <dbl>       <dbl> <int>
## 1  136.  6869. -13462. -12325.    1017.      28110. 28246
```

```
glance(gam3_update1)
```

```
## # A tibble: 1 x 7
##      df logLik     AIC     BIC deviance df.residual  nobs
##   <dbl>  <dbl>   <dbl>   <dbl>    <dbl>       <dbl> <int>
## 1  162. 12771. -25216. -23871.     670.      28084. 28246
```

```
gam.check(gam3_update1)
```

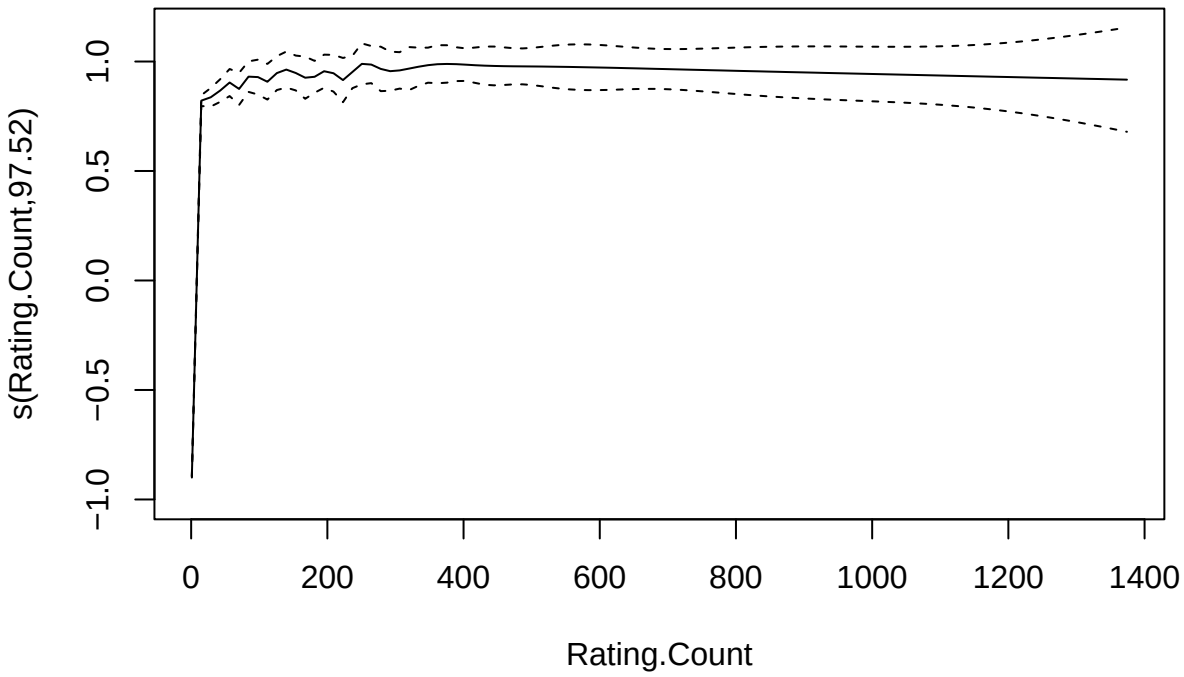


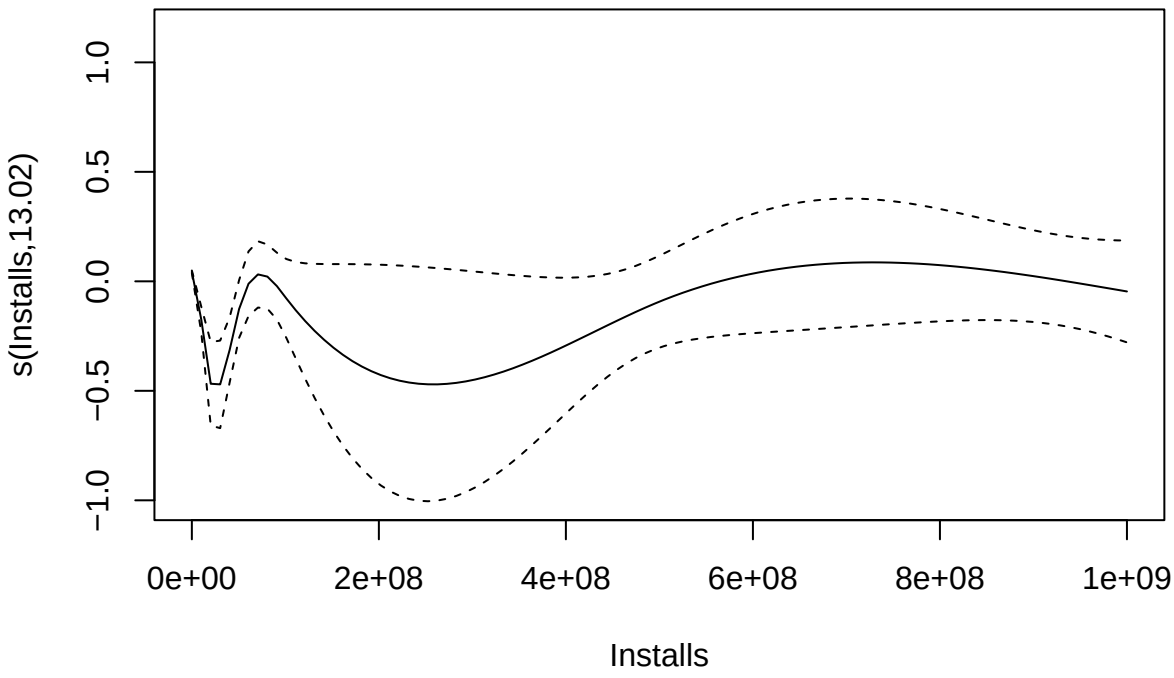

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 15 iterations.
## Gradient range [-0.0005655895,0.0005650734]
## (score -12207.09 & scale 0.02383959).
## Hessian positive definite, eigenvalue range [1.163578,14097.66].
## Model rank =  219 / 219
##
## Basis dimension (k) checking 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(Rating.Count) 149.0  97.5    0.42  <2e-16 ***
## s(Installs)      19.0  13.0    1.01    0.68
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#plot model fit
par(c(5,5))
```

```
## NULL
```

```
plot.gam(gam3_update1)
```





To further support our decision making, we performed an anova test to decide which model to select.

Looking at this side-by-side view, we observe important details. The updated gam3 uses more degrees of freedom (df), as indicated by the decrease in residual degrees of freedom, which indicates more complexity. As noted, the residuals are lower in the updated gam3 model which tell us that the model fits the data better. Lastly, the p-value indicates that the difference between the models is significant. This leads us to select 'gam3_update1' as our preferred model.

```
anova_result <- anova(gam3, gam3_update1, test="Chisq")
print(anova_result)
```

```
## Analysis of Deviance Table
##
## Model 1: Rating ~ s(Rating.Count, bs = "cs", k = 100) + s(Installs, bs = "cs",
##     k = 15) + Category + Ad.Supported + In.App.Purchases + Price
## Model 2: Rating ~ s(Rating.Count, bs = "cs", k = 150) + s(Installs, bs = "cs",
##     k = 20) + Category + Ad.Supported + In.App.Purchases + Price
##   Resid. Df Resid. Dev     Df Deviance  Pr(>Chi)
## 1     28100    1016.85
## 2     28070     669.52 29.993   347.33 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model Selection To finalize the selection of a model, we combined the summary statistics for easy comparison.

Our updated model 3 performs the best. It has the lowest AIC, BIC and also has the highest Log-Likelihood. It also has the lowest deviance. Given the high performance of model 3, we will use this to perform our predictions.

```
gam_model_comp <- bind_rows(glance(gam1), glance(gam2), glance(gam3_update1), glance(gam4))
gam_names1 <- c("gam1","gam2","gam3","gam4")
gam_model_comp <- cbind(model.build = gam_names1, gam_model_comp)
gam_model_comp
```

```
##   model.build        df     logLik       AIC       BIC  deviance df.residual
## 1        gam1  67.44675 -16172.288  32481.90  33048.26 5197.4995    28178.55
## 2        gam2 137.15845   6869.018 -13461.21 -12319.50 1016.8441    28108.84
## 3        gam3 161.54426  12770.952 -25215.76 -23870.61  669.5219    28084.46
## 4        gam4  70.30065 -23714.476  47571.99  48161.94 8865.9092    28175.70
##    nobs
## 1 28246
## 2 28246
## 3 28246
## 4 28246
```

Model Results

From the model selection, improved GAM model 3 was selected as the final model. The final model has the highest R^2 of 96%. This high R^2 does support the need of the spline on the variables "Rating.count" and "STARS", as the R^2 provided once the number of knots increased on these features splines. The team wanted to see if these features are of high importance in the model, so the coefficients were reviewed below.

Variable Importance: From the final model, the team reviewed the current collection of features and their coefficients in the regression model. Ad.SupportedTRUE, CategoryArcade, CategoryBooks & Reference, and CategoryBusiness were identified as statistically significant in the regression model.

The feature of ad supported is not a surprise as Wondwesen (2023) saw ad supported ads were more likely to have lower rating compared to the paid version in the app store.

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Rating ~ s(Rating.Count, bs = "cs", k = 150) + s(Installs, bs = "cs",
##     k = 20) + Category + Ad.Supported + In.App.Purchases + Price
##
## Parametric coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   0.8378909  0.0087042  96.263  < 2e-16 ***
## CategoryAdventure             0.0143087  0.0125410   1.141  0.25390
## CategoryArcade                0.0290436  0.0105846   2.744  0.00607 **
## CategoryArt & Design          0.0038759  0.0135418   0.286  0.77471
## CategoryAuto & Vehicles       0.0094248  0.0136696   0.689  0.49053
## CategoryBeauty                0.0293035  0.0156618   1.871  0.06135 .
## CategoryBoard                -0.0069876  0.0157647  -0.443  0.65760
## CategoryBooks & Reference     0.0433708  0.0094400   4.594 4.36e-06 ***
```

```
## CategoryBusiness                 0.0250091  0.0094191   2.655  0.00793 **
## CategoryCard                    -0.0055891  0.0188815  -0.296  0.76722
## CategoryCasino                  -0.0292344  0.0228092  -1.282  0.19996
## CategoryCasual                   0.0067884  0.0104955   0.647  0.51777
## CategoryComics                   0.0035115  0.0290545   0.121  0.90380
## CategoryCommunication            0.0268396  0.0108117   2.482  0.01305 *
## CategoryDating                  -0.0171642  0.0198066  -0.867  0.38617
## CategoryEducation                0.0365254  0.0090252   4.047 5.20e-05 ***
## CategoryEducational              0.0222154  0.0130591   1.701  0.08893 .
## CategoryEntertainment            0.0172096  0.0093426   1.842  0.06548 .
## CategoryEvents                   0.0050359  0.0153192   0.329  0.74236
## CategoryFinance                  0.0029961  0.0101766   0.294  0.76844
## CategoryFood & Drink             0.0241659  0.0100252   2.411  0.01594 *
## CategoryHealth & Fitness         0.0124245  0.0098597   1.260  0.20763
## CategoryHouse & Home            -0.0265805  0.0143989  -1.846  0.06490 .
## CategoryLibraries & Demo        -0.0030949  0.0208859  -0.148  0.88220
## CategoryLifestyle                0.0235936  0.0095072   2.482  0.01308 *
## CategoryMaps & Navigation        0.0034424  0.0119037   0.289  0.77244
## CategoryMedical                  0.0226132  0.0117043   1.932  0.05336 .
## CategoryMusic                   -0.0261506  0.0205105  -1.275  0.20232
## CategoryMusic & Audio            0.0378139  0.0092128   4.105 4.06e-05 ***
## CategoryNews & Magazines         0.0239415  0.0109534   2.186  0.02884 *
## CategoryParenting                0.0499695  0.0259084   1.929  0.05378 .
## CategoryPersonalization          0.0508263  0.0097007   5.239 1.62e-07 ***
## CategoryPhotography              0.0003155  0.0113322   0.028  0.97779
## CategoryProductivity             0.0159239  0.0099022   1.608  0.10782
## CategoryPuzzle                   0.0214401  0.0106079   2.021  0.04327 *
## CategoryRacing                  -0.0288681  0.0161029  -1.793  0.07303 .
## CategoryRole Playing             0.0072576  0.0164215   0.442  0.65852
## CategoryShopping                 0.0288859  0.0099774   2.895  0.00379 **
## CategorySimulation              -0.0107649  0.0125616  -0.857  0.39147
## CategorySocial                   0.0127934  0.0107196   1.193  0.23270
## CategorySports                   0.0028399  0.0108505   0.262  0.79353
## CategoryStrategy                -0.0099616  0.0169991  -0.586  0.55788
## CategoryTools                    0.0052591  0.0093355   0.563  0.57320
## CategoryTravel & Local           0.0232253  0.0100886   2.302  0.02134 *
## CategoryTrivia                   0.0237025  0.0168629   1.406  0.15985
## CategoryVideo Players & Editors -0.0087638  0.0143979  -0.609  0.54274
## CategoryWeather                  0.0200530  0.0199513   1.005  0.31486
## CategoryWord                     0.0348833  0.0179074   1.948  0.05143 .
## Ad.SupportedTRUE                 0.0141255  0.0021654   6.523 6.99e-11 ***
## In.App.PurchasesTRUE             0.0048077  0.0037113   1.295  0.19518
## Price                           -0.0001817  0.0003259  -0.558  0.57714
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df       F p-value
## s(Rating.Count) 97.52    149 2305.29  <2e-16 ***
## s(Installs)     13.02     19   68.81  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.964   Deviance explained = 96.4%
```

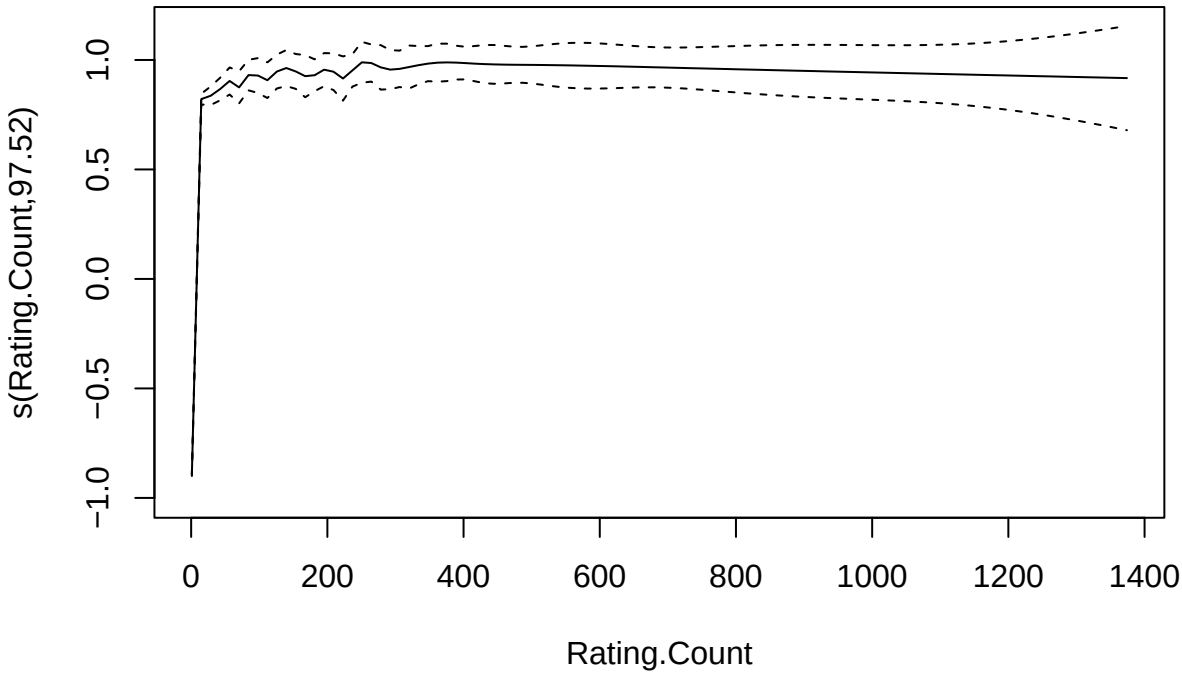
```
## -REML = -12207  Scale est. = 0.02384   n = 28246
```

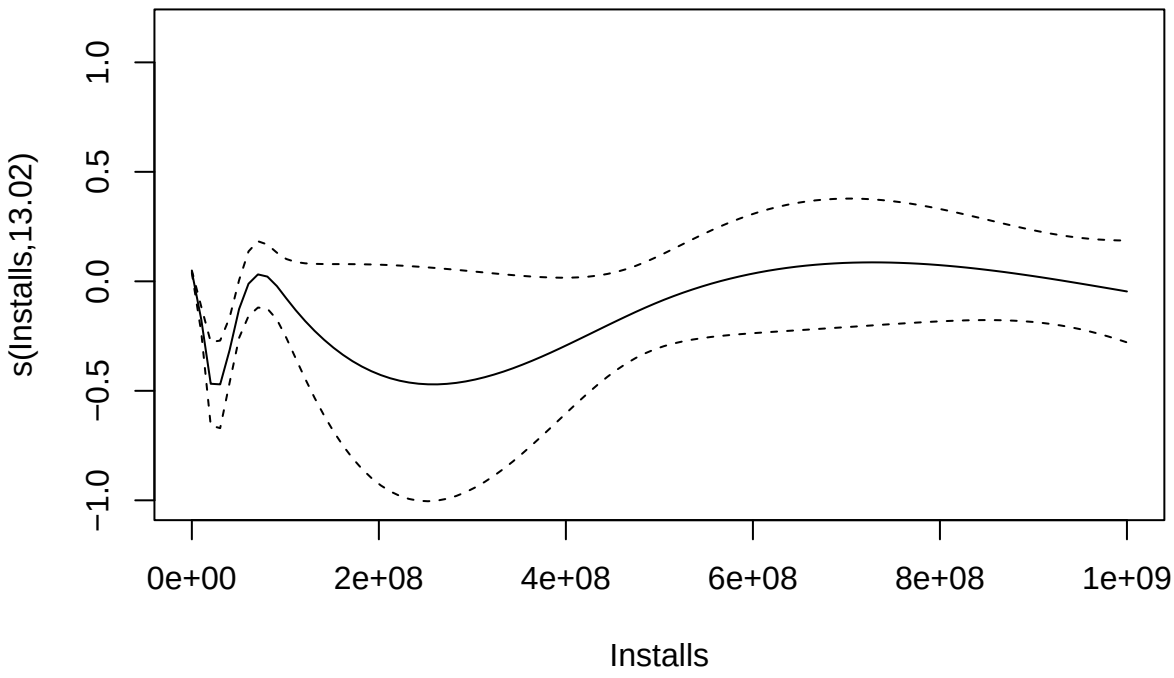
Predictions of The Final Model In the predictions seen below, the team examined the predicted values of the predictions versus the actual values in the data set. The team noticed that none of the predicted values fall outside the confidence interval.

Looking at the fit of the regression model, the team noticed rating count had a larger confidence interval compared to installs. The spline of rating count shows there's still a large variance with its splines compared to installs. This could signify that rating count may need additional support in its spline creation.

```
##
##     0
## 28246
```

```
## NULL
```






```
## # A tibble: 6 x 7
##      pred  se.fit     fit upper.bound lower.bound actual is.conf
##     <dbl>   <dbl>   <dbl>       <dbl>       <dbl>  <dbl>   <dbl>
## 1 0.00219 0.00953 0.00219      0.0209   -0.0165     0          0
## 2 1.52    0.00829 1.52         1.54      1.50       1.69       0
## 3 0.0126  0.00513 0.0126       0.0227    0.00256    0          0
## 4 1.26    0.00736 1.26         1.28      1.25       1.79       0
## 5 0.0103  0.00435 0.0103       0.0188    0.00176    0          0
## 6 0.0143  0.00758 0.0143       0.0292   -0.000559   0          0
```

```
## [1] "RSME of Model 3 is: 0.154"
```

Final Thoughts

The results of the data set highlighted the importance of generalized additive model in regression. The regression model better fit the data set as the features do not follow the assumption of normality like linear regression. The winning GAM models provided a regression model with a R^2 of 96%, a RSME of 15%, and a the lowest deviance of 669. The model's current results could be improved through a different method of splines as the summary results do highlight the need of more knots in the spline function.

References

Tafesse, Wondwesen. (2023). The differential effects of developers' app store strategy on the performance of niche and popular mobile apps. Journal of Marketing Analytics. 11. 1-14. 10.1057/s41270-023-00216-8.1

Lee, Gunwoong & Santanam, Raghu. (2014). Determinants of Mobile Apps' Success: Evidence from the App Store Market. Journal of Management Information Systems. 31. 133-170. 10.2753/MIS0742-1222310206.
Kapoor, Anuj & Vij, Madhu. (2020). How to Boost your App Store Rating? An Empirical Assessment of Ratings for Mobile Banking Apps. Journal of theoretical and applied electronic commerce research. 15. 10.4067/S0718-18762020000100108.
Wu, Huayao & Deng, Wenjun & Niu, Xintao & Nie, Changhai. (2021). Identifying Key Features from App User Reviews. 922-932. 10.1109/ICSE43902.2021.00088.

Appendix

```
library(tidyverse) #data wrangling
library(lubridate) #data wrangling - date formatting
library(MASS) #box-cox function
library(car)
library(forecast)

data_url <- "https://raw.githubusercontent.com/Vy4thewin/criticalthinking3/main/raw_appstore_data.csv"
raw.app.data <-read.csv(data_url)

#Removed columns that are not fit for a linear regression model. (i.e: observation identifying columns)

# BRL   CAD   EUR   GBP   INR   KRW   USD   VND   XXX
#     1     1     1     1     2     1 28963     1   582

#Create list of columns with blank values
blank_val_cols <- c("Released", "Last.Updated", "Minimum.Android")

#Create list of date cols
date_val_cols <- c("Released", "Last.Updated")

#Create list of columns to convert to factor type
factor_list <- c("Category", "Content.Rating")

app.data <- raw.app.data |>
  dplyr::select(-c(ï..App.Name, App.Id, Developer.Id, Developer.Website, Developer.Email, Privacy.Policy
  mutate_at(blank_val_cols, ~na_if(., "")) |> #convert blank values to NA to be better handled by na.om
  na.omit()|> #drops 1577 observations (approx. 35 of the original dataset)
  mutate_at(factor_list, factor) |> #convert cols to factor type
  mutate(Size = str_replace_all(Size, "[^0-9]","")) |> #remove non digits from 'Size' col
  mutate(Size = as.numeric(Size)) |>
  filter(!is.na(Size)) |>
  mutate(Installs = str_replace_all(Installs, "[^0-9]","")) |> #clean 'Installs' col
  mutate(Installs = as.numeric(Installs)) |>
  mutate(across(all_of(date_val_cols), ~mdy(.), .names = "{.col}")) #use lubridate to convert character

library(ggpubr)

plot.func <- function(data, col){
  #compute mean for Rating
  mean_rating <- data |>
  pull(col) |>
```

```
  mean() |>
  signif(6)

  var.dist.plt <- ggplot(
    data=app.data, aes(x=data[[col]])) +
    geom_density()+
    labs(y = "Density", x = data[[col]]) +
    geom_vline(xintercept=mean_rating, linewidth=1.2, color= "red")

  return (var.dist.plt)
}

g1 <- plot.func(app.data, "Rating")
g2 <-plot.func(app.data, "Rating.Count")
g3 <- plot.func(app.data, "Installs")
g4 <- plot.func(app.data, "Minimum.Installs")
g5 <- plot.func(app.data, "Maximum.Installs")
g6 <- plot.func(app.data, "Price")
g7 <- plot.func(app.data, "Size")

plt <- ggarrange(g1,g2,g3,g4,g5,g6,g7,ncol =3, nrow = 3)
plt


#Using box-cox transformations (MASS Package)
lamb.Rating <- boxcox((app.data$Rating+1)~1)
lamb.Rating.Count <- boxcox((app.data$Rating.Count+1)~1)


#retrieving the exact lambda for transformation
lamb.Rating <- lamb.Rating$x[which.max(lamb.Rating$y)]#0.14
lamb.Rating.Count <- lamb.Rating.Count$x[which.max(lamb.Rating.Count$y)]#-0.22


#app.data <- app.data %>% mutate(Rating = (Rating^lamb.Rating-1/lamb.Rating))
app.data <- app.data%>%mutate(Rating=log(Rating+1))
app.data <- app.data%>%mutate(Rating.Count=sqrt(Rating.Count+1))


# lambda.Rating <- BoxCox.lambda(app.data$Rating)
# app.data$Rating = BoxCox(app.data$Rating, lambda.Rating)
#
#
# #compute mean for Rating
# mean_rating <- app.data |>
#   pull(Rating) |>
#   mean() |>
#   signif(6)
#
#
# var.dist.plt <- ggplot(
#   data=app.data, aes(x=Rating)) +
#   geom_density()+
```

```
#   labs(y = "Density", x = "Rating")# +
#   #geom_vline(xintercept=mean_rating, linewidth=1.2, color= "red")
#
# plot(var.dist.plt)

library(corrplot)
#checking for highly correlated variables
numeric.data <- app.data %>%
  select_if(is.numeric)


corrplot(cor(numeric.data),method = "number",type="lower", tl.srt = .71,number.cex=0.75)


#Setting a 70/30 split of the app.store
library(caTools)
temp<-sample.split(app.data$Rating,SplitRatio =0.7)
app.store.train<-subset(app.data,sample=TRUE)
app.store.test<-subset(app.data,sample=FALSE)

library(mgcv)

gam.mod <- gam(Rating ~  Category +
                 s(Rating.Count) #+
                 #Installs +
                 #Minimum.Installs +
                 #Maximum.Installs +
                 #Free +
                 #Price +
                 #Size +
                 #Minimum.Android +
                 #Released +
                 #Last.Updated +
                 #Content.Rating +
                 #Ad.Supported +
                 #In.App.Purchases +
                 #Editors.Choice
               , data = app.store.train)

summary(gam.mod)
par(mfrow = c(2, 2))
plot(gam.mod)

#checking gam stats
gam.check(gam.mod)

#reviewing previous model, apply cubic splines and updating method to reml
gam1<- gam(Rating ~s(Rating.Count,bs ='cs') +
                 s(Installs,bs ='cs')+Category+Ad.Supported+In.App.Purchases
               , data = app.store.train,method = "REML")

summary(gam1)
par(mfrow = c(2, 2))
```

```r
plot(gam1)

par(mfrow = c(2, 2))
gam.check(gam1)


#Increasing the basis functions in the smoothing items
gam2<- gam(Rating ~s(Rating.Count,bs="cs",k=100) +
                  s(Installs,bs="cs",k=15)+Category+Ad.Supported+In.App.Purchases, data = app.store.trai
summary(gam2)
gam.check(gam2)

#Increasing the basis functions in rating count and add extra penalty
gam3<- gam(Rating ~s(Rating.Count,bs="cs",k=100) +
                  s(Installs,bs="cs",k=15)+Category+Ad.Supported+In.App.Purchases+Price, data = app.stor

summary(gam3)
gam.check(gam3)

#remove smoothing on rating count and add in editors choice
gam4<- gam(Rating ~Rating.Count+s(Installs,bs="cs",k=15)+Category+Ad.Supported+In.App.Purchases+Editors
#summary(gam4)
gam.check(gam4)

#check AIC
AIC(gam3,gam4)

library(broom)

glance(gam3)
glance(gam4)


#Increased k folds to improve metrics
gam3_update1<- gam(Rating ~s(Rating.Count,bs="cs",k=150) +
                  s(Installs,bs="cs",k=20)+Category+Ad.Supported+In.App.Purchases+Price, data = app.stor

summary(gam3_update1)
gam.check(gam3_update1)

glance(gam3)
glance(gam3_update1)
gam.check(gam3_update1)

#plot model fit
par(c(5,5))
plot.gam(gam3_update1)

glance(gam3)
glance(gam3_update1)
gam.check(gam3_update1)

#plot model fit
```

```r
par(c(5,5))
plot.gam(gam3_update1)

anova_result <- anova(gam3, gam3_update1, test="Chisq")
print(anova_result)


gam_model_comp <- bind_rows(glance(gam1), glance(gam2), glance(gam3_update1), glance(gam4))
gam_names1 <- c("gam1","gam2","gam3","gam4")
gam_model_comp <- cbind(model.build = gam_names1, gam_model_comp)
gam_model_comp

#look at features with significant p-values
summary.gam(gam3_update1)

#predicting the ratings with the most updated
gam_predict<-predict(gam3_update1,newdata = app.store.test,se.fit = TRUE,type="response")

#Take the upper and lower bounds of the confidence interval
upper<-gam_predict$fit+1.96*gam_predict$se.fit
lower<-gam_predict$fit-1.96*gam_predict$se.fit

#store results
test.results<-data.frame(
    pred<-gam_predict,
    fit<-gam_predict$fit,
    upper<-upper,
    lower<-lower,
    act<-app.store.test$Rating
)
#Show head of the table
test.results<-test.results%>%rename(pred=fit,upper.bound="upper....upper",lower.bound="lower....lower",a

#check that prediction is within the confidence level
test.results<-test.results%>%mutate(is.conf=if_else(pred>=lower.bound && pred<=upper.bound,0,1 ))

#view count of observation outside confidence level
table(test.results$is.conf)

# #see model fit
# test.results%>%ggplot(aes(x=act,y=pred))+geom_ribbon(aes(ymin=lower,ymax=upper,fill="lightblue"))+geo
par(c(5,5))
plot.gam(gam3_update1)

#review results of prediction
as_tibble(head(test.results))
```