Broadband Usage in the U.S.

Shanley Camara

2023/01/13

Introduction

Internet accessibility in the United States is greatly important for people to thrive in our society. Many rural communities may not even have internet services available at all, let alone the ability to pay and access those services. Broadband allows people to access the internet at much faster speeds than other more outdated methods and is accessible at any time. In this project we will be examining broadband availability and usage by county across the United States, and then predict broadband usage based on county census data using several machine learning techniques.

Data sources:

Broadband by county: https://data.world/amberthomas/broadband-usage-in-us (https://data.world/amberthomas/broadband-usage-in-us)

County census data: https://www.kaggle.com/datasets/mmattson/us-broadband-availability?resource=download&select=broadband_access.csv (https://www.kaggle.com/datasets/mmattson/us-broadband-availability?resource=download&select=broadband_access.csv)

Data and Packages

```
library(tidyverse)
library(caret)
library(broom)
library(tidymodels)
library(vip)
library(stringr)
library(DMwR2)
library(forcats)
library(randomForest)
library(e1071)
library(pls)
library(glmnet)

set.seed(42)

census <- read.csv("C:\\Users\\shans\\Downloads\\IMLS_county_data.csv", header = T, sep = ",",co
lClasses = c(FIPS_County = "character"))

broadband <- read.csv("C:\\Users\\shans\\Downloads\\broadband_by_county.csv", header = T, sep =
",", colClasses = c(COUNTY.ID = "character"))
```

```
# Prepare datasets and join by unique county fips codes

broadband <- broadband %>%
    rename(county_fips = COUNTY.ID)

census$county_fips <- paste(census$FIPS_State, '', census$FIPS_County) %>% str_remove_all(" ")

# Joining the data using inner_join with broadband first
broadband_census <- broadband %>% inner_join(census, by = "county_fips")
```

Data Cleaning

Let’s start by taking a look at our table after we joined by county:

```
broadband_census %>% head()
```

```
##   ST county_fips    COUNTY.NAME BROADBAND.AVAILABILITY.PER.FCC BROADBAND.USAGE
## 1 AL        1001 Autauga County                         0.8057           0.391
## 2 AL        1003 Baldwin County                         0.8362           0.452
## 3 AL        1005 Barbour County                         0.6891           0.324
## 4 AL        1007    Bibb County                         0.3368           0.136
## 5 AL        1009  Blount County                          0.758           0.199
## 6 AL        1011 Bullock County                         0.9363           0.157
##          GEO_ID FIPS_State FIPS_County                    NAME  County   State
## 1 0500000US01001          1         001 Autauga County, Alabama Autauga Alabama
## 2 0500000US01003          1         003 Baldwin County, Alabama Baldwin Alabama
## 3 0500000US01005          1         005 Barbour County, Alabama Barbour Alabama
## 4 0500000US01007          1         007    Bibb County, Alabama    Bibb Alabama
## 5 0500000US01009          1         009  Blount County, Alabama  Blount Alabama
## 6 0500000US01011          1         011 Bullock County, Alabama Bullock Alabama
##   Stabr Population_2019 Unemployment.rate.2019 Percent.w.o.Health.insurance
## 1    AL           55869                   2.7                          7.1
## 2    AL          223234                   2.7                         10.2
## 3    AL           24686                   3.8                         11.2
## 4    AL           22394                   3.1                          7.9
## 5    AL           57826                   2.7                         11.0
## 6    AL           10101                   3.6                         10.8
##   MOE....w.o.health.ins. Poverty.Rate.... MOE..Poverty.Rate....
## 1                    0.9             15.4                   2.1
## 2                    0.8             10.6                   0.9
## 3                    1.6             28.9                   2.9
## 4                    1.9             14.0                   3.8
## 5                    1.3             14.4                   1.7
## 6                    3.4             31.4                   6.1
##   Percent.received.SNAP..2018. MOE.SNAP Percent.with.no.home.computer..2018.
## 1                         12.7      1.8                                 13.0
## 2                          7.5      0.7                                 11.4
## 3                         27.4      2.6                                 23.9
## 4                         12.4      3.2                                 23.7
## 5                          9.5      1.6                                 21.3
## 6                         25.9      5.5                                 27.1
##   MOE.No.Computer Percent.with.no.home.Internet..2018. MOE.no.Internet
## 1             1.7                                20.9             1.9
## 2             0.8                                21.3             1.3
## 3             2.2                                38.9             2.8
## 4             3.6                                33.8             3.9
## 5             2.0                                30.6             1.9
## 6             5.3                                40.1             5.7
##   Percent.with.home.Broadband..2018. MOE.Broadband
## 1                               78.9           1.9
## 2                               78.1           1.3
## 3                               60.4           2.7
## 4                               66.1           3.9
## 5                               68.5           1.9
## 6                               58.9           5.6
##   Number.of.Broadband.providers..2019.
## 1                                    0
## 2                                    0
```

```
## 3                                     4
## 4                                     0
## 5                                     0
## 6                                     1
##   Population.for.whom.broadband.available..2019....
## 1                                                 0
## 2                                               0.0
## 3                                              99.2
## 4                                               0.0
## 5                                               0.0
## 6                                              40.1
##   Lowest.broadband.cost.per.month..2019....
## 1                                       N/A
## 2                                       N/A
## 3                                     74.99
## 4                                       N/A
## 5                                       N/A
## 6                                     57.99
```

We have some repeated columns and inconsisent column names, so let's start by cleaning those up:

```
broadband_census <- broadband_census %>%
  rename(state = ST, county_name = COUNTY.NAME, broadband_availability = BROADBAND.AVAILABILITY.
PER.FCC, broadband_usage = BROADBAND.USAGE, population = Population_2019, unemployment_rate = Un
employment.rate.2019, perc_no_health_ins = Percent.w.o.Health.insurance, pov_rate = Poverty.Rat
e...., perc_snap = Percent.received.SNAP..2018., perc_no_computer = Percent.with.no.home.compute
r..2018., perc_no_internet = Percent.with.no.home.Internet..2018.) %>%
  select(-GEO_ID, -FIPS_State, -FIPS_County, -NAME, -County, -State, -Stabr, -MOE....w.o.health.
ins., -MOE..Poverty.Rate...., -MOE.SNAP, -MOE.No.Computer, -MOE.no.Internet, -Percent.with.home.
Broadband..2018., -MOE.Broadband, -Number.of.Broadband.providers..2019., -Population.for.whom.br
oadband.available..2019...., -Lowest.broadband.cost.per.month..2019....)

broadband_census %>% head()
```

```
##   state county_fips    county_name broadband_availability broadband_usage
## 1    AL        1001 Autauga County                 0.8057           0.391
## 2    AL        1003 Baldwin County                 0.8362           0.452
## 3    AL        1005 Barbour County                 0.6891           0.324
## 4    AL        1007    Bibb County                 0.3368           0.136
## 5    AL        1009  Blount County                  0.758           0.199
## 6    AL        1011 Bullock County                 0.9363           0.157
##   population unemployment_rate perc_no_health_ins  pov_rate perc_snap
## 1      55869          2.7                     7.1 15.4           12.7
## 2     223234          2.7                    10.2 10.6            7.5
## 3      24686          3.8                    11.2 28.9           27.4
## 4      22394          3.1                     7.9 14.0           12.4
## 5      57826          2.7                    11.0 14.4            9.5
## 6      10101          3.6                    10.8 31.4           25.9
##   perc_no_computer perc_no_internet
## 1             13.0             20.9
## 2             11.4             21.3
## 3             23.9             38.9
## 4             23.7             33.8
## 5             21.3             30.6
## 6             27.1             40.1
```

Get column information:

```
broadband_census %>% str()
```

```
## 'data.frame':    3142 obs. of  12 variables:
##  $ state                 : chr  "AL" "AL" "AL" "AL" ...
##  $ county_fips           : chr  "1001" "1003" "1005" "1007" ...
##  $ county_name           : chr  "Autauga County" "Baldwin County" "Barbour County" "Bibb Coun
ty" ...
##  $ broadband_availability: chr  "0.8057" "0.8362" "0.6891" "0.3368" ...
##  $ broadband_usage       : chr  "0.391" "0.452" "0.324" "0.136" ...
##  $ population            : chr  "55869" "223234" "24686" "22394" ...
##  $ unemployment_rate     : chr  "2.7     " "2.7     " "3.8     " "3.1     " ...
##  $ perc_no_health_ins    : num  7.1 10.2 11.2 7.9 11 10.8 10.2 9.4 10.8 8.3 ...
##  $ pov_rate              : chr  "15.4     " "10.6     " "28.9     " "14.0     " ...
##  $ perc_snap             : num  12.7 7.5 27.4 12.4 9.5 25.9 18.6 17.9 14.7 15.9 ...
##  $ perc_no_computer      : num  13 11.4 23.9 23.7 21.3 27.1 29.9 15.3 24.2 19.9 ...
##  $ perc_no_internet      : num  20.9 21.3 38.9 33.8 30.6 40.1 37 26.6 33 31 ...
```

We can see here that several of our variables are categorized as characters when they should really be numerical. Here can change the data type of those variables:

```
broadband_census <- broadband_census %>% mutate(broadband_availability = as.numeric(paste(broadb
and_availability)), broadband_usage = as.numeric(paste(broadband_usage)), population = as.numeri
c(paste(population)), unemployment_rate = as.numeric(paste(unemployment_rate)), pov_rate = as.nu
meric(paste(pov_rate)))

broadband_census %>% str()
```

```
## 'data.frame':    3142 obs. of  12 variables:
##  $ state                 : chr  "AL" "AL" "AL" "AL" ...
##  $ county_fips           : chr  "1001" "1003" "1005" "1007" ...
##  $ county_name           : chr  "Autauga County" "Baldwin County" "Barbour County" "Bibb Coun
ty" ...
##  $ broadband_availability: num  0.806 0.836 0.689 0.337 0.758 ...
##  $ broadband_usage       : num  0.391 0.452 0.324 0.136 0.199 0.157 0.183 0.419 0.501 0.125
...
##  $ population            : num  55869 223234 24686 22394 57826 ...
##  $ unemployment_rate     : num  2.7 2.7 3.8 3.1 2.7 3.6 3.6 3.5 2.9 2.9 ...
##  $ perc_no_health_ins    : num  7.1 10.2 11.2 7.9 11 10.8 10.2 9.4 10.8 8.3 ...
##  $ pov_rate              : num  15.4 10.6 28.9 14 14.4 31.4 23.5 18.6 16.6 15 ...
##  $ perc_snap             : num  12.7 7.5 27.4 12.4 9.5 25.9 18.6 17.9 14.7 15.9 ...
##  $ perc_no_computer      : num  13 11.4 23.9 23.7 21.3 27.1 29.9 15.3 24.2 19.9 ...
##  $ perc_no_internet      : num  20.9 21.3 38.9 33.8 30.6 40.1 37 26.6 33 31 ...
```

Now that our numeric variables are indeed numeric, we can find the summary statistics for each of our variables and see if there are any null values:

```
broadband_census %>% summary()
```

```
##    state           county_fips        county_name
##  Length:3142        Length:3142        Length:3142
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  broadband_availability broadband_usage    population       unemployment_rate
##  Min.   :0.0002         Min.   :0.0010   Min.   :      86   Min.   : 0.700
##  1st Qu.:0.7700         1st Qu.:0.1960   1st Qu.:   10902   1st Qu.: 3.000
##  Median :0.9242         Median :0.3680   Median :   25726   Median : 3.700
##  Mean   :0.8404         Mean   :0.3896   Mean   :  104468   Mean   : 4.001
##  3rd Qu.:0.9839         3rd Qu.:0.5660   3rd Qu.:   68073   3rd Qu.: 4.600
##  Max.   :1.0000         Max.   :1.0000   Max.   :10039107   Max.   :19.300
##  NA's   :9              NA's   :11                          NA's   :1
##  perc_no_health_ins    pov_rate      perc_snap     perc_no_computer
##  Min.   : 1.70      Min.   : 2.3   Min.   : 0.00   Min.   : 1.40
##  1st Qu.: 6.20      1st Qu.:11.0   1st Qu.: 8.60   1st Qu.:11.90
##  Median : 9.20      Median :14.7   Median :12.40   Median :15.70
##  Mean   :10.08      Mean   :15.6   Mean   :13.22   Mean   :16.58
##  3rd Qu.:12.68      3rd Qu.:19.1   3rd Qu.:16.80   3rd Qu.:20.40
##  Max.   :45.60      Max.   :55.1   Max.   :59.90   Max.   :61.70
##                     NA's   :1
##  perc_no_internet
##  Min.   : 4.20
##  1st Qu.:20.50
##  Median :25.50
##  Mean   :26.58
##  3rd Qu.:31.80
##  Max.   :74.10
##
```

It looks like there are a few null values in our dataset but not too many, meaning our data is not sparse. We will impute these later with knn once the data is split into testing and training sets. There does appear to be 11 nulls in the broadband_usage column, which is the variable we eventually want to predict. We will remove these from the data set for now, and come back to them later when we have a tried and tested model. The other columns with missing values can be imputed with k-nearest neighbors (knn) imputation.

```
broadband_missing <- broadband_census[is.na(broadband_census$broadband_usage), ]

broadband_census <- broadband_census[!(is.na(broadband_census$broadband_usage)), ]
```

We will want to finish preprocessing and split the data into training and testing sets before we impute the other missing values. This is because we do not want any influence from the training set to affect the testing set, so the imputation process will occur in each set separately.

Splitting training and testing data

```
sample_size = floor(0.8*nrow(broadband_census))

picked = sample(seq_len(nrow(broadband_census)),size = sample_size)
broadband_train = broadband_census[picked,]
broadband_test = broadband_census[-picked,]
```

```
# retrieve mean and standard deviation of broadband usage before scaling for later use
broadband_mean = mean(broadband_train$broadband_usage)
broadband_std = sd(broadband_train$broadband_usage)
```

Impute missing values

```
impute_train <- preProcess(x = broadband_train,
                           method = c("knnImpute"),
                           k = 10,
                           knnSummary = median)
broadband_train <- predict(impute_train, broadband_train, na.action = na.pass)

impute_test <- preProcess(x = broadband_test,
                           method = c("knnImpute"),
                           k = 10,
                           knnSummary = median)
broadband_test <- predict(impute_test, broadband_test, na.action = na.pass)

impute_missing <- preProcess(x = broadband_missing[,c(1:4,6:12)],
                           method = c("knnImpute"),
                           k = 10,
                           knnSummary = median)
broadband_missing <- predict(impute_missing, broadband_missing, na.action = na.pass)


anyNA(broadband_train)
```

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

```
anyNA(broadband_test)
```

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

Scaling the data

Scaling the data is an important preprocessing step to ensure our prediction models perform correctly. We will use a standard scaler on the training data and then apply that scale to our test set and missing values set.

```
scale_parameters <- preProcess(broadband_train, method = c("center", "scale"))

broadband_train <- predict(scale_parameters, broadband_train)
broadband_test <- predict(scale_parameters, broadband_test)
broadband_missing <- predict(scale_parameters, broadband_missing)
```

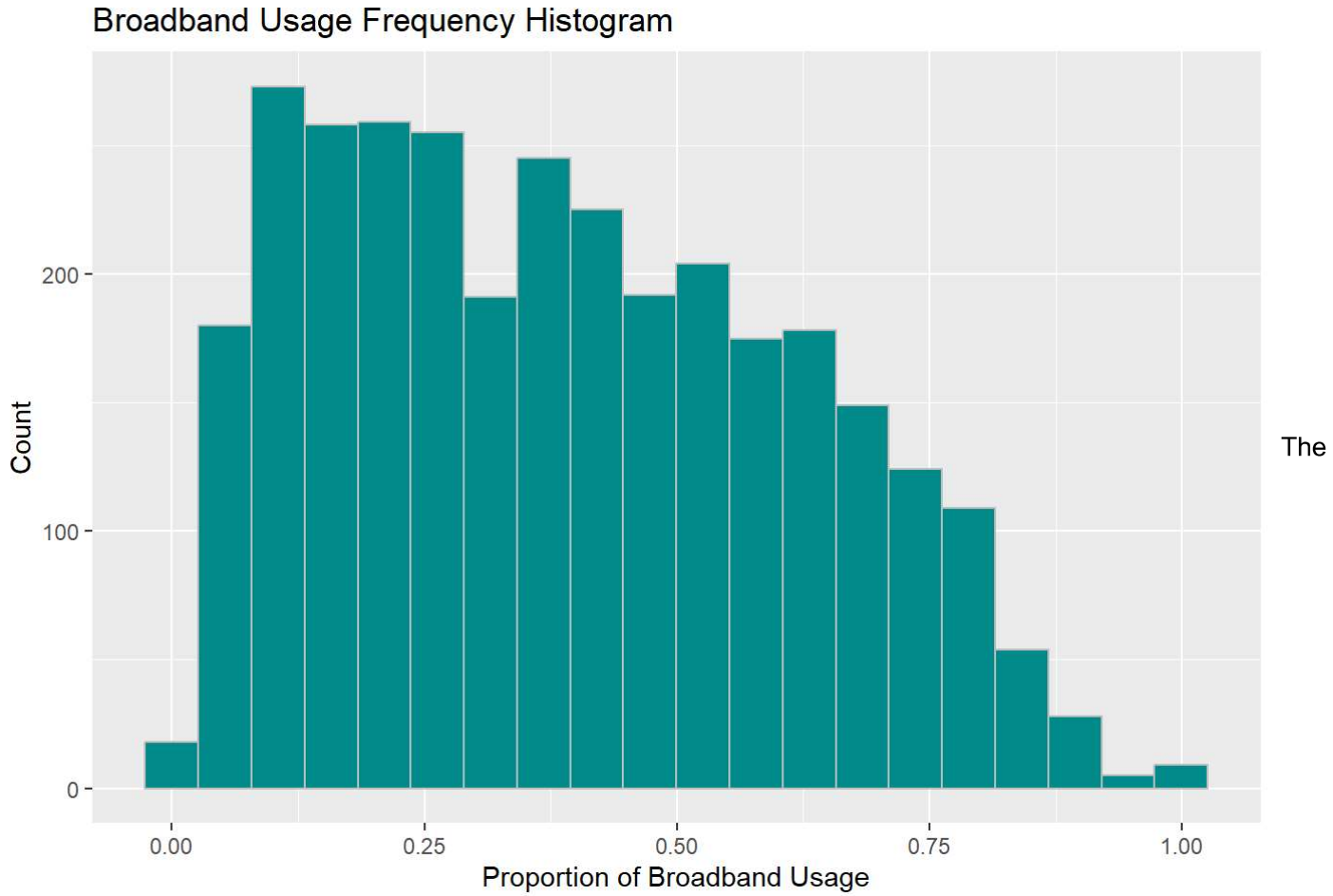
Now that our data is cleaned up and split, let's perform some exploratory analysis and visualizations before we move on to modeling.

Exploratory Analysis

Graphs

Broadband usage histogram shows us the distribution of broadband usage across our entire data set:

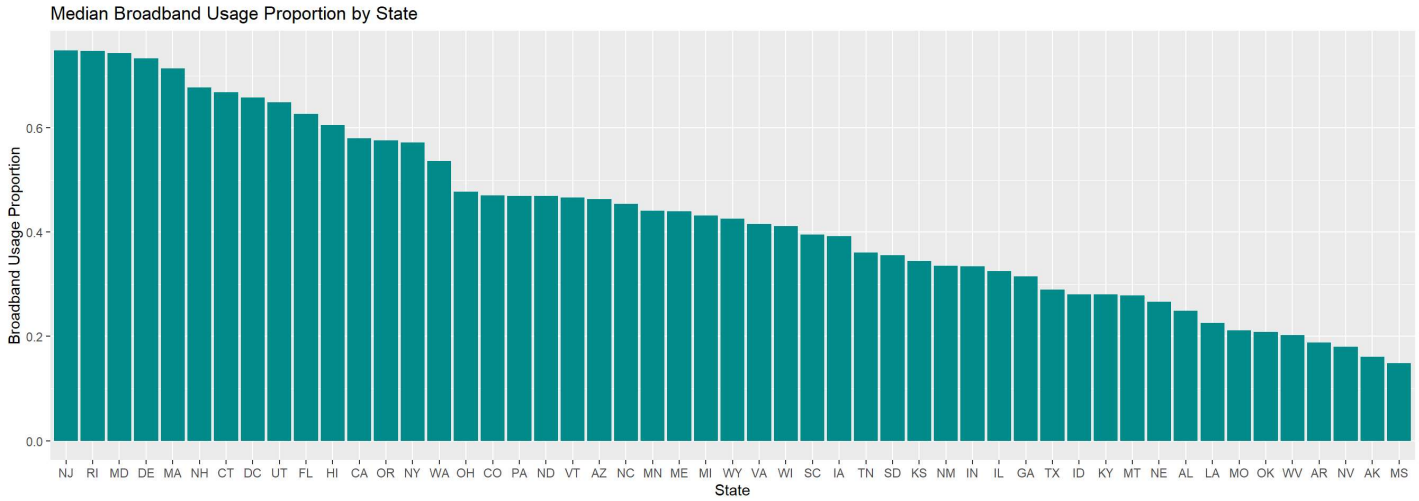
```
broadband_census %>% ggplot(aes(x = broadband_usage)) +
  geom_histogram(bins = 20, fill = 'cyan4', col='grey') +
  labs(title = "Broadband Usage Frequency Histogram", x = "Proportion of Broadband Usage", y =
"Count")
```



The distribution of broadband usage appears to be a bit right skewed, with a peak of 0.1-0.15 proportion of broadband usage.

Broadband usage proportion by state:

```
aggregate_state <- aggregate(broadband_census$broadband_usage ~ broadband_census$state, data = b
roadband_census, median)

aggregate_state %>% ggplot(aes(x = fct_rev(fct_reorder(`broadband_census$state`, `broadband_cens
us$broadband_usage`)), y = `broadband_census$broadband_usage`)) +
  geom_col(fill = 'cyan4') +
  labs(title = "Median Broadband Usage Proportion by State", x = "State", y = "Broadband Usage P
roportion")
```



Our plot shows that New Jersey has the highest median broadband usage, while Mississippi has the lowest.

Modeling

We will be attempting several models to predict the broadband_usage. These models are multivariable linear, k-nearest neighbors regression, regularized regression, and principle component regression.

Simple Linear Regression Model

This model fits our data linearly to predict the broadband usage proportion.

```
# Fitting Simple Linear Regression
# to the Training set
lm.r = lm(formula = broadband_usage ~ .,
                         data = broadband_train[,c(4:12)])
coef(lm.r)
```

```
##            (Intercept) broadband_availability            population
##           4.303680e-17           2.904972e-01          1.678743e-01
##      unemployment_rate     perc_no_health_ins              pov_rate
##          -3.714276e-02          -1.316093e-02          1.154096e-01
##              perc_snap       perc_no_computer       perc_no_internet
##           1.401795e-02          -8.989857e-02          -4.337433e-01
```

```
summary(lm.r)
```

```
##
## Call:
## lm(formula = broadband_usage ~ ., data = broadband_train[, c(4:12)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0837 -0.4621 -0.0210  0.4492  3.0655
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             4.304e-17  1.382e-02   0.000   1.0000
## broadband_availability  2.905e-01  1.629e-02  17.834  < 2e-16 ***
## population              1.679e-01  1.451e-02  11.568  < 2e-16 ***
## unemployment_rate      -3.714e-02  1.724e-02  -2.154   0.0313 *
## perc_no_health_ins     -1.316e-02  1.608e-02  -0.819   0.4131
## pov_rate                1.154e-01  2.702e-02   4.270 2.02e-05 ***
## perc_snap               1.402e-02  2.523e-02   0.556   0.5785
## perc_no_computer       -8.990e-02  3.590e-02  -2.504   0.0123 *
## perc_no_internet       -4.337e-01  3.812e-02 -11.377  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6915 on 2495 degrees of freedom
## Multiple R-squared:  0.5234, Adjusted R-squared:  0.5219
## F-statistic: 342.5 on 8 and 2495 DF,  p-value: < 2.2e-16
```

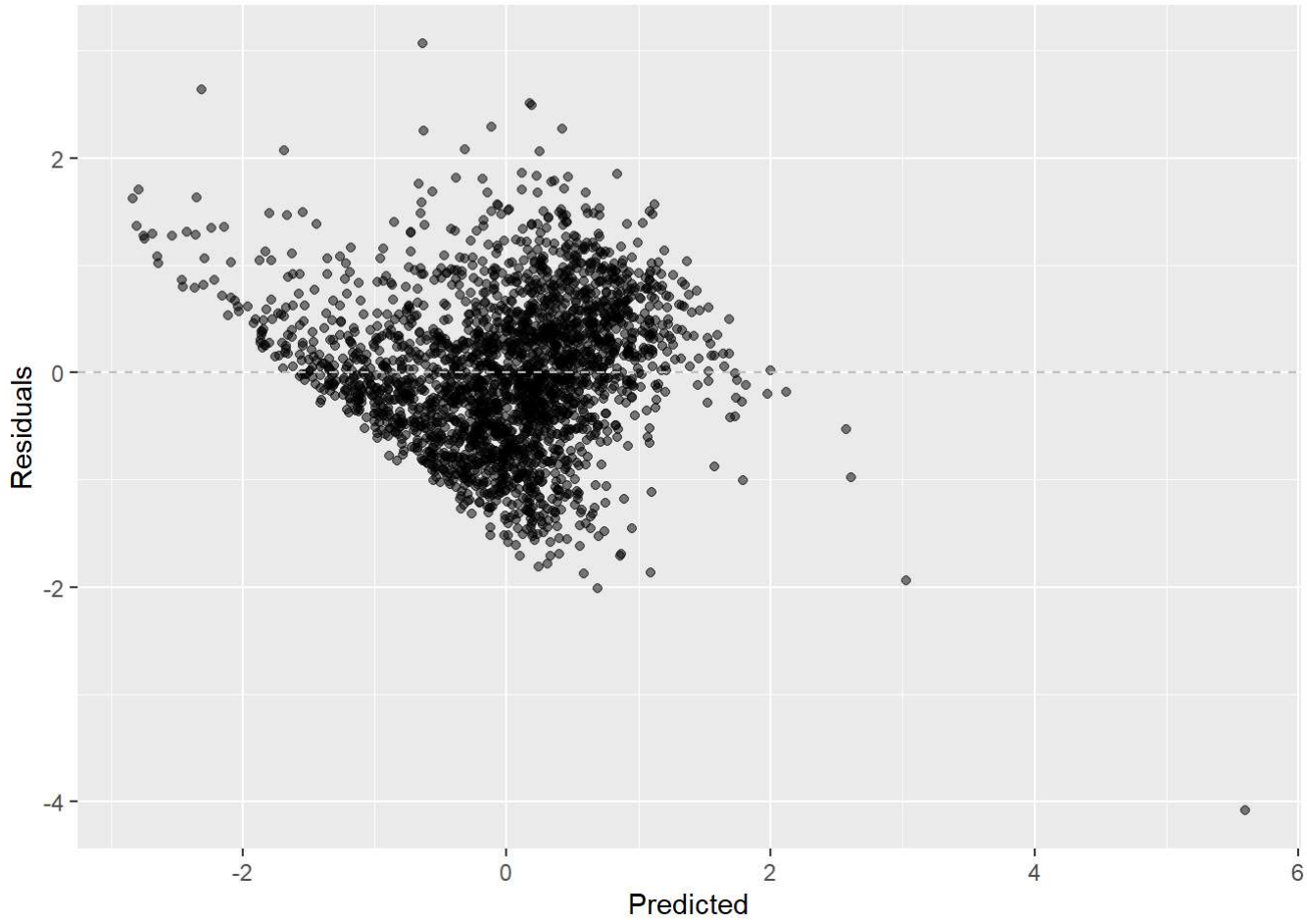
```
# Predicting the Test set results
ypred = predict(lm.r, newdata = broadband_train)
```

Correlation = 0.5234 P-value ~ 0

Our results show that there is a moderate, positive linear correlation between our explanatory and response variables. A low p-value suggests that this is statistically significant.

Residuals vs Predicted:

```
lm.r_aug <- augment(lm.r)
ggplot(lm.r_aug, mapping = aes(x = .fitted, y = .resid)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, color = "gray", lty = "dashed") +
  labs(x = "Predicted", y = "Residuals")
```



This plot of the residual values versus the predicted values show that there may be some nonconstant variance in our dataset. This may mean that our data may not be linear, and so this model may not suit our needs.

Mean Square Error (MSE) and Root Mean Square Error (RMSE) ways of determining error in the model:

```
MSE = (1/96)*sum((abs(lm.r$residuals))^2)
MSE
```

```
## [1] 12.4259
```

```
RMSE = sqrt(MSE)
RMSE
```

```
## [1] 3.525039
```

Test model on testing data:

```
test_pred <- data.frame(broadband_usage = broadband_test$broadband_usage, pred = predict(lm.r, b
roadband_test[,c(4,6:12)])) %>%
  mutate(resid = broadband_usage - pred) %>% filter(!is.na(pred))

MSE = (1/96)*sum((abs(test_pred$resid))^2)
MSE
```

```
## [1] 3.402928
```

```
RMSE = sqrt(MSE)
RMSE
```

```
## [1] 1.844703
```

```
compare_models <- data.frame(Model = "Linear Regression", MSE = MSE, RMSE = RMSE, stringsAsFacto
rs=FALSE)
```

Using this model on the test set yields a lower MSE and RMSE, so this may indicate that the model is effective at predicting broadband usage.

K-Nearest Neighbors Regression

The K-Nearest Neighbors model predicts our broadband usage based on the similarity within the other variables. KNN works by finding the distance between an input value and all other examples in the data, and then finds the average of the k values with the smallest distance to the input. In this model we will be testing several different k values (number of neighbors) and use the model with the least errors.

Trying multiple values of k:

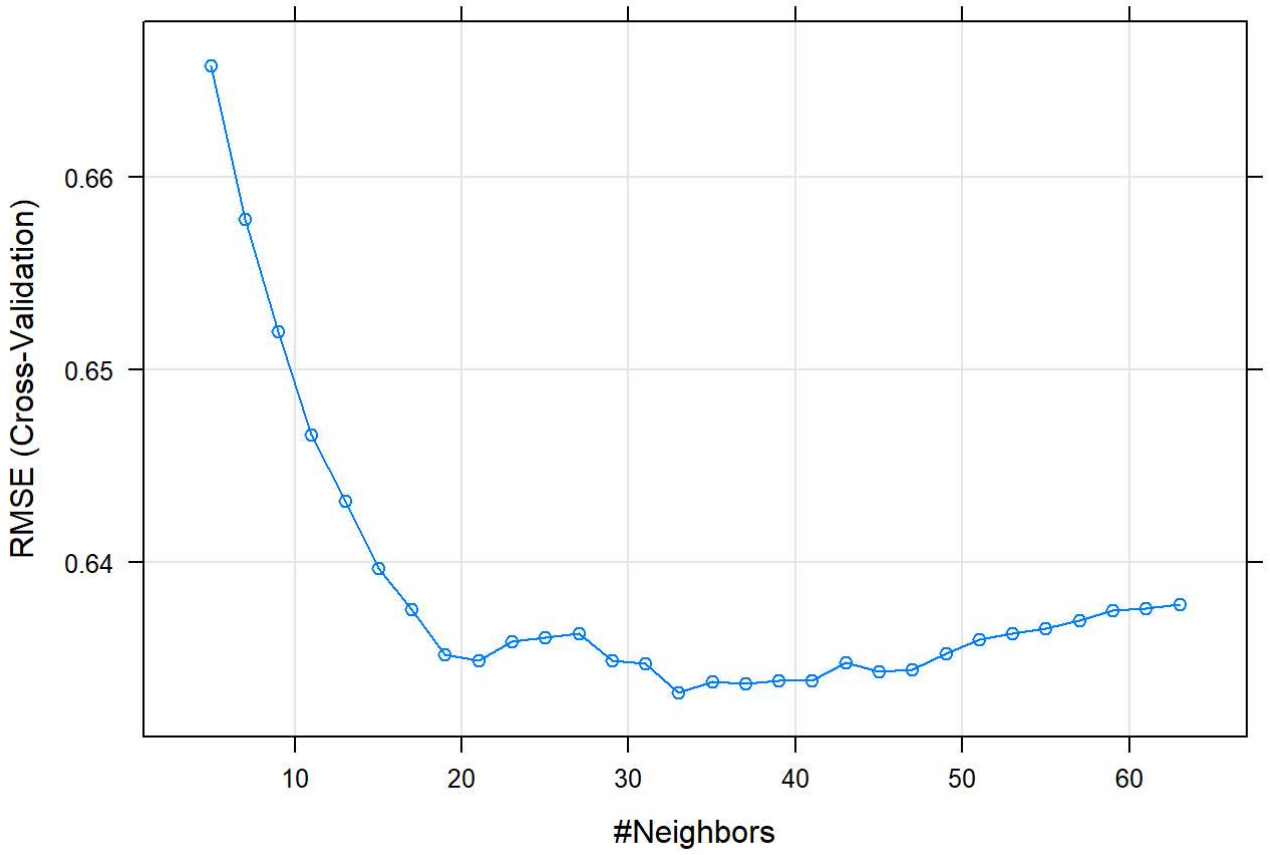
```
# Fit the model on the training set
model <- train(
  x = broadband_train[,c(4,6:12)],
  y = broadband_train[,5],
  method = "knn",
  trControl = trainControl("cv", number = 10),
  tuneLength = 30
  )

model
```

```
## k-Nearest Neighbors
##
## 2504 samples
##    8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2253, 2255, 2253, 2253, 2254, 2253, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    5  0.6658259  0.5610144  0.5077969
##    7  0.6578400  0.5690876  0.4999288
##    9  0.6520071  0.5759659  0.4975248
##   11  0.6466189  0.5826098  0.4927933
##   13  0.6431850  0.5870566  0.4925810
##   15  0.6397163  0.5914905  0.4910865
##   17  0.6375747  0.5945058  0.4912953
##   19  0.6352328  0.5977681  0.4905061
##   21  0.6349092  0.5985001  0.4912426
##   23  0.6359027  0.5978282  0.4928887
##   25  0.6361010  0.5976998  0.4928919
##   27  0.6363049  0.5979000  0.4924695
##   29  0.6349146  0.6000270  0.4914738
##   31  0.6347635  0.6004303  0.4915344
##   33  0.6332320  0.6026762  0.4907551
##   35  0.6338238  0.6021704  0.4913785
##   37  0.6337409  0.6024126  0.4917049
##   39  0.6338637  0.6024642  0.4917724
##   41  0.6338656  0.6028749  0.4920819
##   43  0.6347884  0.6020644  0.4928718
##   45  0.6343258  0.6031403  0.4928688
##   47  0.6344675  0.6033224  0.4932610
##   49  0.6353044  0.6024456  0.4944643
##   51  0.6360122  0.6018244  0.4950399
##   53  0.6363339  0.6017372  0.4952864
##   55  0.6365877  0.6018181  0.4955787
##   57  0.6369934  0.6015550  0.4962237
##   59  0.6375282  0.6010545  0.4970463
##   61  0.6376509  0.6010971  0.4973986
##   63  0.6378333  0.6011751  0.4980097
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 33.
```

```
model$bestTune
```

```
##     k
## 15 33
```

```
# Plot model accuracy vs different values of k
plot(model)
```



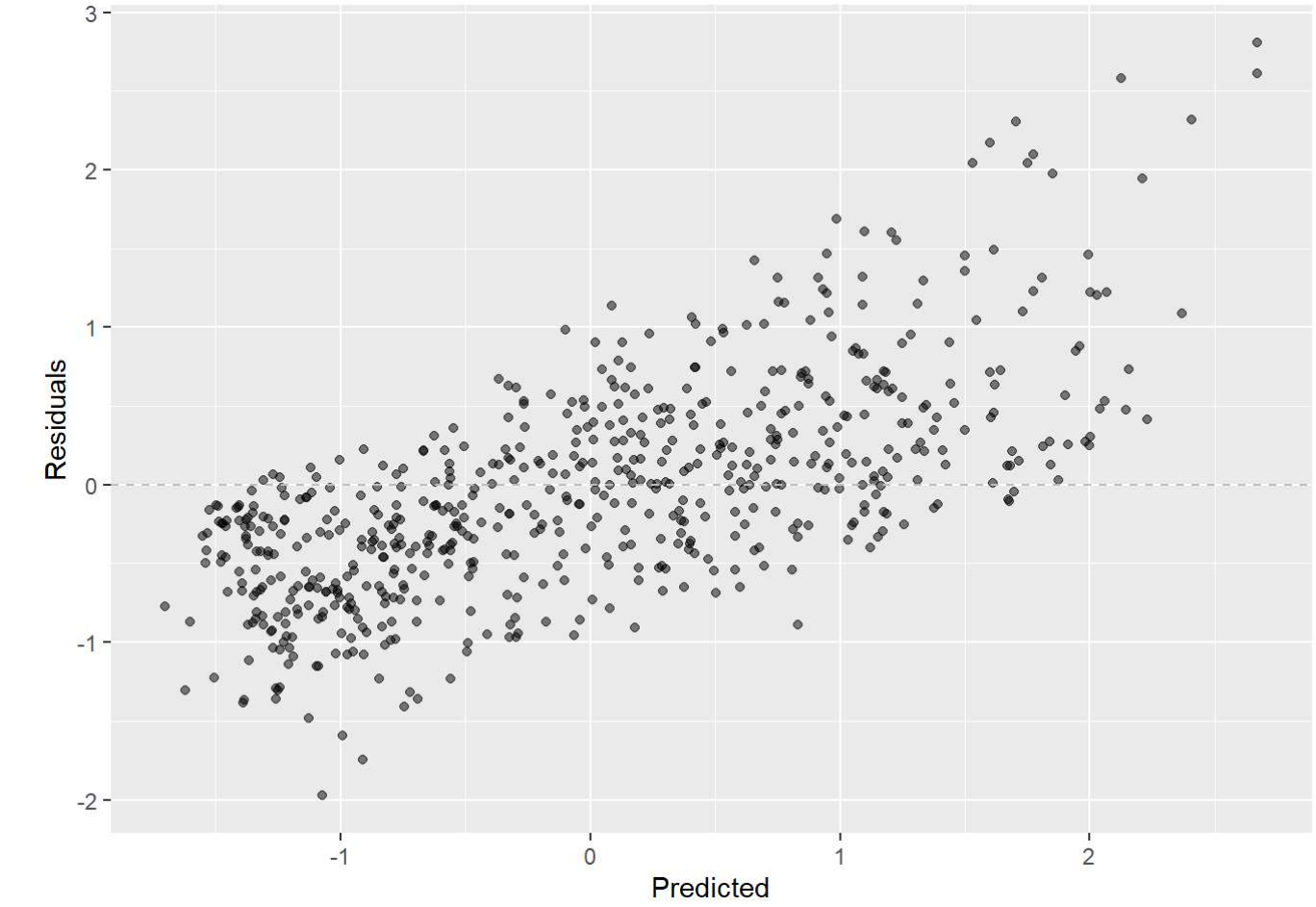

```
postResample(predict(model), broadband_train$broadband_usage)
```

```
##      RMSE  Rsquared       MAE
## 0.6169422 0.6230492 0.4767463
```

The above shows that 33 neighbors yields a model with the least error, so we will fit a model on 33 neighbors.

Fit the model:

```
fit <- knnreg(broadband_train[,c(4,6:12)], broadband_train[,5], k = 33)

fittedY <- predict(fit, broadband_test[,c(4,6:12)])

# plot of residuals
ggplot(broadband_test, mapping = aes(x = broadband_usage, y = (broadband_test$broadband_usage -
fittedY))) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, color = "gray", lty = "dashed") +
  labs(x = "Predicted", y = "Residuals")
```




```
summary(fit)
```

```
##         Length Class  Mode
## learn   2      -none- list
## k       1      -none- numeric
## theDots 0      -none- list
```

MSE and RMSE:

```
MSE = (1/96)*sum((abs(broadband_test$broadband_usage - predict(fit, broadband_test[,c(4,6:1
2)])))^2)
MSE
```

```
## [1] 3.061379
```

```
RMSE = sqrt(MSE)
RMSE
```

```
## [1] 1.74968
```

```
compare_models <- rbind(compare_models, list('KNN', MSE, RMSE))
```

The reduction in error is a good sign, however the nonrandom pattern on the residual plot may mean that this model is not quite where we want it to be.

Regularized regression (glmnet)

Regularized regression (also called glm for generalized linear model) is a sort of penalized multivariable linear regression. The penalties are applied based on the idea of "maximum likelihood." We will be using elastic net regression, which is a mix of lasso and ridge regularization.

Cross validation to find optimal alpha value and lambda value:

```
# find alpha with lowest mse
alpha <- seq(0.01, 0.99, 0.01)
best <- list(a=NULL, mse=NULL)

for (i in 1:length(alpha))
{
   cvg <- cv.glmnet(as.matrix(broadband_train[,c(4,6:12)]), as.matrix(broadband_train[,5]), fami
ly = "gaussian", alpha = alpha[i])
   best$a <- c(best$a, alpha[i])
   best$mse <- c(best$mse, min(cvg$cvm))
}

index <- which(best$mse==min(best$mse))
best_alpha <- best$a[index]
best_mse <- best$mse[index]

# train with best alpha to find best lambda
elastic_cv <- cv.glmnet(as.matrix(broadband_train[,c(4,6:12)]), as.matrix(broadband_train[,5]),
family = "gaussian", alpha = best_alpha)

best_lambda <- elastic_cv$lambda.min

cat("alpha:", best_alpha, " mse:", best_mse, " lambda:", best_lambda)
```

```
## alpha: 0.99  mse: 0.486319  lambda: 0.009017733
```

Now that we have found the optimal values for our model, let's make a final model:

```
elastic_mod <- glmnet(as.matrix(broadband_train[,c(4,6:12)]), as.matrix(broadband_train[,5]), fa
mily = "gaussian", alpha = best_alpha, lambda = best_lambda)
coef(elastic_mod)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                  s0
## (Intercept)             3.108060e-17
## broadband_availability  2.883997e-01
## population              1.637298e-01
## unemployment_rate      -1.591816e-02
## perc_no_health_ins      .
## pov_rate                8.649772e-02
## perc_snap               .
## perc_no_computer       -7.768754e-02
## perc_no_internet       -4.245498e-01
```

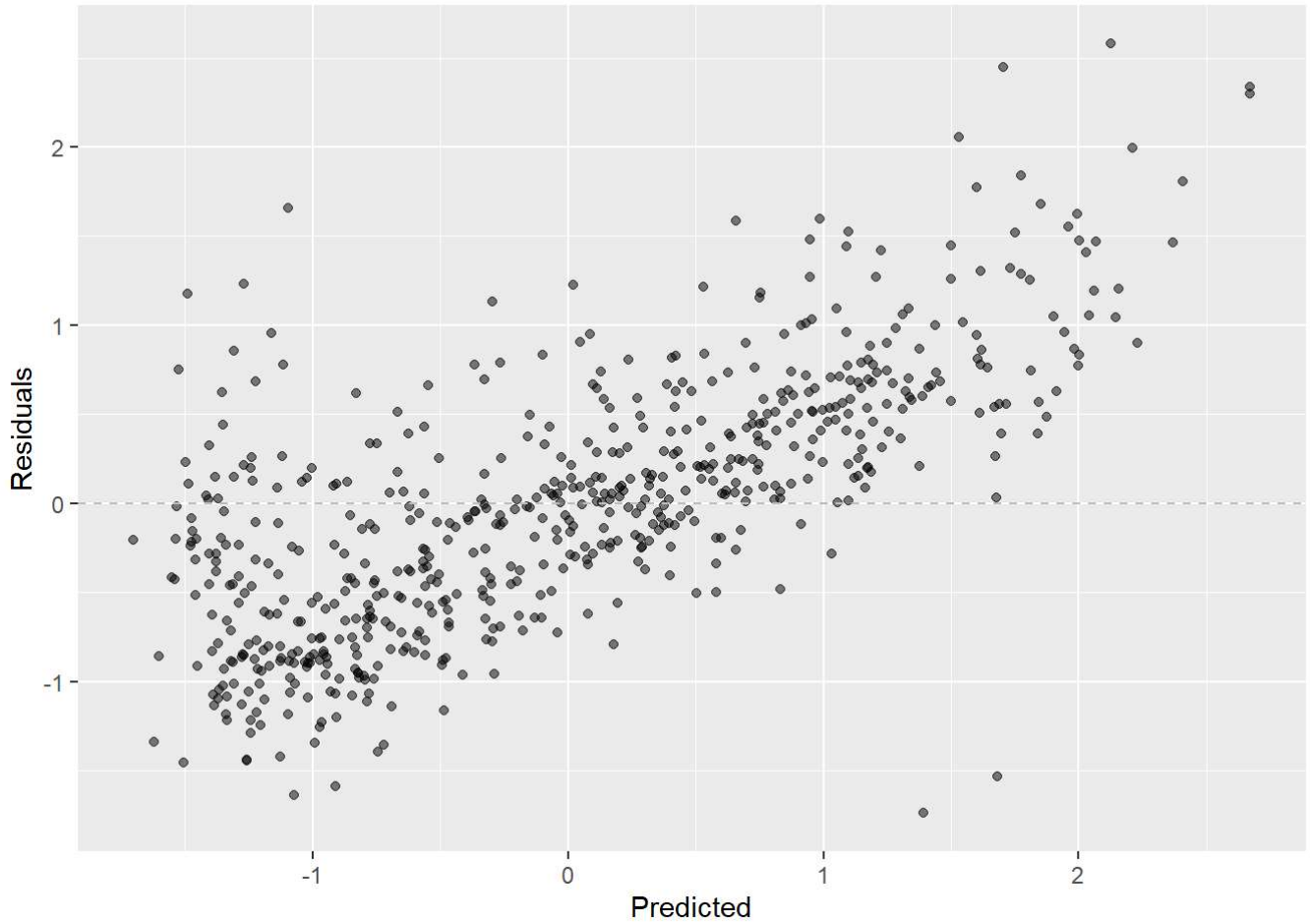
Now let's test on testing data and retrieve mse, rmse, and r-squared values:

```
pred <- predict(elastic_mod, as.matrix(broadband_test[,c(4,6:12)]))

rmse <- sqrt(mean((pred-as.matrix(broadband_test[,5]))^2))
R2 <- 1 - (sum((as.matrix(broadband_test[,5])-pred )^2)/sum((as.matrix(broadband_test[,5])-mean
(as.matrix(broadband_test[,5])))^2))
mse <- mean((as.matrix(broadband_test[,5]) - pred)^2)

compare_models <- rbind(compare_models, list('Elastic Net', mse, rmse))

cat(" RMSE:", rmse, "\n", "R-squared:", R2, "\n", "MSE:", mse)
```

```
##  RMSE: 0.7225156
##  R-squared: 0.4771372
##  MSE: 0.5220289
```

This model appears quite strong with a low MSE and RMSE on the testing data. Let's check the plot of residuals to see if this model gives us constant variance.

```
# plot of residuals
ggplot(broadband_test, mapping = aes(x = broadband_usage, y = (broadband_test$broadband_usage -
pred))) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, color = "gray", lty = "dashed") +
  labs(x = "Predicted", y = "Residuals")
```



Looks like we are still getting some nonrandom variance in our model.

Principle Component Regression

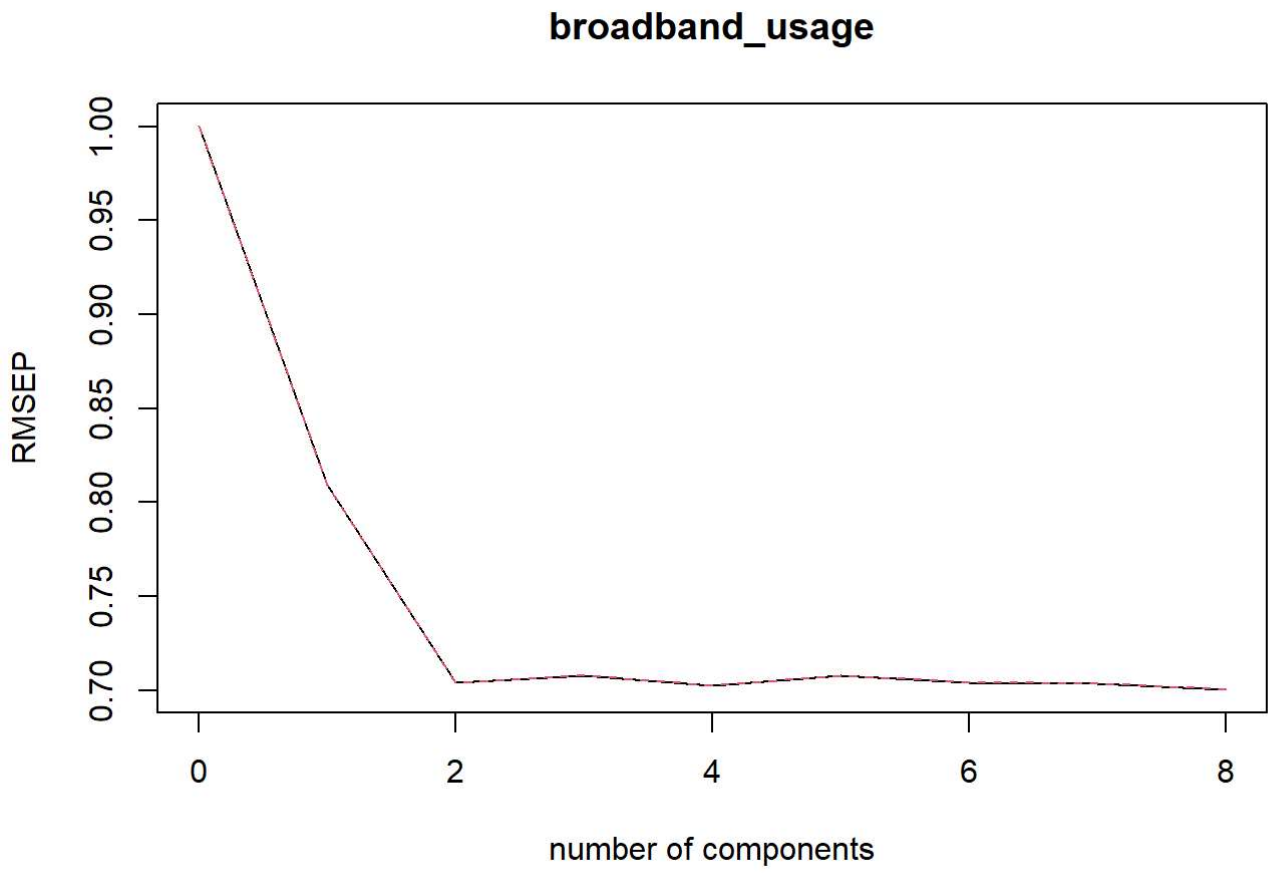
Principle component regression (PCR) expands on the multivariable linear regression by accounting for multicollinearity of predictor variables, which is when multiple predictors are correlated to each other. Multicollinearity effects the accuracy of multivariable linear regression, so PCR should improve the accuracy of this model.

Fit the model on training data:

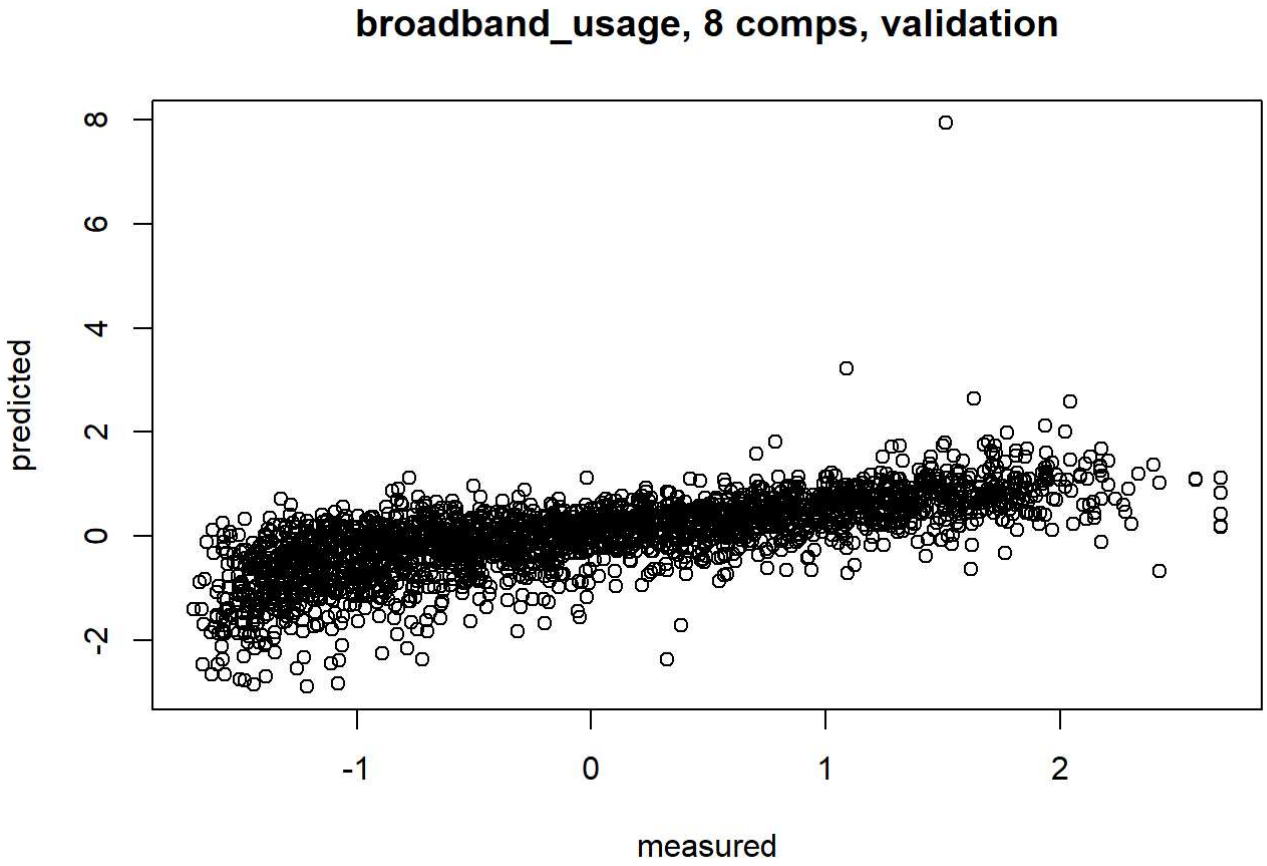
```
pcr_model <- pcr(formula = broadband_usage ~ .,
                         data = broadband_train[,c(4:12)], validation = "LOO")
summary(pcr_model)
```

```
## Data:    X dimension: 2504 8
##  Y dimension: 2504 1
## Fit method: svdpc
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 2504 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV               1   0.8093   0.7043   0.7076   0.7024   0.7077   0.7041
## adjCV            1   0.8093   0.7043   0.7076   0.7024   0.7077   0.7041
##        7 comps  8 comps
## CV      0.7036   0.7003
## adjCV   0.7036   0.7003
##
## TRAINING: % variance explained
##                  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X                  49.19    63.69    75.26    84.72    91.94    96.94    99.06
## broadband_usage    34.59    48.96    49.04    51.26    51.28    51.73    51.88
##                  8 comps
## X                 100.00
## broadband_usage    52.34
```

```
validationplot(pcr_model, val.type = "RMSE")
```



```
plot(pcr_model)
```



Using the elbow method, it appears that 2 components is sufficient for predicting broadband usage, as most additional components seems to add little to the model. Let's test a PCR model on 2 components:

```
pcr_pred <- predict(pcr_model, data = broadband_test, ncomp = 2)

rmse <- sqrt(mean((pcr_pred-broadband_test[,5])^2))
mse <- mean((broadband_test[,5] - pcr_pred)^2)

compare_models <- rbind(compare_models, list('PCR', mse, rmse))

cat(" RMSE:", rmse, "\n", "MSE:", mse)
```

```
##  RMSE: 1.211207
##  MSE: 1.467023
```

These RMSE and MSE values are fairly low, meaning this model may be pretty good.

Comparing Models

Lets take a look at our RMSE and MSE values for each model and for the training and testing data.

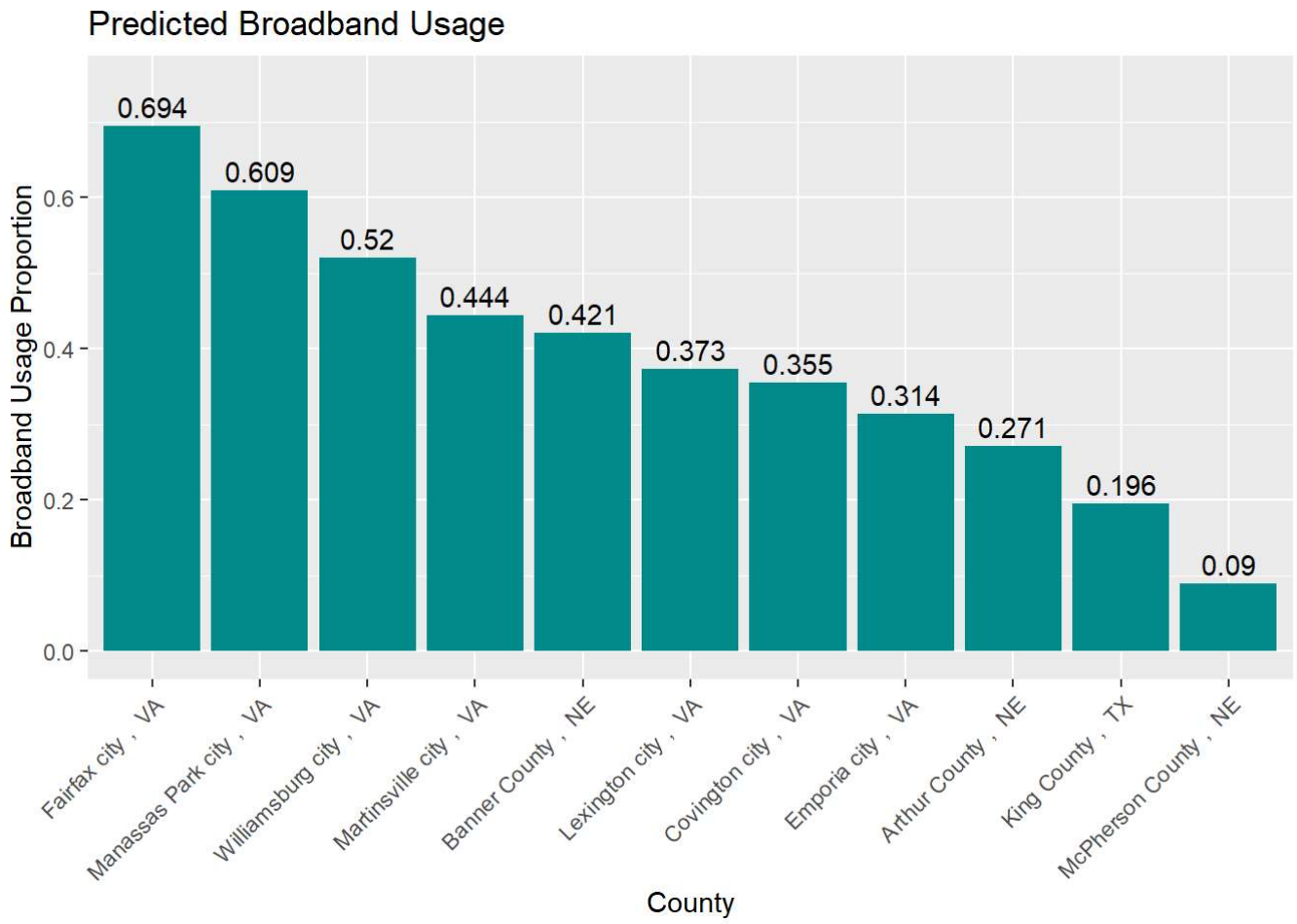
```
compare_models
```

```
##               Model       MSE      RMSE
## 1 Linear Regression 3.4029278 1.8447026
## 2               KNN 3.0613792 1.7496797
## 3       Elastic Net 0.5220289 0.7225156
## 4               PCR 1.4670233 1.2112074
```

Our model with the smallest error after predicting on the testing data is the Elastic Net Regularized Regression Model. We want a value as close to 0 as possible for RMSE, and a value of 0.726 is very close for a model based on real world data. I would recommend this model for predicting the broadband usage for a given U.S. county.

Let's try this model out on our 11 counties that were missing broadband usage data:

```
broadband_predict <- broadband_missing %>% mutate(broadband_usage = predict(elastic_mod, as.matr
ix(broadband_missing[,c(4,6:12)])))

broadband_predict_transform <- broadband_predict %>% select(state, county_name, broadband_usage)
broadband_predict_transform <- broadband_predict_transform %>% mutate(broadband_usage = (broadba
nd_usage * broadband_std) + broadband_mean)

broadband_predict_transform %>% ggplot(aes(x = fct_rev(fct_reorder(paste(county_name,', ', stat
e), broadband_usage)), y = broadband_usage)) +
  geom_col(fill = 'cyan4') +
  labs(title = "Predicted Broadband Usage", x = "County", y = "Broadband Usage Proportion") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = round(broadband_usage, digits = 3)), vjust = -0.4) +
  ylim(0, 0.75)
```


Predicted Broadband Usage
Broadband Usage Proportion
0.0
0.2
0.4
0.6
0.694
0.609
0.52
0.444
0.421
0.373
0.355
0.314
0.271
0.196
0.09
Fairfax city , VA
Manassas Park city , VA
Williamsburg city , VA
Martinsville city , VA
Banner County , NE
Lexington city , VA
Covington city , VA
Emporia city , VA
Arthur County , NE
King County , TX
McPherson County , NE
County