MGMT 6203 Group Project

Code

MGMT6203 Team 26

Preface

Initial data joins and clean-up were done in Python prior to this work in R. In the case of missing graduation rates for a subset, the overall graduation rate was substituted in its place.

In the case of missing teacher salary information or missing operational expenditures, the row was dropped. Three rows were dropped for this.

In the case of missing ‘mean income if on public assistance’, the imputation value was the average of the column. This applied to 12 rows.

Importing Data into R & Loading Required Packages

Hide

```
# Loading R libraries used throughout this analysis
library(ggplot2)
```

```
RStudio Community is a great place to get help:
https://community.rstudio.com/c/tidyverse
```

Hide

```
library(dplyr)
```

```

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

Hide

```
library(reshape)
```

```
Attaching package: ‘reshape’

The following object is masked from ‘package:dplyr’:

    rename
```

Hide

```
library(car)
```

```
Loading required package: carData

Attaching package: ‘car’

The following object is masked from ‘package:dplyr’:

    recode
```

Hide

```
library(tidyverse)
```

```
Registered S3 methods overwritten by 'dbplyr':
  method         from
  print.tbl_lazy
  print.tbl_sql
-- Attaching packages ------------------------------ tidyverse 1.3.1 --
âˆš tibble  3.1.6     âˆš purrr   0.3.4
âˆš tidyr   1.2.0     âˆš stringr 1.4.0
âˆš readr   2.1.2     âˆš forcats 0.5.1
-- Conflicts ---------------------------------- tidyverse_conflicts() --
x tidyr::expand()   masks reshape::expand()
x dplyr::filter()   masks stats::filter()
x dplyr::lag()      masks stats::lag()
x car::recode()     masks dplyr::recode()
x reshape::rename() masks dplyr::rename()
x purrr::some()     masks car::some()
```

Hide

```
library(pls)
```

```
Attaching package: ‘pls’

The following object is masked from ‘package:stats’:

    loadings
```

Hide

```
library(randomForest)
```

```
randomForest 4.7-1
Type rfNews() to see new features/changes/bug fixes.

Attaching package: ‘randomForest’

The following object is masked from ‘package:dplyr’:

    combine

The following object is masked from ‘package:ggplot2’:

    margin
```

Hide

```
library(tibble)
library(tidyr)
library(dplyr)
library(corrplot)
```

```
corrplot 0.92 loaded

Attaching package: ‘corrplot’

The following object is masked from ‘package:pls’:

    corrplot
```

Hide

```
options(max.print = 10000)

# File pathway where the datafile is stored
path = "C:\\Users\\sudip\\OneDrive\\Desktop\\MGMT Project Data\\master_datafile.csv"

# Loading the data from a csv file
data_import <- read.csv(path, header=TRUE, stringsAsFactors=FALSE)
# Dropping the column that contains a row index
data_import <- subset(data_import, select=c(2:100))
```

Data Dictionary

If you were to look at the data imported, you would see 8 different dependent variables (the graduation rate for each school as well as graduation rates for key groups of students, e.g., hispanic students, economically disadvantaged students, female students, etc.) and 91 potential independent variables to use in our regression model. As the names for these variables can be quite long, an alias will be used in their place. The dictionary below provides a reference for what each variable references.

y1 = GRAD_RATE_OVERALL,
y2 = GRAD_RATE_BLACK,
y3 = GRAD_RATE_ASIAN,
y4 = GRAD_RATE_HISPANIC,
y5 = GRAD_RATE_WHITE,
y6 = GRAD_RATE_ECONOMIC_DISADVANTAGE,
y7 = GRAD_RATE_FEMALE,
y8 = GRAD_RATE_MALE,
x1 = TOTAL_OP_EXPENDITURE,
x2 = OP_EXPENDITURE_PER_STUDENT,
x3 = FTE_COUNT,
x4 = TOTAL_SALARY_SPEND,
x5 = AVG_TEACHER_SALARY,
x6 = YEAR,
x7 = TOTAL_POP,
x8 = PERCENT_URBAN,
x9 = PERCENT_RURAL,
x10 = TOTAL_HOUSING_AVAILABLE,
x11 = PERCENT_HOUSING_OCCUPIED,
x12 = MOBILE_HOMES_PERCENTAGE_OF_HOUSING,
x13 = PERCENTAGE_OF_HOMES_OWNER_OCCUPIED,
x14 = PERCENTAGE_OF_HOMES_RENTED,
x15 = AVERAGE_HOUSEHOLD_SIZE_OWNED,
x16 = AVERAGE_HOUSEHOLD_SIZE_RENTED,
x17 = PERCENT_OF_HOMES_W_NO_VEHICLE,
x18 = PERCENT_OF_HOMES_VALUED_LESS_THAN_50000,
x19 = PERCENT_OF_HOMES_VALUED_50000_to_99999,
x20 = PERCENT_OF_HOMES_VALUED_100000_TO_149999,
x21 = PERCENT_OF_HOMES_VALUED_150000_TO_199999,
x22 = PERCENT_OF_HOMES_VALUED_200000_TO_299999,
x23 = PERCENT_OF_HOMES_VALUED_300000_TO_499999,
x24 = PERCENT_OF_HOMES_VALUED_500000_TO_999999,
x25 = PERCENT_OF_HOMES_VALUED_1000000_OR_MORE,
x26 = MEDIAN_HOME_VALUE,
x27 = PERCENTAGE_OF_HOMES_W_MORTGAGE,
x28 = PERCENTAGE_OF_HOMES_W_NO_MORTGAGE,
x29 = PERCENTAGE_OF_RENTERS_PAYING_LESS_THAN_500,
x30 = PERCENTAGE_OF_RENTERS_PAYING_500_TO_999,
x31 = PERCENTAGE_OF_RENTERS_PAYING_1000_TO_1499,
x32 = PERCENTAGE_OF_RENTERS_PAYING_1500_TO_1999,
x33 = PERCENTAGE_OF_RENTERS_PAYING_2000_TO_2499,
x34 = PERCENTAGE_OF_RENTERS_PAYING_2500_TO_2999,
x35 = PERCENTAGE_OF_RENTERS_PAYING_3000_OR_MORE,
x36 = MEDIAN_RENT,
x37 = RENT_AS_PERCENT_OF_INCOME,
x38 = POP_16_YEAR_AND_OVER,

x39 = PERCENT_OF_LABOR_16_YEAR_AND_OVER,
x40 = PERCENT_UNEMPLOYED_16_YEAR_AND_OVER,
x41 = PERCENT_OF_LABOR_FEMALE_AND_16_AND_OVER,
x42 = PERCENT_EMPLOYED_FEMALE_AND_16_AND_OVER,
x43 = PERCENT_OF_HOMES_WITH_CHILDREN_UNDER_6_BOTH_PARENTS_WORK,
x44 = NUM_OF_HOMES_WITH_CHILDREN_6_TO_17_YEARS,
x45 = PERCENT_OF_HOMES_WITH_CHILDREN_6_TO_17_BOTH_PARENTS_WORK,
x46 = PERCENT_W_INCOME_LESS_THAN_10000,
x47 = PERCENT_W_INCOME_10000_TO_14999,
x48 = PERCENT_W_INCOME_15000_TO_24999,
x49 = PERCENT_W_INCOME_25000_To_34999,
x50 = PERCENT_W_INCOME_35000_TO_49999,
x51 = PERCENT_W_INCOME_50000_TO_74999,
x52 = PERCENT_W_INCOME_75000_TO_99999,
x53 = PERCENT_W_INCOME_100000_TO_149999,
x54 = PERCENT_W_INCOME_150000_TO_199999,
x55 = PERCENT_W_INCOME_200000_OR_MORE,
x56 = MEDIAN_HOUSEHOLD_INCOME,
x57 = MEAN_HOUSEHOLD_INCOME,
x58 = PERCENT_ON_SOCIAL_SECURITY,
x59 = PERCENT_HOUSEHOLDS_RETIRED,
x60 = PERCENT_RECIEVING_PUBLIC_ASSISTANCE,
x61 = MEAN_INCOME_IF_PUBLIC_ASSISTANCE,
x62 = PERCENT_RECEIVING_FOOD_STAMPS,
x63 = PERCENT_NO_HEALTH_INSURANCE,
x64 = PERCENT_CHILDREN_NO_HEALTH_INSURANCE,
x65 = PERCENT_FAMILIES_W_CHILDREN_BELOW_POVERTY,
x66 = PERCENT_POP_MALE,
x67 = PERCENT_POP_FEMALE,
x68 = PERCENT_POP_UNDER_5,
x69 = PERCENT_POP_5_TO_9,
x70 = PERCENT_POP_10_TO_14,
x71 = PERCENT_POP_15_TO_19,
x72 = PERCENT_POP_20_TO_24,
x73 = PERCENT_POP_25_34,
x74 = PERCENT_POP_35_TO_44,
x75 = PERCENT_POP_45_TO_54,
x76 = PERCENT_POP_55_TO_59,
x77 = PERCENT_POP_60_TO_64,
x78 = PERCENT_POP_65_TO_74,
x79 = PERCENT_POP_75_TO_84,
x80 = PERCENT_POP_85_OR_OLDER,
x81 = MEDIAN_POP_AGE,
x82 = PERCENT_POP_WHITE,
x83 = PERCENT_POP_BLACK,
x84 = PERCENT_POP_AMINDIAN_NATIVE,
x85 = PERCENT_POP_ASIAN,
x86 = PERCENT_POP_HAWAII_PAC_ISL,
x87 = PERCENT_POP_OTHER,

x88 = PERCENT_POP_HISPANIC,
x89 = DISTRICT_TYPE_Major.Suburban,
x90 = DISTRICT_TYPE_Major.Urban,
x91 = DISTRICT_TYPE_Non.metropolitan.Stable

Hide

```
# Rename all data columns as per the data dictionary

data_import <- dplyr::rename(data_import, y1 = GRAD_RATE_OVERALL,
y2 = GRAD_RATE_BLACK,
y3 = GRAD_RATE_ASIAN,
y4 = GRAD_RATE_HISPANIC,
y5 = GRAD_RATE_WHITE,
y6 = GRAD_RATE_ECONOMIC_DISADVANTAGE,
y7 = GRAD_RATE_FEMALE,
y8 = GRAD_RATE_MALE,
x1 = TOTAL_OP_EXPENDITURE,
x2 = OP_EXPENDITURE_PER_STUDENT,
x3 = FTE_COUNT,
x4 = TOTAL_SALARY_SPEND,
x5 = AVG_TEACHER_SALARY,
x6 = YEAR,
x7 = TOTAL_POP,
x8 = PERCENT_URBAN,
x9 = PERCENT_RURAL,
x10 = TOTAL_HOUSING_AVAILABLE,
x11 = PERCENT_HOUSING_OCCUPIED,
x12 = MOBILE_HOMES_PERCENTAGE_OF_HOUSING,
x13 = PERCENTAGE_OF_HOMES_OWNER_OCCUPIED,
x14 = PERCENTAGE_OF_HOMES_RENTED,
x15 = AVERAGE_HOUSEHOLD_SIZE_OWNED,
x16 = AVERAGE_HOUSEHOLD_SIZE_RENTED,
x17 = PERCENT_OF_HOMES_W_NO_VEHICLE,
x18 = PERCENT_OF_HOMES_VALUED_LESS_THAN_50000,
x19 = PERCENT_OF_HOMES_VALUED_50000_to_99999,
x20 = PERCENT_OF_HOMES_VALUED_100000_TO_149999,
x21 = PERCENT_OF_HOMES_VALUED_150000_TO_199999,
x22 = PERCENT_OF_HOMES_VALUED_200000_TO_299999,
x23 = PERCENT_OF_HOMES_VALUED_300000_TO_499999,
x24 = PERCENT_OF_HOMES_VALUED_500000_TO_999999,
x25 = PERCENT_OF_HOMES_VALUED_1000000_OR_MORE,
x26 = MEDIAN_HOME_VALUE,
x27 = PERCENTAGE_OF_HOMES_W_MORTGAGE,
x28 = PERCENTAGE_OF_HOMES_W_NO_MORTGAGE,
x29 = PERCENTAGE_OF_RENTERS_PAYING_LESS_THAN_500,
x30 = PERCENTAGE_OF_RENTERS_PAYING_500_TO_999,
x31 = PERCENTAGE_OF_RENTERS_PAYING_1000_TO_1499,
x32 = PERCENTAGE_OF_RENTERS_PAYING_1500_TO_1999,
x33 = PERCENTAGE_OF_RENTERS_PAYING_2000_TO_2499,
x34 = PERCENTAGE_OF_RENTERS_PAYING_2500_TO_2999,
x35 = PERCENTAGE_OF_RENTERS_PAYING_3000_OR_MORE,
x36 = MEDIAN_RENT,
x37 = RENT_AS_PERCENT_OF_INCOME,
x38 = POP_16_YEAR_AND_OVER,
x39 = PERCENT_OF_LABOR_16_YEAR_AND_OVER,
x40 = PERCENT_UNEMPLOYED_16_YEAR_AND_OVER,
x41 = PERCENT_OF_LABOR_FEMALE_AND_16_AND_OVER,
x42 = PERCENT_EMPLOYED_FEMALE_AND_16_AND_OVER,
```

```
x43 = PERCENT_OF_HOMES_WITH_CHILDREN_UNDER_6_BOTH_PARENTS_WORK,
x44 = NUM_OF_HOMES_WITH_CHILDREN_6_TO_17_YEARS,
x45 = PERCENT_OF_HOMES_WITH_CHILDREN_6_TO_17_BOTH_PARENTS_WORK,
x46 = PERCENT_W_INCOME_LESS_THAN_10000,
x47 = PERCENT_W_INCOME_10000_TO_14999,
x48 = PERCENT_W_INCOME_15000_TO_24999,
x49 = PERCENT_W_INCOME_25000_To_34999,
x50 = PERCENT_W_INCOME_35000_TO_49999,
x51 = PERCENT_W_INCOME_50000_TO_74999,
x52 = PERCENT_W_INCOME_75000_TO_99999,
x53 = PERCENT_W_INCOME_100000_TO_149999,
x54 = PERCENT_W_INCOME_150000_TO_199999,
x55 = PERCENT_W_INCOME_200000_OR_MORE,
x56 = MEDIAN_HOUSEHOLD_INCOME,
x57 = MEAN_HOUSEHOLD_INCOME,
x58 = PERCENT_ON_SOCIAL_SECURITY,
x59 = PERCENT_HOUSEHOLDS_RETIRED,
x60 = PERCENT_RECIEVING_PUBLIC_ASSISTANCE,
x61 = MEAN_INCOME_IF_PUBLIC_ASSISTANCE,
x62 = PERCENT_RECEIVING_FOOD_STAMPS,
x63 = PERCENT_NO_HEALTH_INSURANCE,
x64 = PERCENT_CHILDREN_NO_HEALTH_INSURANCE,
x65 = PERCENT_FAMILIES_W_CHILDREN_BELOW_POVERTY,
x66 = PERCENT_POP_MALE,
x67 = PERCENT_POP_FEMALE,
x68 = PERCENT_POP_UNDER_5,
x69 = PERCENT_POP_5_TO_9,
x70 = PERCENT_POP_10_TO_14,
x71 = PERCENT_POP_15_TO_19,
x72 = PERCENT_POP_20_TO_24,
x73 = PERCENT_POP_25_34,
x74 = PERCENT_POP_35_TO_44,
x75 = PERCENT_POP_45_TO_54,
x76 = PERCENT_POP_55_TO_59,
x77 = PERCENT_POP_60_TO_64,
x78 = PERCENT_POP_65_TO_74,
x79 = PERCENT_POP_75_TO_84,
x80 = PERCENT_POP_85_OR_OLDER,
x81 = MEDIAN_POP_AGE,
x82 = PERCENT_POP_WHITE,
x83 = PERCENT_POP_BLACK,
x84 = PERCENT_POP_AMINDIAN_NATIVE,
x85 = PERCENT_POP_ASIAN,
x86 = PERCENT_POP_HAWAII_PAC_ISL,
x87 = PERCENT_POP_OTHER,
x88 = PERCENT_POP_HISPANIC,
x89 = DISTRICT_TYPE_Major.Suburban,
x90 = DISTRICT_TYPE_Major.Urban,
x91 = DISTRICT_TYPE_Non.metropolitan.Stable)
```

Hide

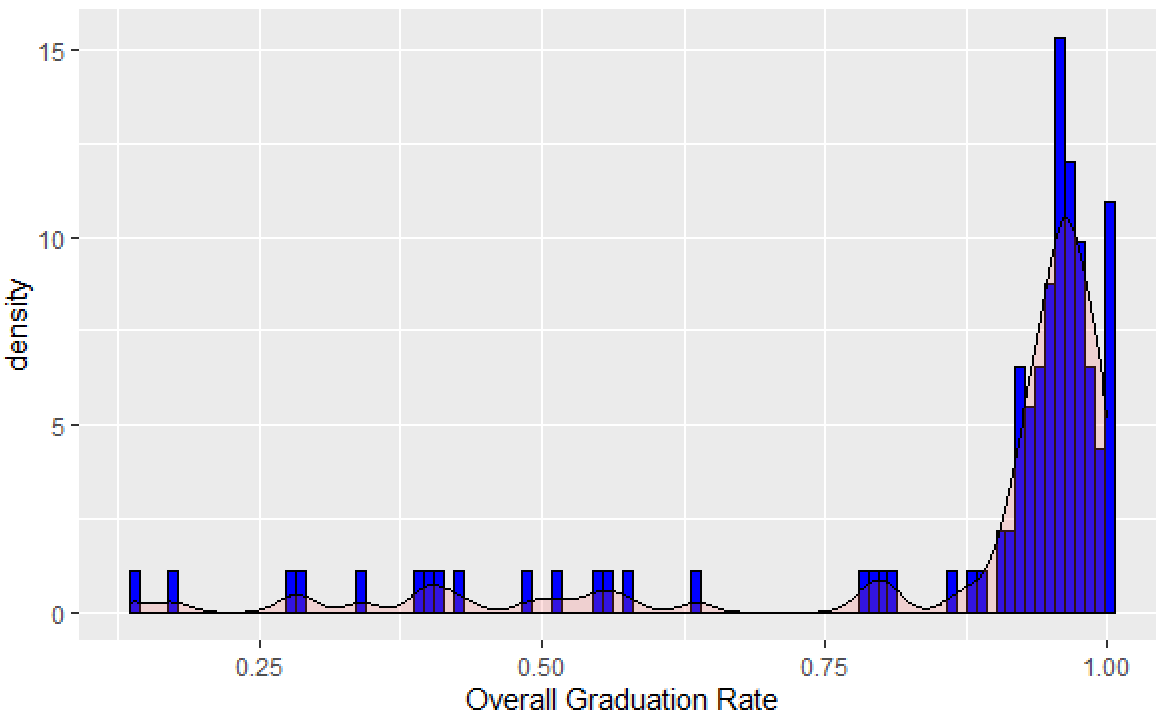
```
# Separate out the different dependent variables
y_variables <- subset(data_import, select=c(1:8))
# Separate out all the possible predictors
x_variables <- subset(data_import, select=c(9:99))
```

Checking the Distribution of the Target Variable

Now, we check the distribution of our target variable (graduation rates). This is done independently for each graduation rate subset (white graduation rate, female graduation rate, etc.)

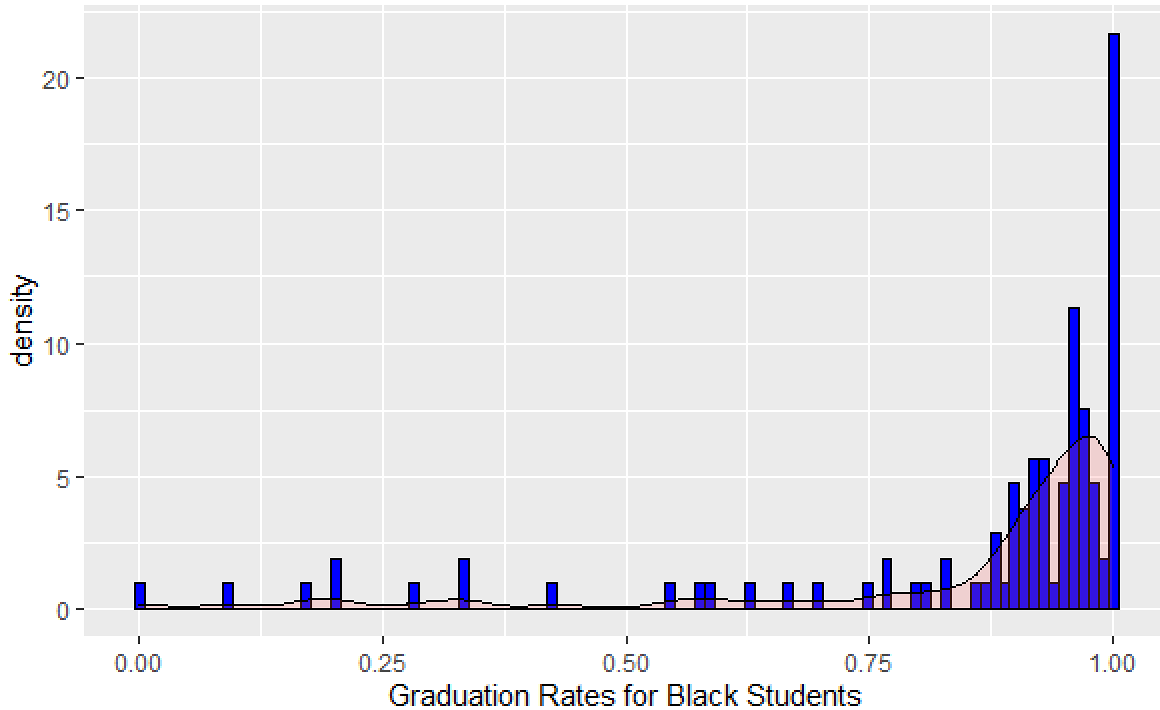
Hide

```
ggplot(data=y_variables, aes(y1)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "Overall Graduation Rate")
```



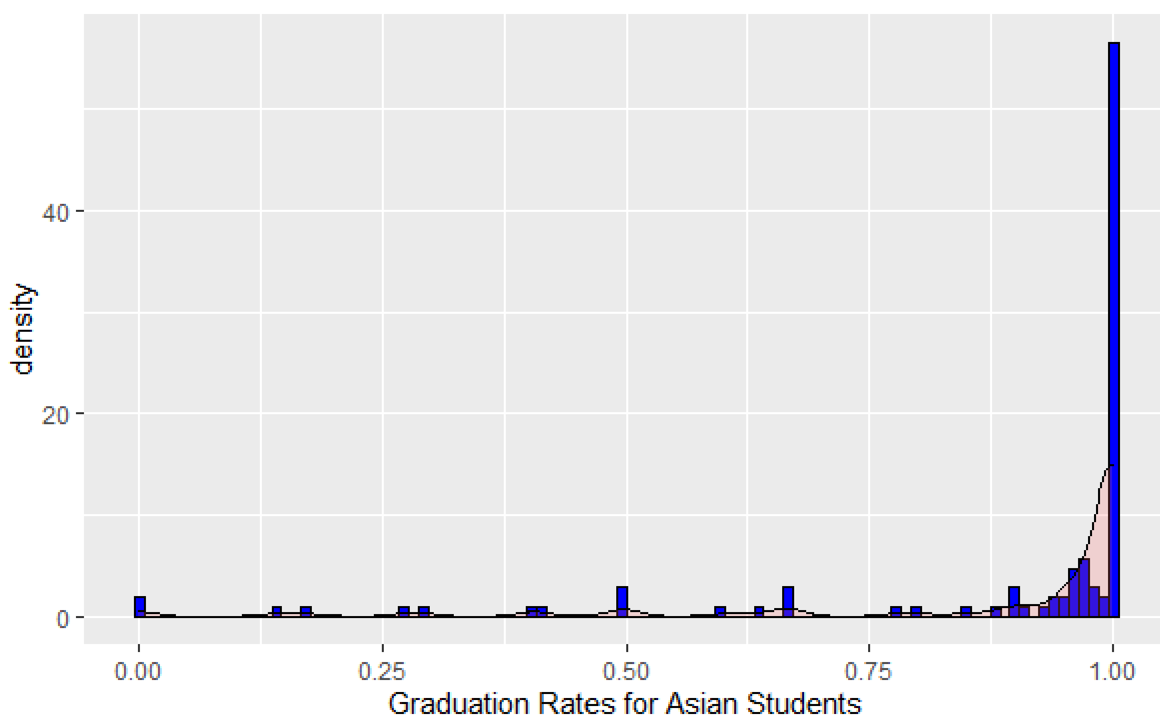
Hide

```
ggplot(data=y_variables, aes(y2)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "Graduation Rates for Black Students")
```



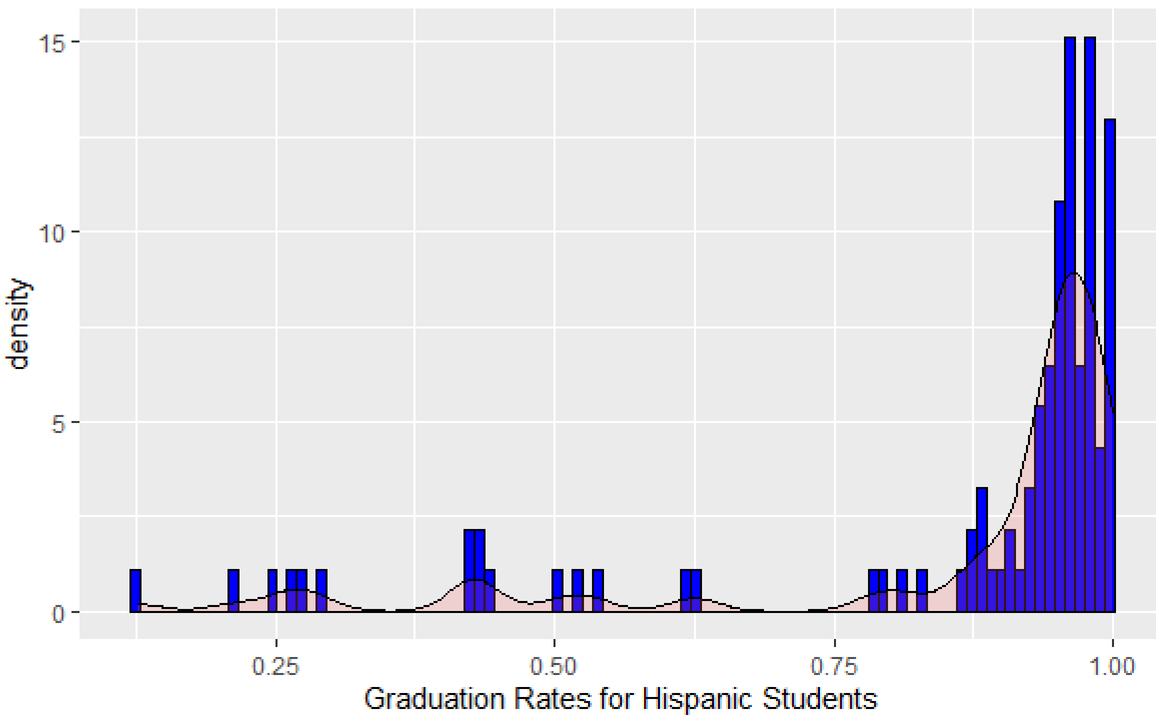
Hide

```
ggplot(data=y_variables, aes(y3)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "Graduation Rates for Asian Students")
```



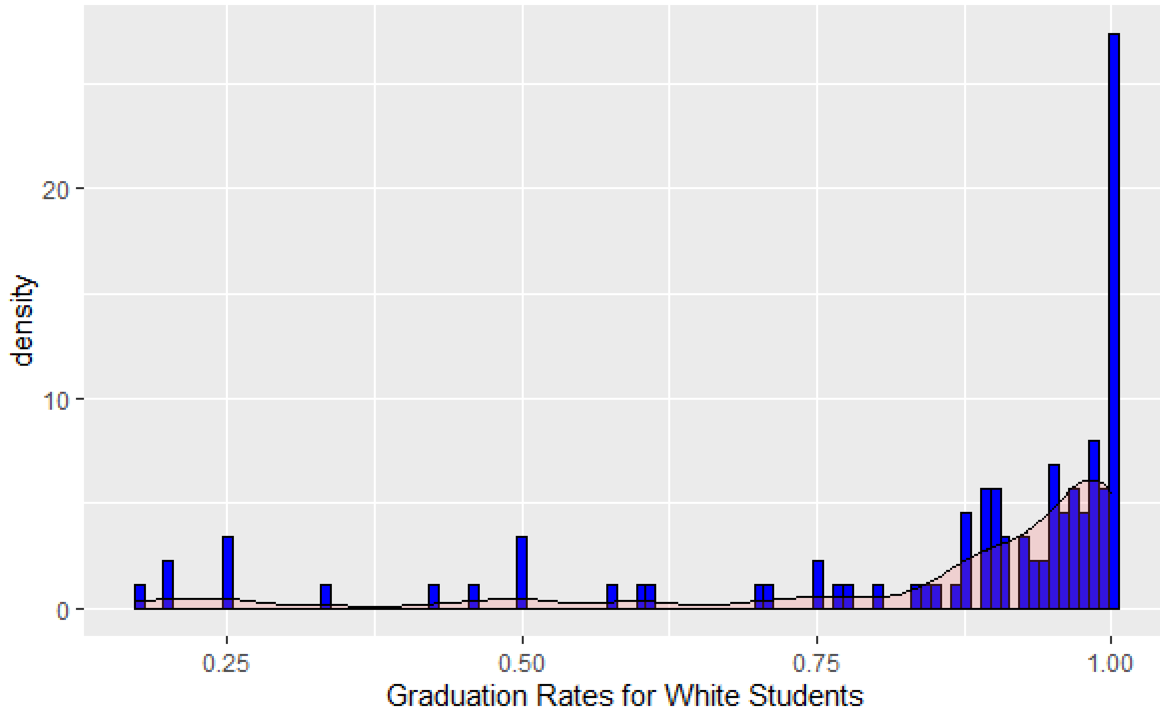
Hide

```
ggplot(data=y_variables, aes(y4)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "Graduation Rates for Hispanic Students")
```



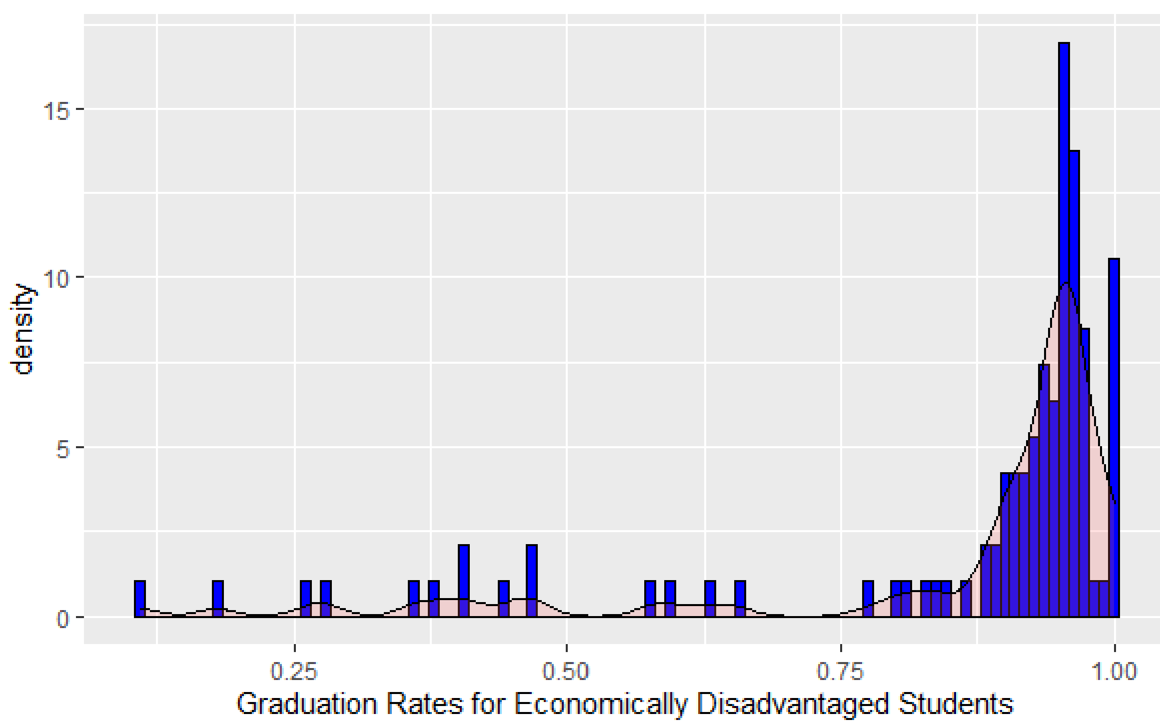
Hide

```
ggplot(data=y_variables, aes(y5)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "Graduation Rates for White Students")
```



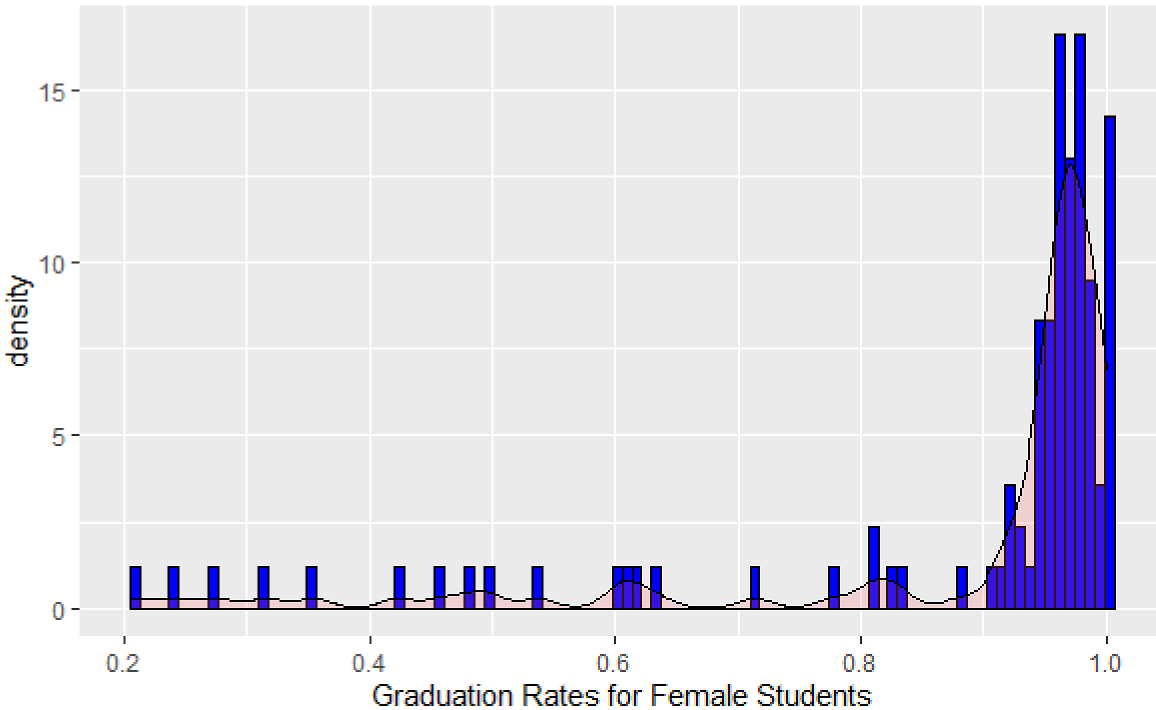
Hide

```
ggplot(data=y_variables, aes(y6)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "Graduation Rates for Economically Disadvantaged Students")
```



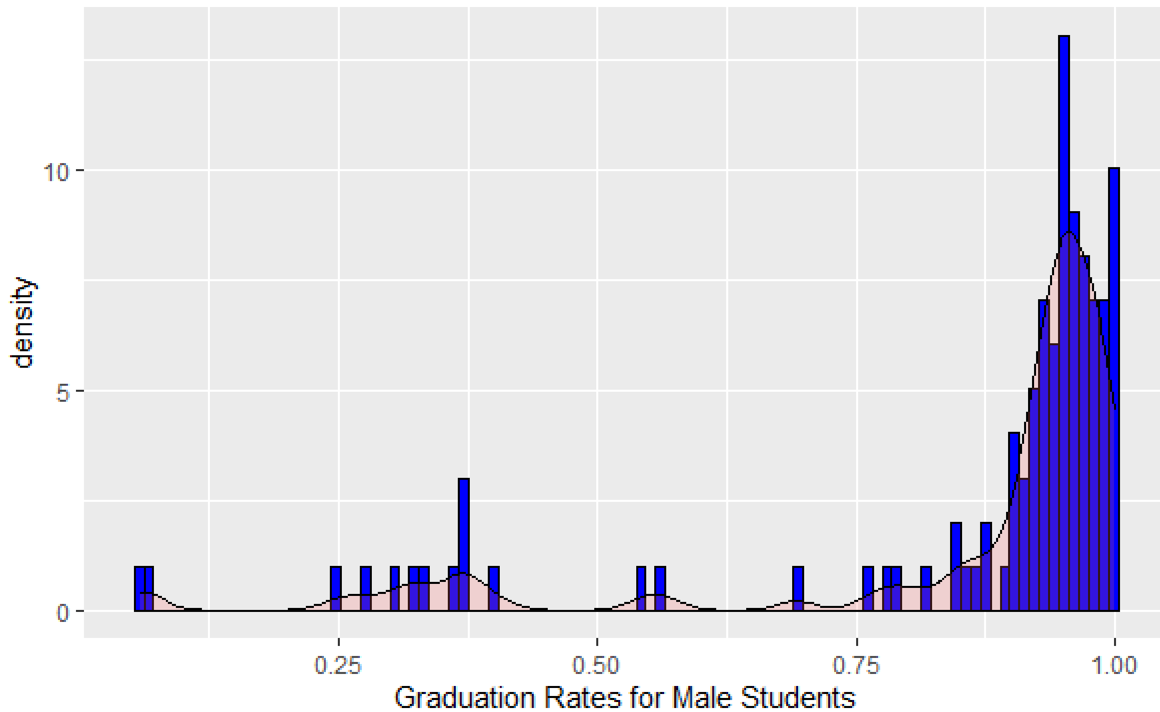
Hide

```
ggplot(data=y_variables, aes(y7)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "Graduation Rates for Female Students")
```



Hide

```
ggplot(data=y_variables, aes(y8)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "Graduation Rates for Male Students")
```

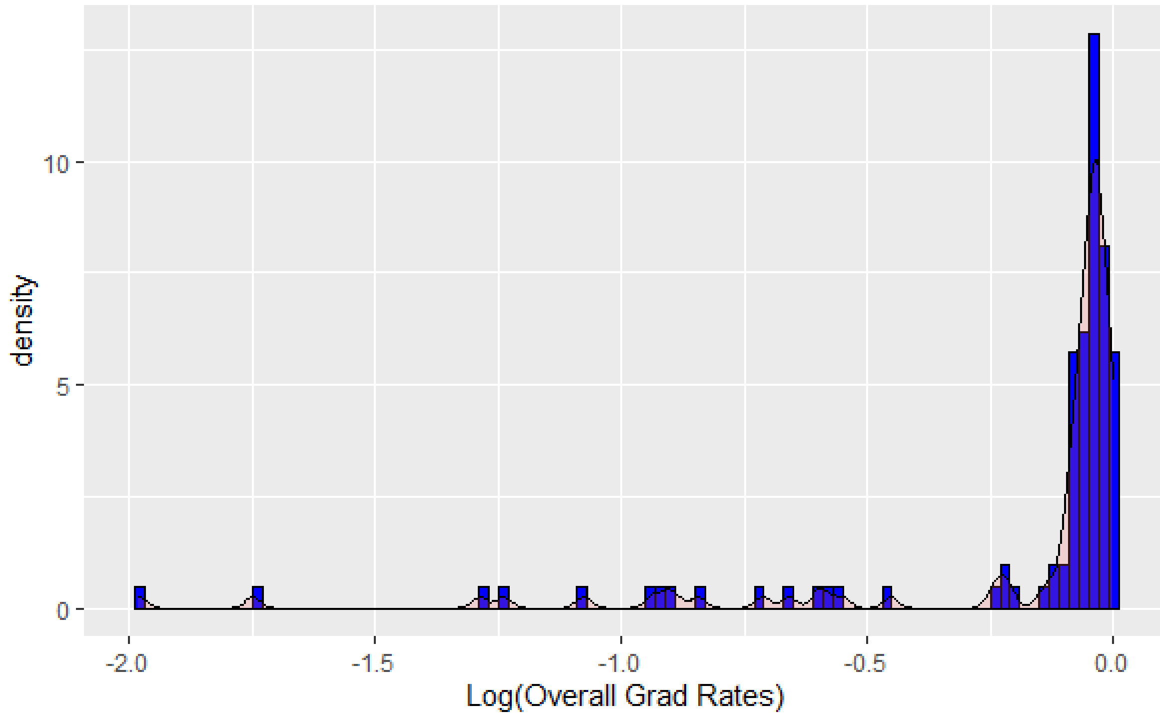


Tranformation of the Target Variable

As can be observed, all the target variables show significant left-skew. This would suggest that a transformation would be appropriate. Traditional transformations considered for a left-skewed data set include a log transform, a square root transform, or a cube root transform. Below, we see if any of these transformations make our dependent variable appear normally distributed.

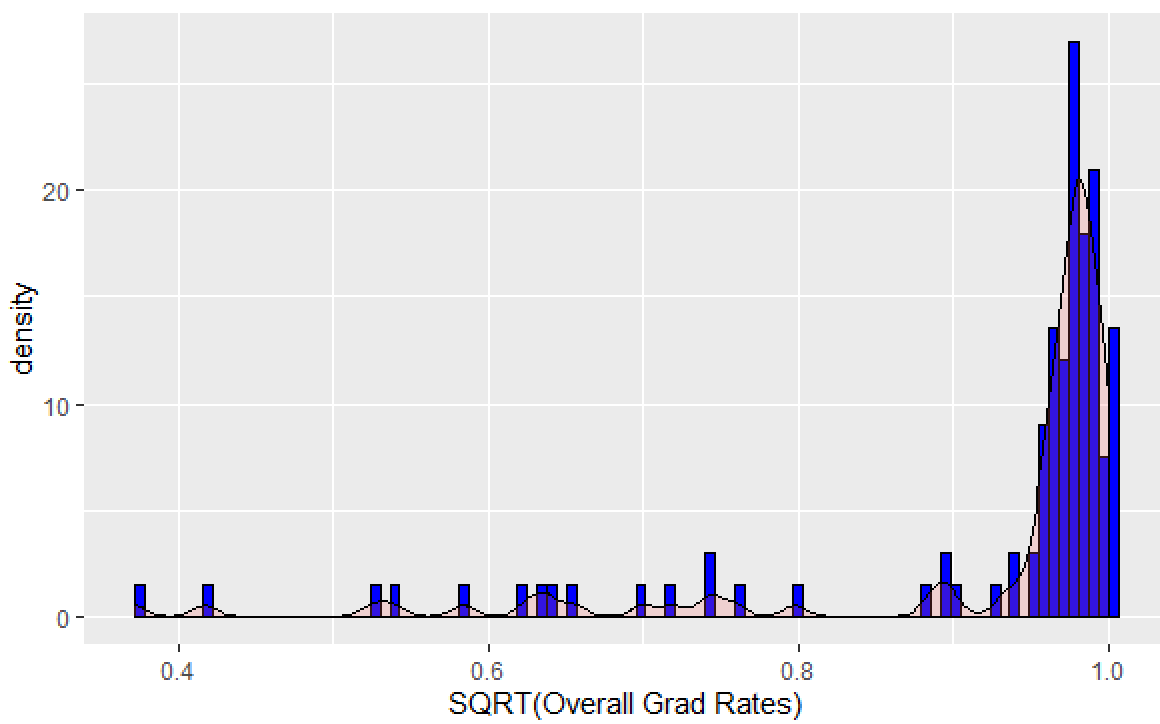
Hide

```
# Natural Log transformation
log_grad_rates <- log(y_variables['y1'])

ggplot(data=log_grad_rates, aes(y1)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "Log(Overall Grad Rates)")
```



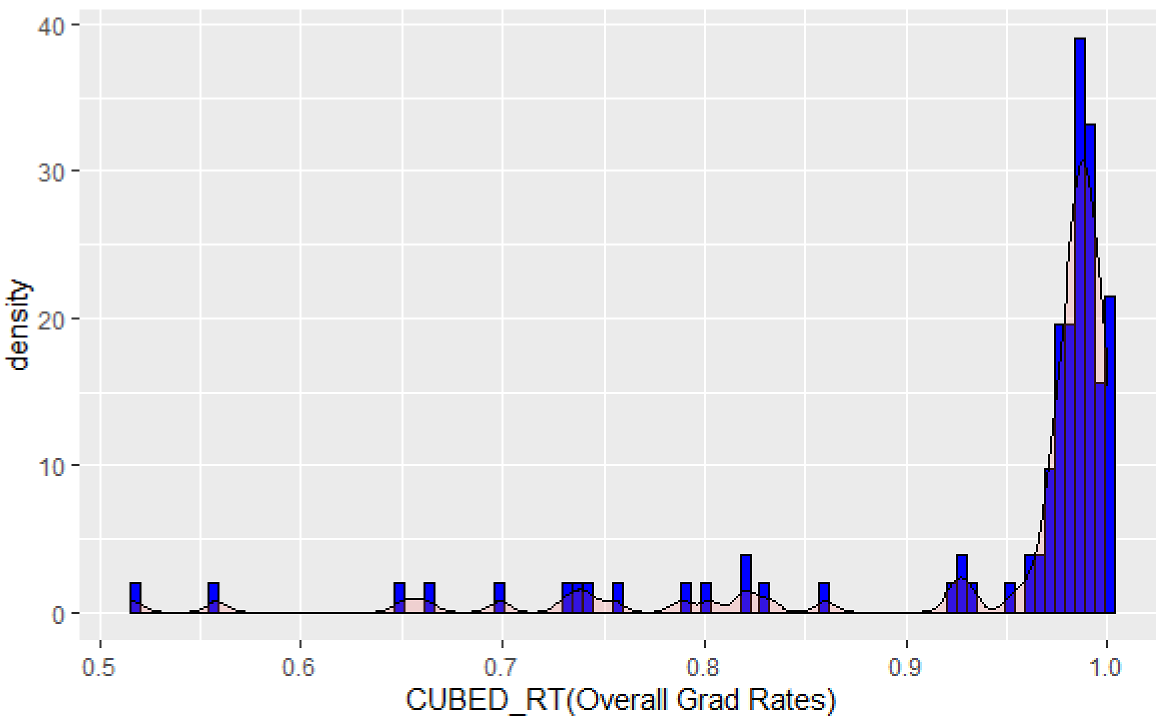
Hide

```
# Square root transformation
sqrt_grad_rates <- '^'(y_variables['y1'], 1/2)

ggplot(data=sqrt_grad_rates, aes(y1)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "SQRT(Overall Grad Rates)")
```



Hide

```
# Cubed Root transformation
cubert_grad_rates <- '^'(y_variables['y1'], 1/3)

ggplot(data=cubert_grad_rates, aes(y1)) +
  geom_histogram(aes(y =..density..), color="black", fill = "blue", bins=100) +
  geom_density(alpha = 0.2, fill = "#FF6666") +
  labs(x = "CUBED_RT(Overall Grad Rates)")
```



Regardless of the transformation used, the distribution of the dependent variable cannot be changed to a normal distribution. The left-skew is something that must be lived with and considered in the final analysis.

Checking Outliers Using Boxplots

For each numerical variable, a boxplot was constructed to visualize the distribution of that variable. If points lie beyond whispers, then outlier values are present. However, the presence of an outlier does not automatically suggest a data point should be excluded from the overall data set.

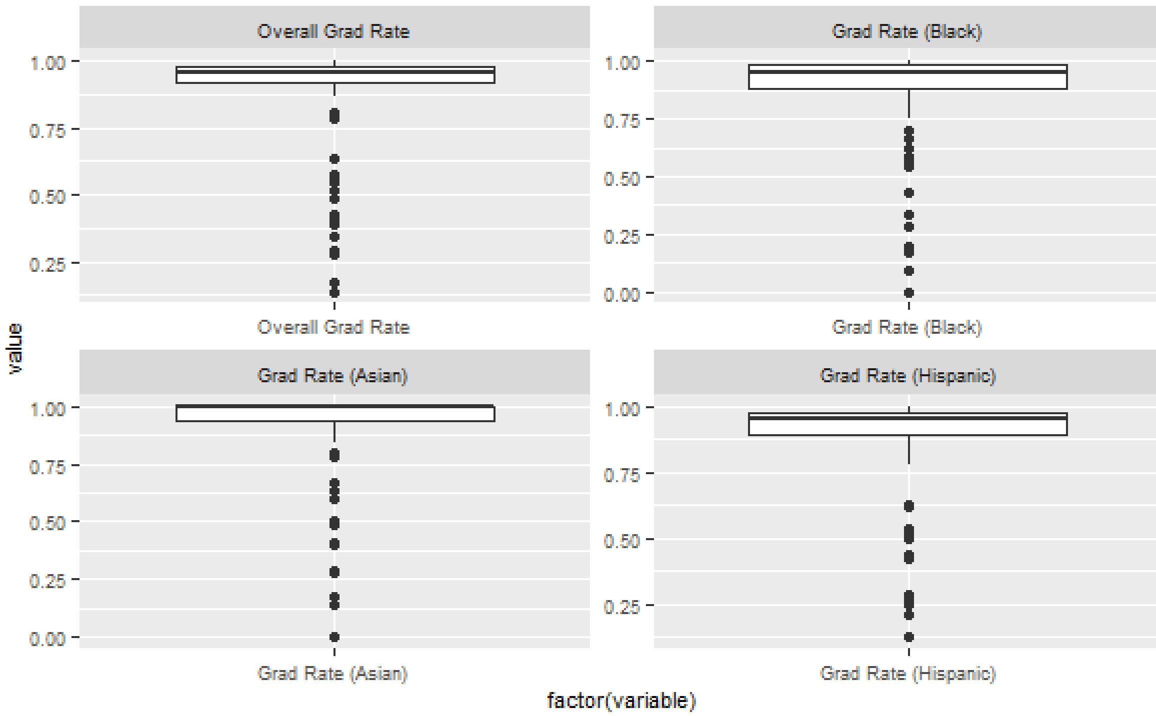
Hide

```
df <- subset(y_variables, select=c(1:4))
colnames(df) <- c("Overall Grad Rate",
                  "Grad Rate (Black)",
                  "Grad Rate (Asian)",
                  "Grad Rate (Hispanic)")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



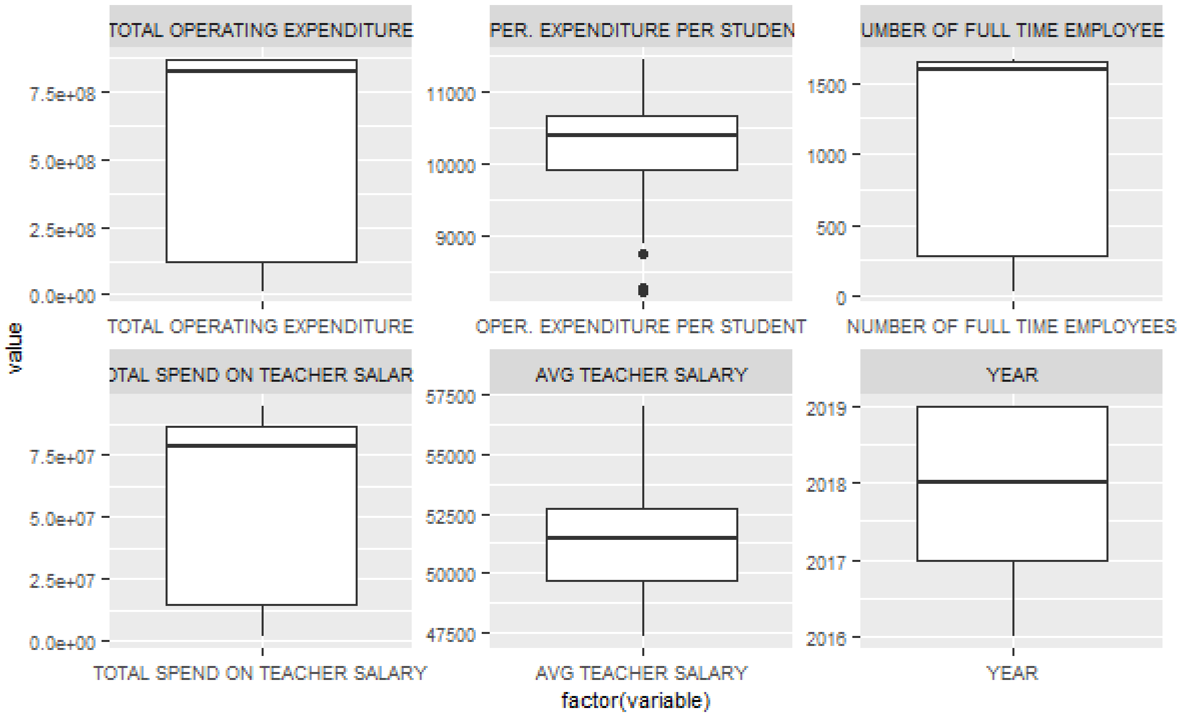
Hide

```
df <- subset(x_variables, select=c(1:6))
colnames(df) <- c("TOTAL OPERATING EXPENDITURE",
      "OPER. EXPENDITURE PER STUDENT",
      "NUMBER OF FULL TIME EMPLOYEES",
      "TOTAL SPEND ON TEACHER SALARY",
      "AVG TEACHER SALARY",
      "YEAR")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



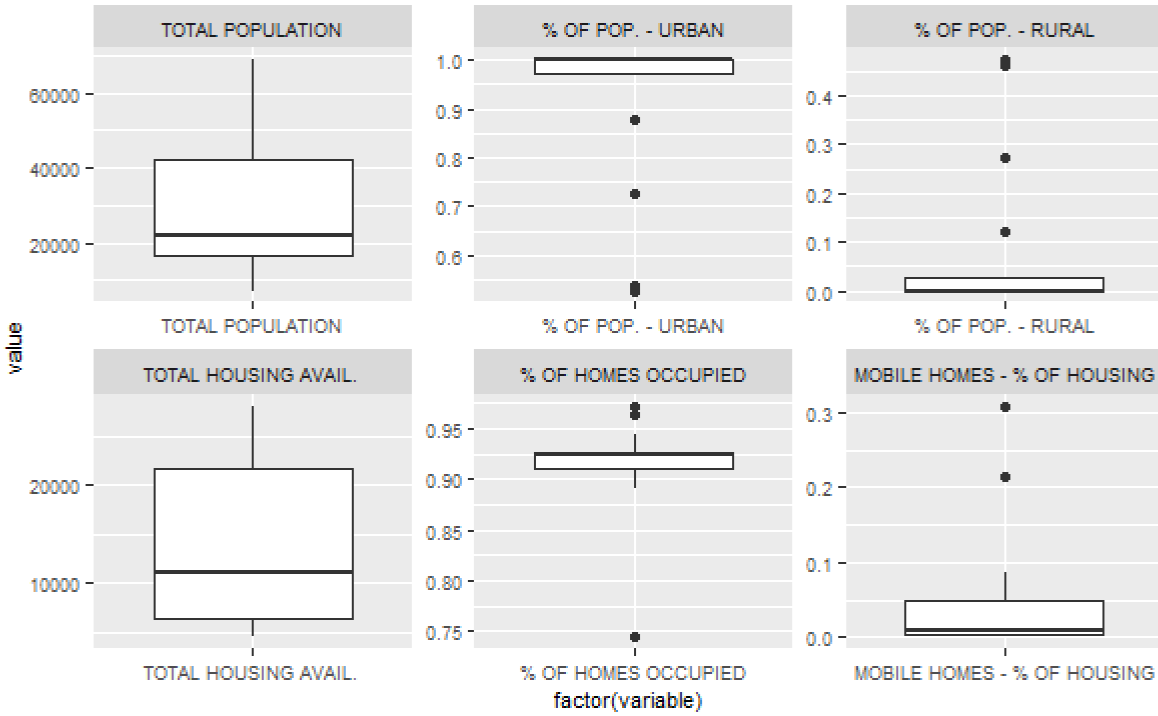
Hide

```
df <- subset(x_variables, select=c(7:12))
colnames(df) <- c("TOTAL POPULATION",
                  "% OF POP. - URBAN",
                  "% OF POP. - RURAL",
                  "TOTAL HOUSING AVAIL.",
                  "% OF HOMES OCCUPIED",
                  "MOBILE HOMES - % OF HOUSING")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



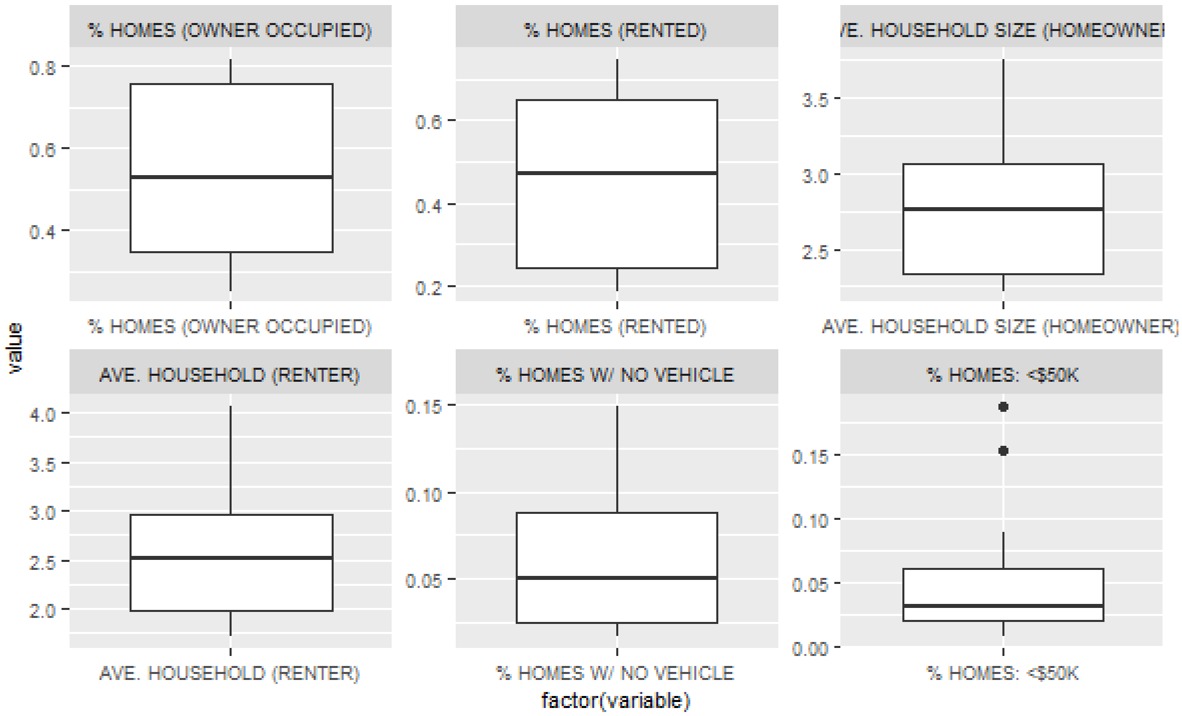
Hide

```
df <- subset(x_variables, select=c(13:18))
colnames(df) <- c("% HOMES (OWNER OCCUPIED)",
                  "% HOMES (RENTED)",
                  "AVE. HOUSEHOLD SIZE (HOMEOWNER)",
                  "AVE. HOUSEHOLD (RENTER)",
                  "% HOMES W/ NO VEHICLE",
                  "% HOMES: <$50K")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



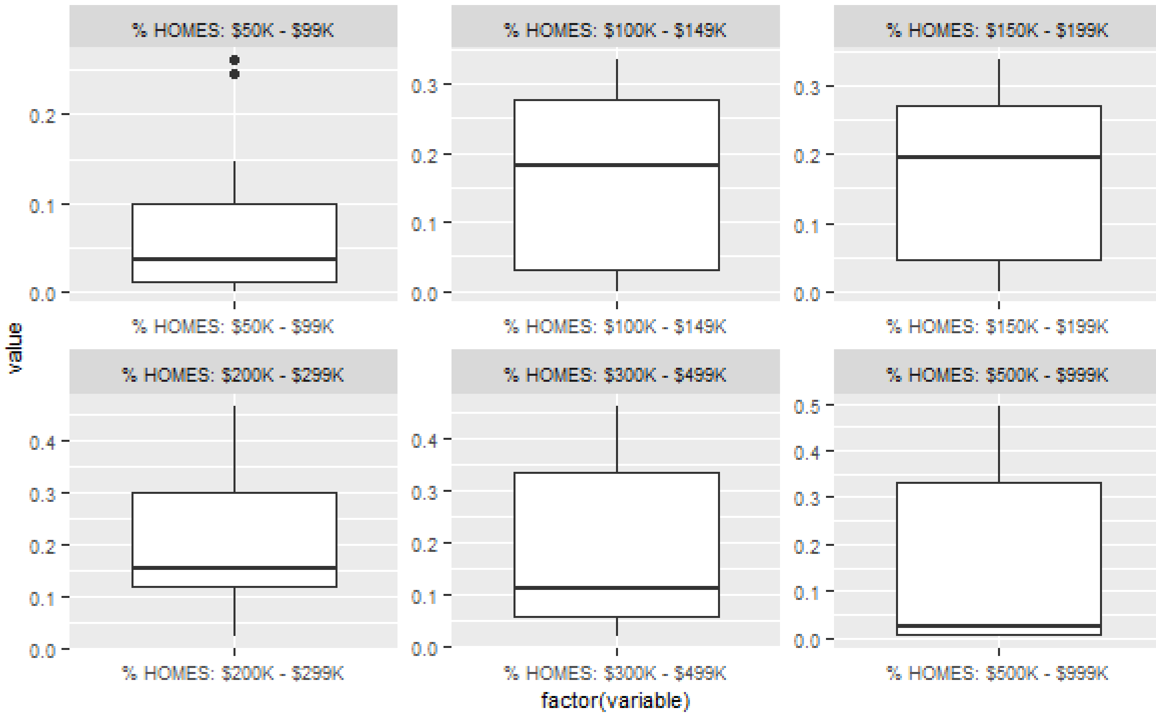
Hide

```
df <- subset(x_variables, select=c(19:24))
colnames(df) <- c("% HOMES: $50K - $99K",
                  "% HOMES: $100K - $149K",
                  "% HOMES: $150K - $199K",
                  "% HOMES: $200K - $299K",
                  "% HOMES: $300K - $499K",
                  "% HOMES: $500K - $999K")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



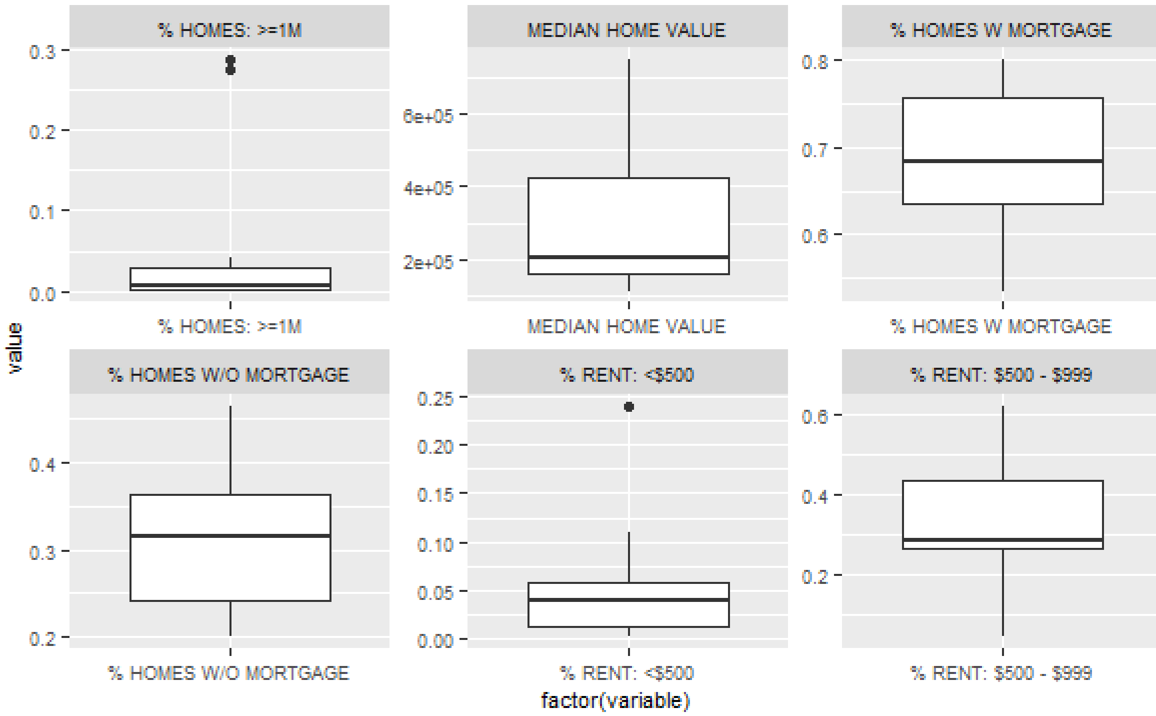
Hide

```
df <- subset(x_variables, select=c(25:30))
colnames(df) <- c("% HOMES: >=1M",
                  "MEDIAN HOME VALUE",
                  "% HOMES W MORTGAGE",
                  "% HOMES W/O MORTGAGE",
                  "% RENT: <$500",
                  "% RENT: $500 - $999")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



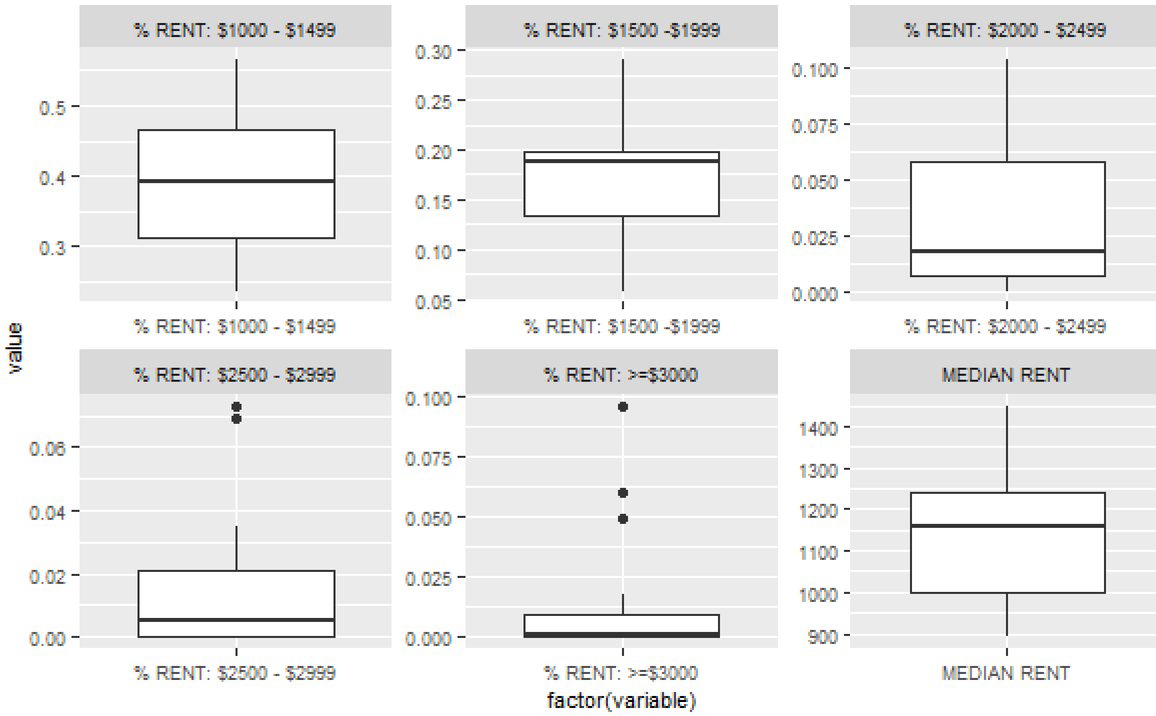
Hide

```
df <- subset(x_variables, select=c(31:36))
colnames(df) <- c("% RENT: $1000 - $1499",
                  "% RENT: $1500 -$1999",
                  "% RENT: $2000 - $2499",
                  "% RENT: $2500 - $2999",
                  "% RENT: >=$3000",
                  "MEDIAN RENT")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



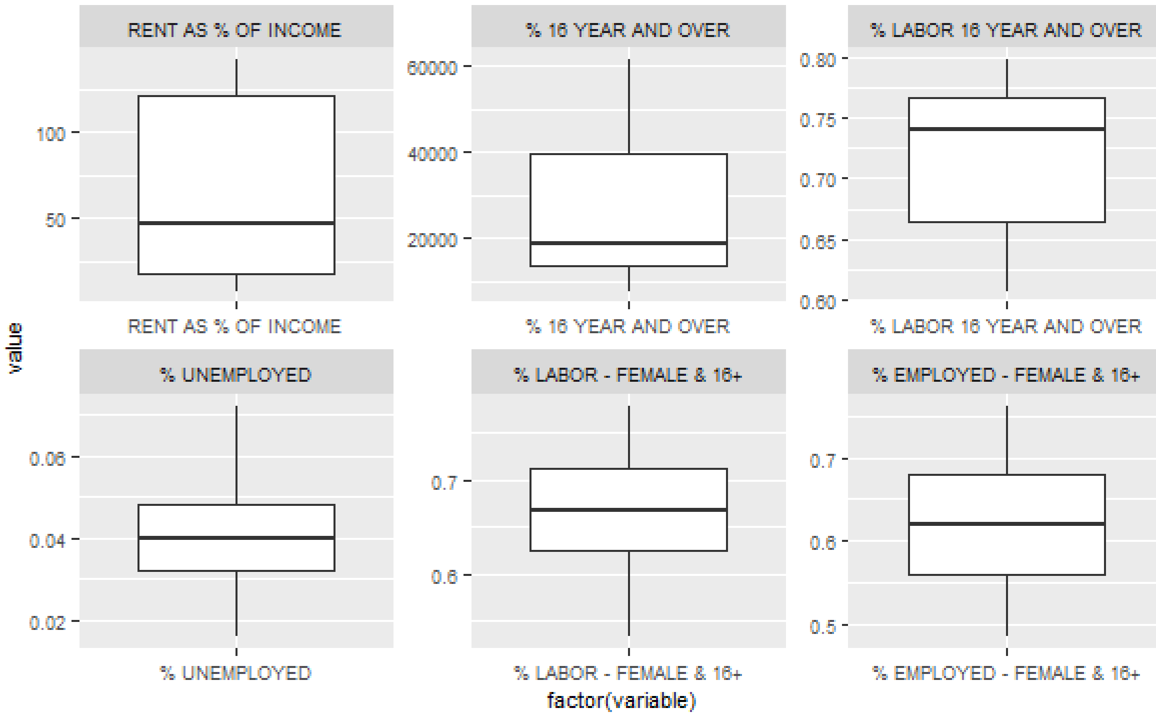
Hide

```
df <- subset(x_variables, select=c(37:42))
colnames(df) <- c("RENT AS % OF INCOME",
                  "% 16 YEAR AND OVER",
                  "% LABOR 16 YEAR AND OVER",
                  "% UNEMPLOYED",
                  "% LABOR - FEMALE & 16+",
                  "% EMPLOYED - FEMALE & 16+")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



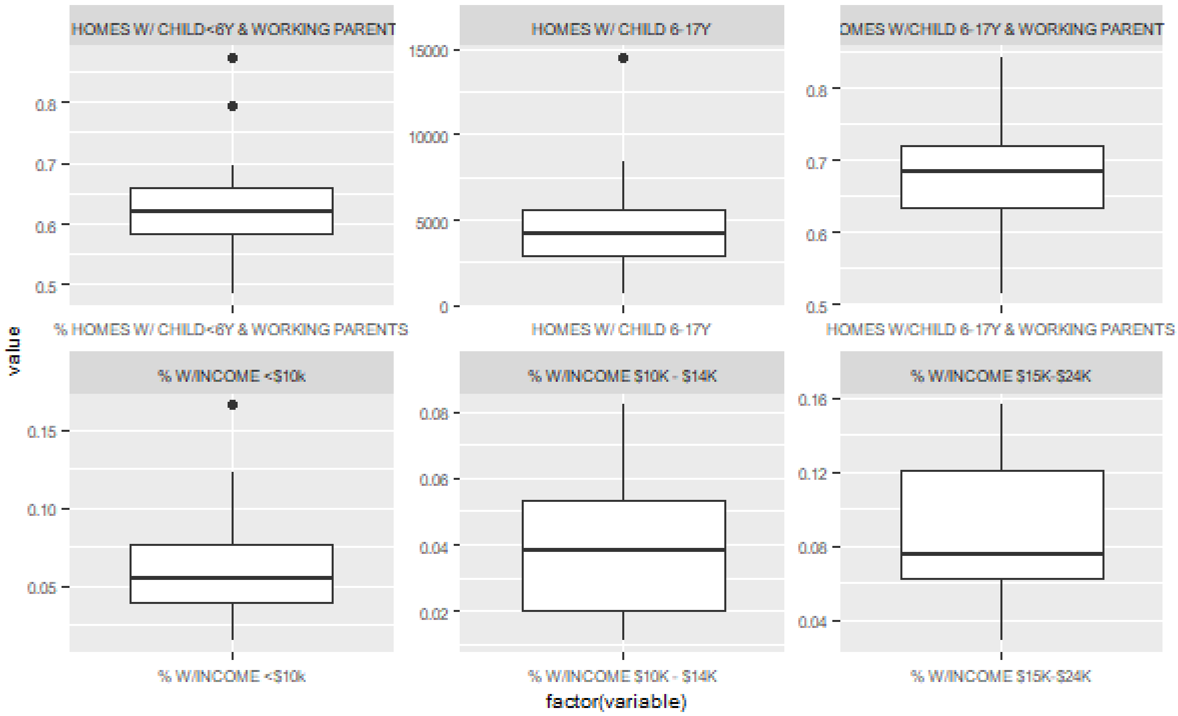
Hide

```
df <- subset(x_variables, select=c(43:48))
colnames(df) <- c("% HOMES W/ CHILD<6Y & WORKING PARENTS",
                  "HOMES W/ CHILD 6-17Y",
                  "HOMES W/CHILD 6-17Y & WORKING PARENTS",
                  "% W/INCOME <$10k",
                  "% W/INCOME $10K - $14K",
                  "% W/INCOME $15K-$24K")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=7.5))
```



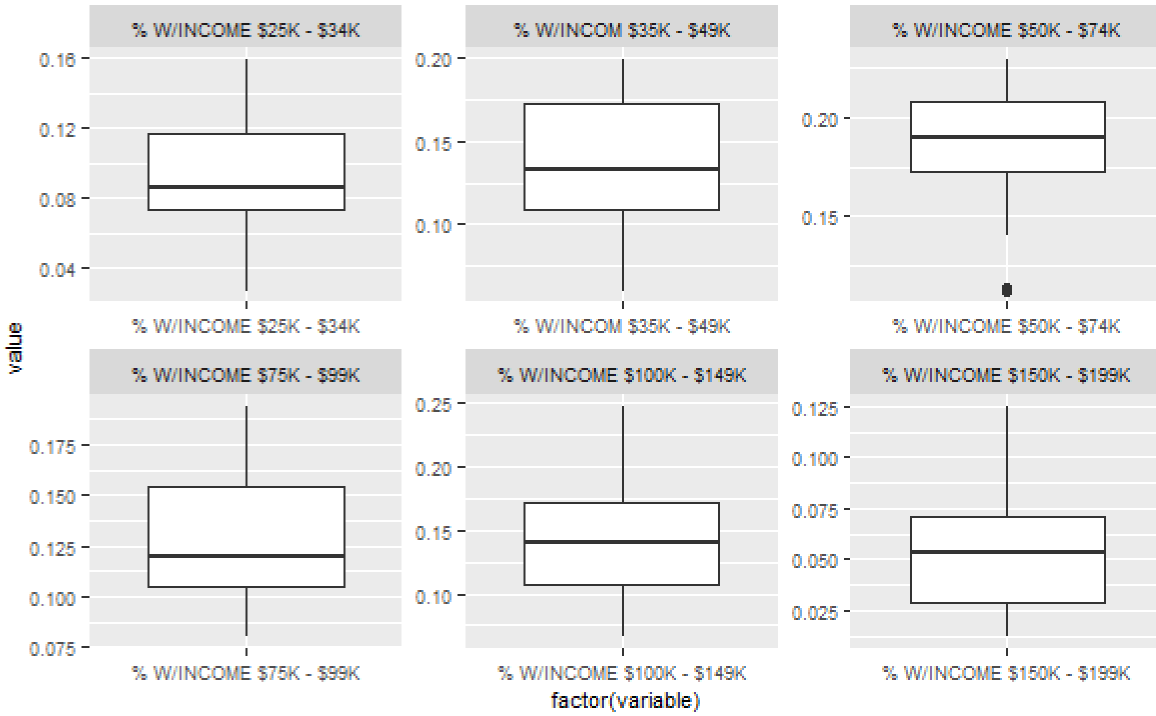
Hide

```
df <- subset(x_variables, select=c(49:54))
colnames(df) <- c("% W/INCOME $25K - $34K",
                  "% W/INCOM $35K - $49K",
                  "% W/INCOME $50K - $74K",
                  "% W/INCOME $75K - $99K",
                  "% W/INCOME $100K - $149K",
                  "% W/INCOME $150K - $199K")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



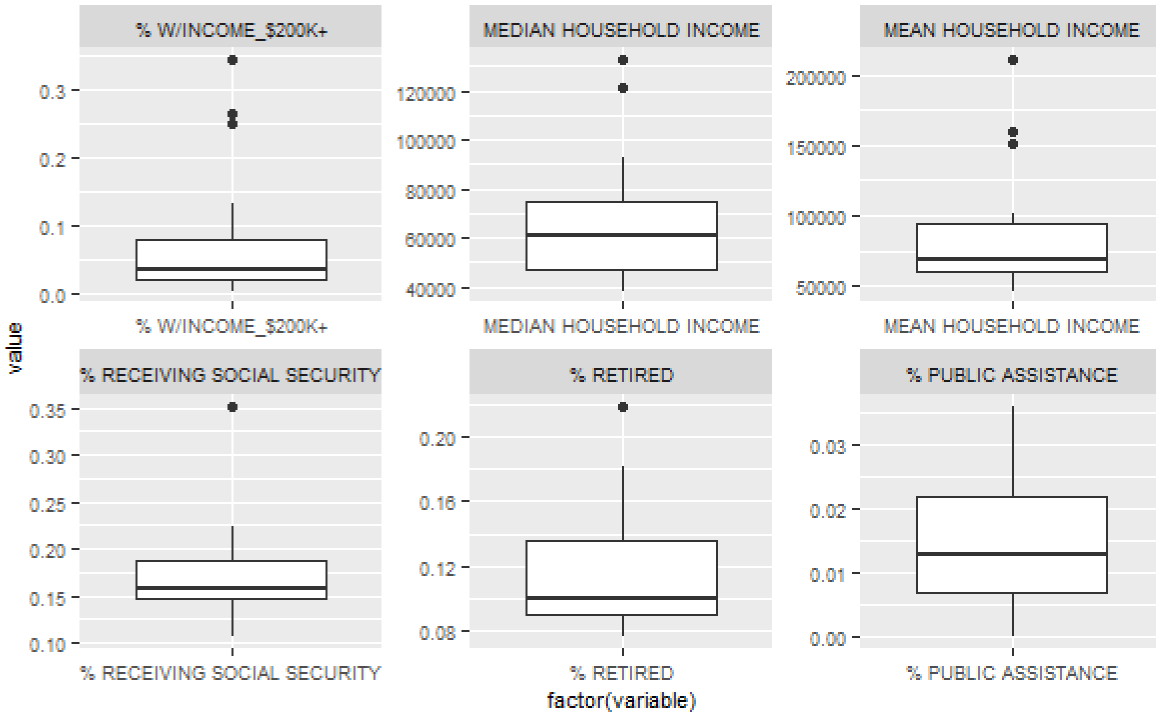
Hide

```
df <- subset(x_variables, select=c(55:60))
colnames(df) <- c("% W/INCOME_$200K+",
                  "MEDIAN HOUSEHOLD INCOME",
                  "MEAN HOUSEHOLD INCOME",
                  "% RECEIVING SOCIAL SECURITY",
                  "% RETIRED",
                  "% PUBLIC ASSISTANCE")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



Hide

```
df <- subset(x_variables, select=c(61:66))
colnames(df) <- c("MEAN INCOME (IF PUBLIC ASSISTANCE)",
                  "% RECEIVING FOOD STAMPS",
                  "% W/O HEALTH INSURANCE",
                  "% CHILDREN W/O HEALTH INSURANCE",
                  "% HOMES W/CHILD BELOW POVERTY",
                  "% POP. MALE")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```

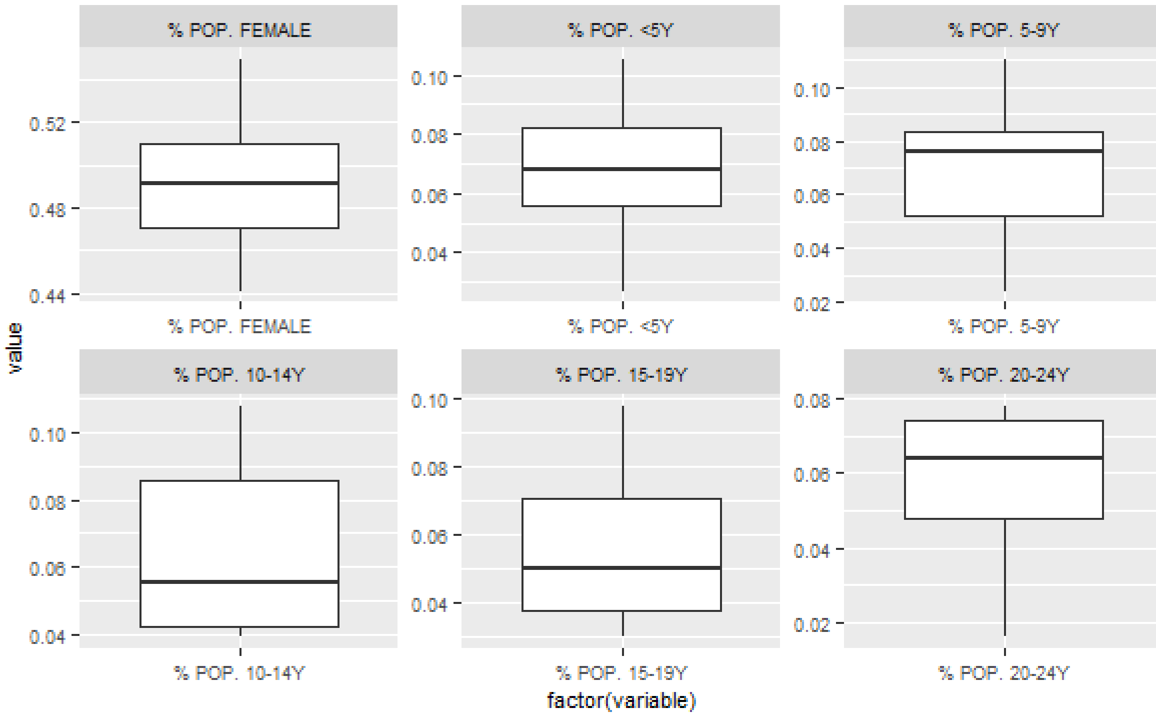
Hide

```
df <- subset(x_variables, select=c(67:72))
colnames(df) <- c("% POP. FEMALE",
                  "% POP. <5Y",
                  "% POP. 5-9Y",
                  "% POP. 10-14Y",
                  "% POP. 15-19Y",
                  "% POP. 20-24Y")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



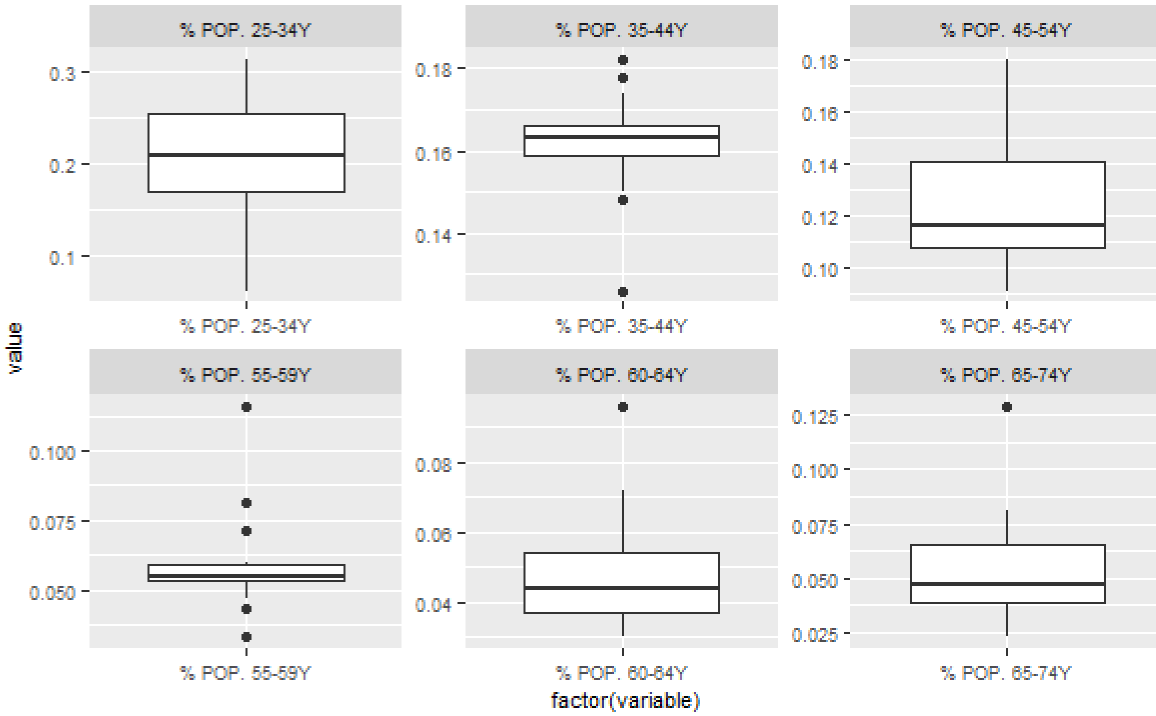
Hide

```
df <- subset(x_variables, select=c(73:78))
colnames(df) <- c("% POP. 25-34Y",
                  "% POP. 35-44Y",
                  "% POP. 45-54Y",
                  "% POP. 55-59Y",
                  "% POP. 60-64Y",
                  "% POP. 65-74Y")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



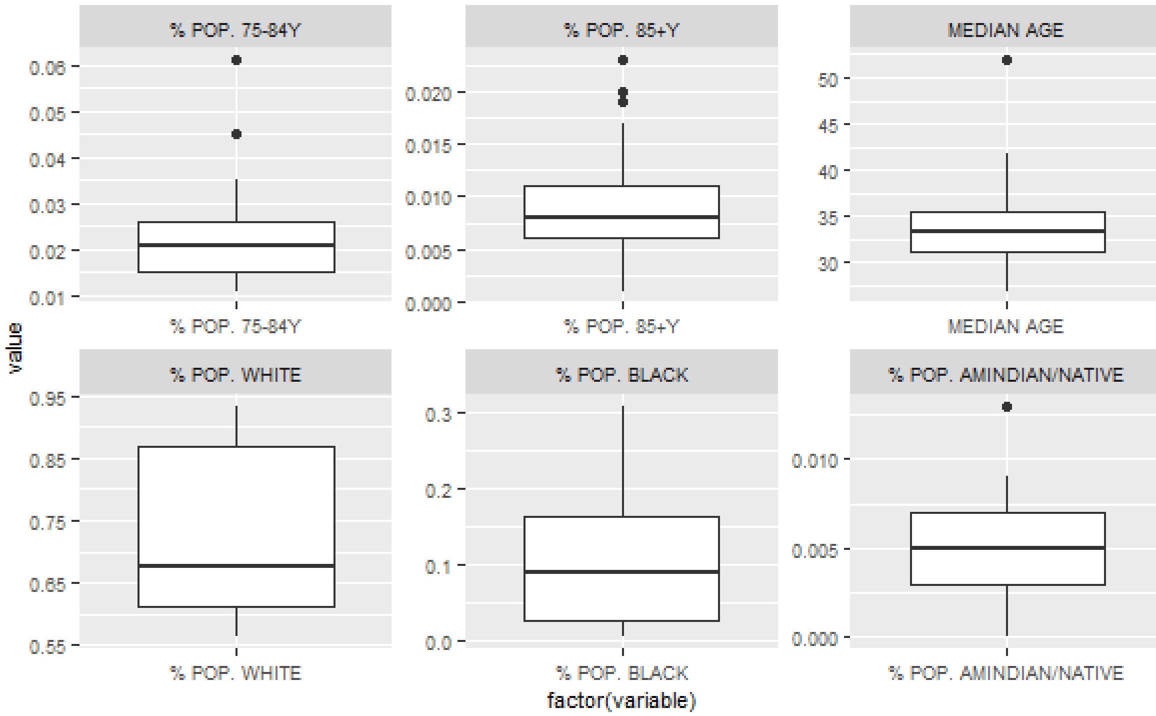
Hide

```
df <- subset(x_variables, select=c(79:84))
colnames(df) <- c("% POP. 75-84Y",
                  "% POP. 85+Y",
                  "MEDIAN AGE",
                  "% POP. WHITE",
                  "% POP. BLACK",
                  "% POP. AMINDIAN/NATIVE")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



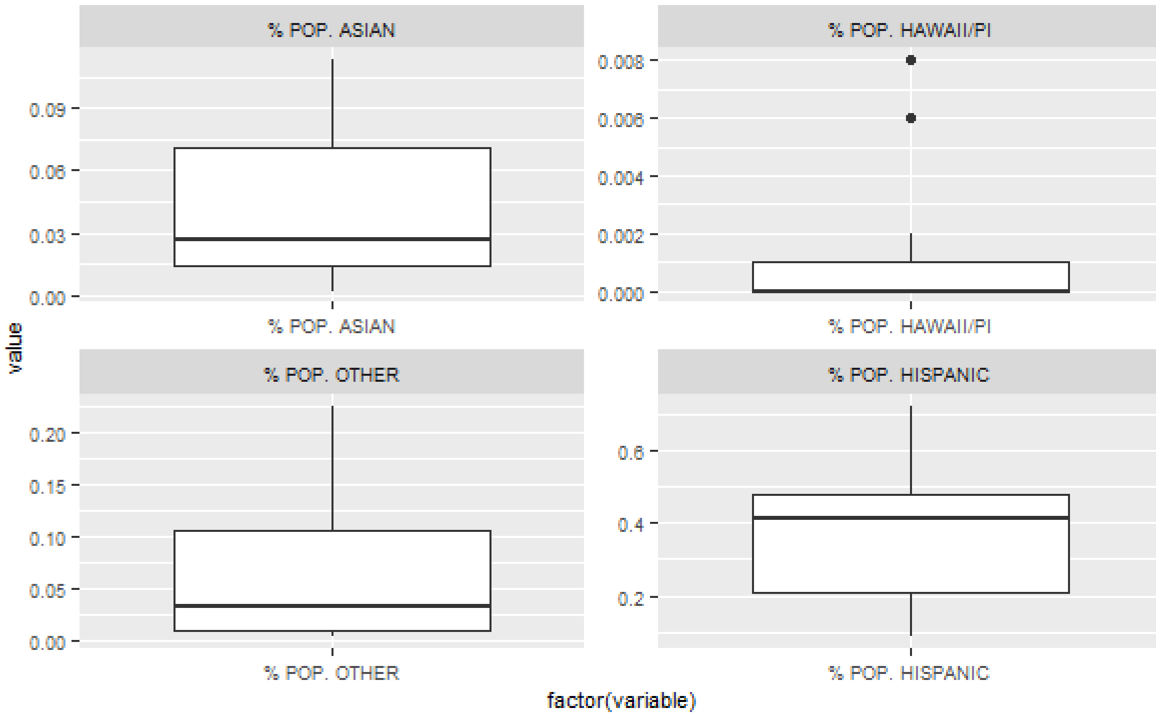
Hide

```
df <- subset(x_variables, select=c(85:88))
colnames(df) <- c("% POP. ASIAN",
                  "% POP. HAWAII/PI",
                  "% POP. OTHER",
                  "% POP. HISPANIC")
meltData <- melt(df)
```

```
Using  as id variables
```

Hide

```
p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+ theme(text=element_text(size=8))
```



Next, we create a correlation matrix in order to determine which predictors are heavily correlated.

From the correlation matrix, we find a few variables that are so strongly correlated, they basically provide the same information.

x8/x9 : -1.00 (% Urban, % Rural)
x13/x14: -1.00 (% Homes Owned, % Homes Rented)
x27/x28: -1.00 (% Homes w/Mortgage, % Homes w/o Mortgage)
x66/x67: -1.00 (% Pop. Male, % Pop. Female)

These inverse relationships are completely understandable as each variable pair is essentially the inverse of the other. The following variables are dropped as potential predictors: x8, x13, x27, x66.

x1/x3 : 0.999 (TOTAL_OP_EXPENDITURE, FTE_COUNT)
x1/x4 : 0.994 (TOTAL_OP_EXPENDITURE, TOTAL_SALARY_SPEND)
x3/x4 : 0.994 (FTE_COUNT, TOTAL_SALARY_SPEND)

We see that these three predictors track strongly to each other, which is a reasonable observance. If there isn't much variance in pay among employees, the total salary spend would effectively be the number of full-time employees (FTE) multiplied by the nominal salary. Similarly, if the operational expenditures budget is dominated by the amount spent on employee salaries, it would be understandable for these variables to also be strongly correlated. To simplify our model, the number of full-time employees variable will be kept while total_salary_soend and total_op_expenditures will be dropped.

x7/x38 : 0.995 (TOTAL_POP,POP_16_YEAR_AND_OVER)

This is a reasonable observation if fraction of the total population who are adults is similar/identical across Austin.

x41/x42: 0.989 (PERCENT_OF_LABOR_FEMALE_AND_16_AND_OVER, PERCENT_EMPLOYED_FEMALE_AND_16_AND_OVER)

Another reasonable observation. The number of adult women who are employed would reasonably track the number of adult women in the labor force. In this case, x41 will be dropped and x42 will be kept as 'employed' is a clearer descriptor than participating in the labor force, which has a number of caveats.

x55/x57: 0.978 (PERCENT_W_INCOME_200000_OR_MORE, MEAN_HOUSEHOLD_INCOME)

Austin is a relatively well-off city with a booming tech sector. These high income employees likely are skewing the mean household income.

x10/x38: 0.965 (TOTAL_HOUSING_AVAILABLE, POP_16_YEAR_AND_OVER)

The amount of housing available tracks with population. This seems an uncontroversial relationship.

x56/x57: 0.951 (MEDIAN_HOUSEHOLD_INCOME, MEAN_HOUSEHOLD_INCOME)

The relationship between median and mean is well-explained.

x7/x10 : 0.950 (TOTAL_POP, TOTAL_HOUSING_AVAILABLE)

This relationship is similar to the one between x10/x38.

x15/x71: 0.940 (AVERAGE_HOUSEHOLD_SIZE_OWNED, PERCENT_POP_15_TO_19)

A reasonable hypothesis is that the average household size of a homeowner is related to the number of teenage children who still live at home. Conversely, parents of young children (who are generally younger and earlier in their careers) may not be able to afford to own a home and still rent. This is an interesting observation as we are considering high school graduation rates, where students are generally aged 15-19 years of age. This possibly suggests that those students are generally coming from income-stable homes in Austin.

x24/x26: 0.944 (PERCENT_OF_HOMES_VALUED_500000_TO_999999, MEDIAN_HOME_VALUE)

As mentioned earlier, Austin does have a booming tech sector and thus has seen an influx of high-paid employees coming into the city. Able to afford nicer, more expensive homes, they likely are skewing the median home value upwards.

x78/x81: 0.947 (PERCENT_POP_65_TO_74, MEDIAN_POP_AGE)

Likely this indicates a significant elderly contingent in Austin, skewing the median population age upwards.

x63/x65: -0.926 (PERCENT_NO_HEALTH_INSURANCE, PERCENT_FAMILIES_W_CHILDREN_BELOW_POVERTY)

While it is unsurprising that households that are below the poverty line are also unable to afford health insurance for their adult members, it is notable that these variables do not also track with the percentage of uninsured children. Possibly children living in poverty are successful in being caught by state-wide safety nets?

x26/x55: 0.915 (MEDIAN_HOME_VALUE, PERCENT_W_INCOME_200000_OR_MORE)

People who earn more buy more expensive houses.

x12/x18: 0.907 (MOBILE_HOMES_PERCENTAGE_OF_HOUSING, PERCENT_OF_HOMES_VALUED_LESS_THAN_50000)

Homes in Austin are very expensive due to demand outstripping supply. It would appear that very cheap homes are largely of the mobile home variety.

x18/x64: 0.903 (PERCENT_OF_HOMES_VALUED_LESS_THAN_50000, PERCENT_CHILDREN_NO_HEALTH_INSURANCE)

The best hypothesis I have for this relationship is that within the working poor demographic, there is a population who makes too much money for social safety nets (and thus can afford the lowest tier of home ownership) but insufficient income to afford health insurance without assistance.

x30/x36: -0.900 (PERCENTAGE_OF_RENTERS_PAYING_500_TO_999, MEDIAN_RENT)

The percentage of renters paying 500 to 999 dollars a month are numerous enough to skew the median rent value.

x55/x56: 0.905 (PERCENT_W_INCOME_200000_OR_MORE, MEDIAN_HOUSEHOLD_INCOME)

The richest Austinites are numerous enough to skew the median household income.

x64/x88: 0.909 (PERCENT_CHILDREN_NO_HEALTH_INSURANCE, PERCENT_POP_HISPANIC)

Any attempt to explain this relationship is pure conjecture and, more importantly, just makes me sad to think about.

x76/x81: 0.910 (PERCENT_POP_55_TO_59, MEDIAN_POP_AGE)

This age bracket consists of the oldest Gen X-ers and the youngest of the Baby Boomers. Reasonbly, this would be senior managers, etc. within the working population. Apparently, they are numerous enough to skew the overall median age in Austin.

In considering what variables to drop due to collinearity, aggregates of multiple variables were favored over variables describing a sub-category (e.g., median age vs. percentage aged 65-74). In the end, the following variables were dropped as predictors: x1, x4, x10, x15, x18, x24, x30,x38, x41, x55, x63, x76, x78.

Unfortunately, even dropping these highly correlated variables does not allow us to compute VIF within R. As such, we turn to the 'alias' function to find which variables are considered to be linearly dependent. These will be removed and the model re-run.

Hide

```
# Remove highly correlated variables from the predictor list
x_reduced = subset(x_variables, select=-c(x8, x13, x27, x66, x1, x4, x10, x15,
                                          x18, x24, x30,x38, x41, x55, x63, x76,
                                          x78))
# Bind a column with the target variable "overall grad rate"
df <- cbind(x_reduced, y_variables['y1'])
# Create a linear model object
model <- lm(y1~., data=df)

ld.vars <- attributes(alias(model)$Complete)$dimnames[[1]]
ld.vars
```

```
 [1] "x33" "x34" "x35" "x36" "x37" "x39" "x40" "x42" "x43" "x44" "x45"
[12] "x46" "x47" "x48" "x49" "x50" "x51" "x52" "x53" "x54" "x56" "x57"
[23] "x58" "x59" "x60" "x61" "x62" "x64" "x65" "x67" "x68" "x69" "x70"
[34] "x71" "x72" "x73" "x74" "x75" "x77" "x79" "x80" "x81" "x82" "x83"
[45] "x84" "x85" "x86" "x87" "x88" "x89" "x90" "x91"
```

Hide

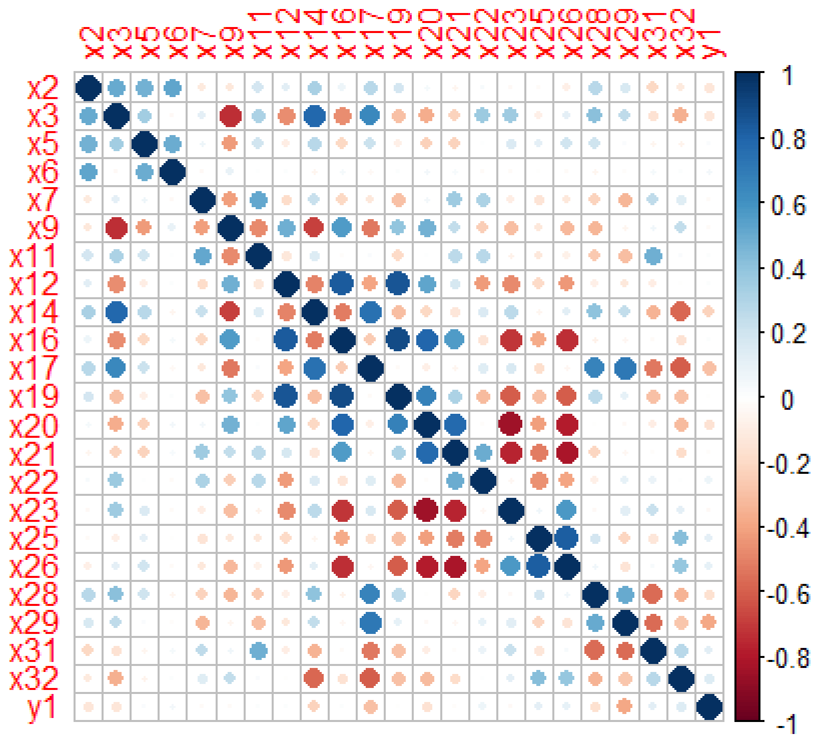
```
# Remove additional linearly dependent variables.
df <- subset(df, select=-c(x33,x34,x35,x39,x42,x43,x44,x46,x47,x48,x49,x50,x51,
                           x52,x53,x54,x58,x61,x68,x69,x70,x72,x73,x74,x75,x77,
                           x79,x80,x84,x86,x87,x36,x37,x40,x45,x56,x57,x59,x60,
                           x62,x64,x65,x67,x71,x81,x82,x83,x85,x88))
```

Hide

```
# Remove categorical variables which are dependent on each other
df <- subset(df, select=-c(x89,x90,x91))
```

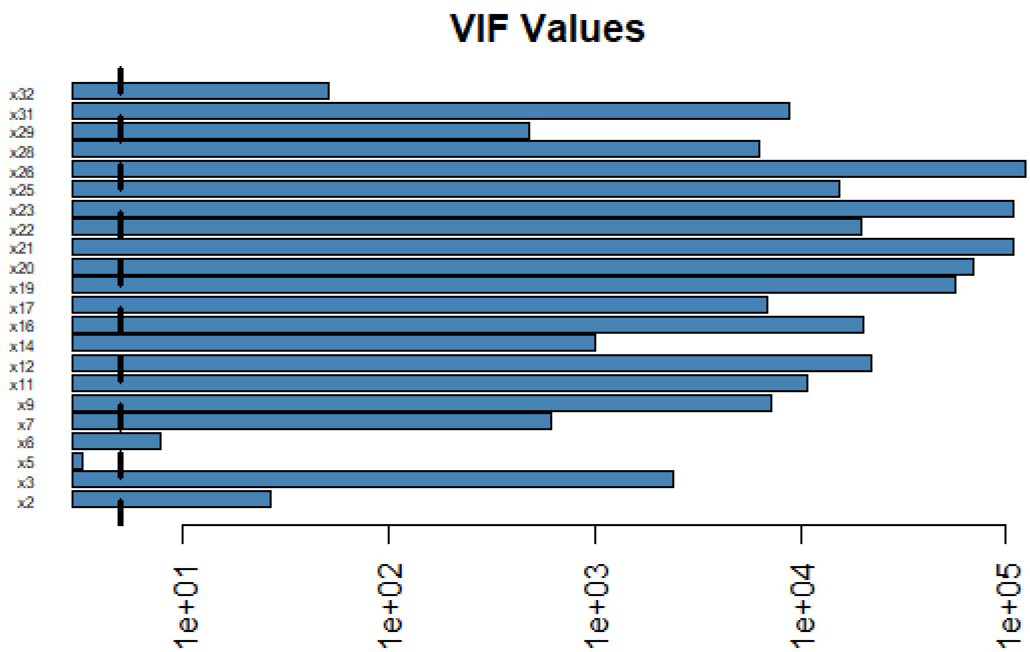
Hide

```
model <- lm(y1~., data=df)

# Create a correlation matrix
corrplot(cor(df))
```



Hide

```
# Create vector of VIF values
vif_values <- vif(model)

# Create horizontal bar chart to display each VIF value
barplot(vif_values, main = "VIF Values", horiz = TRUE, col = "steelblue",las=2, cex.names=.5, lo
g="x")

# Add vertical line at 5
abline(v = 5, lwd = 3, lty = 2)
```



As can be seen in the plot of VIF values, there is massive multicollinearity in play. This will have to be dealt with via principal component analysis.

Principal Component Analysis (PCA)

Hide

```
#calculate principal components
x_reduced <- subset(df, select=-c(y1))
# Variables are scalled to a mean of zero and a SD of 1
results <- prcomp(x_reduced, scale = TRUE)

# Eigenvectors in R point in the negative direction by default, so we'll
# multiply by -1 to reverse the signs.
results$rotation <- -1*results$rotation

# Display principal components
results$rotation
```

```
             PC1         PC2          PC3          PC4          PC5
x2  -0.050063586  0.21735467 -0.042475186  0.492848367 -0.170013393
x3  -0.271809067  0.26165221  0.114770275  0.080202180  0.013232635
x5  -0.148031494  0.09902085  0.003950352  0.490964438 -0.075945711
x6   0.007538492  0.03209771 -0.005133091  0.488037824 -0.392281064
x7  -0.039423957 -0.02228767  0.411120083  0.039323882  0.287356315
x9   0.293430091 -0.14775166 -0.173379508 -0.075692351 -0.305021252
x11 -0.048077176  0.01424854  0.405170152  0.232354000  0.218185127
x12  0.308132340  0.01040934 -0.163899174  0.202270440  0.123087668
x14 -0.257557555  0.28892247  0.073714670 -0.019190670  0.157638133
x16  0.371096882  0.10277273 -0.044118874  0.106530493  0.038540975
x17 -0.186511011  0.38466751 -0.072041708 -0.117439036 -0.002575341
x19  0.300833199  0.19016589 -0.193734367  0.124627646  0.137564891
x20  0.333023695  0.18174180  0.069993718  0.012285738  0.130111432
x21  0.258104104  0.16522651  0.311122597 -0.083894214  0.017970420
x22 -0.042022504  0.12632156  0.374174191 -0.163096489 -0.296473513
x23 -0.295455976 -0.12936662 -0.059501128 -0.072875934 -0.317203983
x25 -0.148378126 -0.22132892 -0.246073805  0.144432115  0.381519059
x26 -0.286772518 -0.25608419 -0.197784047  0.074823994  0.168526934
x28 -0.108480480  0.29252891 -0.263107131 -0.002626547  0.200590617
x29 -0.066146192  0.28223062 -0.207732243 -0.212292667 -0.295780719
x31  0.020271066 -0.26951993  0.295675885  0.103740818 -0.124236482
x32 -0.001925007 -0.35664240  0.007493792  0.082617431 -0.068940676
             PC6          PC7          PC8         PC9        PC10
x2   0.040937934 -0.212981103  0.384509765 -0.32551116  0.18782724
x3  -0.146597545 -0.233578874  0.253597868 -0.20726421 -0.13633488
x5   0.058422538  0.067326863 -0.448532070  0.20781697 -0.31266859
x6   0.241102035  0.377515456 -0.050344060  0.09609942  0.09850349
x7   0.144101408  0.081538273 -0.463772978 -0.31004835  0.30767032
x9   0.078330332  0.111504935  0.235170010 -0.11336364  0.12225878
x11 -0.168500939 -0.257536109  0.168888638  0.37724985  0.33325071
x12 -0.266563693 -0.205370521 -0.229281108 -0.25115435  0.07802589
x14 -0.078193625  0.290334173  0.127153092 -0.31032036  0.08497880
x16 -0.124497845 -0.147224146  0.023450052  0.12739931 -0.03599356
x17 -0.007315599  0.033423502  0.009476478  0.27823062  0.23463323
x19 -0.184730701 -0.165244403 -0.124350131 -0.02459077 -0.09955962
x20  0.095874367  0.214859956  0.148175339  0.05913889  0.08725857
x21  0.305814055  0.082339604  0.061802445  0.08067163  0.03441623
x22  0.282071467 -0.308774683  0.035639058 -0.06883394 -0.43760145
x23 -0.369152625 -0.022514198 -0.163940018 -0.11689671  0.11027249
x25  0.339676335 -0.009415741  0.244568094  0.13031367 -0.02409649
x26  0.072488654  0.039952216  0.085304606  0.08266061  0.06750066
x28  0.128047803 -0.192165014 -0.081720320  0.17697373 -0.25036891
x29  0.131728679 -0.232396695 -0.200846505  0.25957073  0.46440331
x31 -0.373863712  0.008315502  0.149865586  0.36027655 -0.07552643
x32  0.347041565 -0.499570581 -0.099403940 -0.11410055  0.19179469
           PC11          PC12         PC13         PC14         PC15
x2  -0.06867979  0.1314550570 -0.083978857  0.248481513 -0.067885830
x3  -0.03294444  0.0117538704 -0.182703262 -0.492530224  0.227054748
x5   0.30233658  0.5211290407  0.001699694  0.019985898  0.054790704
x6  -0.20713452 -0.5084992867  0.049635135 -0.152335781 -0.005095607
x7  -0.32826691  0.0307338056 -0.173665358  0.090560067  0.238066169
```

```
x9  -0.08465850  0.4008985243  0.096428038  0.252121876  0.242294259
x11  0.17443059 -0.0974796252  0.195507529  0.279987320 -0.182106218
x12  0.13117097 -0.1231997006 -0.196724074  0.082268848 -0.194657160
x14  0.05059309  0.1629656278  0.102172886  0.025083331  0.077351545
x16  0.06215767 -0.0939486408  0.029614067  0.006683232  0.463522228
x17  0.01357152  0.0367181919  0.386520799 -0.158356548  0.178376024
x19 -0.07689288 -0.1443958303  0.180701407 -0.107231107  0.195017468
x20 -0.16984599  0.3205739501 -0.048172375 -0.217169827 -0.458564407
x21  0.11576490  0.0205790509  0.100479202  0.141207118  0.257012225
x22  0.06335698 -0.1464283871 -0.093406275  0.214929690 -0.080106629
x23 -0.11977704 -0.0001777069  0.222576717  0.306767126  0.116212508
x25  0.04350708 -0.0419082476 -0.263895314  0.144679159  0.343254403
x26  0.09537720 -0.0484096703 -0.037440020  0.012863294 -0.105068602
x28 -0.64339035  0.0286223310  0.107389929  0.295717192 -0.131280573
x29  0.06389238  0.0650455304 -0.514970825 -0.026901481  0.008428159
x31 -0.42511262  0.2028412851 -0.279799694 -0.130837043  0.143178536
x32 -0.12022480  0.1832491452  0.388087174 -0.379601485 -0.027566465
            PC16          PC17          PC18          PC19          PC20
x2  -0.242929405 -0.387818165  0.160650730  0.054073983  0.079774614
x3   0.524596414  0.049536698 -0.150588815  0.050872350  0.008503789
x5   0.064076831  0.005167273 -0.016680603 -0.023872536 -0.001503574
x6   0.084110763  0.206947486 -0.073897327 -0.015947685 -0.042027123
x7   0.089733093 -0.165256375  0.066058594 -0.023524967 -0.249582781
x9   0.300096534  0.119719745 -0.310865422 -0.011883588 -0.381919188
x11  0.231703312  0.158202381 -0.154173527 -0.194784723 -0.073678483
x12  0.002928159  0.345429373 -0.086713408  0.481976339 -0.122887570
x14 -0.417441911  0.562013488 -0.193300932 -0.135072074  0.057391557
x16 -0.026600755  0.193406366  0.363668760  0.089956224  0.009383299
x17 -0.131267965 -0.112458051  0.220181989  0.368687283 -0.409885226
x19 -0.182146229 -0.186909258 -0.266669087 -0.569696009 -0.154091366
x20  0.185187280  0.136062723  0.368399552 -0.176470230 -0.062082238
x21  0.032827948 -0.085517707 -0.277339025  0.268718789  0.478343465
x22 -0.135952579  0.186863606  0.108590008 -0.073299537 -0.403447511
x23  0.232818974  0.137170906  0.314267157 -0.135306793  0.209745813
x25  0.031027912  0.219862143  0.278978465 -0.138535139 -0.011688491
x26 -0.034787780 -0.128261768 -0.278989014  0.150086308 -0.296756338
x28  0.116288250  0.111696522 -0.157288684  0.143318413  0.130065866
x29 -0.071123531  0.098348774 -0.111210390 -0.176023321  0.085521139
x31 -0.353345270  0.051136422 -0.091148569  0.124926725 -0.021819824
x32 -0.154134739  0.219854601 -0.001621883  0.005018674  0.125066578
            PC21          PC22
x2   0.024578205  0.0012112677
x3   0.005752465 -0.0648170863
x5  -0.001911451  0.0012752896
x6  -0.012048622 -0.0001652251
x7   0.065859903  0.0295164182
x9  -0.050030311  0.1121447807
x11 -0.019151591  0.1385205559
x12 -0.245332935 -0.1920043653
x14  0.114166095  0.0348753805
x16  0.594568697  0.1454937189
x17 -0.254948194 -0.0970153108
```

```
x19 -0.134824613 -0.3151034120
x20  0.097650564 -0.3481735956
x21 -0.043799925 -0.4381160318
x22  0.014942781 -0.1871186318
x23 -0.003551659 -0.4375180470
x25 -0.358957071 -0.1477422506
x26  0.562468571 -0.4550197277
x28  0.095800043  0.1040377788
x29  0.072008468 -0.0251229734
x31 -0.103355850 -0.1233204466
x32  0.011831209 -0.0082385026
```

Looking at these scores, it seems the first principal component (PC1) has relatively high scores for x9 (*% Rural*), x12 (*Mobile homes as % of total housing*), x16 (*Avg. household size for renters*), x19 (*% of homes valued between $50,000 to $99,999*), x20 (*% of homes valued between $100,000 and $149,999*), and x21 (*% of homes valued between $150,000 and $199,999*). This should mean that PC1 describes the most variation in these variables.

PC2 has the highest score for x17 (*% of homes with no vehicle*), which indicates this principal component puts most of its emphasis on that variable.

Hide

```
summary(results)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6
Standard deviation     2.4963 2.0795 1.8001 1.43683 1.19016 1.09053
Proportion of Variance 0.2832 0.1966 0.1473 0.09384 0.06439 0.05406
Cumulative Proportion  0.2832 0.4798 0.6271 0.72093 0.78532 0.83938
                          PC7     PC8    PC9    PC10    PC11    PC12
Standard deviation     0.9096 0.80121 0.7534 0.70321 0.62266 0.50607
Proportion of Variance 0.0376 0.02918 0.0258 0.02248 0.01762 0.01164
Cumulative Proportion  0.8770 0.90616 0.9320 0.95444 0.97206 0.98370
                          PC13    PC14    PC15    PC16    PC17    PC18
Standard deviation     0.34571 0.29616 0.22639 0.21775 0.16036 0.11707
Proportion of Variance 0.00543 0.00399 0.00233 0.00216 0.00117 0.00062
Cumulative Proportion  0.98913 0.99312 0.99545 0.99760 0.99877 0.99940
                          PC19    PC20     PC21     PC22
Standard deviation     0.09273 0.06807 0.006627 0.001329
Proportion of Variance 0.00039 0.00021 0.000000 0.000000
Cumulative Proportion  0.99979 1.00000 1.000000 1.000000
```

Hide

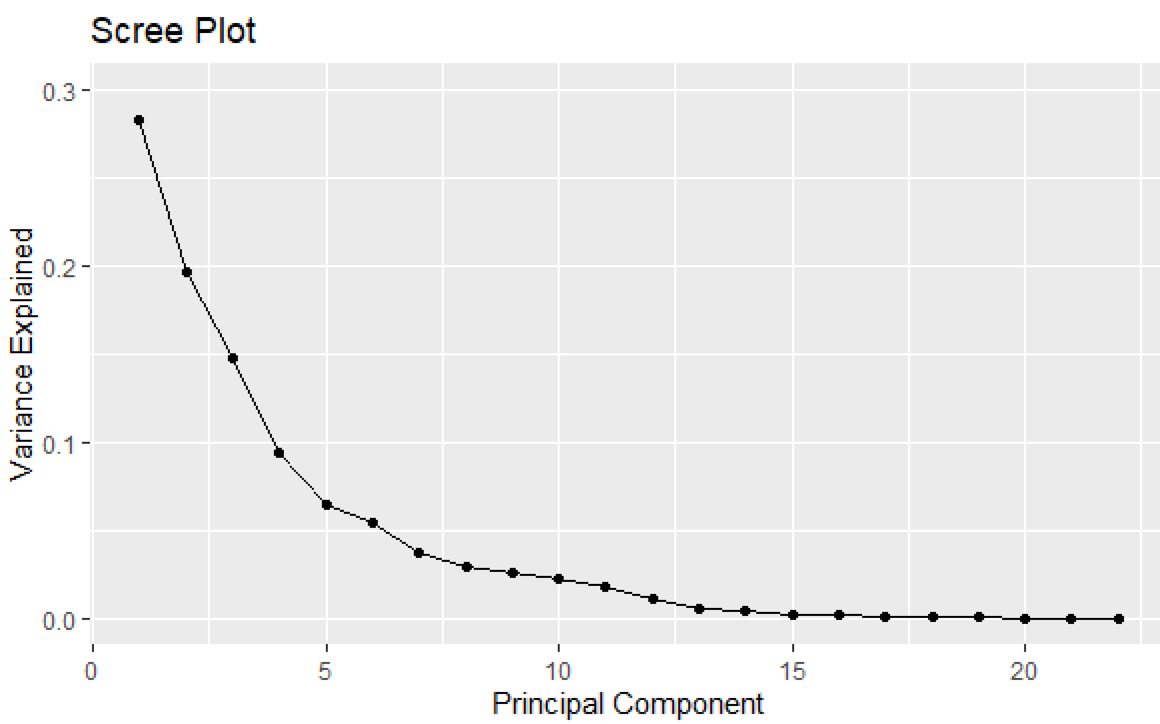
```
# Reverse the signs of the scores
results$x <- -1*results$x

# Calculate total variance explained by each principal component
var_explained = results$sdev^2 / sum(results$sdev^2)

var_explained
```

```
 [1] 2.832446e-01 1.965595e-01 1.472883e-01 9.383955e-02 6.438589e-02
 [6] 5.405726e-02 3.760403e-02 2.917869e-02 2.580116e-02 2.247717e-02
[11] 1.762310e-02 1.164126e-02 5.432444e-03 3.986720e-03 2.329646e-03
[16] 2.155326e-03 1.168853e-03 6.229268e-04 3.908377e-04 2.105996e-04
[21] 1.996426e-06 8.032749e-08
```

The first principal component explains 28.3% of the total variance in the dataset, the second principal component explains 19.7% of the total variance in the dataset, the third principal component explains 14.7% of the total variance in the dataset, the fourth principal component explains 9.4% of the total variance in the dataset, the fifth component explains 5.4% of the total variance in the dataset, etc.

Hide

```
#create scree plot
qplot(c(1:22), var_explained) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, .3)
```



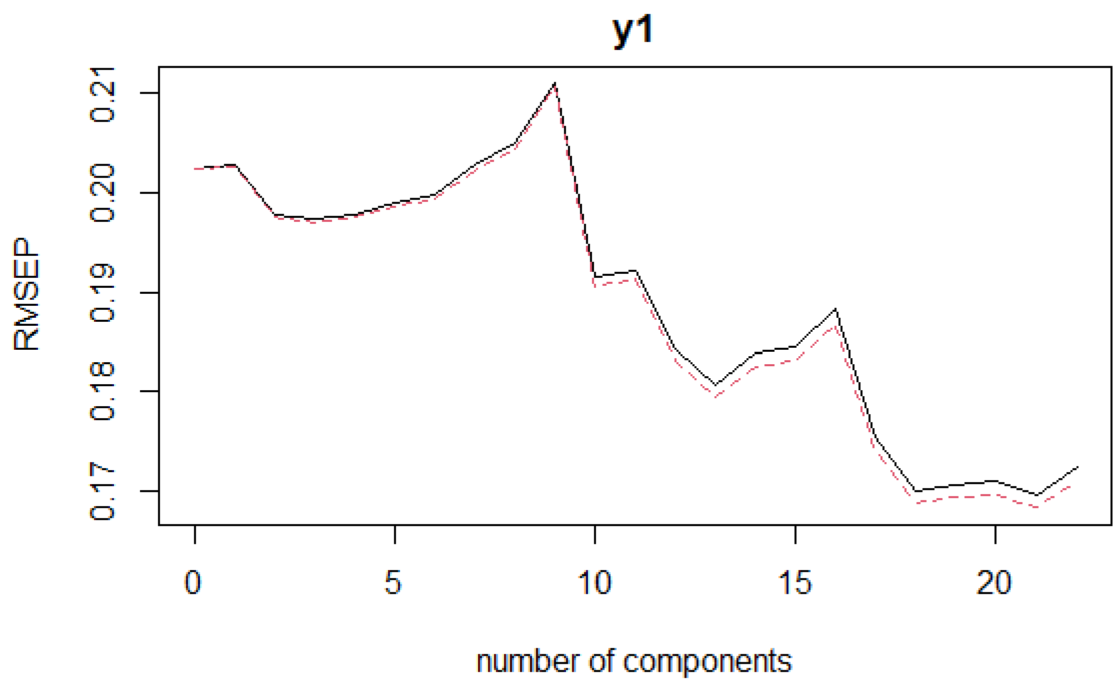
Hide

```
set.seed(7)


model_pca <- pls::pcr(y1~., data=df, scale=TRUE, validation="CV")
summary(model_pca)
```

```
Data: 	X dimension: 105 22 
	Y dimension: 105 1
Fit method: svdpc
Number of components considered: 22

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
CV          0.2023   0.2028   0.1978   0.1973   0.1978   0.1989
adjCV       0.2023   0.2027   0.1975   0.1970   0.1975   0.1985
       6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
CV      0.1998   0.2028   0.2050   0.2110    0.1916    0.1921
adjCV   0.1993   0.2023   0.2045   0.2107    0.1905    0.1913
       12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
CV       0.1842    0.1806    0.1838    0.1845    0.1883    0.1754
adjCV    0.1831    0.1794    0.1825    0.1830    0.1868    0.1740
       18 comps  19 comps  20 comps  21 comps  22 comps
CV       0.1700    0.1706    0.1709    0.1696    0.1724
adjCV    0.1687    0.1693    0.1696    0.1683    0.1710

TRAINING: % variance explained
    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X   28.3245    47.98   62.709   72.093   78.532   83.938   87.698
y1   0.1739     7.17    7.604    8.477    8.657    8.774    8.794
    8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
X    90.616   93.196     95.44     97.21     98.37     98.91     99.31
y1    8.811    8.811     23.01     24.41     32.90     35.97     36.22
    15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
X      99.54     99.76     99.88     99.94     99.98    100.00
y1     36.62     36.74     43.04     45.71     45.73     45.82
    21 comps  22 comps
X     100.00    100.00
y1     46.57     46.57
```

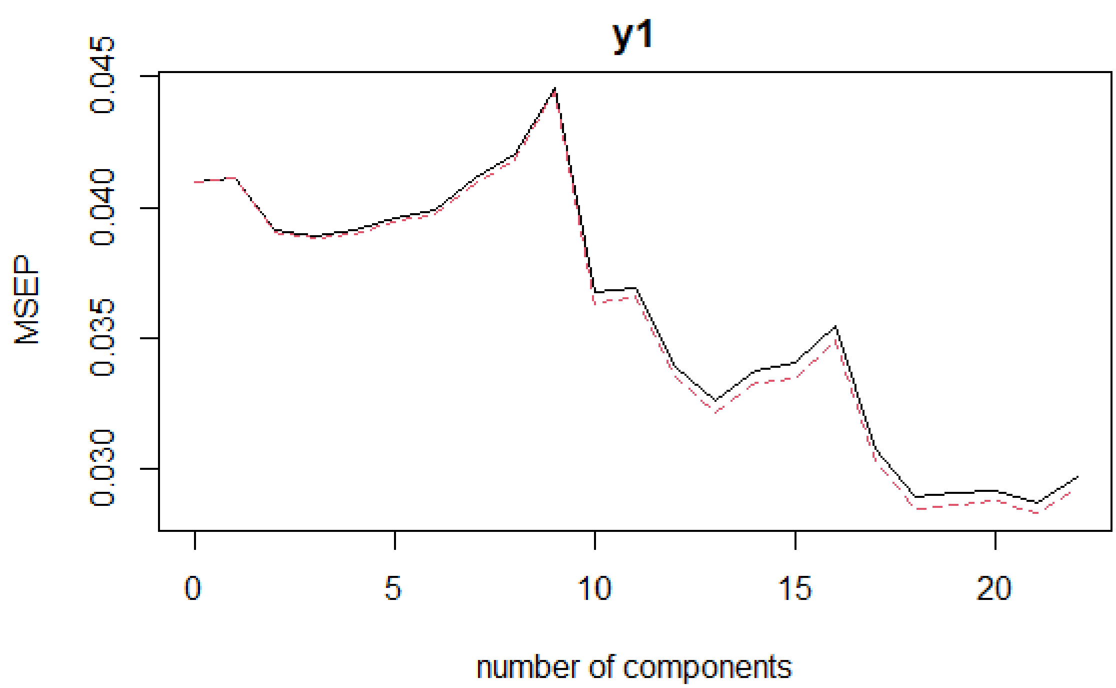
Hide

```
validationplot(model_pca)
```



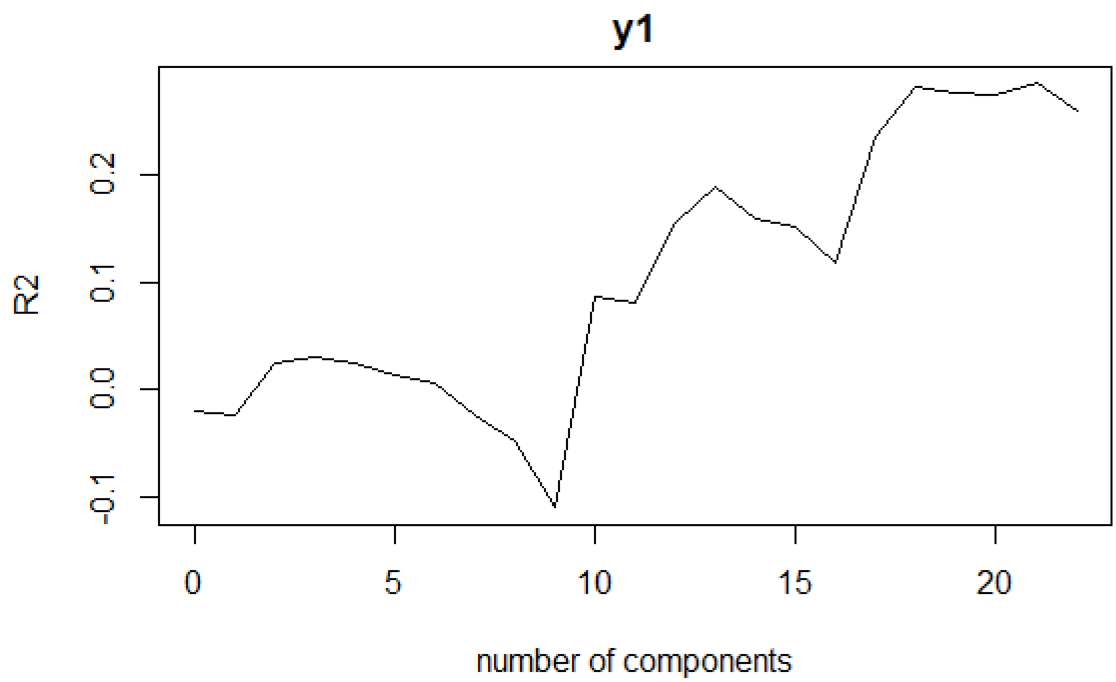
Hide

```
validationplot(model_pca, val.type="MSEP")
```



Hide

```
pls::validationplot(model_pca, val.type="R2")
```



Hide

```
#Use 70% of dataset as training set and remaining 30% as testing set
set.seed(7)

split1<- sample(c(rep(0, 0.7 * nrow(df)), rep(1, 0.3 * nrow(df))))
train <- df[split1 == 0, ]
test <- df[split1== 1, ]
y_test <- subset(test, select=c(y1))
test <- subset(test, select=-c(y1))
```

Hide

```
# Use model to make predictions on a test set
# 2 Principal Component Case
# By setting the parameter scale equal to TRUE the data is standardized before
# running the pcr algorithm on it. You can also perform validation by setting
# the argument validation. In this case I chose to perform 10 fold
# cross-validation and therefore set the validation argument to “CV”.

model <- pcr(y1~., data=train, scale=TRUE, validation="CV")
pcr_pred <- predict(model, test, ncomp=2)

#calculate RMSE
RMSE <- cbind(y_test, round(pcr_pred,4))
RMSE['SqDiff'] <- (-RMSE['y1'] + RMSE['y1.2 comps'])^2
Model_RMSE <- sqrt(mean(RMSE$SqDiff))
Model_RMSE
```

```
[1] 0.1882663
```

Hide

```
# Calculate adjusted R2
R2 <- 0.0717
adj_r2 <- 1-(1-R2)*(74-1)/(74-2-1)
adj_r2
```

```
[1] 0.0455507
```

Hide

```
# Note that it is possible to get a negative R-square for equations that do # not contain a cons
tant term. Because R-square is defined as the proportion # of variance explained by the fit, if
 the fit is actually worse than just
# fitting a horizontal line then R-square is negative. In this case,
# R-square cannot be interpreted as the square of a correlation. Such
# situations indicate that a constant term should be added to the model.
```

Hide

```
# Use model to make predictions on a test set
# 12 Principal Component Case
model <- pcr(y1~., data=train, scale=TRUE, validation="CV")
pcr_pred <- predict(model, test, ncomp=12)

#calculate RMSE
RMSE <- cbind(y_test, round(pcr_pred,4))
RMSE['SqDiff'] <- (-RMSE['y1'] + RMSE['y1.12 comps'])^2
Model_RMSE <- sqrt(mean(RMSE$SqDiff))
Model_RMSE
```

```
[1] 0.2036243
```

Hide

```
# Calculate adjusted R2
R2 <- 0.329
adj_r2 <- 1-(1-R2)*(74-1)/(74-12-1)
adj_r2
```

```
[1] 0.197
```

Hide

```
# Use model to make predictions on a test set
# 16 Principal Component Case
model <- pcr(y1~., data=train, scale=TRUE, validation="CV")
pcr_pred <- predict(model, test, ncomp=16)

#calculate RMSE
RMSE <- cbind(y_test, round(pcr_pred,4))
RMSE['SqDiff'] <- (-RMSE['y1'] + RMSE['y1.16 comps'])^2
Model_RMSE <- sqrt(mean(RMSE$SqDiff))
Model_RMSE
```

```
[1] 0.2051205
```

Hide

```
# Calculate adjusted R2
R2 <- 0.3674
adj_r2 <- 1-((1-R2)*(74-1)/(74-16-1))
adj_r2
```

```
[1] 0.1898281
```

Random Forest

Hide

```
library(caret)
```

```
Warning: package ‘caret’ was built under R version 4.1.3
Loading required package: lattice

Attaching package: ‘caret’

The following object is masked from ‘package:pls’:

    R2

The following object is masked from ‘package:purrr’:

    lift
```

Hide

```
# Create the forest

# By using the 'rf' method in caret, we also incorporate a lasso model
# for variable selection. Though not absolutely necessary, cross-validation
# is used here with 10-folds.

set.seed(7)
ctrl <- trainControl(
  method = "cv",
  number = 10,
)

# Train the random forest on the reduced variable dataset
rf <- train(
   y1 ~ .,
   data = train,
   method = 'rf',
   preProcess = c("center", "scale"),
   trControl = ctrl
   )
summary(rf)
```

```
                Length Class      Mode
call              4    -none-     call
type              1    -none-     character
predicted        74    -none-     numeric
mse             500    -none-     numeric
rsq             500    -none-     numeric
oob.times        74    -none-     numeric
importance       22    -none-     numeric
importanceSD      0    -none-     NULL
localImportance   0    -none-     NULL
proximity         0    -none-     NULL
ntree             1    -none-     numeric
mtry              1    -none-     numeric
forest           11    -none-     list
coefs             0    -none-     NULL
y                74    -none-     numeric
test              0    -none-     NULL
inbag             0    -none-     NULL
xNames           22    -none-     character
problemType       1    -none-     character
tuneValue         1    data.frame list
obsLevels         1    -none-     logical
param             0    -none-     list
```

Hide

```
rf_pred <- predict(rf, newdata = test)

# RMSE
RMSE <- cbind(y_test, round(rf_pred,4))
RMSE['SqDiff'] <- (-RMSE['y1'] + RMSE[,c(2)])^2
Model_RMSE <- sqrt(mean(RMSE$SqDiff))
Model_RMSE
```

```
[1] 0.1952697
```

Hide

```
R2 <- 0.4681217
n <- 74
p <- 1
adjR2 <- 1-(1-R2)*(n-1)/(n-p-1)
adjR2
```

```
[1] 0.4607345
```

Hide

```
# Checking variable importance
varImp(rf)
```

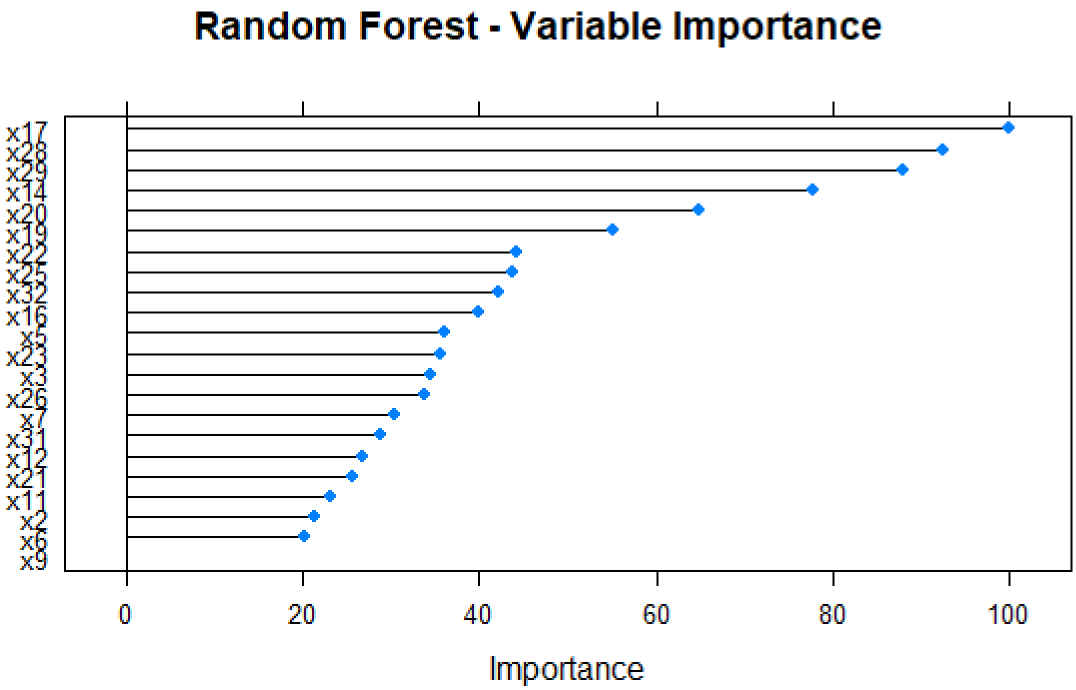
```
rf variable importance

  only 20 most important variables shown (out of 22)
```

	Overall <dbl>
x17	100.00000
x28	92.52116
x29	87.90637
x14	77.79387
x20	64.84714
x19	55.11545
x22	44.30435
x25	43.69935
x32	42.29745
x16	39.96782

1-10 of 20 rows Previous **1** 2 Next

Hide

```
# Plot of variable importance
plot(varImp(rf), main = "Random Forest - Variable Importance")
```



Hide

```
# Create a random forest using all the original predictors

rf_large = cbind(x_variables, y_variables['y1'])
split2<- sample(c(rep(0, 0.7 * nrow(rf_large)), rep(1, 0.3 * nrow(rf_large))))
train_rf <- rf_large[split1 == 0, ]
test_rf <- rf_large[split1== 1, ]
y_rf <- subset(test_rf, select=c(y1))
test_rf <- subset(test_rf, select=-c(y1))
```

Hide

```
set.seed(7)
ctrl <- trainControl(
  method = "cv",
  number = 10,
)
# Create the forest
# Again, by using the 'rf' method in caret, we also incorporate a lasso model
rf2 <- train(
   y1 ~ .,
   data = train_rf,
   method = 'rf',
   preProcess = c("center", "scale"),
   trControl = ctrl
   )
summary(rf2)
```

```
                Length Class      Mode
call              4    -none-     call
type              1    -none-     character
predicted        74    -none-     numeric
mse             500    -none-     numeric
rsq             500    -none-     numeric
oob.times        74    -none-     numeric
importance       91    -none-     numeric
importanceSD      0    -none-     NULL
localImportance   0    -none-     NULL
proximity         0    -none-     NULL
ntree             1    -none-     numeric
mtry              1    -none-     numeric
forest           11    -none-     list
coefs             0    -none-     NULL
y                74    -none-     numeric
test              0    -none-     NULL
inbag             0    -none-     NULL
xNames           91    -none-     character
problemType       1    -none-     character
tuneValue         1    data.frame list
obsLevels         1    -none-     logical
param             0    -none-     list
```

Hide

```
rf_pred <- predict(rf2, newdata = test_rf)

# RMSE
RMSE <- cbind(y_rf, round(rf_pred,4))
RMSE['SqDiff'] <- (-RMSE['y1'] + RMSE[,c(2)])^2
Model_RMSE <- sqrt(mean(RMSE$SqDiff))
Model_RMSE
```

```
[1] 0.1901425
```

Hide

```
# Calculate adjusted R2
R2 <- 0.4654447
n <- 74
p <- 1
adjR2 <- 1-(1-R2)*(n-1)/(n-p-1)
adjR2
```

```
[1] 0.4580203
```

Hide

```
# Checking variable importance
varImp(rf2)
```

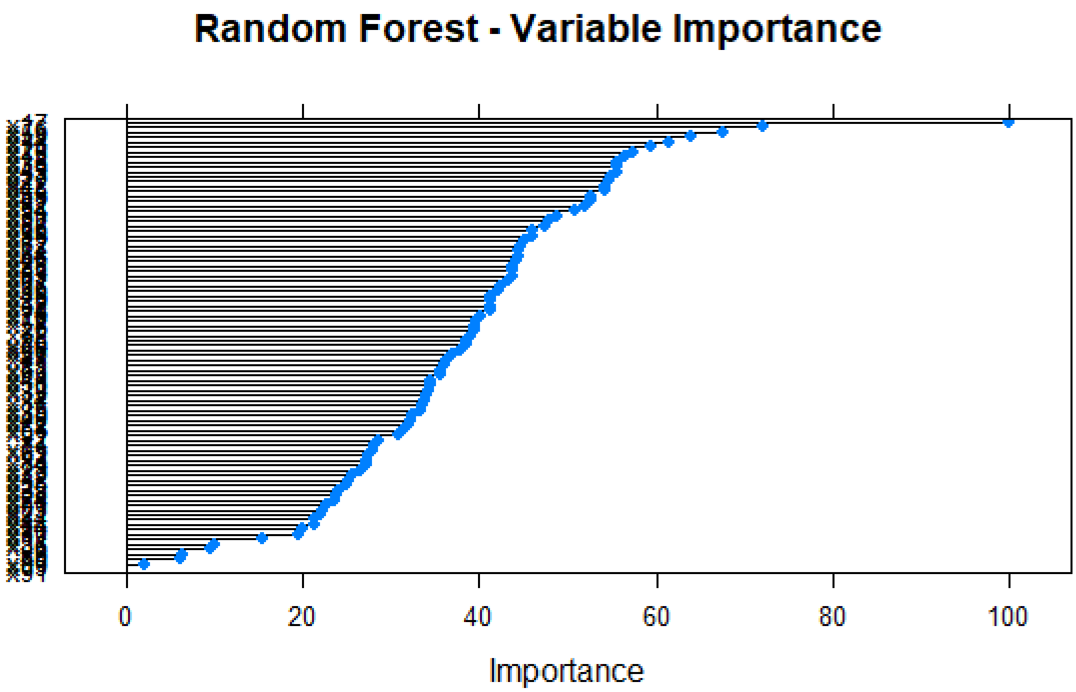
```
rf variable importance

  only 20 most important variables shown (out of 91)
```

	Overall <dbl>
x17	100.00000
x76	72.04629
x59	67.47708
x14	63.90382
x73	61.52924
x20	59.31001
x75	57.34514
x13	56.41976
x36	55.62298
x43	55.52221

1-10 of 20 rows Previous **1** 2 Next

Hide

```
# Plot of variable importance
plot(varImp(rf2), main = "Random Forest - Variable Importance")
```



Stepwise Factor Selection

Though we have tried to address the issue of multicollinearity through PCA, which is a technique designed to accommodate highly correlated predictor variables, it is worth looking at other common solutions.

A simple approach to try is simply to remove the correlated variables. This is the quickest fix in most cases and is often an acceptable solution because the variables you're removing are redundant anyway and add little unique or independent information the model.

Forward selection and bidirectional elimination will be tried find a set of independent variables that significantly influence the dependent variable and, hopefully, are not highly correlated.

Hide

```
# Forward stepwise
# Create a clean dataframe to work from
set.seed(7)
stepwise <- cbind(x_variables, y_variables['y1'])

# Define intercept-only model
intercept_only <- lm(y1~1, data=stepwise)

# Define full model
all <- lm(y1~., data=stepwise)

# Perform forward stepwise regression
forward <- step(intercept_only, direction='forward', scope=formula(all), trace=0)

# View results of forward stepwise regression
forward$anova
```

Step <S3: AsIs>	Df <dbl>	Deviance <dbl>	Resid. Df <dbl>	Resid. Dev <dbl>	AIC <dbl>
	NA	NA	104	4.216775	-335.5634
+ x29	-1	0.61169437	103	3.605080	-350.0197
+ x66	-1	0.34340119	102	3.261679	-358.5304
+ x31	-1	0.15371637	101	3.107963	-361.5993
+ x86	-1	0.14269560	100	2.965267	-364.5343
+ x25	-1	0.22190994	99	2.743357	-370.7017
+ x60	-1	0.13973563	98	2.603622	-374.1910
+ x85	-1	0.14209541	97	2.461526	-378.0838
+ x53	-1	0.05170977	96	2.409816	-378.3130

9 rows

Hide

```
# See final model
forward$coefficients
```

```
(Intercept)         x29         x66         x31         x86 
   4.501598   -3.401911   -5.375139   -0.993087   26.433430 
        x25         x60         x85         x53 
  -1.232727   -9.828022   -1.113957   -0.929767 
```

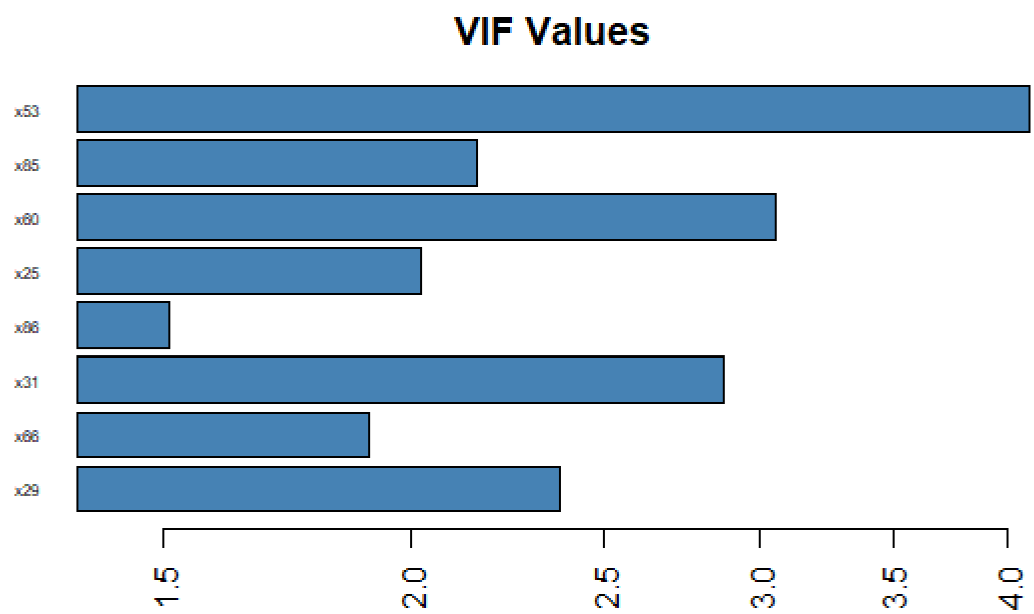
Hide

```
# VIF on the linear regression model
vif(forward)
```

```
     x29      x66      x31      x86      x25      x60      x85 
2.375435 1.902150 2.875429 1.507427 2.021312 3.053653 2.157817 
     x53 
4.104442 
```

Hide

```
# Create vector of VIF values
vif_values <- vif(forward)

# Create horizontal bar chart to display each VIF value
barplot(vif_values, main = "VIF Values", horiz = TRUE, col = "steelblue",las=2, cex.names=.5, lo
g="x")

# Add vertical line at 5
abline(v = 5, lwd = 3, lty = 2)
```



With VIF scores under 5, we now have acceptable levels of multicollinearity.

Hide

```
#perform both-direction stepwise regression
both <- step(intercept_only, direction='both', scope=formula(all), trace=0)

#view results of backward stepwise regression
both$anova
```

Step <S3: AsIs>	Df <dbl>	Deviance <dbl>	Resid. Df <dbl>	Resid. Dev <dbl>	AIC <dbl>
	NA	NA	104	4.216775	-335.5634
+ x29	-1	0.61169437	103	3.605080	-350.0197
+ x66	-1	0.34340119	102	3.261679	-358.5304
+ x31	-1	0.15371637	101	3.107963	-361.5993
+ x86	-1	0.14269560	100	2.965267	-364.5343

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
<S3: AsIs>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
+ x25	-1	0.22190994	99	2.743357	-370.7017
+ x60	-1	0.13973563	98	2.603622	-374.1910
+ x85	-1	0.14209541	97	2.461526	-378.0838
+ x53	-1	0.05170977	96	2.409816	-378.3130

9 rows

Hide

```
#view final model
both$coefficients
```

```
(Intercept)         x29         x66         x31         x86 
   4.501598   -3.401911   -5.375139   -0.993087   26.433430 
        x25         x60         x85         x53 
  -1.232727   -9.828022   -1.113957   -0.929767 
```

Both forward stepwise selection and bidirectional elimination approaches resulted in the same set of predictors. It would appear that this subset of predictors also elimates the multicollinearity problem and improves our R2 for the model. However, this model was created using the entire dataset and thus may be overfitted, so let us consider these predictors but use a k-fold cross-validation approach.

Hide

```
set.seed(7)

# Set training control to cross-validation with 10 folds
train_control <- trainControl(method = "cv", number = 10)

# Create a data subset
forward_df <- subset(stepwise, select=c(y1,x29,x66,x31,x86,x25,x60,x85,x53))

# training the model by y1 as the target variable as a function of only
# those variables identified by forward selection
forward_cv<- train(y1~., data = forward_df,
               method = "lm",
               trControl = train_control)
print(forward_cv)
```

```
Linear Regression

105 samples
  8 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 94, 95, 93, 95, 95, 95, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.1482888  0.4327947  0.1015939

Tuning parameter 'intercept' was held constant at a value of TRUE
```

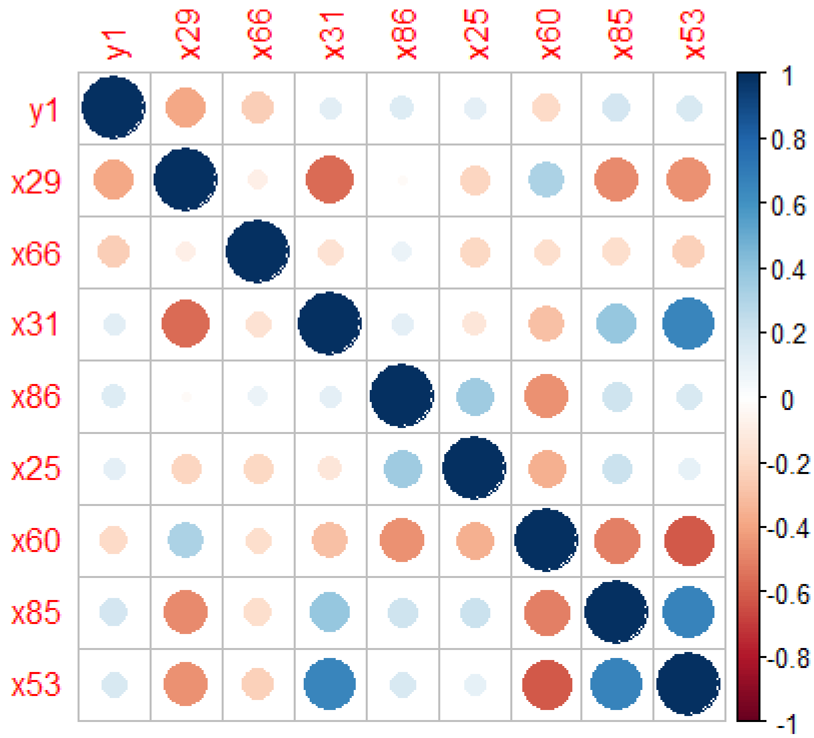
Hide

```
summary(forward_cv)
```

```


Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-0.53818 -0.04132  0.01761  0.05959  0.29711

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.5016     0.5426   8.296 6.69e-13 ***
x29          -3.4019     0.4863  -6.996 3.53e-10 ***
x66          -5.3751     0.8445  -6.365 6.68e-09 ***
x31          -0.9931     0.2819  -3.523 0.000655 ***
x86          26.4334    10.1293   2.610 0.010516 *
x25          -1.2327     0.3032  -4.065 9.82e-05 ***
x60          -9.8280     2.8669  -3.428 0.000897 ***
x85          -1.1140     0.6895  -1.616 0.109448
x53          -0.9298     0.6478  -1.435 0.154464
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1584 on 96 degrees of freedom
Multiple R-squared:  0.4285,    Adjusted R-squared:  0.3809
F-statistic: 8.998 on 8 and 96 DF,  p-value: 3.86e-09
```

Hide

```
# Create a correlation matrix
corrplot(cor(forward_df))
```



y1
x29
x66
x31
x86
x25
x60
x85
x53
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1