# The Post-Affordable Care Act Landscape: A Non-Parametric and Parametric Statistical Analysis

*Adriel Sumathipala*

## Contents

# Introduction and Objectives:

For this project, I will analyze trends in health outcomes in the post-ACA (Affordable Care Act) environment. Specifically, I will examine the healthcare landscape of my home state, Virginia. To better understand the healthcare landscape in Virginia, I will use both non-parametric and parametric statistical models to quantify healthcare trends and make projections about future healthcare premiums. With a growing elderly population, rising per-capita healthcare costs, and highly unequal health outcomes, my analysis aims to make projections about the future of healthcare in my state and understand how current healthcare policies, such as the Affordable Care Act, are being used and improving health outcomes.

Through this project, I will employ rigorous analysis to large, complex healthcare datasets available on a range of Federal agencies' webpages. Specifically, I will use the Qualifying Health Plan Selections by Metal Level and County datasets from the Centers for Medicare and Medicaid (CMS) (Source: data.cms.gov). These datasets provides the selection of the ACA's qualifying health plans, which are the heavily-subsidized insurance plans available on the state exchanges created by the ACA. I will join this table with 2016 Educational Attainment Data, 2016 US Census Income Data, and 2016 US Medicare Expenditures, to better understand the social and economic factors underlying the ACA's impact on Virginia's healthcare landscape. From these datasets I will select only the counties that lie within Virginia.

I anticipate that non-parametric models will be more useful in predicting future premium prices than parametric models, because non-parametric models make fewer assumptions about the data. Given the large number of factors that affect health outcomes, I believe making fewer assumptions about the data will improve my models.

Finally, this project is made possible only by the enormous wealth of data that has been generously provided by several federal agencies. In the future, I hope to use more datasets from different agencies through this project and in later courses at Yale.

# Part 1: Pre-Processing and Describing Data

## Importing Data:

In the code below, I have taken several steps to pre-process the data and imported all necessary R libraries. First, I read in nearly 1GB (917 mb) worth of text data from spreadsheets created by querying federal databases. These datasets contain data for all 50 US states. I am importing the following data sets:

- 2017, 2016, 2015, 2014, 2013 Qualified Health Plan Data
- Each of these datasets includes 150 features:
  - Monthly Premiums in every FIPS county for every age group, gender, and family household size
    - The health insurance exchanges created by the ACA set premiums based off age, gender, and family household size charactersitics
  - Drug Deductibles, Maximum Out of Pocket Expenditures, Medical Deductibles
  - Co-Pays for Primary Care Physicians, Specialists, Emergency Room visits, Inpatient Facilities, Generic Drugs, Preferred Brand Drugs, Specialty Drugs
- 2016 US Medicare Expenditure, Enrollee, and Reimbursement Data
- 2016 US Census Income by Household Type, Marriage, Race, Sex, Age

## Pre-Processing Data

In the code here, I have taken some early pre-processing steps. I first filter the QHP datasets by selecting only the entries from Virginia. Across all the datasets, the data are organized by county FIPS (Federal Information Processing Standards) codes. FIPS codes are the standard geographic unit used by federal agencies to organize data. Accordingly, I have created a vector that stores unique Virginia FIPS codes. Because the data I am using comes from federal databases, it contains data on all 50 states, whereas I am only interested in my home state of Viriginia. As a result, I will use this vector containing unique Virginia FIPS codes with `dplyr` package to select the data relevant to this project, using functions such as `inner_join()`, `left_join()`, etc.

Continuing my work to pre-process the data, I wrote a function that strips commas and dollar signs from matrix columns and applied it to datasets using the `apply()` function to speed up implementation of code. I also wrote code to reconcile differences in the FIPS system from 2013-2017 as regional and county demarcations within my home state of Virginia have changed. I wrote a function that enables the plotting of monthly premiums, given the plan level, age/family household structure, and year using the `choroplethr` package. This function strips the "$" sign, and selects the appropriate dataset by year, plan level, and age/household structure. Because there are several plans, each with their own monthly premium price, for each year, plan level, and age/household structure in each FIPS district, I then use the `dplyr summarise()` function to compute the mean premium price in each FIPS district. Finally, I connect the FIPS code with their associated monthly premium so they can be plotted appropriately.

## Exploring dimensionality of the data:

*QHP Dataset:* To get a sense of the dimensionality of the Qualified Health Plan (QHP) data, I have printed out the names of the variables in the 2017 QHP dataset here. This dataset includes a few variables that will not be useful such as "Plan Marketing Name" or "Network URL".

*Medicare Dataset:* In this project, I will also use the Medicare dataset to better understand cost, healthcare utilization, and federal healthcare spending within each FIPS district. I have printed out the names of 10 variables within this dataset here.

*Education Dataset:* In this project, I will also use the 2016 Educational Attainment dataset published by the US Census. I am interested in this dataset to better understand the relationship between education and monthly premiums. There are two possible theories regarding healthcare and education. The first is that higher education leads to lower premiums because people with higher levels of educations have healthier lifestyles/diets and don't live/work in dangerous environments. The second is that higher education leads to higher premiums because people with higher educations live in more urban areas with higher costs of living and are more able to seek expensive specialist care when they need it. As a result, their overall per-capita healthcare expenditures are higher which is then reflected in the premiums they pay. I have printed out the names of the 10 variables within the 2016 Educational Attainment dataset (which contains 771 total variables) here to provide a high-level overview of the data structure.

*Income Dataset:* I will also use the 2016 Income dataset from the US Census. I suspect income has a strong relationship with premiums as insurers will price their premiums, in part, based off what consumers can pay. I have printed out 10 variables from the 2016 US Census Income dataset here.

## Choropleth Mapping of Data:

After pre-processing data, I have plotted monthly premium data for Virginia counties. I have plotting the average monthly premium price for **Bronze Plans** for **21-year-old non-married adults** from 2013-2017 here. I have also plotted 2016 income data by FIPS county codes here. As seen here, incomes are generally higher in the northern part of Virginia, near DC, where many suburbanites live and commute to DC. The poorest parts of the state are the south and southwest regions which are former industrial and coal mining

regions. In addition to income, I have plotted the percentage of the population with a HS or Bachelor's degree by FIPS county here. As we can see there is a strong correlation between the income and education. Again, the northern part of state is both much wealthier and better educated and the southwestern part of the state is both poor and not well-educated. Additionally, with regards to the percentage of the population with a Bachelor's degree we see a massive disparity between districts with only 8% of the population having a Bachelor's degree with districts where 80% of people have a Bachelor's degree.

# Part 2: Comparison of Non-Parametric and Parametric Models in Low-Dimensions

## Non-Parametric vs. Parametric Analysis of Healthcare Premiums and Medicare Reimbursements

In this section, I will compare non-parametric and parametric estimators using insurance premium and Medicare reimbursement data. I will test two non-parametric regression models and one parametric models to compare their relative performance:

- Local Regression (Non-Parametric)
  - First-Degree Local Polynomial Regression (Local Linear Regression)
  - Second-Degree Local Polynomial Regression
- Cubic Smoothing Spline (Non-Parametric)
- Linear Regression (Parametric)

In theory, there shouldn't be a strong relationship between Medicare reimbursements and monthly premiums because public healthcare spending does not have a strong effect on private insurance markets. Indeed, the scattered data reflect this theory as seen here. I have fitted a non-parametric regression function, predicting Bronze Plan monthly premiums for 21-year-old adults using the average Medicare Enrollee reimbursement in each FIPS county. I built a local regression model using the `locfit` package. I have selected the bandwidth through cross-validation and implemented 95% confidence intervals. This model can be viewed here.

In the plot here, we can see that the confidence bands (black lines) around the local linear regression function (red). The local linear regression function has a wave-like shape, whereas the true relationship between these two variables should be a horizontal line as the variables are uncorrelated, as we will see later. The 95% confidence interval in this plot demonstrates high confidence that the expected value of the predicted $E(\hat{r}(x))$ function lies within this confidence interval. This does not, however, guarantee that the true regression function $r(x)$ lies within this confidence interval.

Here, I have constructed a `locfit` model with second-degree local polynomial regression. The curvature of the fit has increased substantially with the second-degree local polynomial regression from the local linear regression model fitted earlier. From prior knowledge of the data, there should be no meaningful relationship between these two variables and thus, in this case, the local regression function is fitting to noise. The true regression function here would be a straight line, as the two variables are uncorrelated, because public insurance expenditures for retirees should not have a substantial effect on premiums for young adults in private insurance markets. Yet, as we can see, the second-degree local polynomial regression function takes a wave-like fit to the data, with the most significant curvature happening towards the boundaries of the dataset.

Overall, we see the confidence bands for the second-degree local polynomial regression are wider than the confidence bands for the local linear regression. Increasing the degree of the model, in this case, appears to lower bias at the expense of increasing variance which in turns leads to larger overall confidence bands. We can see in both the local linear regression and second-order local polynomial regression, the confidence bands widen dramatically towards the boundaries of the data. This shows that both local regression functions have high boundary bias.

The cubic smoothing spline (non-parametric) and linear regression model (parametric), will have lower boundary bias and alleviate this problem. In this plot, we can also see that the confidence intervals widen towards the boundary of the function, which we will see in both the cubic smoothing spline and linear regression.

Next, I have implemented the cubic smoothing spline and selected the smoothing parameter using cross-validation. The cubic smoothing spline model can be viewed here. I have implemented asymmetric confidence bands for cubic smoothing splines as developed at Carnegie Mellon University.

Interestingly enough, we see a substantial difference between the second-order local polynomial regression and cubic smoothing spline fits. The second-order local polynomial regression function has a distinct wave shape, whereas the cubic smoothing spline does not. The cubic smoothing spline has a slight downward slope, indicating some bias, but overall approximates the linear regression function as we will see below. It appears that the smoothing spline approximates the linear regression function in cases of high noise. The cubic smoothing spline also eliminated the boundary bias problem by having a straight fit to the data towards the boundary of the dataset. This is in accordance with the idea that higher-order polynomial fits, in general, have reduced boundary bias as we were taught in class. In class, we saw how local regression performed worse than smoothing splines at the boundaries when fitting data generated from a function such as $r(x) = x^2$.

Finally, I have implemented the linear regression model and constructed 95% confidence intervals. This linear regression model can be viewed here. In this plot, we can see there is no real relationship between reimbursement rates and premiums—the data are scattered and random. The nearly horizontal line in the linear regression supports this finding. Although the line is not exactly horizontal, the p-value of the slope is very large (p= 0.444), thus indicating this slight slope downwards is not significant. A printout of the linear model summary can be seen here here.

The 95% confidence interval for linear regression shows smaller confidence intervals towards the mean of the data, where there are a greater number of data points, and larger confidence intervals towards the boundaries, where there are fewer data points.

With this dataset, we see no difference between the linear regression fit and cubic smoothing spline fit. Because of the large noise in this data, the cubic smoothing spline approximates the OLS solution. However, both the linear regression function and cubic smoothing spline performed better than the local regression function which fit a wave-like function to uncorrelated, noisy data, providing lower bias at the expense of much higher variance as we can see with the larger confidence intervals of the local regression fits.

## Non-Parametric vs. Parametric Analysis of Education and Income:

Continuing my comparison of non-parametric and parametric methods, I have tested the performance and properties of parametric and non-parametric estimators using a new dataset—comparing the relationship between education and income.Below, I have fit a local linear regression (non-parametric) and linear regression (parametric) fits to 2016 median household income and percent of population with a high school degree for each county in Virginia. In the plots below, the 95% confidence intervals are marked by the dotted red lines. The local linear regression (non-parametric) and linear regression (parametric) models can be viewed here.

The error of the local linear regression model, with the bandwidth selected by cross-validation, is 1.622168 x $10^8$. The non-parametric MSE is **smaller** than the linear regression MSE which is 1.858521 x $10^8$. We can see in the earlier plots that the non-parametric model is a better fit to the data. However, we can also see that the parametric model has overall tighter confidence intervals than the non-parametric model. In particular, both models have large bias around the boundaries, where the data begin to increase exponentially. As a result, the confidence intervals of both functions increase around the boundaries.

Examining the plots here, it appears the local linear regression fit (non-parametric) takes an exponential form. In fact, the general shape of the data appears to be exponential. With this prior information, I will fit an exponential regression, a parametric method, to the data and compare this exponential fit with the local linear regression (non-parametric).

I built the exponential model by first fitting a linear regression to a log transformation of the data such that:

$$log(Y_i) = b + ax_i$$

I then took the exponential of this linear model, yielding the exponential model of the form:

$$Y_i = e^{b+ax_i}$$

Here, I plotted the exponential model (parametric) with 95% confidence intervals. The exponential model is an overall much better fit to the data than the linear regression model. In fact, the parametric exponential model, is nearly identical to the local linear regression model (non-parametric), as seen in this side-by-side comparison plot here

As we can see, having *a priori* information about the structure of the data allows us to build parametric models that are quite similar to the performance of non-parametric models. Thus, in certain cases, the parametric model can approximate the non-parametric model. Because of the large bandwidth, $h$, used to construct the local linear regression fit, the non-parametric fit is quite smooth and closely approximates the exponential parametric fit.

Moving on from analyzing high school data to bachelor's degree data, I have similarly built a local linear regression (non-parametric) and linear regression (parametric) model for this new data. These models can be viewed here.

As we can see in the plots here, there is little difference between the parametric and non-parametric model. The non-parametric model appears to largely converge to the parametric linear model. The non-parametric model has far wider confidence interval towards the rightmost boundary of the data, as there are fewer datapoints there. As with the earlier example, the non-parametric regression fit can approximate the parametric fit, if the underlying structure of the data is linear, as in this case, or exponential, as in the earlier case.

# Part 3: Predicting Monthly Premiums using Non-Parametric (GAM) and Parametric (LASSO) Methods

## Importance of Monthly Premium Prediction Models:

Within the field of healthcare analytics, there is a great deal of interest in predicting monthly insurance premiums. These monthly premiums have a substantial impact on the healthcare landscape, influencing consumer purchasing decisions, healthcare resource utilization, and the overall risk pool. Thus, being able to predict future monthly premiums is highly useful for both private and public healthcare decisionmakers

Within the private sector, hospitals are interested in knowing the monthly premiums of a particular demographic group to know how many uninsured patients they should expect to treat in the upcoming fiscal year. Such uninsured patients have a greater tendency to not make payments for their treatments and declare bankruptcy. In the public sector, predicting healthcare premiums is important to allocate sufficient funds for the cost-sharing reduction (CSR) program established by the Affordable Care Act (ACA). For lawmakers in Congress, this information is crucial when developing a budget because the state exchanges established by the ACA are reliant on federal funding; if there are not enough funds available, insurers will exit the markets or increase costs for consumers.

## GAM Model Assumptions:

In the code below, I will compare the relative performance of non-parametric and parametric regression models. I will build a generalized additive model (GAM), a non-parametric method, and a LASSO model

(least absolute shrinkage and selection operator), a parametric-method. I am using LASSO because of the large number of variables I am using in this prediction model. The LASSO method is a selection operator and promotes parsimonious models by setting some model coefficients equal to zero.

In the code below, I have constructed a smoothing spline GAM model. This model makes two assumptions:

1.) The errors are normally distributed

2.) $m_i(x)$ is a cubic smoothing spline

I am using the Gaussian family for a description of the error distribution. This assumes the model errors are normally distributed such that the general form of the data is:

$$Y_i = m_1(x_{i,1}) + m_2(x_{i,2}) + ... + m_p(x_{i,p}) + N(0,1)$$

Furthermore, I am assuming that each regression function is of the form of a cubic smoothing spline such that $m_i(x)$ is:

$$m_i(x) = argmin_{\hat{f}(x)} \sum_{i=1}^{n} \{Y_i - \hat{f}(x_i)\}^2 + \lambda \int \hat{f}''(x)^2 dx$$

## Building Spline GAM Model

To prepare for modelling using GAM, I created a dataframe that combines all predictor variables. This data manipulation code can be viewed in the appendix here

In this combined dataset, `ACAdf`, I have extracted only the most important features. This data set contains 16 variables including:

- 8 Medicare Features (Number of Enrollees, Average Reimbursements, etc.)
- 2 Education Features (% with Bachelor's or HS Degree)
- 4 Economic Features (Median Household, Married Couple, Family, and Non-Family annual incomes)
- 1 Healthcare Plan Feature (Annual Deductible)
- 1 Geographic Feature (FIPS County Code)

I will build two types of GAM Models:

- Smoothing Spline GAM Model
- Local Regression GAM Model

I am testing these two methods and comparing their relative performance. I first built a GAM model using the smoothing splines method. In the plots here, I have plotted the spline fit for each variable, with the confidence intervals (dashed lines), and residuals (black points). As we can see, this model has large residuals. Moreover, we can see that most of the variables are uncorrelated and the estimated regression functions are straight lines. The only variables with a decent relationship with premiums is the percentage of population with bachelor's degrees and annual deductibles. We can see that more educated populations tend to have higher premiums—this may be because more educated populations are more likely to take advantage of high-cost specialist care. Interestingly enough, deductibles and premiums appear to have a parabolic relationship, for which there is no obvious explanation. In this plot, I have compared non-parametric and parametric fits to the deductible data. Here, the blue line is the local regression fit (non-parametric) and the red line is the linear fit (parametric). The non-parametric fit is clearly a better fit to the data. Examining this plot and the plots produced by the GAM model, I believe the non-parametric model (GAM) will outperform the parametric model (LASSO). I have visualized the fit of the smoothing spline GAM model by plotting the residuals here. In this plot, we can see that the model has a large boundary bias and is unable to accurately

predict large and small premium prices far away from the mean, plotted as the vertical black line. Before moving onto local regression GAM, I have evaluated the performance of the spline smoothing GAM model using `k-fold` cross-validation. The code for this evaluation of GAM performance can be seen here.

## Building GAM Local Regression

For this GAM model, I have used the changed the earlier GAM model assumptions and I am now using a local linear regression estimator to fit each variable. I will test and see if this improves model performance. This alteration changes the earlier assumption that $m_i(x)$ was a smoothing spline and now assumes that $m_i(x)$ is a local regression function, not a spline. This GAM local regression model can be seen here.

Here, we can see that using local regression did not substantially change the model's fit to the predictor variables. These plots show that most of the variables do not have strong relationship with premium prices as seen with the GAM smoothing spline model. However, as with the GAM smoothing splines, the deductible variable has a strong relationship with monthly premiums, forming an interesting parabolic relationship between deductibles and monthly premiums. As we can see in these plots, the `s()` smoothing spline fit is smoother than the `lo()` local regression fit. However, the overall model predictions from the GAM local regression fit remain similar to the GAM smoothing spline fit as we can see in a side-by-side comparison of the residuals from both models here.

The boundary bias seen in the linear relationship in this plot can be reduced by decreasing the smoothness of the spline or local regression fits to the data. Decreasing the smoothness of the fitted functions will decrease bias at the expense of increasing variance. I have reduced the smoothness of the GAM fits by decreasing the `spar` and `span` model parameters. These new models and their associated residuals can be seen here.

As we can see in this residuals plot, the residuals have been dramatically decreased by reducing the smoothness of the local regression and spline smoothing functions. Moreover, the boundary bias seems to have decreased substantially and residuals are more evenly distributed about 0. Finally, I have calculated the `k-fold` cross-validation MSE to compare the GAM local regression to the GAM smoothing spline model. This code can be viewed here.

The k-fold validation MSE are as follows for the two GAM models:

- GAM local regression = 11431.98
- GAM smoothing spline = 10525.24

From the MSE values calculated above, it would appear that the smoothing spline GAM model outperformed the GAM local regression model.

## Multivariate LASSO Regression Model (Parametric)

After examining the performance of non-parametric prediction models, I will now move on to examine a parametric model—specifically the LASSO regression model. Because of the high number of variables used to predict premiums, I am using the LASSO method to eliminate unnecessary variables and improve overall model performance. The LASSO method behaves as a selector operator and can set model coefficients equal to zero. The LASSO can be defined as the following where $\lambda$ is the penalty parameter and shrinks model coefficients:

$$\hat{m}(x) = argmin_{\beta}||Y - X\beta||_2^2 + \lambda|\beta|_1$$

To construct the LASSO model I will first find the optimal lambda value using k-fold cross-validation (50 iterations). This process and the associated plot of $log(\lambda)$ vs. MSE can be seen here. Having selected the optimal $\lambda$, I will construct the LASSO model using this $\lambda$ value. Several plots showing the mechanics of the

underlying LASSO model and the decay of coefficient size as a function of $\lambda$ can be seen here. To visualize the fit of the LASSO, residuals from the LASSO model can be seen here.

In the residuals plot here, it appears the LASSO model has boundary bias, wherein the model is unable to accurately predict premiums far away from the mean. The solid black line in the plot is the mean of the data. This was also the case with both non-parametric GAM models, as seen in this side-by-side comparison plot. Finally, I have calculated the k-fold cross-validated MSE for the LASSO model. The code for this evaluation of model performance can be seen here.

# Part 4: Conclusions

## The Challenge of Predicting Insurance Premiums:

In general, it remains a difficult, largely unsolved problem to predict healthcare premiums, which is reflected in the subpar GAM model performance as seen above. The difficulty in this problem reflects the complex nature of the American healthcare system which involves 1.) a large range of variables not reflected in the analysis above and 2.) substantial regional care pricing variation. A Yale research study examined the regional variation in healthcare care pricing, finding an enormous amount of variation in pricing between different regions. For instance, an MRI costs 12 times more in Bronx, New York than in Baltimore, Maryland. This regional variation in costs is in turn reflected in the regional variation in premium pricing. As a result of these factors, neither the non-parametric nor parametric models could accurately and reliably predict premium prices.

## Comparing Non-Parametric vs. Parametric Model Performance:

Despite the shortcomings of prediction accuracy, the non-parametric model performed twice as well as the parametric model— the non-parametric model had half the k-fold cross validation MSE as the parametric model:

- **LASSO Model:** 21171.69
- **GAM Local Regression (Non-Parametric):** 11431.98
- **GAM Spline Smoothing (Non-Parametric):** 10525.24

This outcome shows the benefits of using non-parametric models in applications where there are non-linear relationships between predictor variables. Within this dataset, we saw a large amount of noise and non-linear relationships, which is likely why the non-parametric model performed better. Finally, the spline smoothing GAM model performed 8% better than the local regression GAM model. This indicates that increasing smoothness in the fits to the data lowered overall model risk and improved performance.

## Future Work:

This work could be further improved by using the entire national datasets to build more informative models, rather than looking at a single state. Using the national dataset could reduce overall model error by increasing the training dataset size and building models that are less prone to outliers. As Professor Lafferty said on Piazza, "there's no data like more data".

This work could also be improved by looking at a greater range of variables outside of Medicare, Education and Income variables. Further work could examine the influence of Medicaid, public health, and other variables on premium prices. Healthcare insurance premiums are influenced by a wide range of factors not reflected in the data sets used in the analysis and better models could be developed by using a wider array of variables and then using a selection method such as LASSO to identify the most significant features for prediction.

# Appendix:

## Data Processing Code

Return to Pre-Processing Data

```
#Import mapping libraries:
library(choroplethr)
library(choroplethrMaps)
library(dplyr)
library(data.table)
library(locfit)
library(Iso)
library(gam)
library(splines)
library(glmnet)
data("df_pop_county")


#Import QHP (Qualified Healthare Plan) and clean data
plans <- read.csv("2016_Plans.csv")
qhp18 <- data.table::fread("PHY18.csv")
```

```
FALSE
Read 40.0% of 99999 rows
Read 99999 rows and 150 (of 150) columns from 0.075 GB file in 00:00:03
```

```
qhp17 <- data.table::fread("PY2017_Medi-Indi-Land-08_11_2017.csv")
qhp16 <- data.table::fread("PY2016_Med-Indi-Land-07-29-2016.csv")
```

```
FALSE
Read 78.8% of 88857 rows
Read 88857 rows and 146 (of 146) columns from 0.140 GB file in 00:00:03
```

```
qhp15 <- data.table::fread("PY2015_Med-Indi-Land-08-13-2015.csv")
qhp14 <- data.table::fread("Individual_Market_Medical_8_11_14.csv")
countyNames <- data.table::fread("CountyNamesFIPS.csv")[,1:2]
colnames(countyNames) <- c("FIPS County Code", "County Name")
medicareraw <- data.table::fread("medicare.csv")
colnames(medicareraw)[1] <- "FIPS County Code"
medicareVAtmp <- right_join(medicareraw, countyNames)
medicareVAtmp$`County name` <- NULL
medicareVAtmp$`County Name` <- NULL

vaFIPS <-  data.frame(region = unique(filter(qhp18, qhp18$`State Code` == "VA")$`FIPS County Code`))

#Importing 2016 Income Data from US Census
#Data is organized by FIPS County codes:
incomeRaw <- data.table::fread("2016Income.csv")
incomeVA <- select(incomeRaw, c(Id2, `Households; Estimate; Median income (dollars)`, `Nonfamily househo
names(incomeVA) <- c("region", "HouseholdIncome", "NonFamilyIncome", "MarriedCoupleIncome", "FamilyIncon
income <- select(incomeRaw, c(Id2, `Households; Estimate; Median income (dollars)`))
colnames(income) <- c("region", "value")
vaIncome <- left_join(vaFIPS, income)
```

```r
#2016 Educational Attainment Data from US Census
#Data organized by FIPS county codes:
eduRaw <- data.table::fread("2016Education.csv")
```

```r
#Strip commas from CSV files:
stripComma <- function(charVec){
  numTmp <- gsub('[,]', '', charVec)
  num <- as.numeric(numTmp)
  return(num)
}

medicareVA <- data.frame(sapply(medicareVAtmp, stripComma))

#Format FIPS correctly for earlier years
formatFIPS <- function(t, CaseIndicator){
  countyN <- countyNames
  if(CaseIndicator == 1){
    countyN$`County Name` <- toupper(countyN$`County Name`)
  }
  tmp <- data.frame(t[,2])
  colnames(tmp) <- "County Name"
  tmp$`County Name` <- as.character(tmp$`County Name`)
  x <- left_join(tmp, countyN)
  colnames(t)[2] <- "FIPS County Code"
  t$`FIPS County Code` <- x$`FIPS County Code`
  return(t)

}

va18 <- filter(qhp18, qhp18$`State Code` == "VA")
va17 <- filter(qhp17, qhp17$`State Code` == "VA")
va16 <- formatFIPS(filter(qhp16, qhp16$`State Code` == "VA"), 0)
va15 <- formatFIPS(filter(qhp15, qhp15$State == "VA"), 0)
va14 <- formatFIPS(filter(qhp14, qhp14$State == "VA"), 1)

stripDollar <- function(string){
  numeric <- as.numeric(gsub('[$,]', '', string))
  return(numeric)
}

plotByPlan <- function(planType, age, yearDF){
  planDF <- filter(yearDF, yearDF$`Metal Level` == planType)
  indx <- which(colnames(yearDF) == age)
  planDF[,indx] <- as.numeric(gsub('[$,]', '', planDF[,indx]))
  df <- planDF[,c(2, indx)]
  colnames(df) <- c("region", "value")
  tmp <- dplyr::group_by(df, region)
  plt <- dplyr::summarise(tmp, MeanPrice = mean(value))
  colnames(plt) <- c("region", "value")
  return(plt)
}


premiumsDeductible <- function(planType, age, deductible, yearDF){
```

```
  planDF <- filter(yearDF, yearDF$`Metal Level` == planType)
  premiumIndx <- which(colnames(yearDF) == age)
  deductIndx <- which(colnames(yearDF) == deductible)
  planDF[,premiumIndx] <- stripDollar(planDF[,premiumIndx])
  planDF[,deductIndx] <- stripDollar(planDF[,deductIndx])
  df <- planDF[,c(2, premiumIndx,deductIndx)]
  colnames(df) <- c("region", "premiums", "deductibles")
  tmp <- dplyr::group_by(df, region)
  plt <- dplyr::summarise(tmp, premium = mean(premiums), deductible = mean(deductibles))
  return(plt)
}

test <- premiumsDeductible("Bronze", "Premium Adult Individual Age 21", "Medical Deductible - Individual
```

## QHP Variable Names

Return to Exploring dimensionality of the data:

```
names(qhp18)[c(c(1:5), c(30:35))]
```

```
##  [1] "State Code"                      "FIPS County Code"
##  [3] "County Name"                     "Metal Level"
##  [5] "Issuer Name"                     "Premium Adult Individual Age 30"
##  [7] "Premium Adult Individual Age 40" "Premium Adult Individual Age 50"
##  [9] "Premium Adult Individual Age 60" "Premium Couple 21"
## [11] "Premium Couple 30"
```

## Medicare Variable Names

Return to Exploring dimensionality of the data:

```
names(medicareraw)[1:10]
```

```
##  [1] "FIPS County Code"
##  [2] "County name"
##  [3] "Medicare enrollees (2014)"
##  [4] "Total Medicare reimbursements per enrollee (Parts A and B) (2014)"
##  [5] "V5"
##  [6] "Hospital & skilled nursing facility reimbursements per enrollee (2014)"
##  [7] "V7"
##  [8] "Physician reimbursements per enrollee (2014)"
##  [9] "V9"
## [10] "Outpatient facility reimbursements per enrollee (2014)"
```

## 2016 Educational Attainment Variable Names

Return to Exploring dimensionality of the data:

```
names(eduRaw)[15:25]
```

```
##  [1] "Percent Females; Margin of Error; Population 18 to 24 years"
##  [2] "Total; Estimate; Population 18 to 24 years - Less than high school graduate"
##  [3] "Total; Margin of Error; Population 18 to 24 years - Less than high school graduate"
```

```
##  [4] "Percent; Estimate; Population 18 to 24 years - Less than high school graduate"
##  [5] "Percent; Margin of Error; Population 18 to 24 years - Less than high school graduate"
##  [6] "Males; Estimate; Population 18 to 24 years - Less than high school graduate"
##  [7] "Males; Margin of Error; Population 18 to 24 years - Less than high school graduate"
##  [8] "Percent Males; Estimate; Population 18 to 24 years - Less than high school graduate"
##  [9] "Percent Males; Margin of Error; Population 18 to 24 years - Less than high school graduate"
## [10] "Females; Estimate; Population 18 to 24 years - Less than high school graduate"
## [11] "Females; Margin of Error; Population 18 to 24 years - Less than high school graduate"
```

### 2016 US Census Income Data

Return to Exploring dimensionality of the data:

```
names(incomeRaw)[20:30]
```

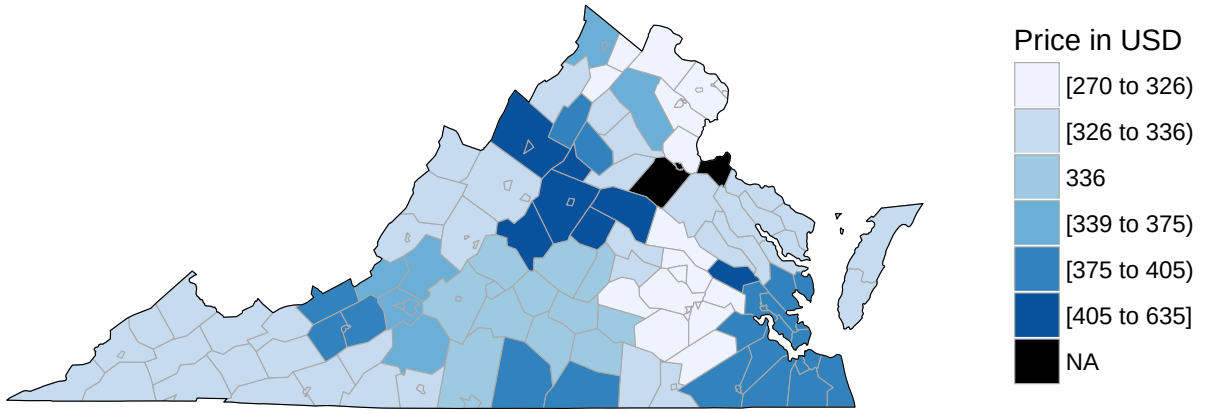
```
##  [1] "Households; Estimate; $10,000 to $14,999"
##  [2] "Households; Margin of Error; $10,000 to $14,999"
##  [3] "Families; Estimate; $10,000 to $14,999"
##  [4] "Families; Margin of Error; $10,000 to $14,999"
##  [5] "Married-couple families; Estimate; $10,000 to $14,999"
##  [6] "Married-couple families; Margin of Error; $10,000 to $14,999"
##  [7] "Nonfamily households; Estimate; $10,000 to $14,999"
##  [8] "Nonfamily households; Margin of Error; $10,000 to $14,999"
##  [9] "Households; Estimate; $15,000 to $24,999"
## [10] "Households; Margin of Error; $15,000 to $24,999"
## [11] "Families; Estimate; $15,000 to $24,999"
```

### Choropleth Mapping of Premium Data

Return to Choropleth Mapping of Data:

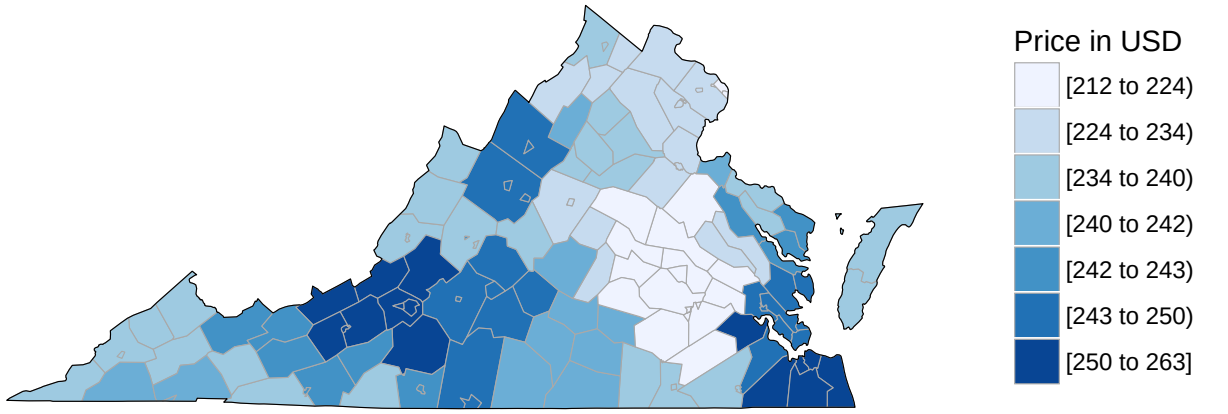
```
#Using custom-written plotByPlan function: (see Appendix)
va18age21Bronze <- plotByPlan("Bronze", "Premium Adult Individual Age 21", va18)
va17age21Bronze <- plotByPlan("Bronze", "Premium Adult Individual Age 21", va17)
va16age21Bronze <- plotByPlan("Bronze", "Premium Adult Individual Age 21", va16)
va15age21Bronze <- plotByPlan("Bronze", "Premium Adult Individual Age 21", va15)
va14age21Bronze <- plotByPlan("Bronze", "Premium Adult Individual Age 21", va14)

county_choropleth(va18age21Bronze, state_zoom = "virginia", title = "Bronze Plan Premiums for 21 Year-ol
```

Bronze Plan Premiums for 21 Year−olds in 2017



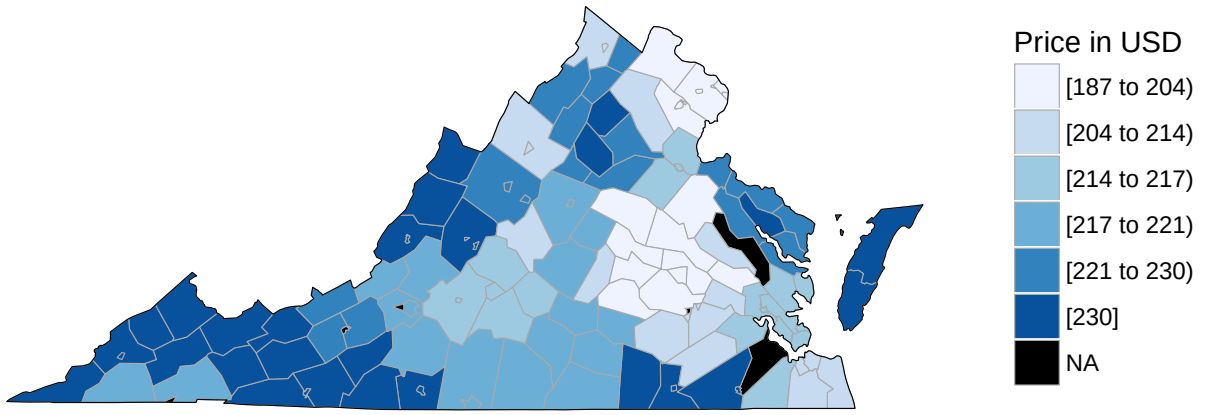

```
county_choropleth(va17age21Bronze, state_zoom = "virginia", title = "Bronze Plan Premiums for 21 Year-ol
```

Bronze Plan Premiums for 21 Year−olds in 2016



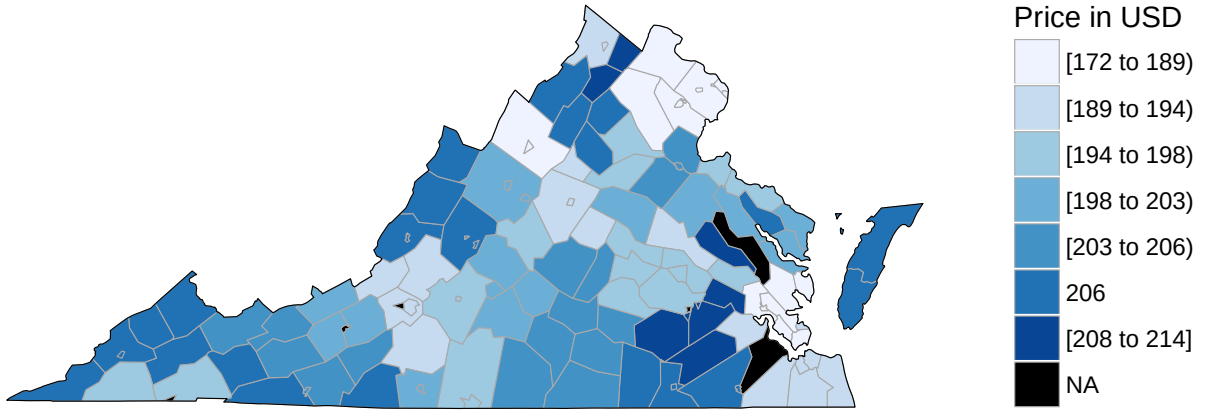

```
county_choropleth(va16age21Bronze, state_zoom = "virginia", title = "Bronze Plan Premiums for 21 Year-ol
```

Bronze Plan Premiums for 21 Year−olds in 2015



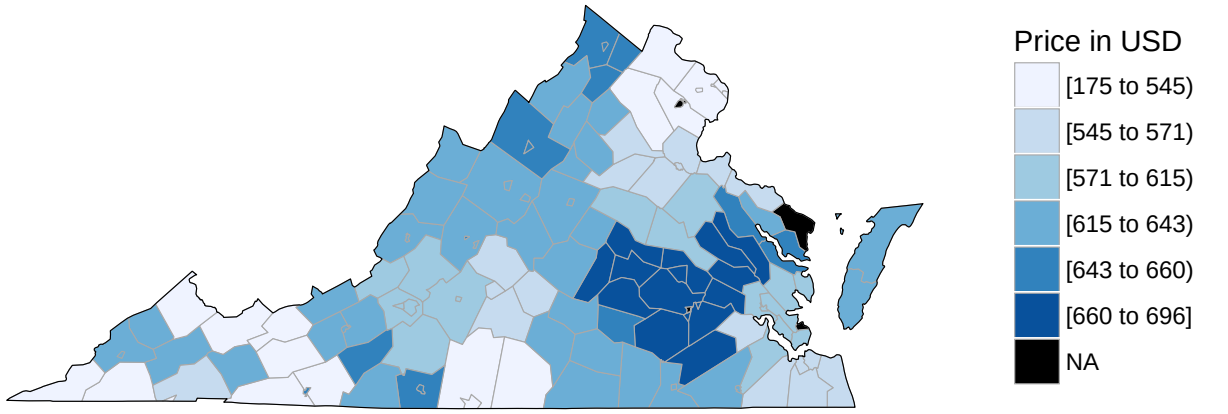

```
county_choropleth(va15age21Bronze, state_zoom = "virginia", title = "Bronze Plan Premiums for 21 Year-ol
```

Bronze Plan Premiums for 21 Year−olds in 2014




```
county_choropleth(va14age21Bronze, state_zoom = "virginia", title = "Bronze Plan Premiums for 21 Year-o
```

Bronze Plan Premiums for 21 Year−olds in 2013

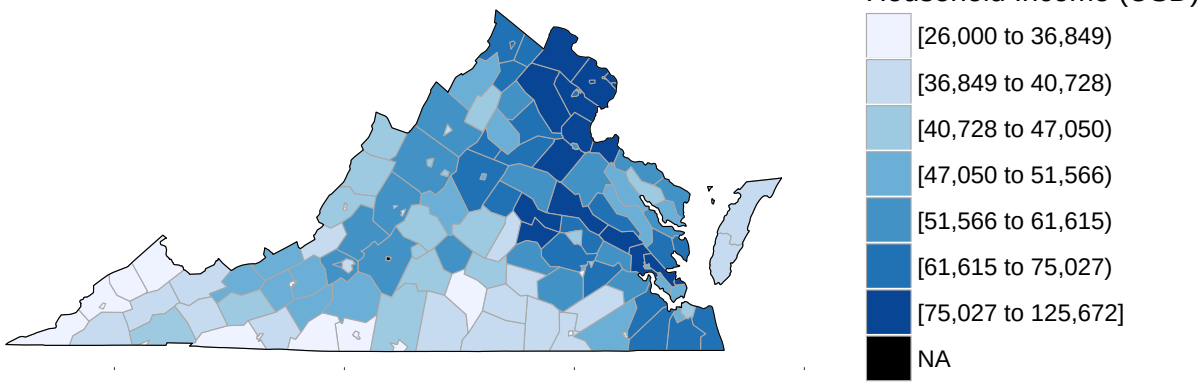



## Choropleth Mapping of Income Data

Return to Choropleth Mapping of Data:

```
county_choropleth(vaIncome, title = "2016 Median Household Income", legend = "Household Income (USD)", s
```

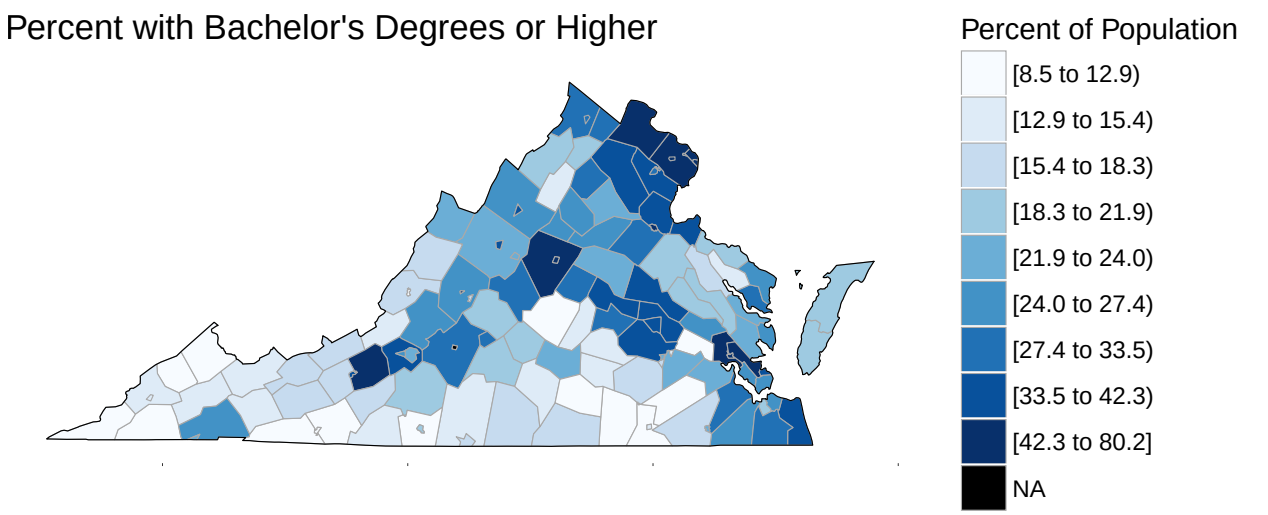
2016 Median Household Income



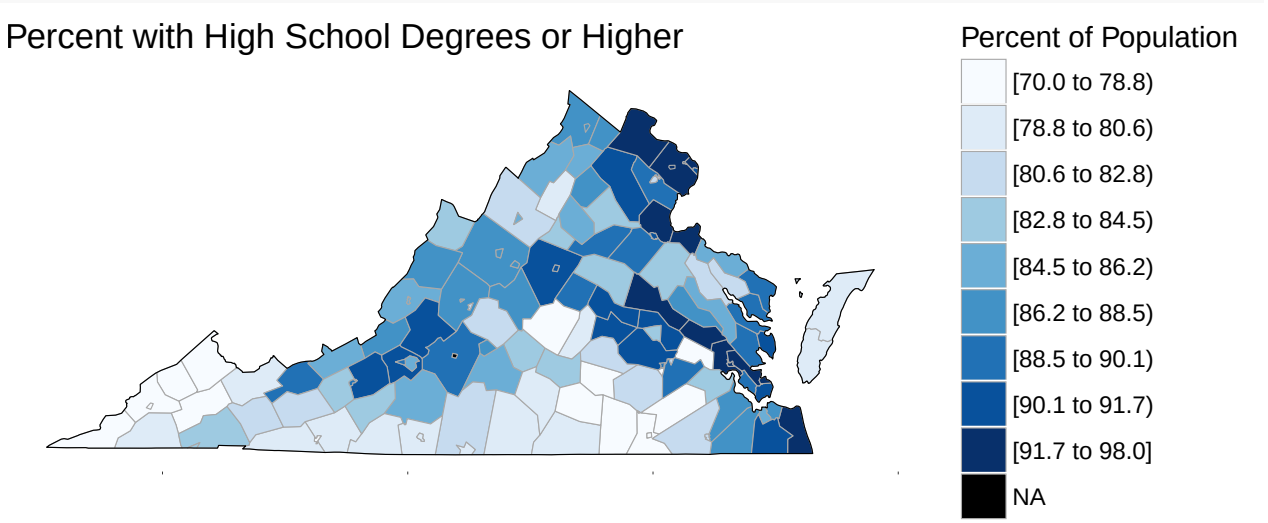
## Choropleth Mapping of Education Data

Return to Choropleth Mapping of Data:

```
edu <- select(eduRaw, c(Id2, `Percent; Estimate; Percent bachelor's degree or higher`, `Percent; Estima
colnames(edu) <- c("region", "bachelor", "hs")
vaEdu <- left_join(vaFIPS, edu)
plotBach <- data.frame(region = vaEdu$region, value = vaEdu$bachelor)
county_choropleth(plotBach, title = "Percent with Bachelor's Degrees or Higher", legend = "Percent of P
```



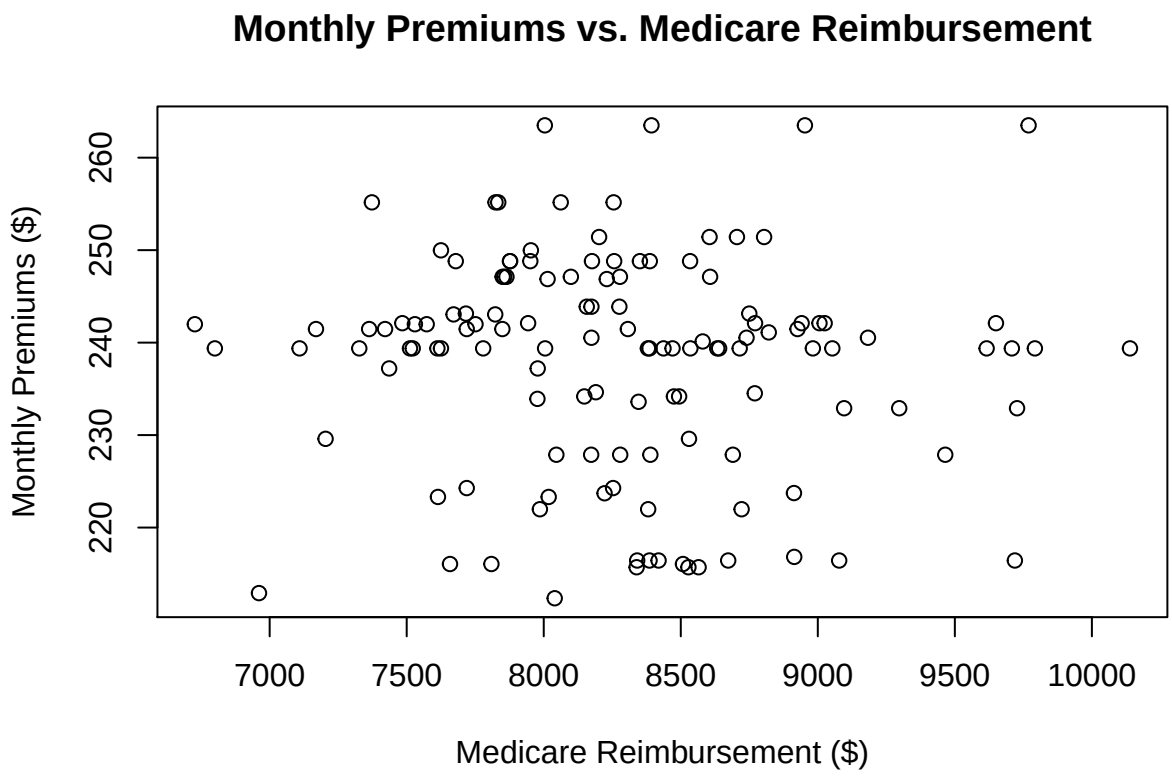

```
plotHS <- data.frame(region = vaEdu$region, value = vaEdu$hs)
county_choropleth(plotHS, title = "Percent with High School Degrees or Higher", legend = "Percent of Po
```




## Plot of Monthly Premiums vs. Medicare Reimbursement

Return to Non-Parametric vs. Parametric Analysis of Healthcare Premiums and Medicare Reimbursements

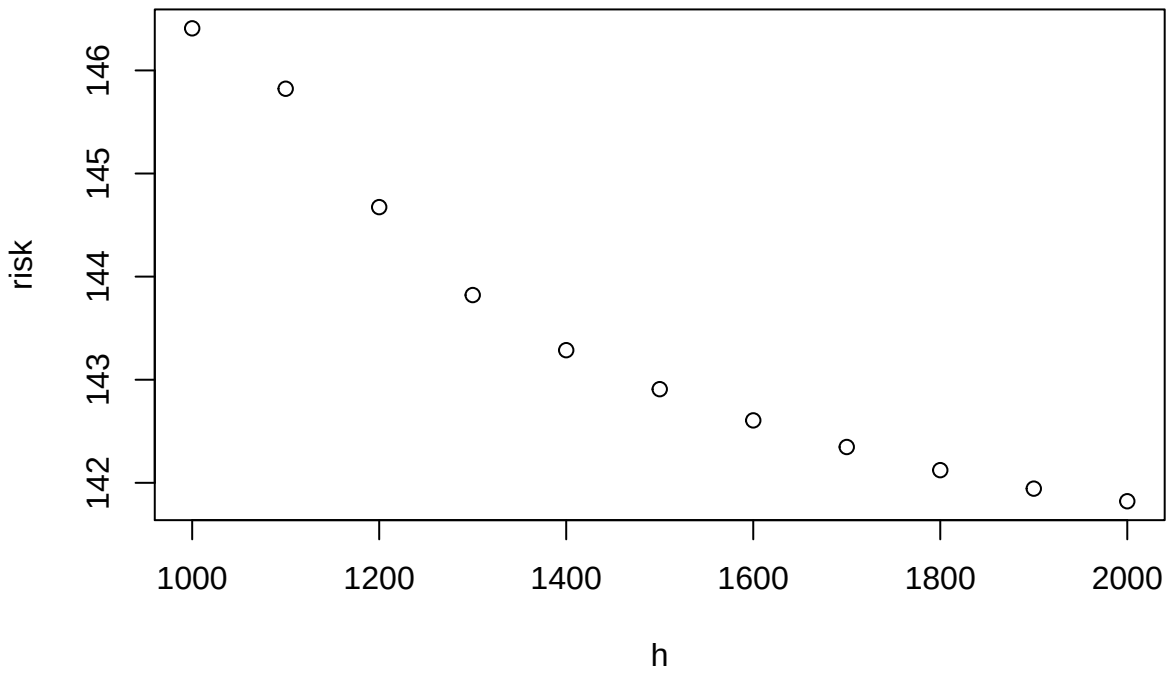
```
dta <- na.omit(left_join(medicareVA, data.frame(va17age21Bronze), by = c("FIPS.County.Code" ="region")))
dta <- dta[order(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.),]
plot(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014., dta$value, xlab = "Medicare
```




### Local Linear Regression on Premiums Data

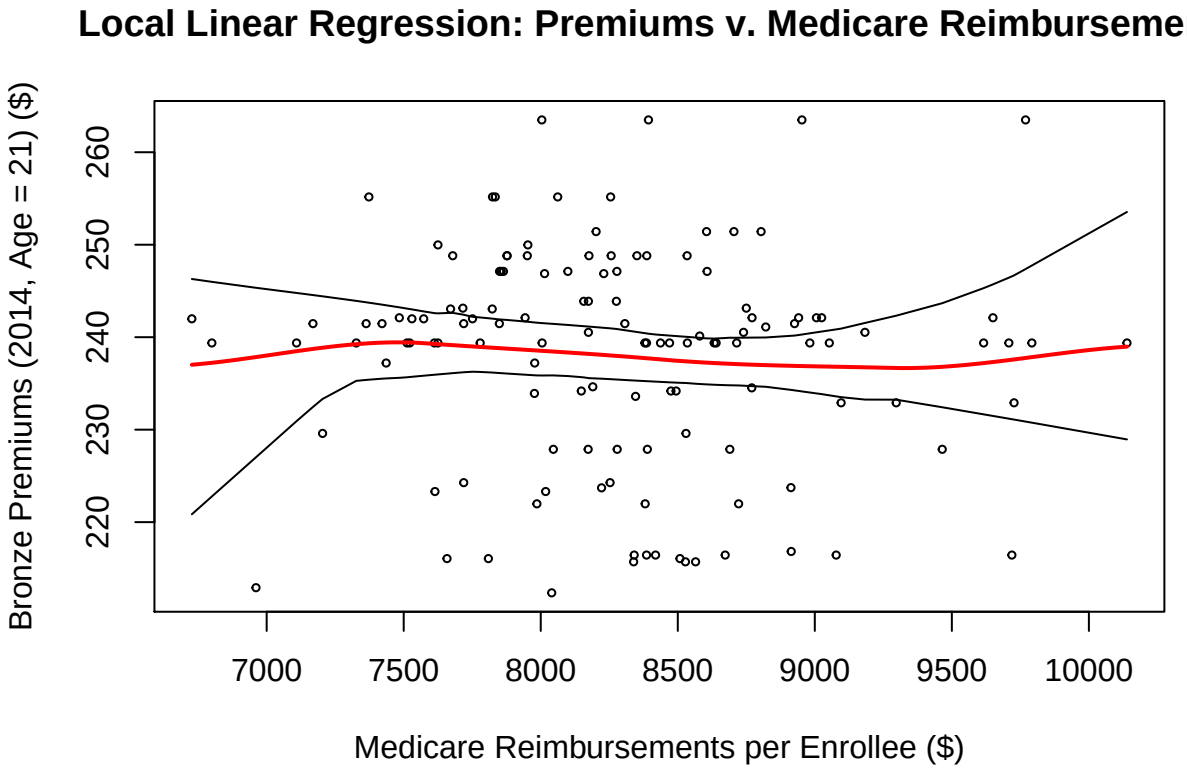
Return to Non-Parametric vs. Parametric Analysis of Healthcare Premiums and Medicare Reimbursements

```
#use to find optimum bandwidth for locfit
optBandwidth <- function(y, x, h_test){
  risk <- rep(NA, length(h_test))
  n <- length(y)
  for(i in 1:length(h_test)){
    model <- locfit(y~x, alpha = c(0, h[i]), deg = 1)
    Lii <- predict(model, where = "data", what = "infl")
    risk[i] <- (1/n) * sum(((y-fitted(model))/(1-Lii))^2)
  }
  return(risk)
}

dta <- na.omit(left_join(medicareVA, data.frame(va17age21Bronze), by = c("FIPS.County.Code" ="region")))
dta <- dta[order(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.),]

#Local Regression:
h <- seq(1000, 2000, 100)
risk <- optBandwidth(dta$value,
             dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
             h)

plot(h, risk, main = "Risk Decay vs. Bandwidth")
```




```
hopt <- h[which.min(risk)]
model <- locfit(dta$value~dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
                alpha = c(0, hopt), deg = 1)
#Creating Confidence Bands:
set.seed(1)
#B = number of iterations
B = 50
#Number of datapoints to sample:
n = 100
conf.mat <- matrix(nrow = length(fitted(model)), ncol = B)
for(i in 1:B){
  indx <- sample(1:length(fitted(model)), n, replace = TRUE)
  trainX <- dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.[indx]
  trainY <- dta$value[indx]
  risk <- optBandwidth(trainY,trainX, h)
  hopt <- h[which.min(risk)]
  tmpModel <- locfit(trainY~trainX, deg = 1, alpha = c(0, hopt))
  pred <- predict(tmpModel, newdata = dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2
  conf.mat[,i] <- pred
}
conf <- apply(conf.mat, 1, quantile, probs = c(0.025, 0.975))
upper <- conf[1,]
lower <- conf[2,]


#Plotting:
{plot(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
      dta$value,
      xlab = "Medicare Reimbursements per Enrollee ($)",
      ylab = "Bronze Premiums (2014, Age = 21) ($)",
```

```
      main = "Local Linear Regression: Premiums v. Medicare Reimbursements"
      ,cex = 0.5
      )
lines(model, col = "red", lwd =2)
points(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014., upper, type = "l")
points(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014., lower, type =  "l")
}
```

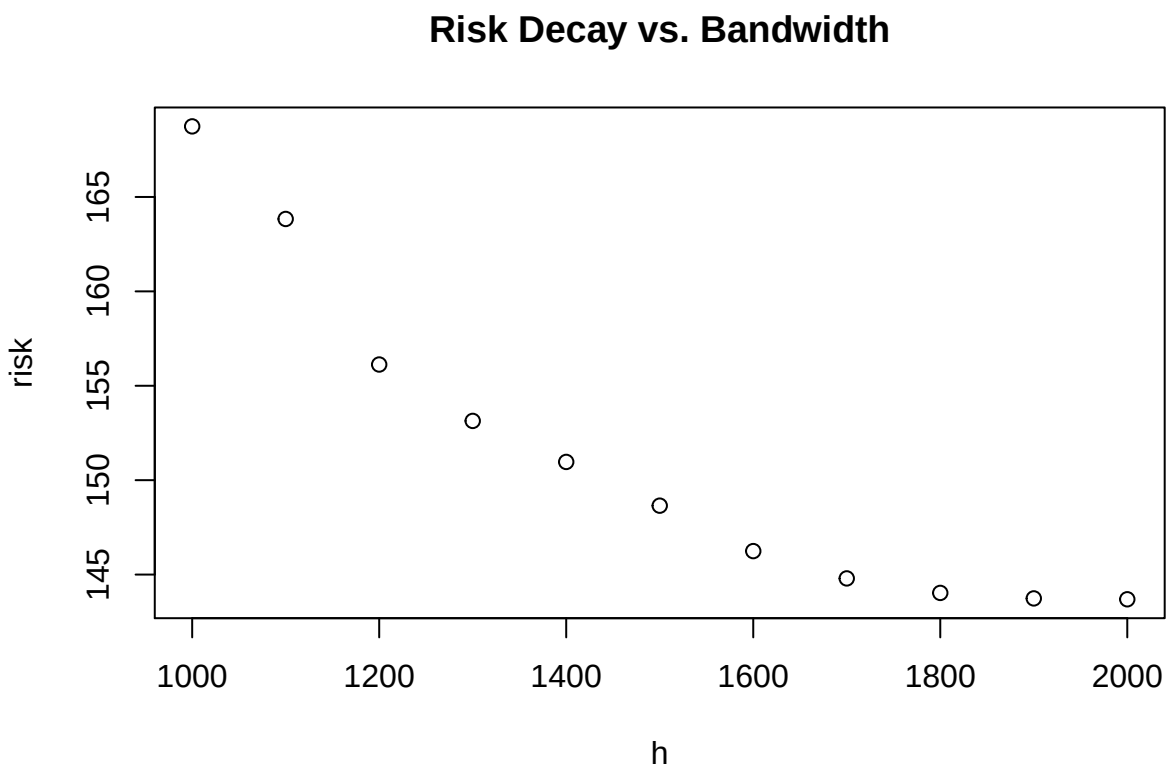



### Second-Degree Local Polynomial Regression on Premiums Data

Return to Non-Parametric vs. Parametric Analysis of Healthcare Premiums and Medicare Reimbursements

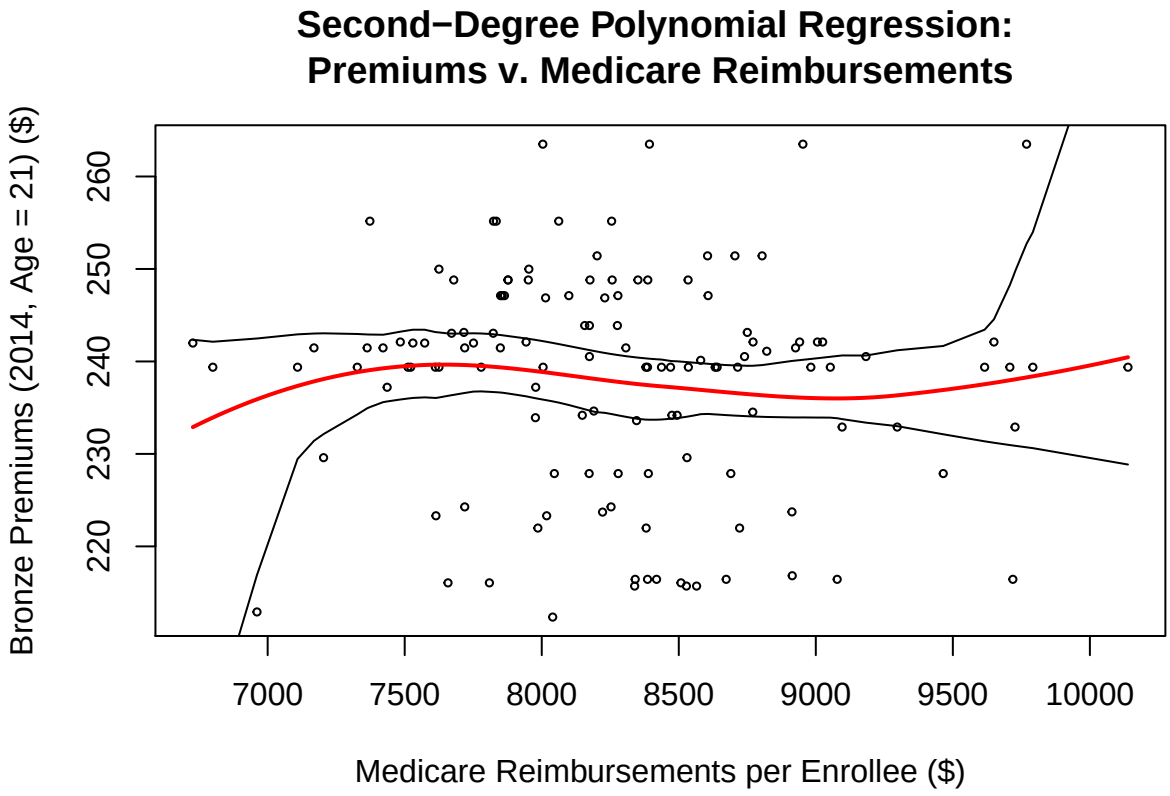
```
#use to find optimum bandwidth for locfit
optBandwidth <- function(y, x, h_test){
  risk <- rep(NA, length(h_test))
  n <- length(y)
  for(i in 1:length(h_test)){
    model <- locfit(y~x, alpha = c(0, h[i]), deg  = 2)
    Lii <- predict(model, where = "data", what = "infl")
    risk[i] <- (1/n) * sum(((y-fitted(model))/(1-Lii))^2)
  }
  return(risk)
}

dta <- na.omit(left_join(medicareVA, data.frame(va17age21Bronze), by = c("FIPS.County.Code" ="region")))
dta <- dta[order(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.),]

#Local Regression:
h <- seq(1000, 2000, 100)
```

```
risk <- optBandwidth(dta$value,
             dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
             h)

plot(h, risk, main = "Risk Decay vs. Bandwidth")
```




```
hopt <- h[which.min(risk)]
model <- locfit(dta$value~dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
                alpha = c(0, hopt), deg = 2)
#Creating Confidence Bands:
set.seed(1)
#B = number of iterations
B = 50
#Number of datapoints to sample:
n = 100
conf.mat <- matrix(nrow = length(fitted(model)), ncol = B)
for(i in 1:B){
  indx <- sample(1:length(fitted(model)), n, replace = TRUE)
  trainX <- dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.[indx]
  trainY <- dta$value[indx]
  risk <- optBandwidth(trainY,trainX, h)
  hopt <- h[which.min(risk)]
  tmpModel <- locfit(trainY~trainX, deg = 2, alpha = c(0, hopt))
  pred <- predict(tmpModel, newdata = dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2
  conf.mat[,i] <- pred
}
conf <- apply(conf.mat, 1, quantile, probs = c(0.025, 0.975))
upper <- conf[1,]
lower <- conf[2,]
```

```r
#Plotting:
{plot(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
      dta$value,
      xlab = "Medicare Reimbursements per Enrollee ($)",
      ylab = "Bronze Premiums (2014, Age = 21) ($)",
      main = "Second-Degree Polynomial Regression: \nPremiums v. Medicare Reimbursements"
      ,cex = 0.5
      )
lines(model, col = "red", lwd =2)
points(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014., upper, type = "l")
points(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014., lower, type =  "l")
}
```

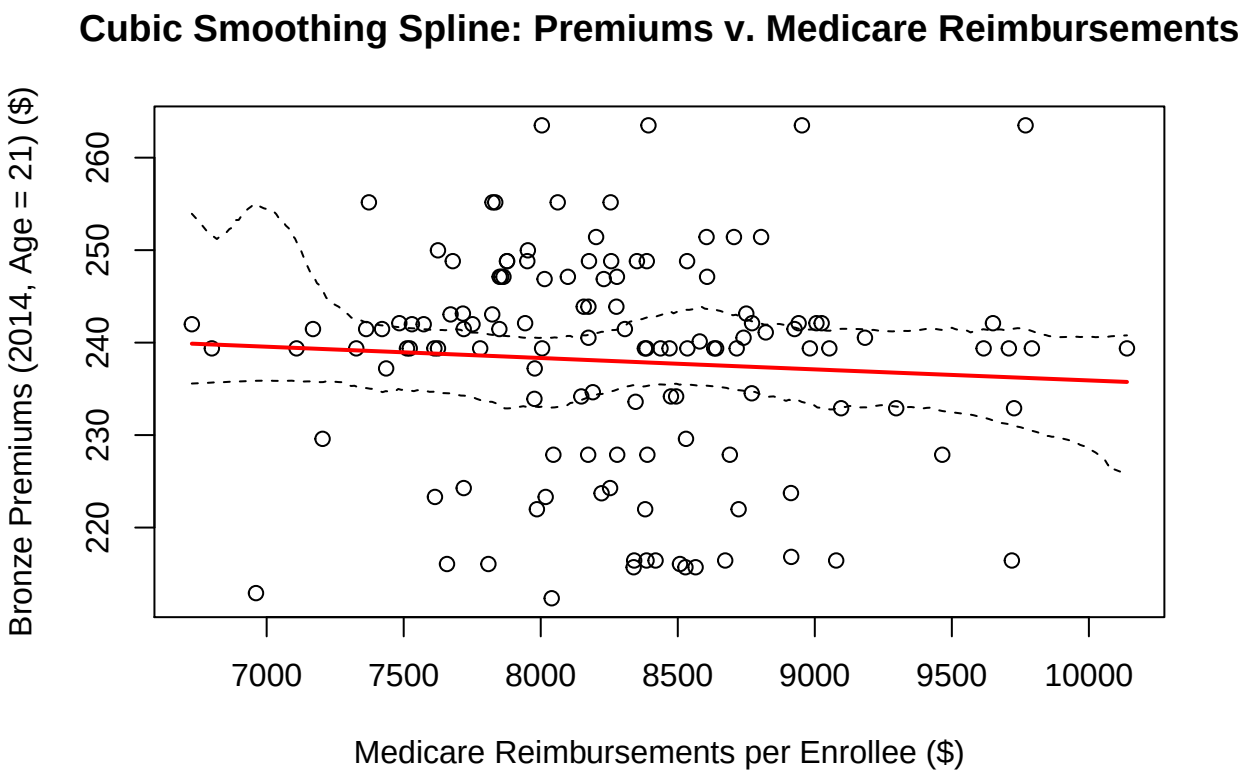



### Smoothing Spline Fit on Premiums Data

Return to Non-Parametric vs. Parametric Analysis of Healthcare Premiums and Medicare Reimbursements

```r
#Smoothing Spline:
model2 <- smooth.spline(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014., dta$valu

#Creating Confidence Bands:
#Source: CMU Data Analysis Course
#URL: http://www.stat.cmu.edu/~cshalizi/402/lectures/11-splines/lecture-11.pdf
resampler <- function(data) {
    n <- nrow(data)
    resample.rows <- sample(1:n,size=n,replace=TRUE)
```

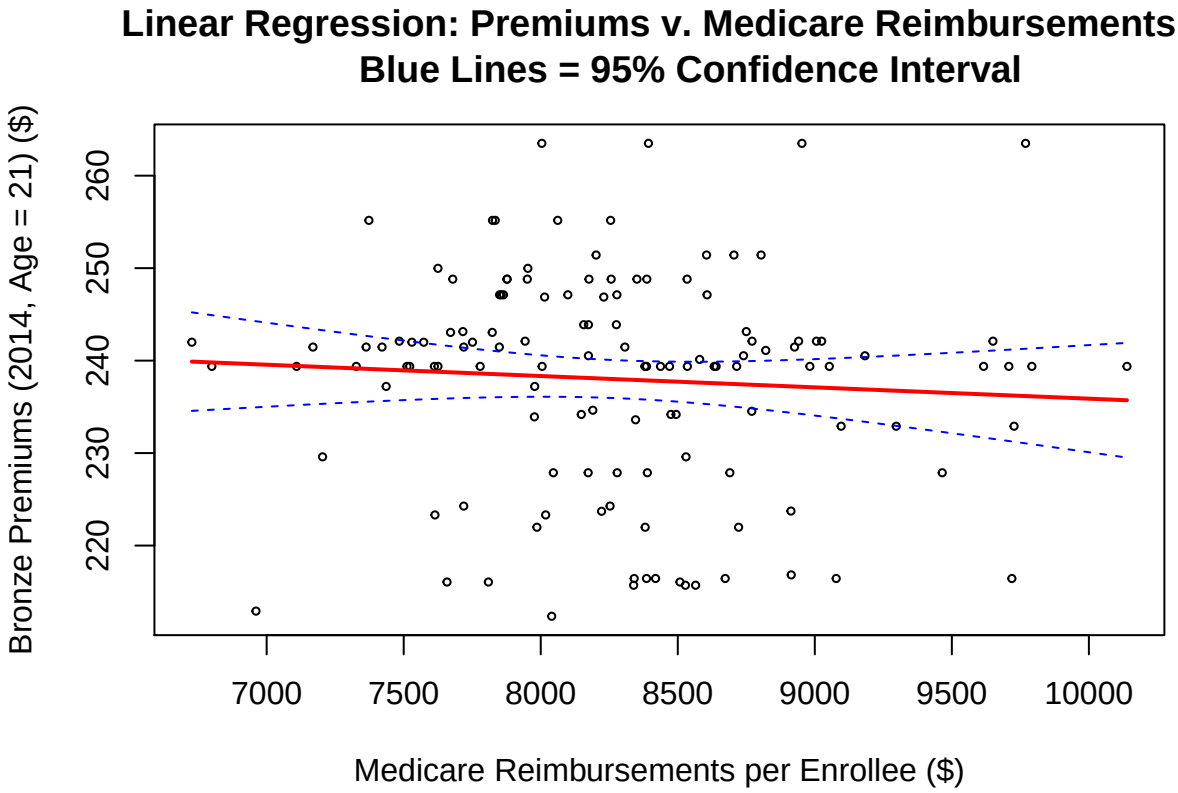
```
    return(data[resample.rows,])
}

spline.estimator <- function(data,m=300) {
    fit <- smooth.spline(x=data[,1],y=data[,2],cv=TRUE)
    eval.grid <- seq(from=min(data[,1]),to=max(data[,1]),length.out=m)
    return(predict(fit,x=eval.grid)$y) # We only want the predicted values
}

spline.cis <- function(data,B,alpha=0.05,m=300) {
    spline.main <- spline.estimator(data,m=m)
    spline.boots <- replicate(B,spline.estimator(resampler(data),m=m))
    cis.lower <- 2*spline.main - apply(spline.boots,1,quantile,probs=1-alpha/2)
    cis.upper <- 2*spline.main - apply(spline.boots,1,quantile,probs=alpha/2)
    return(list(main.curve=spline.main,lower.ci=cis.lower,upper.ci=cis.upper,
    x=seq(from=min(data[,1]),to=max(data[,1]),length.out=m)))
}

dataTmp <- matrix(nrow= dim(dta)[1], ncol = 2)
dataTmp[,1] <- dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.
dataTmp[,2] <- dta$value

#run and plot
sp.cis <- spline.cis(dataTmp, B=1000,alpha=0.05)

{plot(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
      dta$value,
      xlab = "Medicare Reimbursements per Enrollee ($)",
      ylab = "Bronze Premiums (2014, Age = 21) ($)",
      main = "Cubic Smoothing Spline: Premiums v. Medicare Reimbursements"
      )
lines(model2, col = "red", lwd =2)
lines(x=sp.cis$x,y=sp.cis$lower.ci, lty=2)
lines(x=sp.cis$x,y=sp.cis$upper.ci, lty=2)}
```




### Linear Regression on Premiums Data

Return to Non-Parametric vs. Parametric Analysis of Healthcare Premiums and Medicare Reimbursements

```
x <- dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.
model3<- lm(dta$value~x)

confidence <- predict(model3,
                      newdata=data.frame(x=x),
                      interval="confidence", level = 0.95)
{plot(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
      dta$value,
      xlab = "Medicare Reimbursements per Enrollee ($)",
      ylab = "Bronze Premiums (2014, Age = 21) ($)",
      main = "Linear Regression: Premiums v. Medicare Reimbursements,
      Blue Lines = 95% Confidence Interval",
      cex = 0.5
      )
lines(x = dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
      y = model3$fitted.values,col = "red", lwd =2)
lines(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
       confidence[,2], col = "blue", lty = "dashed")
lines(dta$Total.Medicare.reimbursements.per.enrollee..Parts.A.and.B...2014.,
       confidence[,3], col = "blue", lty = "dashed")
}
```




### Linear Regression on Premiums Data Model Summary

Return to Non-Parametric vs. Parametric Analysis of Healthcare Premiums and Medicare Reimbursements

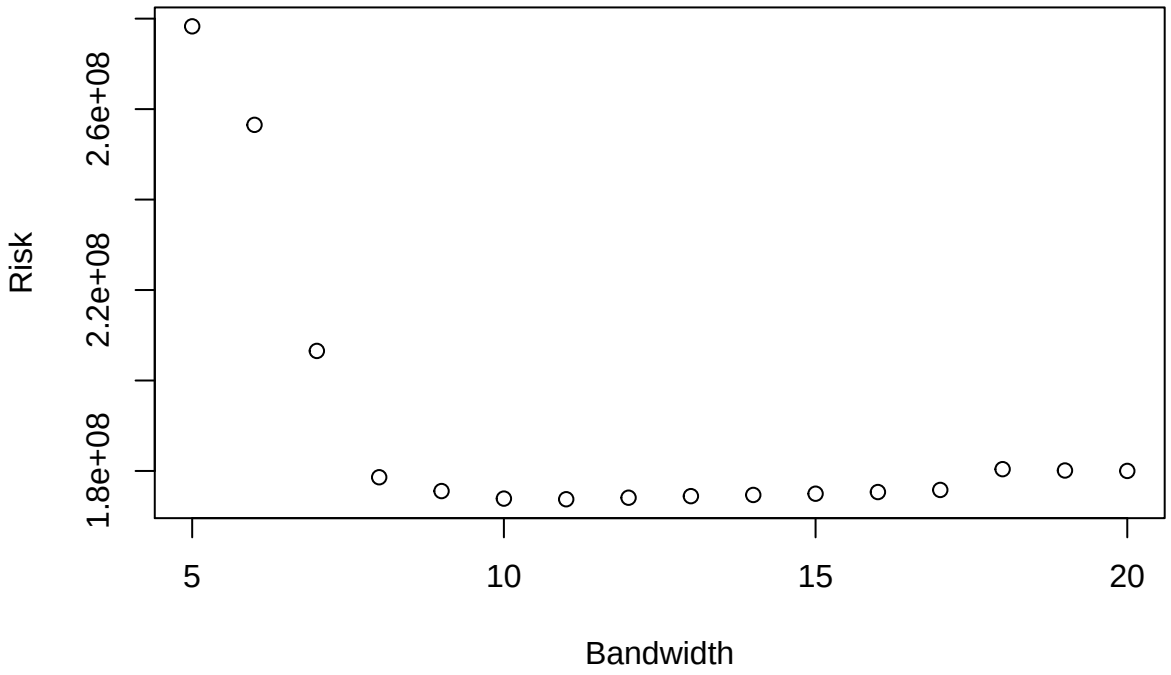
```
summary(model3)
```

```
##
## Call:
## lm(formula = dta$value ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.694  -4.444   2.333   5.888  27.328
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 248.126942  13.251702  18.724   <2e-16 ***
## x            -0.001224   0.001595  -0.768    0.444
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.79 on 127 degrees of freedom
## Multiple R-squared:  0.00462,    Adjusted R-squared:  -0.003218
## F-statistic: 0.5894 on 1 and 127 DF,  p-value: 0.4441
```

### Local Linear Regression & Linear Regression Model on Education & Income Data

Return to Non-Parametric vs. Parametric Analysis of Education and Income:

```
set.seed(1)
#Prepare Data:
vaIncome$value <- as.numeric(vaIncome$value)
vaEdu$hs <- as.numeric(vaEdu$hs)
vaTmp <- vaEdu
vaTmp$hs <- as.numeric(vaEdu$hs)
vaTmp$income <- as.numeric(vaIncome$value)
vaTmp <- vaTmp[rowSums(is.na(vaTmp)) == 0,]
vaTest <- vaTmp[order(vaTmp$hs),]

#use LOOCV to find optimum bandwidth for locfit
degree <- 1
h <- seq(5,20, 1)
n <- nrow(vaTest)
optBandwidth <- function(y, x, h_test){
  risk <- rep(NA, length(h_test))
  n <- length(y)
  for(i in 1:length(h_test)){
    model <- locfit(y~x, alpha = c(0, h[i]), deg  = degree)
    Lii <- predict(model, where = "data", what = "infl")
    risk[i] <- (1/n) * sum(((y-fitted(model))/(1-Lii))^2)
  }
  return(risk)
}
risk <- optBandwidth(vaTest$income, vaTest$hs, h)
plot(h, risk, main = "Risk vs. Bandwidth", xlab = "Bandwidth", ylab = "Risk")
```




```
opth <- h[which(min(risk) == risk)]
HSmodel <- locfit(vaTest$income~vaTest$hs, deg = degree, alpha = c(0, opth))
NPminMSE <- (1/n)*sum((vaTest$income-fitted(HSmodel))^2)
HSParametric <- lm(vaTest$income~vaTest$hs)
PminMSE <- (1/n)*sum((vaTest$income-fitted(HSParametric))^2)
NPminMSE
```

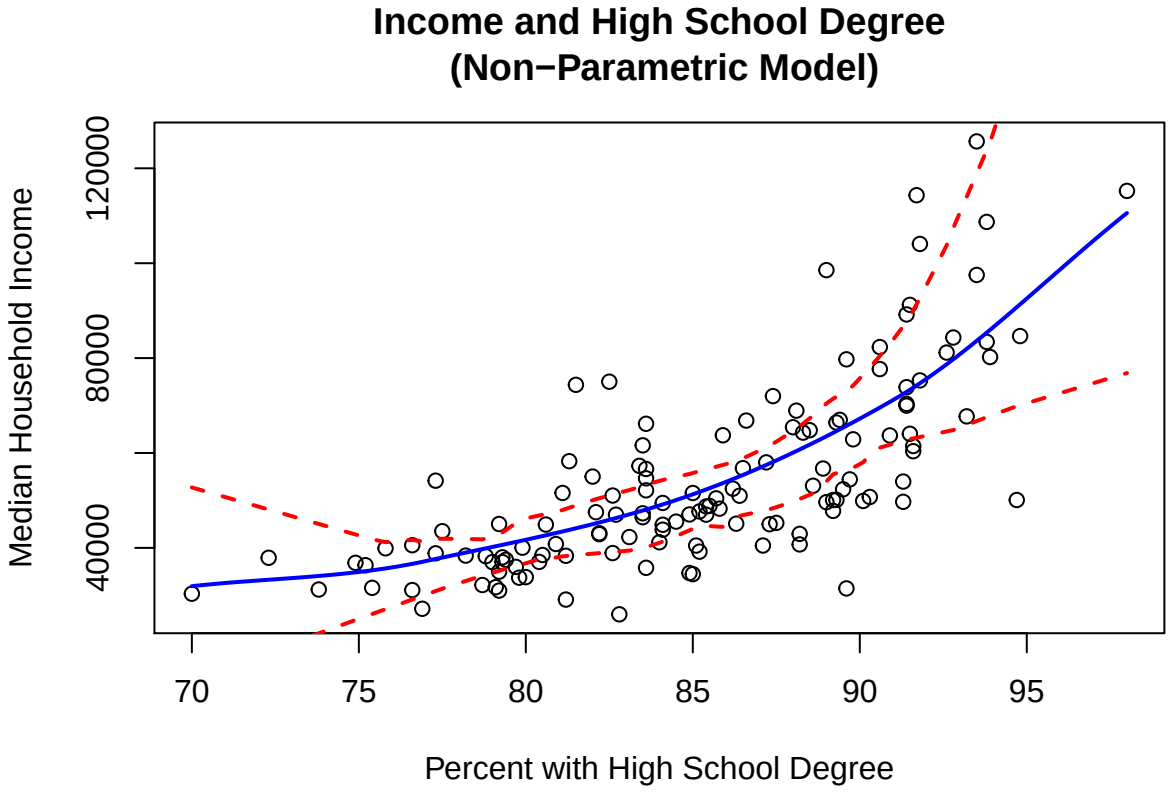
```
## [1] 162216830
```

```
PminMSE
```

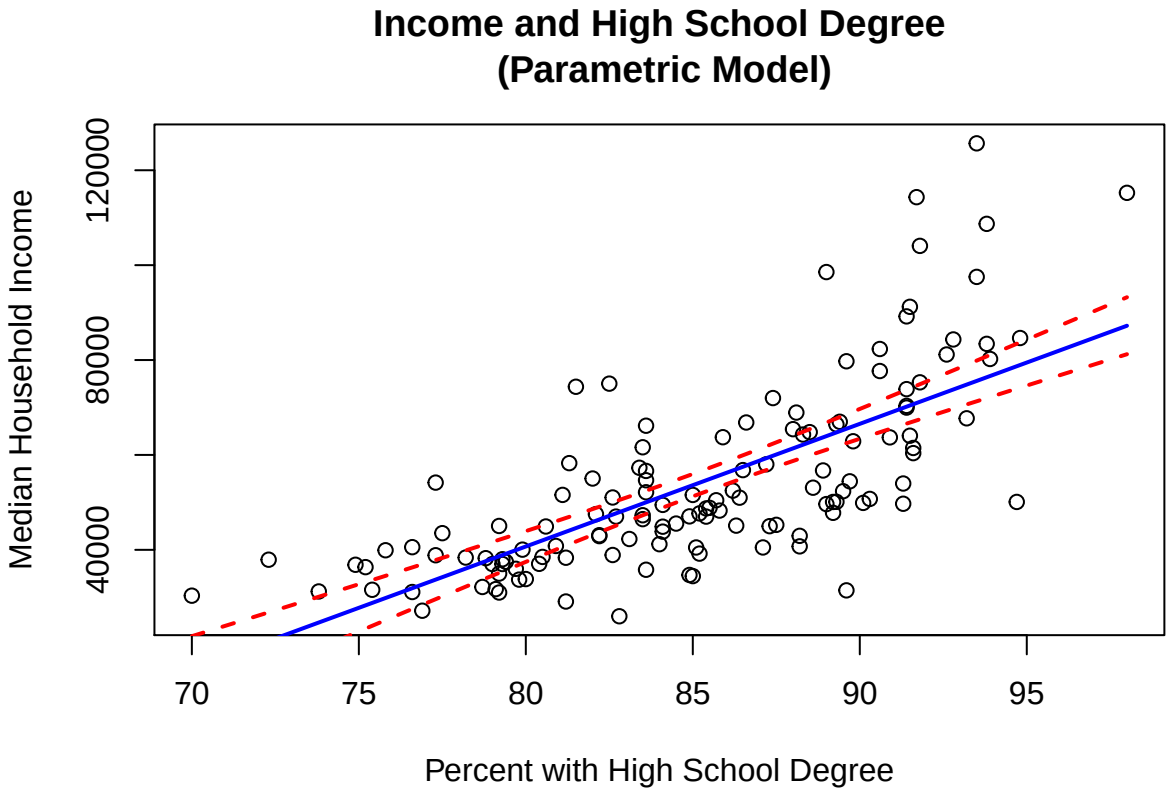
```
## [1] 185852083
```

```
#Creating Local Regression Confidence Bands:
    #B = number of iterations
B = 40
    #Number of datapoints to sample:
n = 30
conf.mat <- matrix(nrow = length(fitted(HSmodel)), ncol = B)
for(i in 1:B){
  indx <- sample(1:length(fitted(model)), n, replace = TRUE)
  trainX <- vaTest$hs[indx]
  trainY <- vaTest$income[indx]
  risk <- optBandwidth(trainY,trainX, h)
  hopt <- h[which.min(risk)]
  tmpModel <- locfit(trainY~trainX, deg = degree, alpha = c(0, hopt))
  pred <- predict(tmpModel, newdata = vaTest$hs)
  conf.mat[,i] <- pred
}
conf <- apply(conf.mat, 1, quantile, probs = c(0.025, 0.975))
upper <- conf[1,]
lower <- conf[2,]
```

```r
#Creating Linear Regression Confidence Bands:
confidence <- predict(HSParametric,
                      newdata=data.frame(x=vaTest$hs),
                      interval="confidence", level = 0.95)

#Plotting Non-Parametric Model:
{plot(vaTest$hs, vaTest$income, ylab = "Median Household Income", xlab  = "Percent with High School Degr
lines(HSmodel, col = "blue", lwd = "2")
lines(vaTest$hs, lower, lty = "dashed", col = "red", lwd = 2)
lines(vaTest$hs, upper, lty = "dashed", col = "red", lwd = 2)
}
```



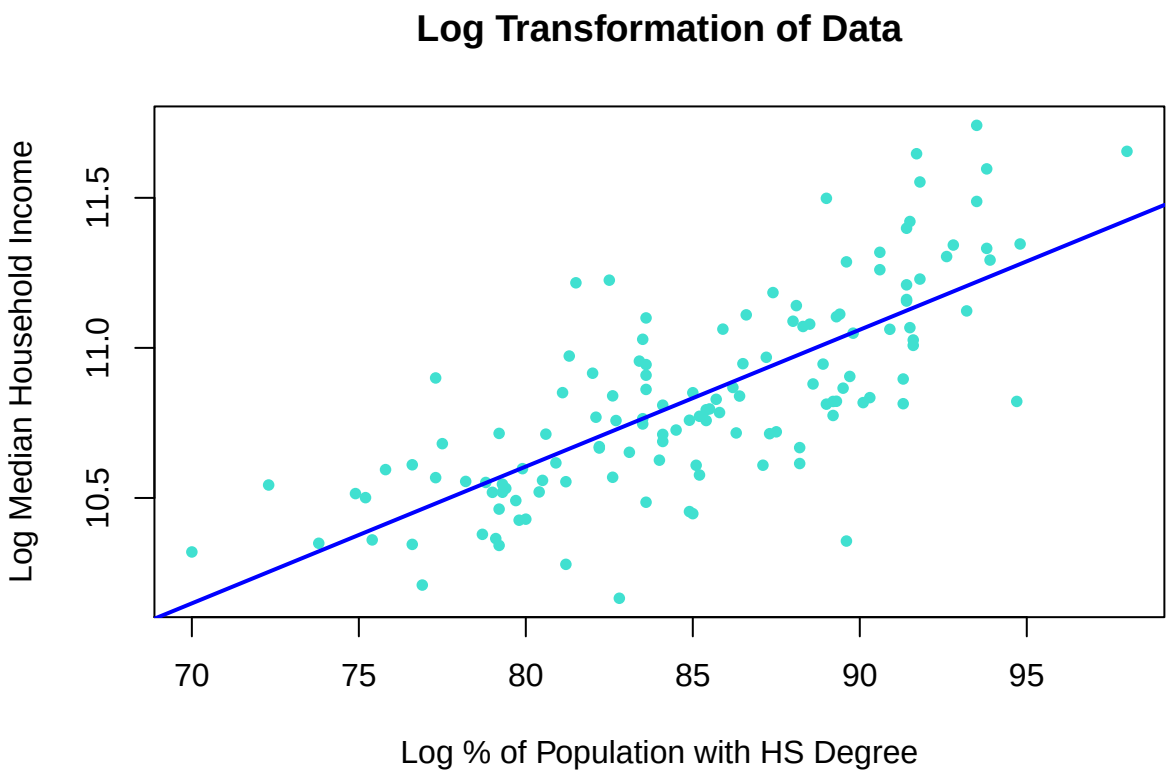

```r
#Plotting Parametric Model:
{
plot(vaTest$hs, vaTest$income, ylab = "Median Household Income", xlab  = "Percent with High School Degre
lines(vaTest$hs, HSParametric$fitted.values, col = "blue", lwd = 2)
lines(vaTest$hs, confidence[,2], lty = "dashed", col = "red", lwd = 2)
lines(vaTest$hs, confidence[,3], lty = "dashed", col = "red", lwd = 2)

}
```




### Exponential Model of Income & Education Data

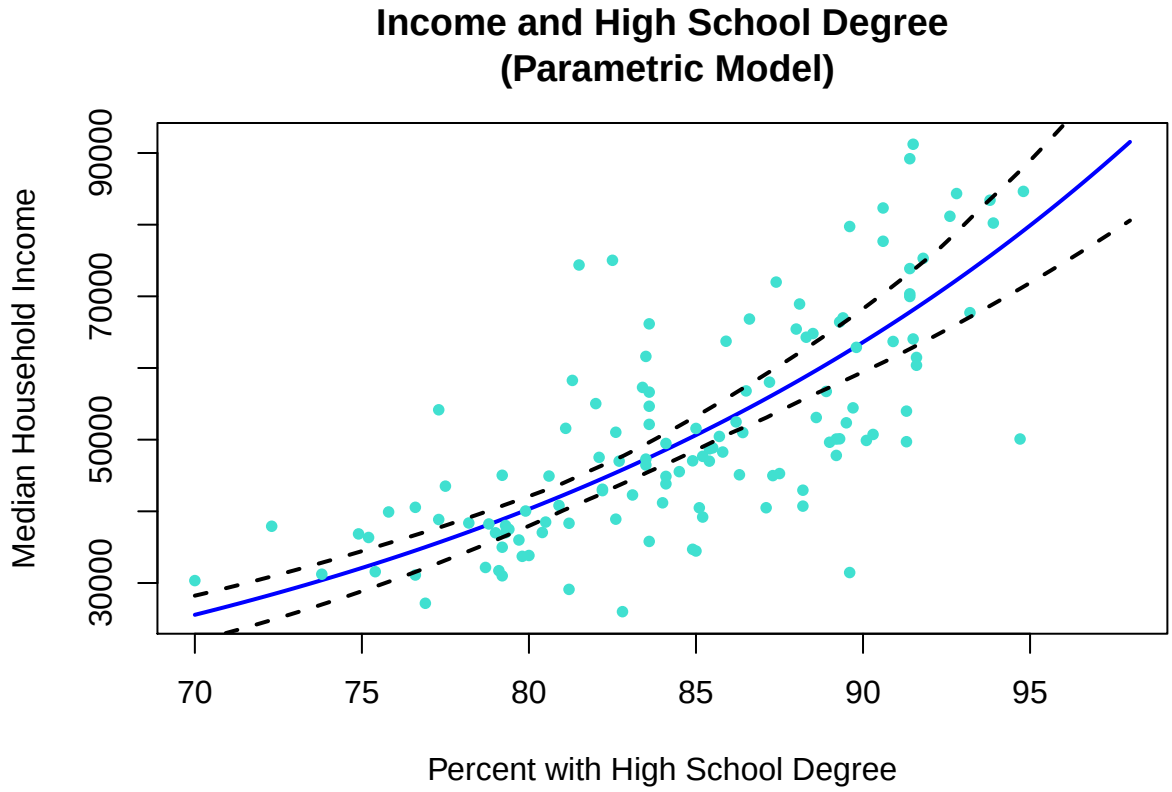
Return to Non-Parametric vs. Parametric Analysis of Education and Income:

```
#take log transformation of data
#logX <- log(vaTest$hs, base = exp(1))
logX <- vaTest$hs
logY <- log(vaTest$income, base = exp(1))
#Fit linear model to log transformation
logModel <- lm(logY~logX)

{plot(logX, logY, main = "Log Transformation of Data", xlab =
       "Log % of Population with HS Degree", ylab = "Log Median Household Income", col = "turquoise", p
  abline(logModel, col = "blue", lwd = 2)}
```




```r
#Re-transform log model into exponential model:
expModel <- function(x){
  fit <- exp(logModel$coefficients[1] + logModel$coefficients[2]*x)
  return(fit)
}

x <- seq(min(vaTest$hs), max(vaTest$hs), by = 0.03)

#Creating Exponential Model Confidence Bands:
    #B = number of iterations
B = 50
    #Number of datapoints to sample:
n = 100
conf.mat <- matrix(nrow = length(x), ncol = B)
error <- rep(NA, B)
for(i in 1:B){
  indx <- sample(1:length(fitted(model)), n, replace = TRUE)
  trainX <- vaTest$hs[indx]
  trainY <- log(vaTest$income[indx])
  tmpModel <- lm(trainY~trainX)
  pred <- exp(tmpModel$coefficients[1] + tmpModel$coefficients[2]*x)
  conf.mat[,i] <- pred
  msePred <- exp(tmpModel$coefficients[1] + tmpModel$coefficients[2]*vaTest$hs)
  error[i] <- (1/length(msePred))*sum((msePred - vaTest$income)^2)
}

#expMSE <- mean(error)
expMSE <- (1/length(logX))*sum((vaTest$income-expModel(vaTest$hs))^2)
conf <- apply(conf.mat, 1, quantile, probs = c(0.025, 0.975))
```

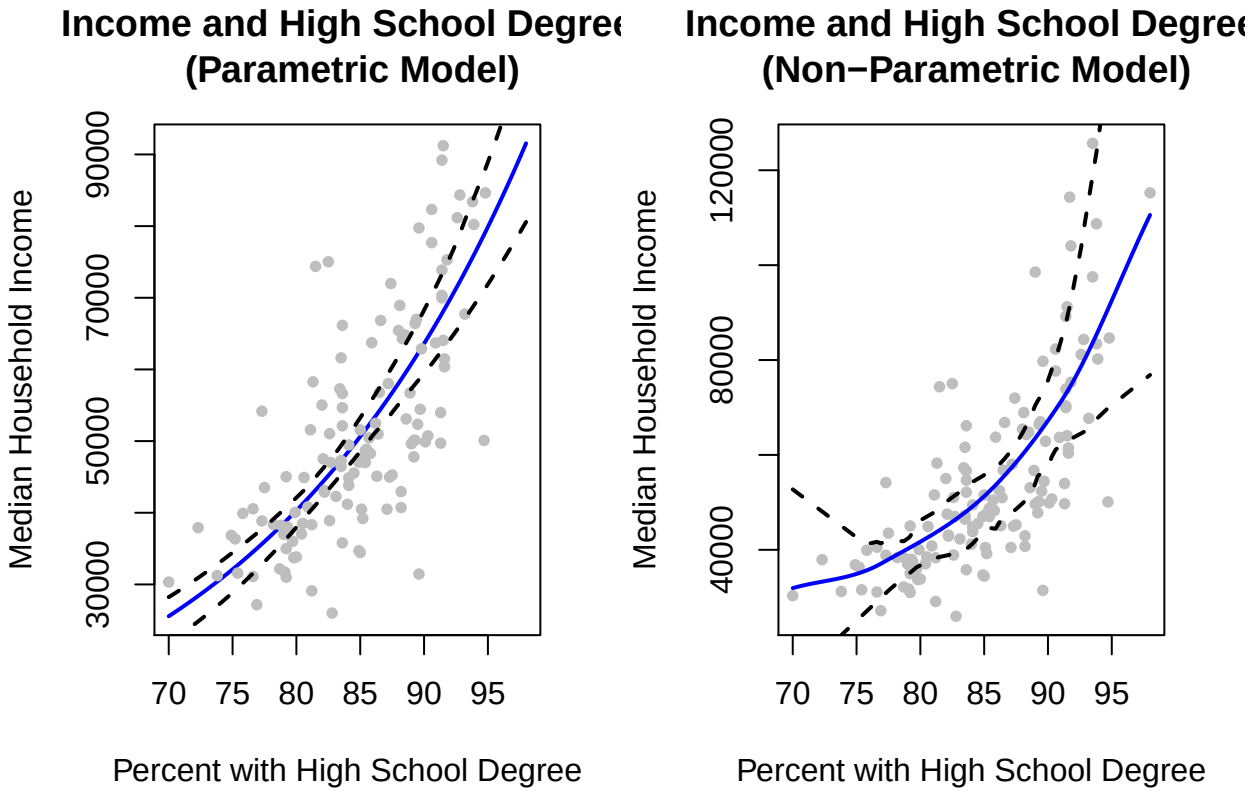
```
upper1 <- conf[1,]
lower1 <- conf[2,]


#Plot Exponential Model:
{plot(x, expModel(x), ylab = "Median Household Income", xlab  = "Percent with High School Degree", main
  points(vaEdu$hs, vaIncome$value, col = "turquoise", pch = 16, cex =0.8)
  lines(x, lower1, lwd = 2, lty = "dashed")
  lines(x, upper1, lwd = 2, lty = "dashed")}
```




### Parametric vs. Non-Parametric Side-by-Side Plot

Return to Non-Parametric vs. Parametric Analysis of Education and Income:

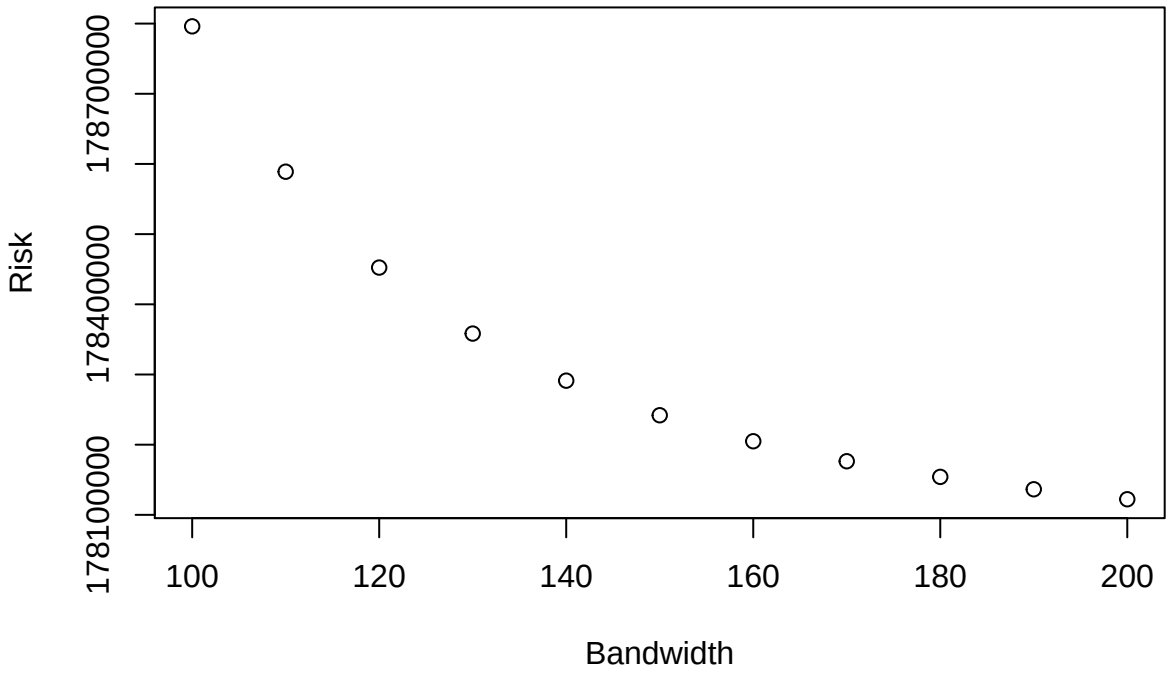
```
#Plot Local Linear and Exponential Model:
{par(mfrow=c(1,2))
  {plot(x, expModel(x), ylab = "Median Household Income", xlab  = "Percent with High School Degree", ma
  points(vaEdu$hs, vaIncome$value, col = "grey", pch = 16, cex =0.8)
  lines(x, lower1, lwd = 2, lty = "dashed")
  lines(x, upper1, lwd = 2, lty = "dashed")}
  plot(vaTest$hs, vaTest$income, ylab = "Median Household Income", xlab  = "Percent with High School De
lines(HSmodel, col = "blue", lwd = "2")
lines(vaTest$hs, lower, lty = "dashed", lwd = 2)
lines(vaTest$hs, upper, lty = "dashed", lwd = 2)
}
```




### Parametric and Non-Parametric Regression on Bachelor's Degree Data

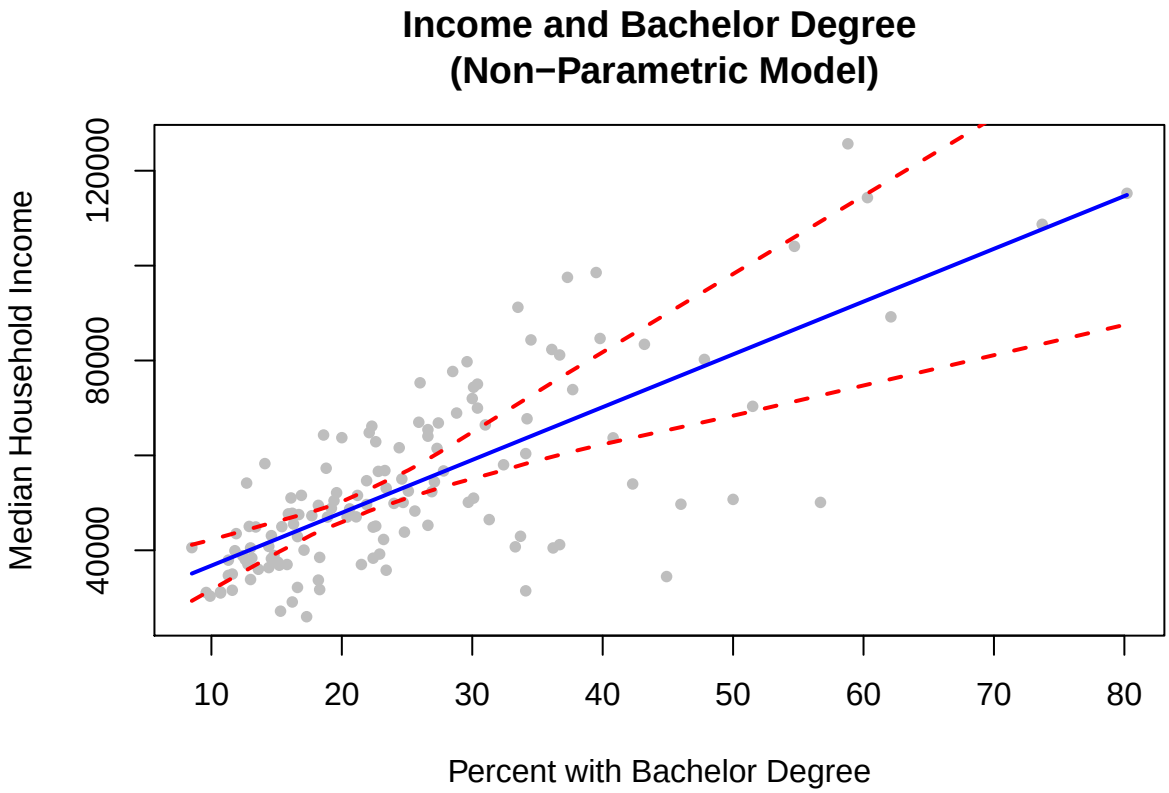
Return to Non-Parametric vs. Parametric Analysis of Education and Income:

```
vaTest2 <- vaTest[order(vaTest$bachelor),]
set.seed(1)
degree <- 1

#use LOOCV to find optimum bandwidth for locfit
h <- seq(100,200, 10)
n <- nrow(vaTest2)
optBandwidth <- function(y, x, h_test){
  risk <- rep(NA, length(h_test))
  n <- length(y)
  for(i in 1:length(h_test)){
    model <- locfit(y~x, alpha = c(0, h[i]), deg  = degree)
    Lii <- predict(model, where = "data", what = "infl")
    risk[i] <- (1/n) * sum(((y-fitted(model))/(1-Lii))^2)
  }
  return(risk)
}

risk <- optBandwidth(vaTest2$income, vaTest2$bachelor, h)
plot(h, risk, main = "Risk vs. Bandwidth", xlab = "Bandwidth", ylab = "Risk")
```

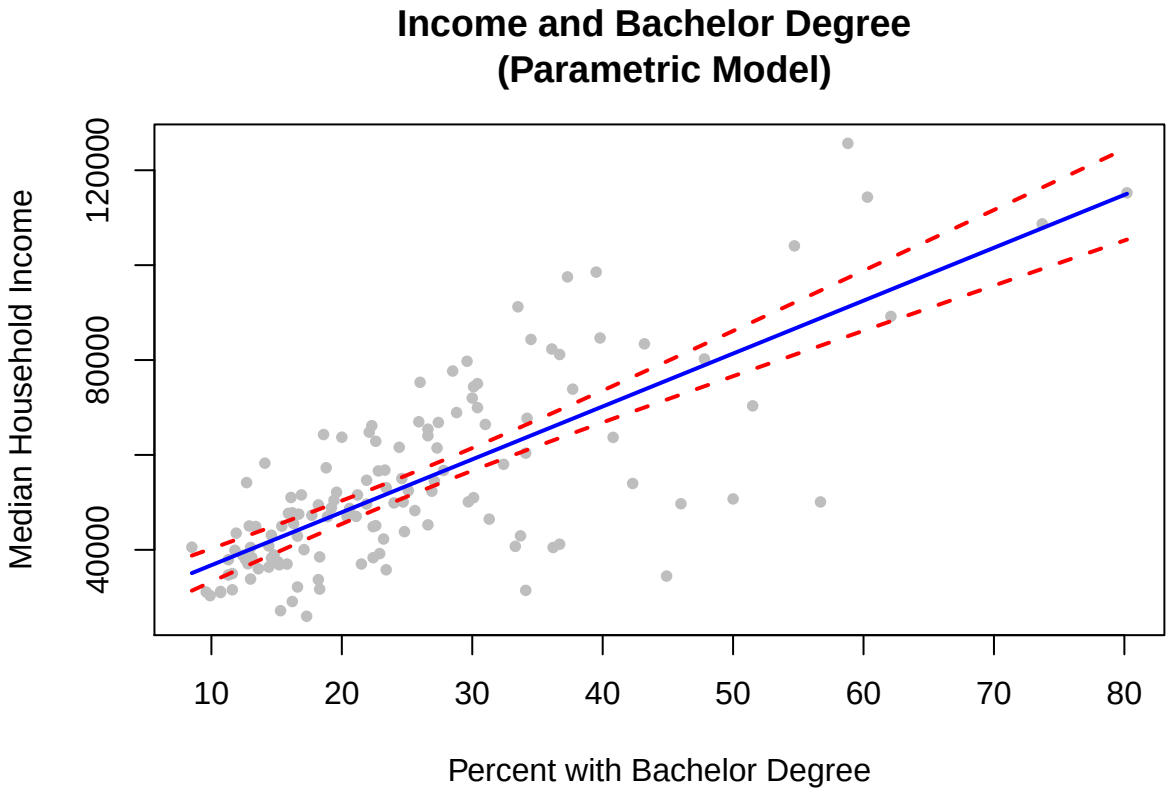



```
opth <- h[which(min(risk) == risk)]
Bachmodel <- locfit(vaTest2$income~vaTest2$bachelor, alpha = c(0, opth), deg = degree)
BachParametric <- lm(vaTest2$income~vaTest2$bachelor)

#Creating Local Regression Confidence Bands:
    #B = number of iterations
B = 60
    #Number of datapoints to sample:
n = 30
conf.mat <- matrix(nrow = nrow(vaTest2), ncol = B)
for(i in 1:B){
  indx <- sample(1:nrow(vaTest2), n, replace = FALSE)
  trainX <- vaTest2$bachelor[indx]
  trainY <- vaTest2$income[indx]
  risk <- optBandwidth(trainY,trainX, h)
  hopt <- h[which.min(risk)]
  tmpModel <- locfit(trainY~trainX, deg = degree, alpha = c(0, hopt))
  pred <- predict(tmpModel, newdata = vaTest2$bachelor)
  conf.mat[,i] <- pred
}
conf <- apply(conf.mat, 1, quantile, probs = c(0.025, 0.975))
upper <- conf[1,]
lower <- conf[2,]

#Creating Linear Regression Confidence Bands:
confidence <- predict(BachParametric,
                      newdata=data.frame(x=vaTest2$bachelor),
                      interval="confidence", level = 0.95)

{plot(vaTest2$bachelor, vaTest2$income, ylab = "Median Household Income", xlab  = "Percent with Bachelo
```

```
lines(Bachmodel, col = "blue", lwd = 2)
lines(vaTest2$bachelor, lower, lty = "dashed", col = "red", lwd = 2)
lines(vaTest2$bachelor, upper, lty = "dashed", col = "red", lwd = 2)
}
```




```
{plot(vaTest2$bachelor, vaTest2$income, ylab = "Median Household Income", xlab  = "Percent with Bachelo
lines(vaTest2$bachelor, BachParametric$fitted.values, col = "blue", lwd = 2)
lines(vaTest2$bachelor, confidence[,2], col = "red", lty = "dashed", lwd = 2)
lines(vaTest2$bachelor, confidence[,3], col = "red", lty = "dashed", lwd = 2)}
```




### Combining Data for GAM Modelling

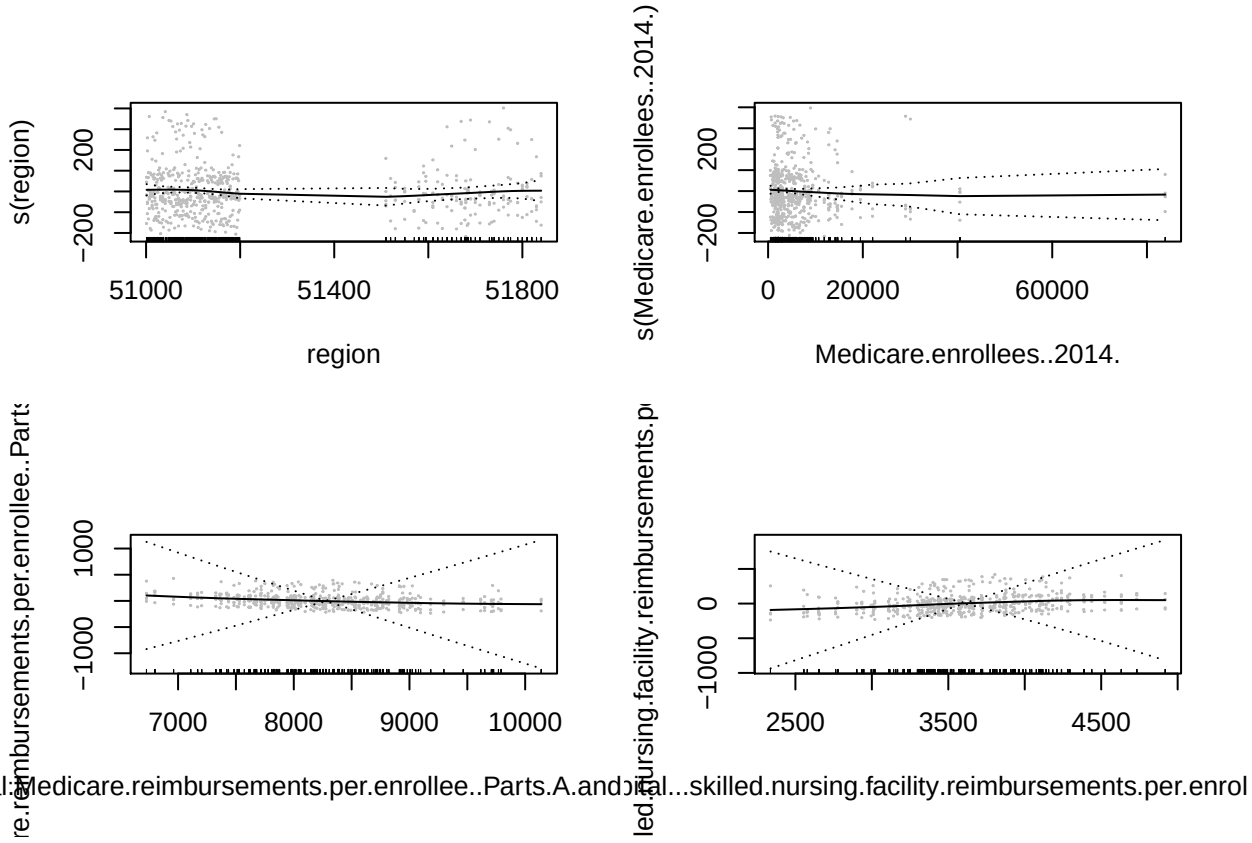
Return to Building Spline GAM Model

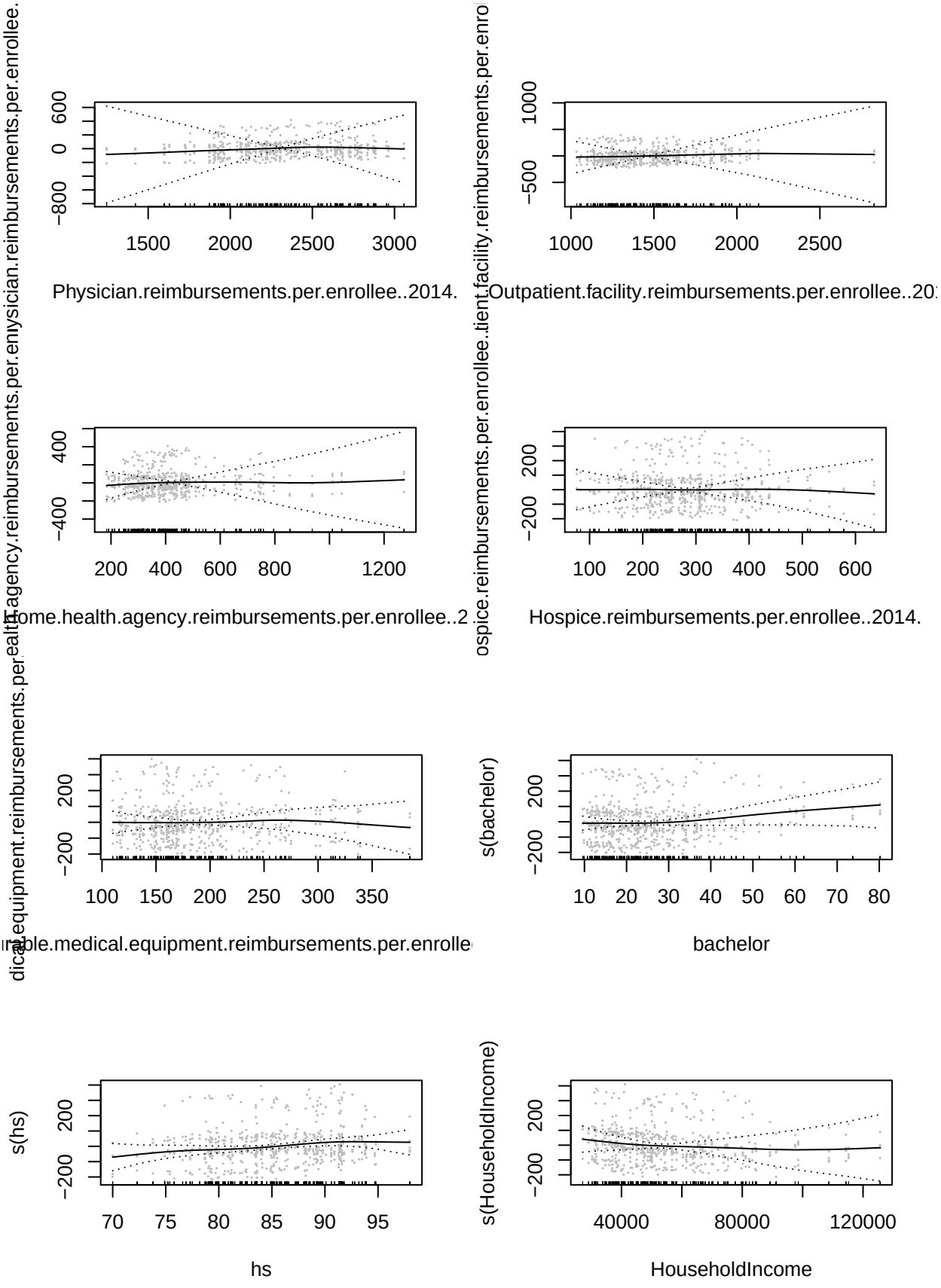
```
va18age21Bronze <- premiumsDeductible("Bronze", "Premium Adult Individual Age 21",
                                      "Medical Deductible - Individual - Standard", va18)
va17age21Bronze <- premiumsDeductible("Bronze", "Premium Adult Individual Age 21",
                                      "Medical Deductible - Individual - Standard", va17)
va16age21Bronze <- premiumsDeductible("Bronze", "Premium Adult Individual Age 21",
                                      "Medical Deductible - Individual - Standard", va16)
va15age21Bronze <- premiumsDeductible("Bronze", "Premium Adult Individual Age 21",
                                      "Medical Deductible-individual-standard", va15)
va14age21Bronze <- premiumsDeductible("Bronze", "Premium Adult Individual Age 21",
                                      "Medical Deductible-individual-standard", va14)

#Merging premium and deductible data:
x <- Reduce(function(...) merge(..., all=TRUE), list(va18age21Bronze,va17age21Bronze,va16age21Bronze,va
names(x) <- c("region","premiums", "deductible")

#Adding in Medicare Data:
z <- inner_join(x, medicareVA,by = c("region" = "FIPS.County.Code"))

#Adding in Education Data:
a <- inner_join(z, vaEdu, by = c("region" = "region"))

#Adding in Income Data:
b <- inner_join(a, incomeVA, by = c("region" = "region"))
ACAdf <- b[rowSums(is.na(b)) == 0,]
```

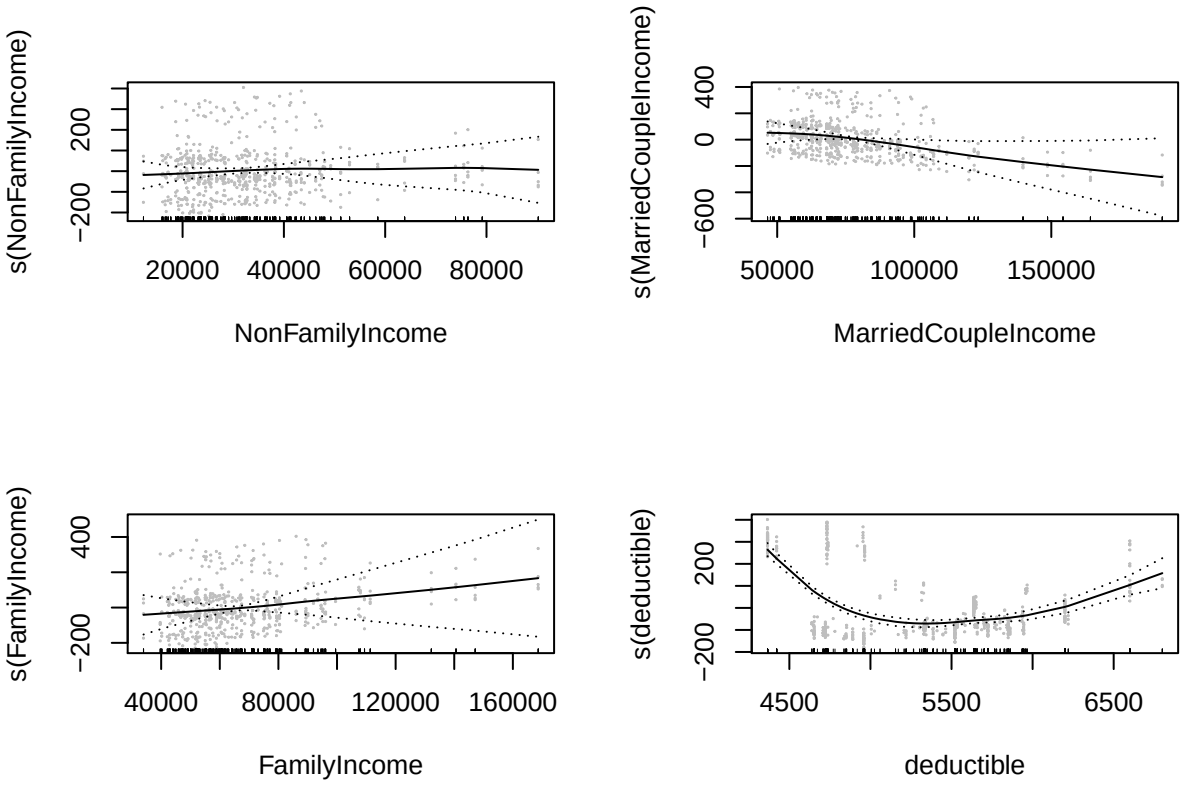
## Spline Smoothing GAM Model

Return to Building Spline GAM Model

```
model <- gam(premiums~s(region) + s(Medicare.enrollees..2014.) + s(Total.Medicare.reimbursements.per.en
par(mfrow = c(2,2))
plot.Gam(model, residuals = TRUE, se = TRUE, cex = 0.1, col = "grey")
```



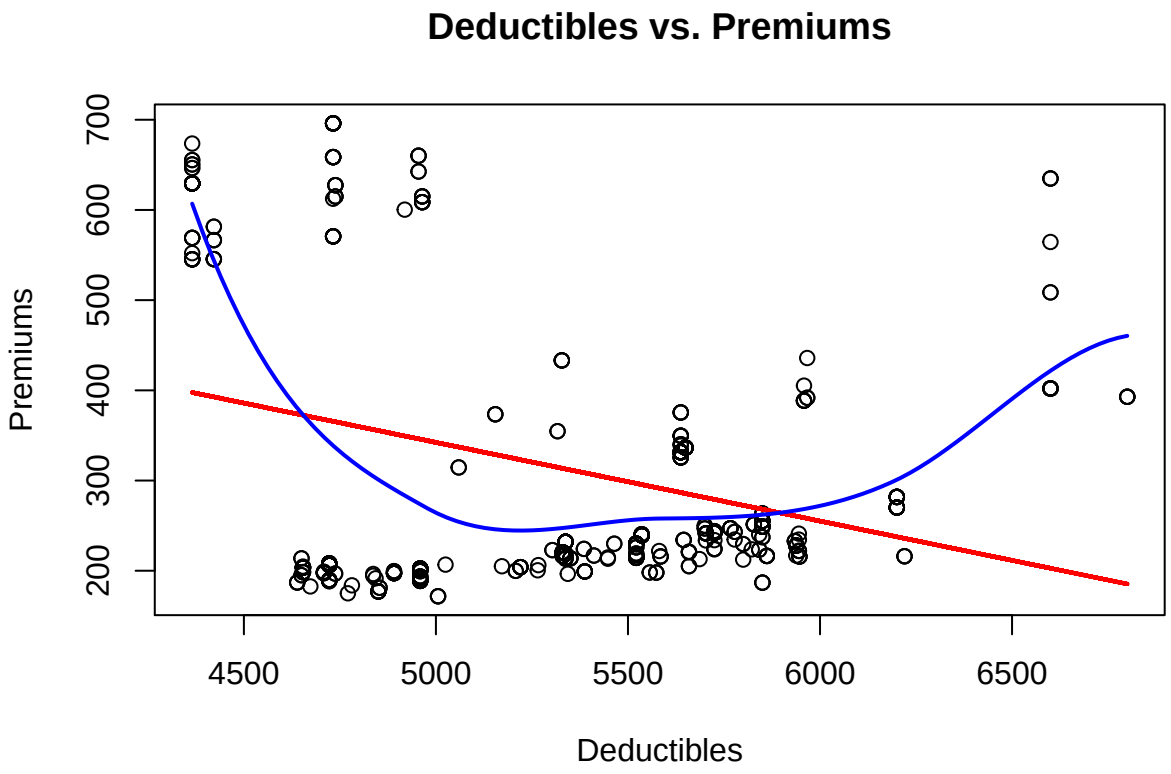

Physician.reimbursements.per.enrollee..2014.
Outpatient.facility.reimbursements.per.enrollee..20
Home.health.agency.reimbursements.per.enrollee..2
Hospice.reimbursements.per.enrollee..2014.
rable.medical.equipment.reimbursements.per.enrolle
bachelor
s(bachelor)
s(hs)
hs
s(HouseholdIncome)
HouseholdIncome




### Insurance Deductible Non-Parametric vs. Parametric Fit

Return to Building Spline GAM Model

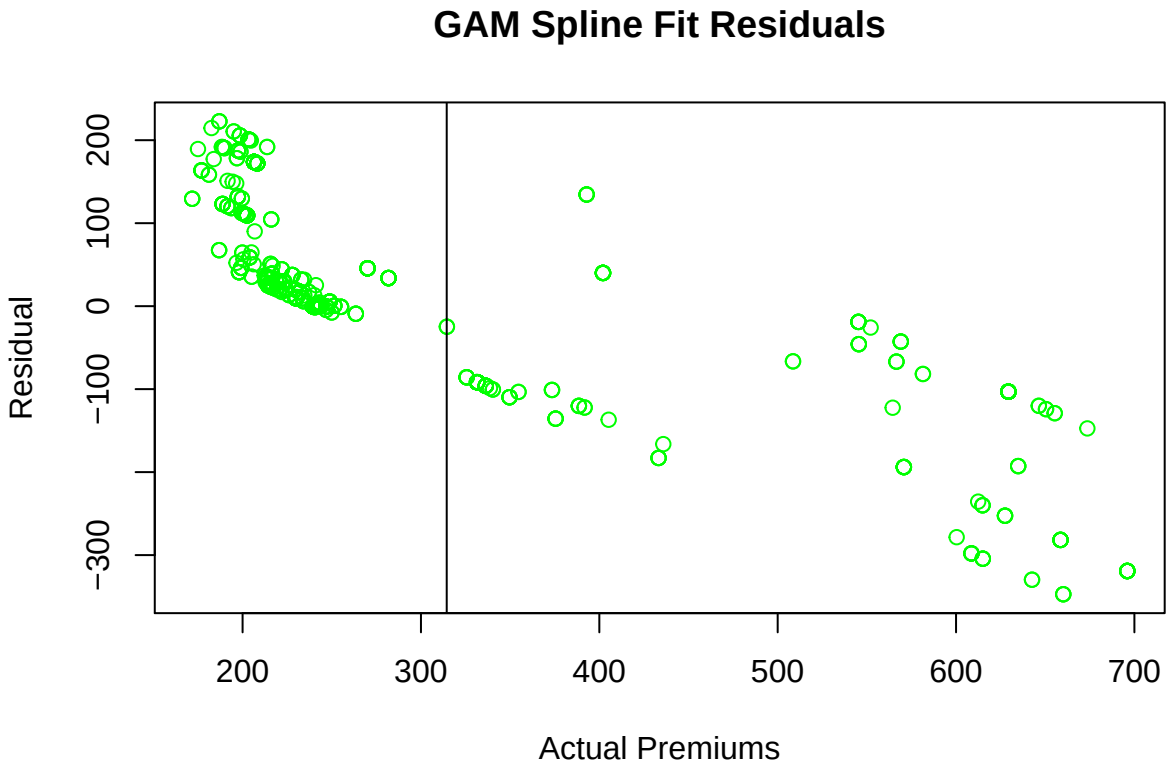
```
set.seed(1)
h <- seq(1000,5000, by = 50)
risk <- rep(NA, length(h))
for(i in 1:length(h)){
  model <- locfit(ACAdf$premiums~ACAdf$deductible, alpha = c(0, h[i]))
  Lii <- predict(model, where = "data", what = "infl")
  risk[i] <- (1/nrow(ACAdf)) * sum(((ACAdf$premiums-fitted(model))/(1-Lii))^2)
}
opth <- h[which(min(risk) == risk)]
linear <- lm(premiums~deductible, data = ACAdf)
np <- locfit(premiums~deductible, data = ACAdf, alpha = c(0, opth))
{plot(ACAdf$deductible, ACAdf$premiums, main = "Deductibles vs. Premiums", xlab = "Deductibles", ylab =
lines(ACAdf$deductible,linear$fitted.values, col = "red", lwd = 2)
lines(np, col = "blue", lwd = 2)
}
```




### Smoothing Spline GAM Residuals Plot

Return to Building Spline GAM Model

```
{plot(ACAdf$premiums, fitted(model) - ACAdf$premiums, main = "GAM Spline Fit Residuals",
     xlab = "Actual Premiums", ylab = "Residual", col = "green")
  abline(v = mean(ACAdf$premiums))
}
```




### K-Fold CV of Smoothing Spline GAM

Return to Building Spline GAM Model

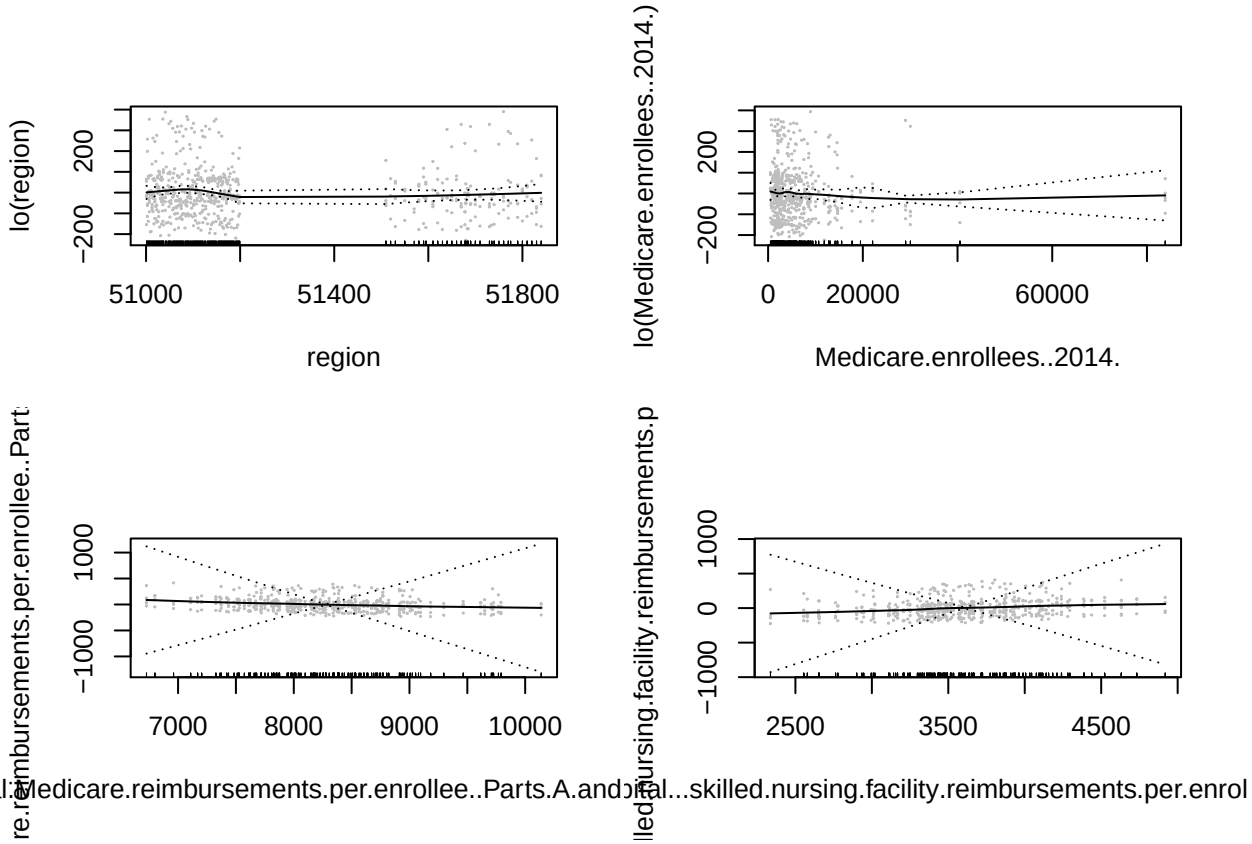
```
set.seed(1)
#n = sample size
#k = number of k-fold validations to run
kFoldSpline <- function(k, n, data) {
  mseVec <- rep(NA, k)
  for(i in 1:k){
    indx <- sample(1:nrow(data), n)
    test <- data[indx,]
    train <- data[-indx,]
    modelTMP <- gam(premiums~s(region) + s(Medicare.enrollees..2014.) + s(Total.Medicare.reimbursements
    predictions <- predict.Gam(modelTMP, newdata = test)
    mseVec[i] <- (1/n)*(sum(predictions - test$premiums)^2)
  }
  return(mean(mseVec))
}

splineMSE <- kFoldSpline(20, 100, ACAdf)
splineMSE
```

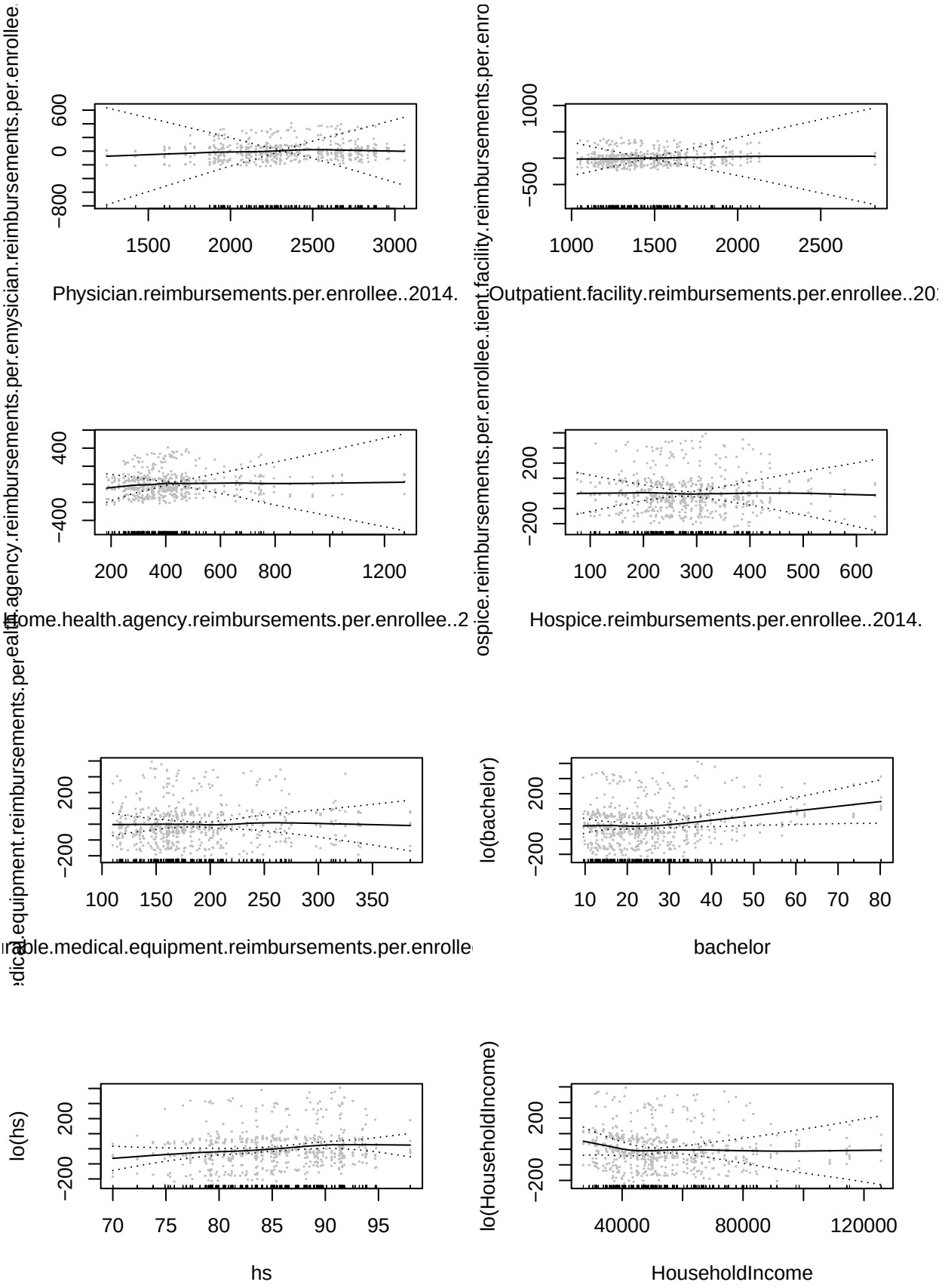
```
## [1] 10525.24
```

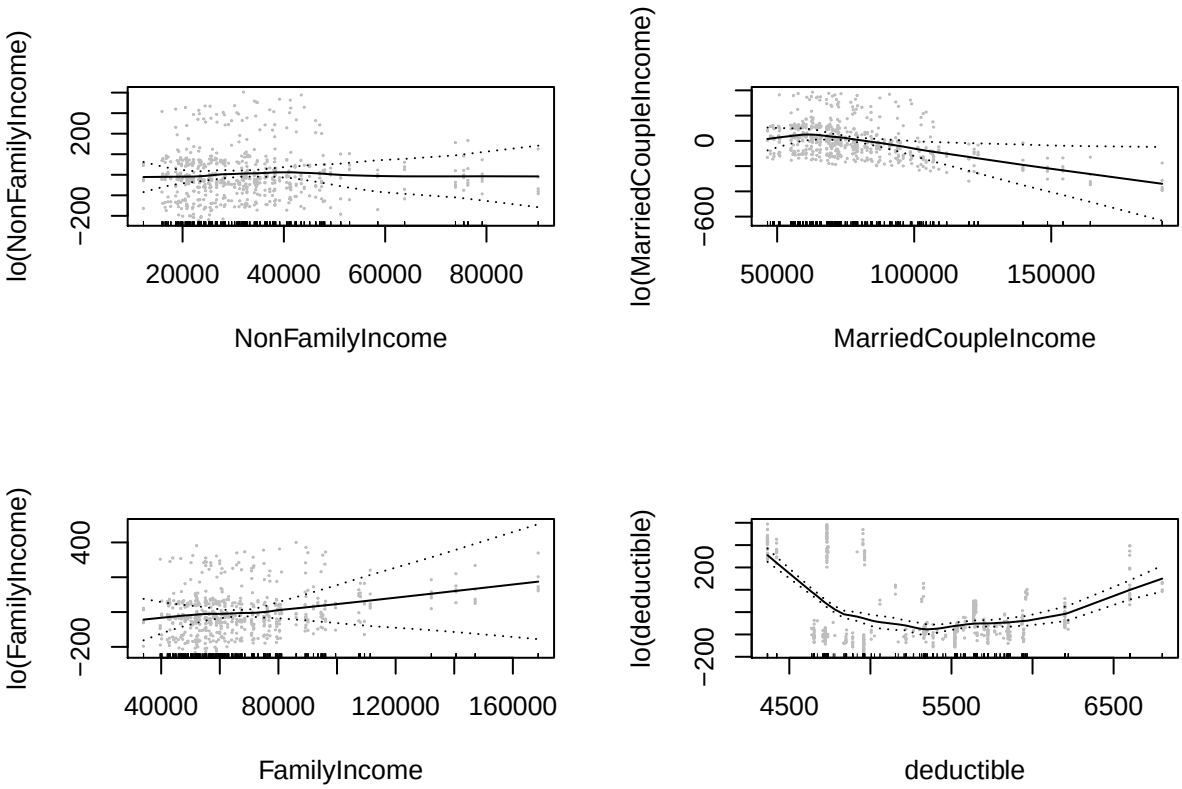
### GAM Local Regression Model

Return to Building GAM Local Regression

```r
modelLocal <- gam(premiums~lo(region) + lo(Medicare.enrollees..2014.) + lo(Total.Medicare.reimbursements

par(mfrow = c(2,2))
plot.Gam(modelLocal, residuals = TRUE, se = TRUE, cex = 0.1, col = "grey")
```



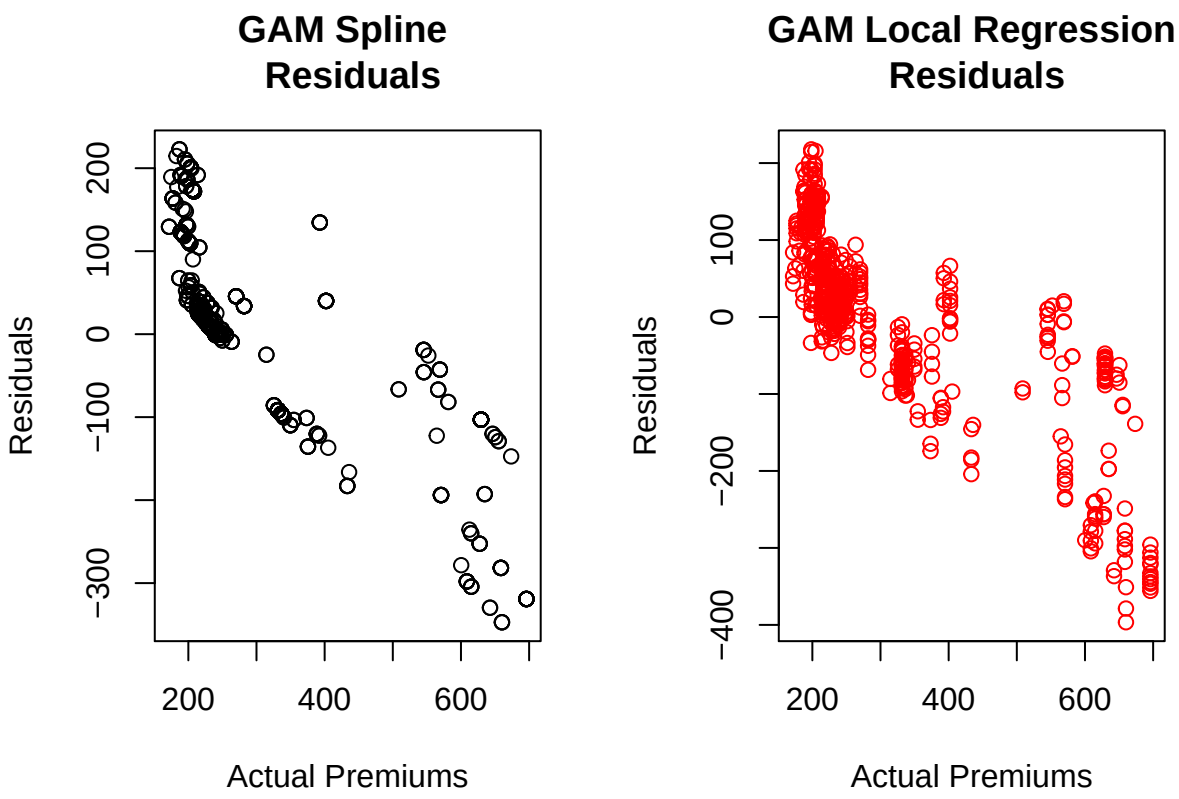

Physician.reimbursements.per.enrollee..2014.
Outpatient.facility.reimbursements.per.enrollee..20
Home.health.agency.reimbursements.per.enrollee..2
Hospice.reimbursements.per.enrollee..2014.
lo(bachelor)
bachelor
lo(hs)
hs
lo(HouseholdIncome)
HouseholdIncome




### Comparison of Residuals from GAM Smoothing Spline and GAM Local Regression

Return to Building GAM Local Regression

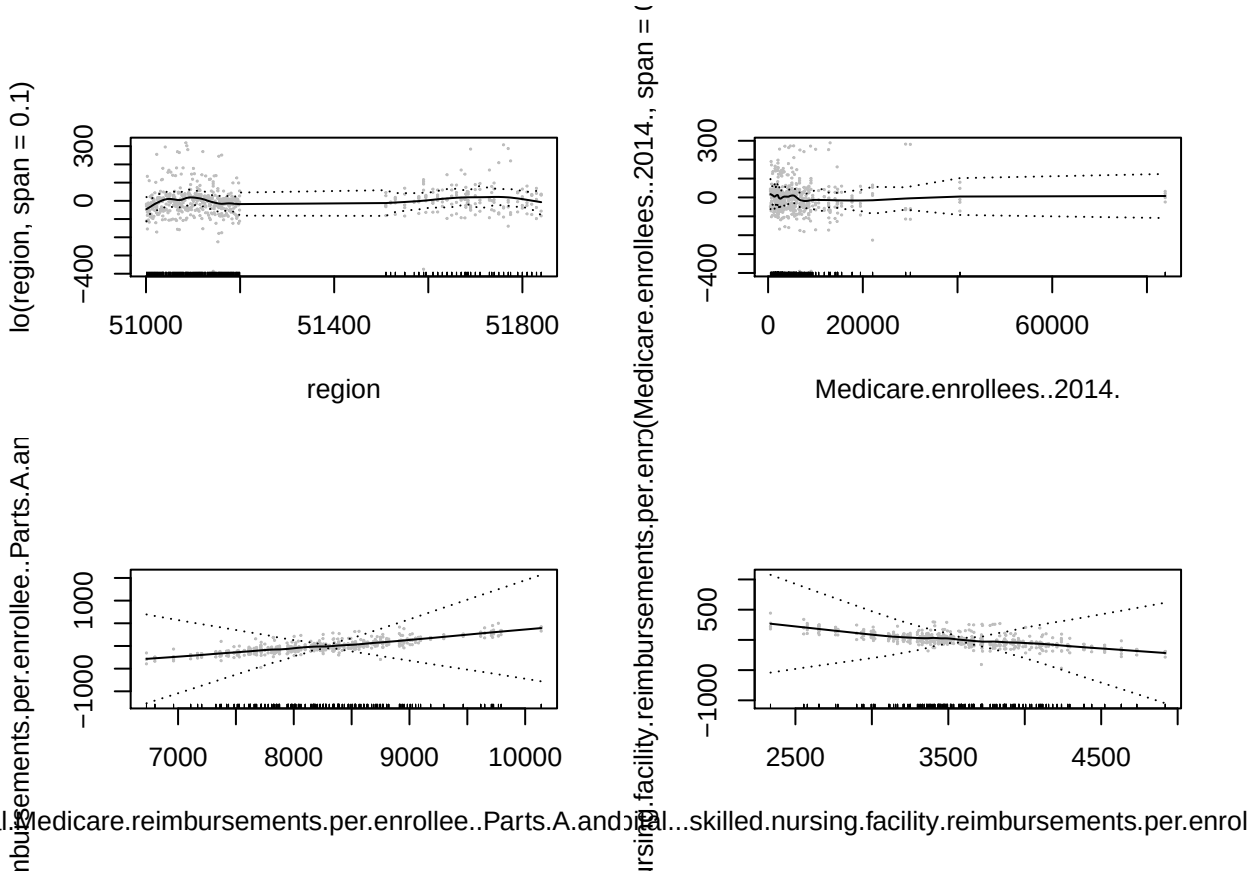
```
par(mfrow = c(1,2))
plot(ACAdf$premiums, fitted(model) - ACAdf$premiums, main = "GAM Spline \n Residuals",
     xlab = "Actual Premiums", ylab = "Residuals")
plot(ACAdf$premiums, fitted(modelLocal) - ACAdf$premiums, main = "GAM Local Regression\n Residuals",
     xlab = "Actual Premiums", ylab = "Residuals", col = "red")
```

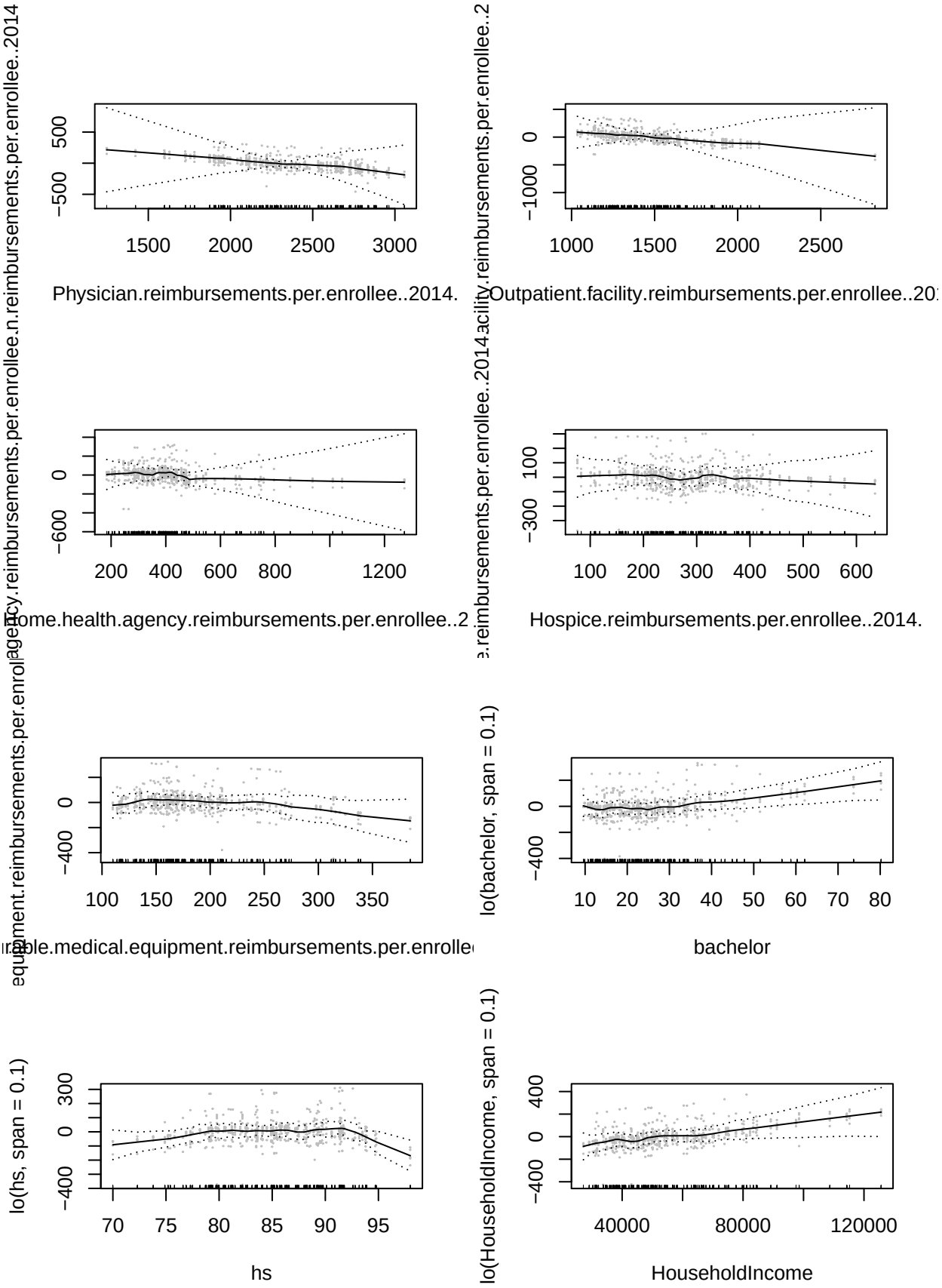


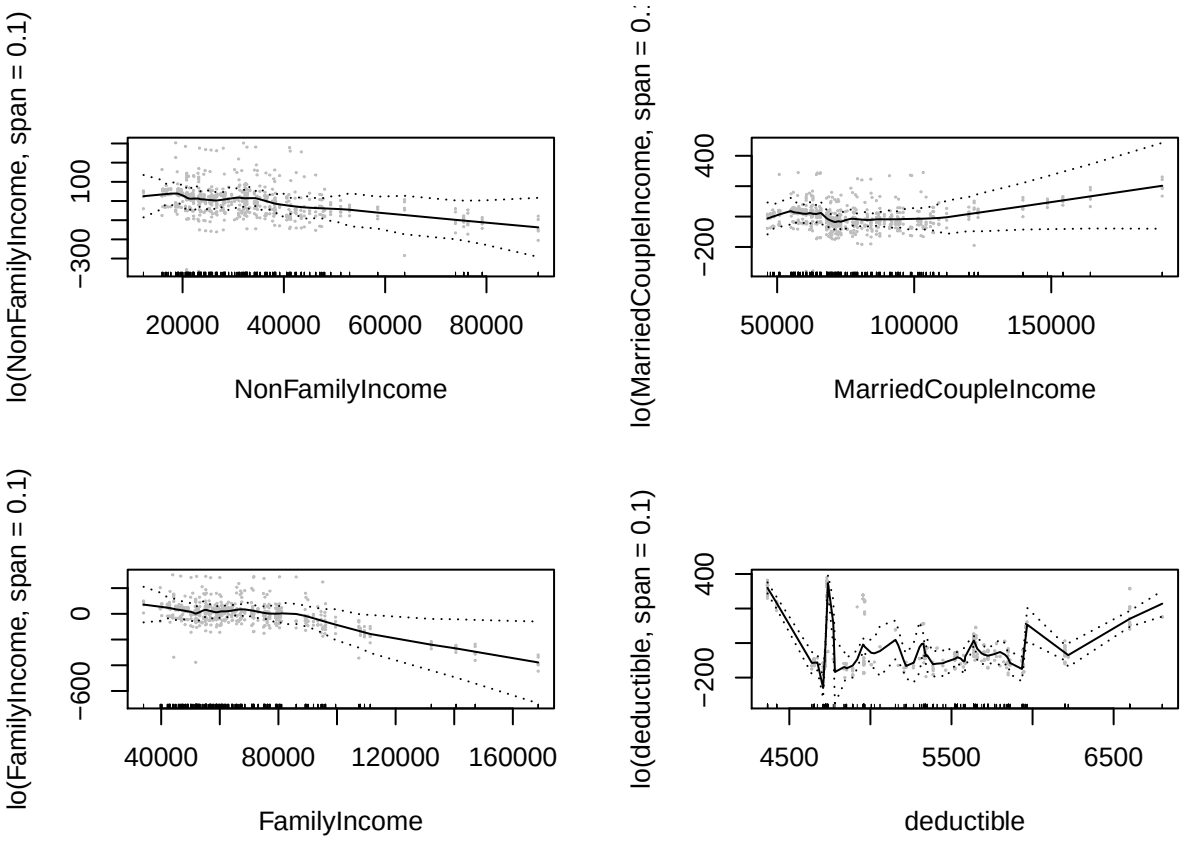

**Tight Fit of GAM Smoothing Spline and Local Regression Residuals Plot**

Return to Building GAM Local Regression

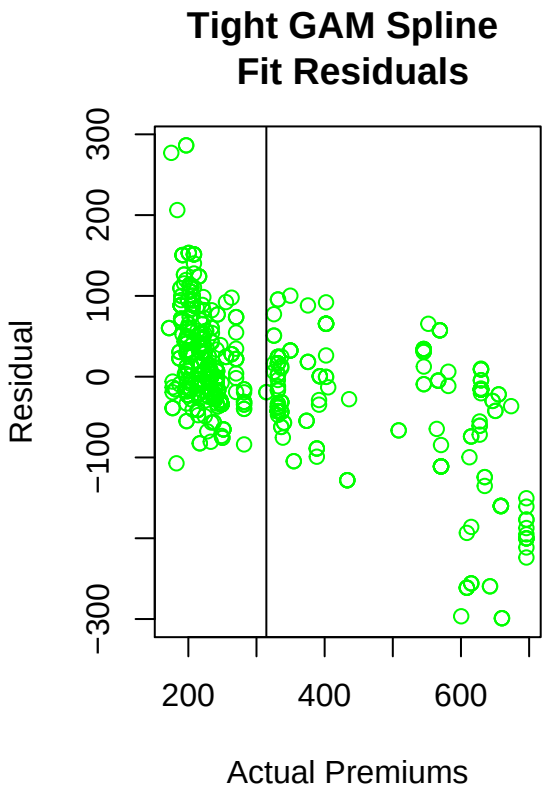
```
modelTight <- gam(premiums~s(region, spar = 0.4) + s(Medicare.enrollees..2014., spar = 0.4) + s(Total.M

modelLocalTight <- gam(premiums~lo(region, span = 0.1) + lo(Medicare.enrollees..2014., span = 0.1 ) + l

{par(mfrow = c(2,2))
plot.Gam(modelLocalTight, residuals = TRUE, se = TRUE, cex = 0.1, col = "grey")}
```

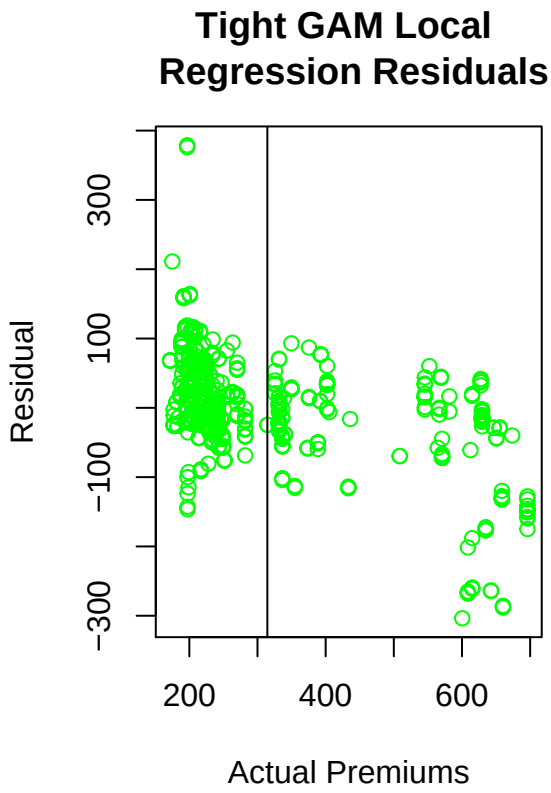

lo(region, span = 0.1)
−400
0
300
51000
51400
51800
region
Medicare.enrollees..2014.
0
20000
60000
nbursements.per.enrollee..Parts.A.an
−1000
1000
7000
8000
9000
10000
l...Medicare.reimbursements.per.enrollee..Parts.A.and.
rsing.facility.reimbursements.per.enro(Medicare.enrollees..2014., span =
−1000
500
2500
3500
4500
...skilled.nursing.facility.reimbursements.per.enrol


Physician.reimbursements.per.enrollee..2014.
Outpatient.facility.reimbursements.per.enrollee..2014.
Home.health.agency.reimbursements.per.enrollee..2014.
Hospice.reimbursements.per.enrollee..2014.
durable.medical.equipment.reimbursements.per.enrollee..2014.
lo(bachelor, span = 0.1)
bachelor
lo(hs, span = 0.1)
hs
lo(HouseholdIncome, span = 0.1)
HouseholdIncome




```
{par(mfrow=c(1,2))
  plot(ACAdf$premiums, fitted(modelTight) - ACAdf$premiums, main = "Tight GAM Spline \n Fit Residuals",
     xlab = "Actual Premiums", ylab = "Residual", col = "green")
  abline(v = mean(ACAdf$premiums))
plot(ACAdf$premiums, fitted(modelLocalTight) - ACAdf$premiums, main = "Tight GAM Local \n Regression Res
     xlab = "Actual Premiums", ylab = "Residual", col = "green")
  abline(v = mean(ACAdf$premiums))
}
```




### K-Fold CV of GAM Local Regression Model

Return to Building GAM Local Regression

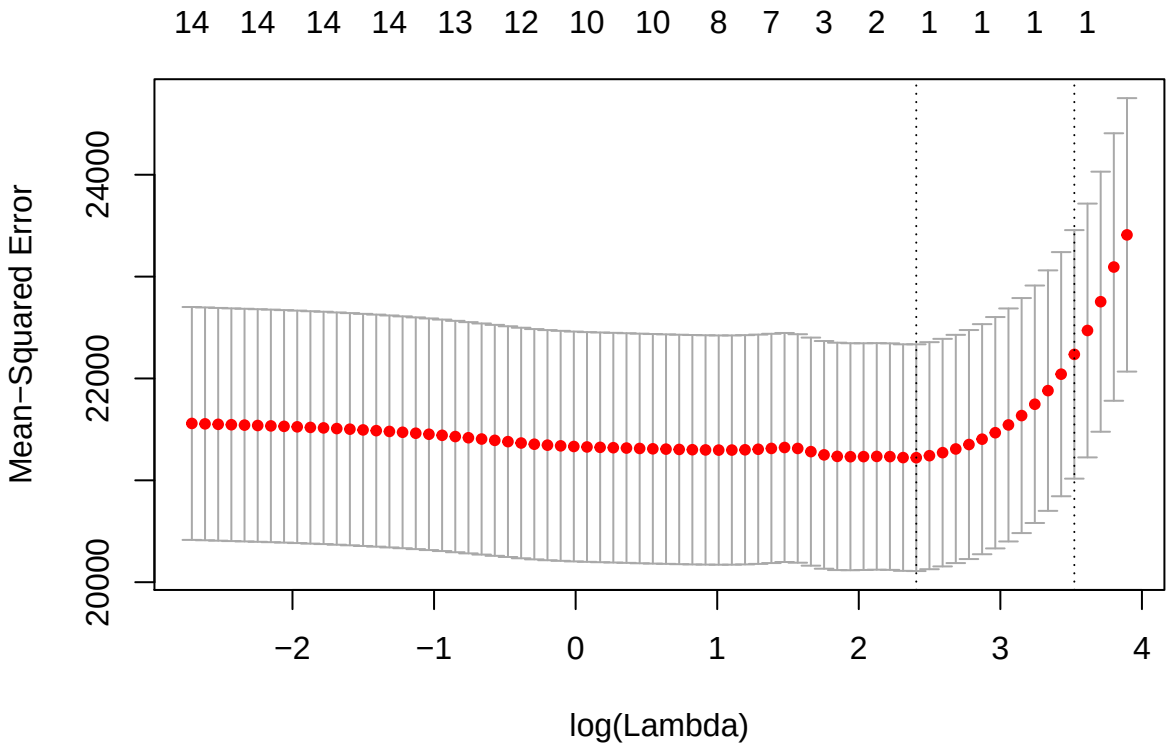
```
set.seed(1)
#n = sample size
#k = number of k-fold validations to run
kFoldLocal <- function(k, n, data) {
  mseVec <- rep(NA, k)
  for(i in 1:k){
    indx <- sample(1:nrow(data), n)
    test <- data[indx,]
    train <- data[-indx,]
    modelTMP <- gam(premiums~lo(region) + lo(Medicare.enrollees..2014.) + lo(Total.Medicare.reimburseme
    predictions <- predict.Gam(modelTMP, newdata = test)
    mseVec[i] <- (1/n)*(sum(predictions - test$premiums)^2)
  }
  return(mean(mseVec))
}

localMSE <- kFoldLocal(20, 100, ACAdf)
localMSE
```

```
## [1] 11431.98
```

### Finding Optimal Lambda Value for LASSO

Return to Multivariate LASSO Regression Model (Parametric)

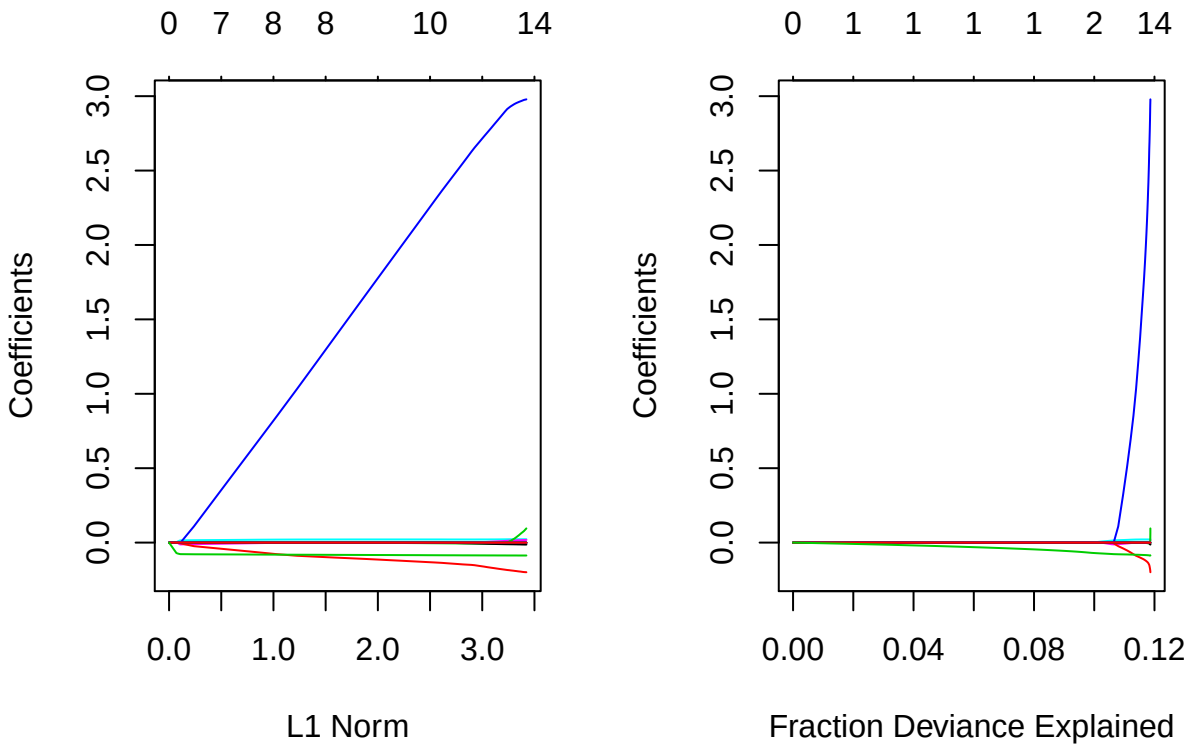
```
set.seed(1)
y <- ACAdf$premiums
x <- as.matrix(select(ACAdf, c(region,Medicare.enrollees..2014., Total.Medicare.reimbursements.per.enrol

optLambda <- cv.glmnet(x, y, family = "gaussian", nfolds = 50, alpha = 1)
plot(optLambda)
```



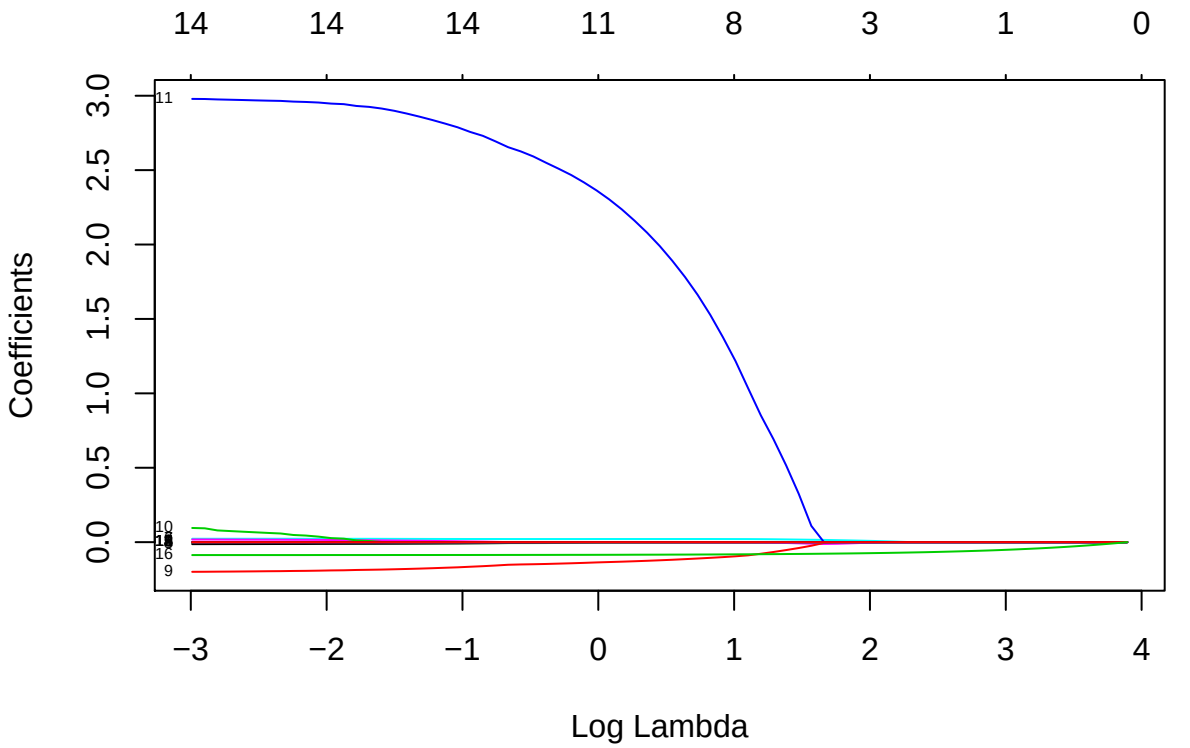

### LASSO Selection Mechanics and Residuals

Return to Multivariate LASSO Regression Model (Parametric)

```
lassoModel <- glmnet(x, y, family = "gaussian", alpha = 1)

{par(mfrow = c(1,2))
  plot(lassoModel, xvar = "norm", label = TRUE)
plot(lassoModel, xvar = "dev", label = TRUE)}
```

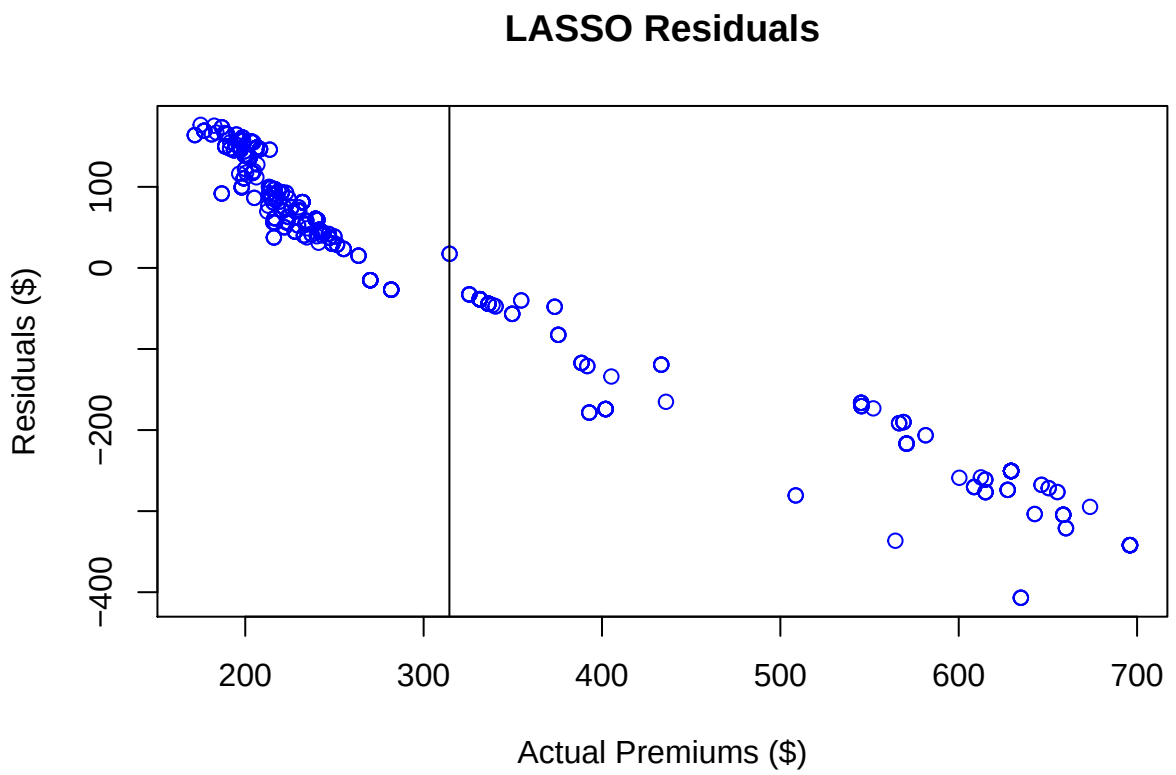



```
par(mfrow = c(1,1))
plot(lassoModel, xvar = "lambda", label = TRUE)
```




```
pred <- predict.glmnet(lassoModel,newx = x, s = optLambda$lambda.min)
resid <- pred - y
par(mfrow = c(1,1))
{plot(ACAdf$premiums, resid, xlab = "Actual Premiums ($)", ylab = "Residuals ($)", main = "LASSO Residua
```

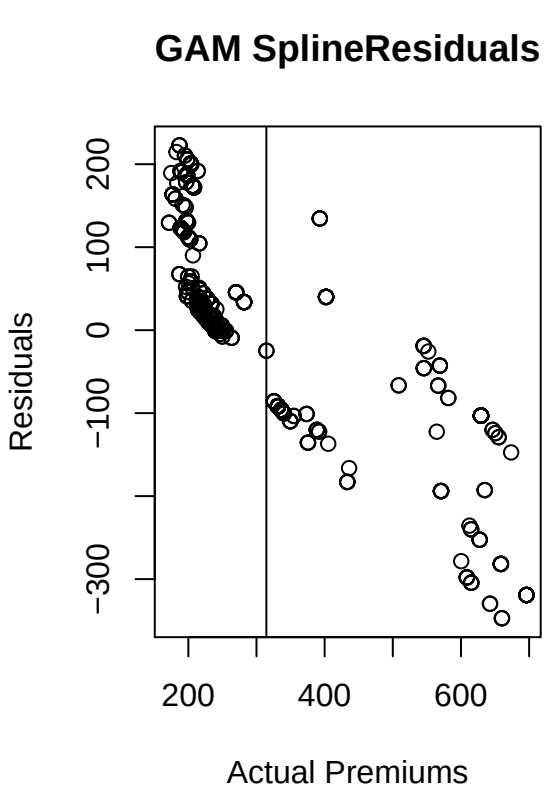
```
abline(v = mean(ACAdf$premiums))}
```

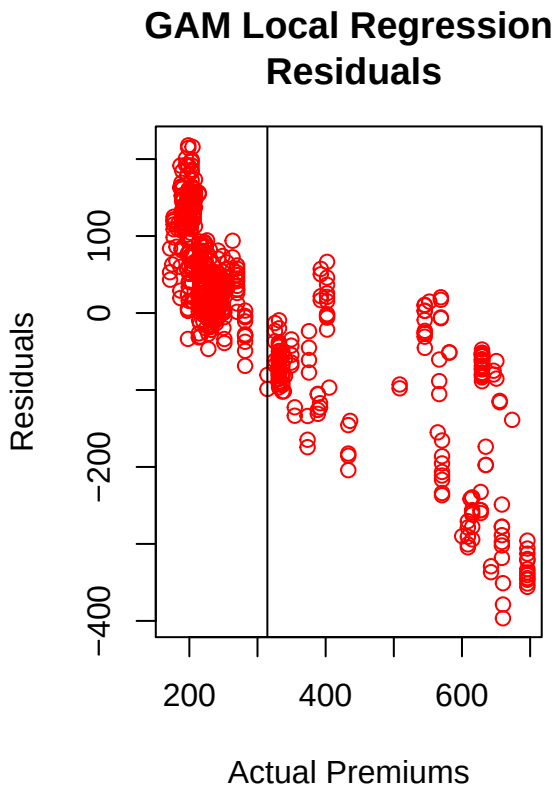



### Comparison of GAM Smoothing Spline and GAM Local Regression Residuals

Return to Multivariate LASSO Regression Model (Parametric)

```
par(mfrow = c(1,2))
{plot(ACAdf$premiums, fitted(model) - ACAdf$premiums, main = "GAM SplineResiduals",
     xlab = "Actual Premiums", ylab = "Residuals")
abline(v = mean(ACAdf$premiums))}
{plot(ACAdf$premiums, fitted(modelLocal) - ACAdf$premiums, main = "GAM Local Regression\n Residuals",
     xlab = "Actual Premiums", ylab = "Residuals", col = "red")
abline(v = mean(ACAdf$premiums))}
```




### K-Fold CV of LASSO Model

Return to Multivariate LASSO Regression Model (Parametric)

```
set.seed(1)
#k = number of k-fold interations
#n = size of test set
lassoKFold <- function(k, n, x, y){
  mseVec <- rep(NA, k)
  for(i in 1:k){
    indx <- sample(1:nrow(x), n)
    trainX <- x[-indx,]
    trainY <- y[-indx]
    testX <- x[indx,]
    testY <- y[indx]
    optLmdTmp <- cv.glmnet(trainX, trainY, family = "gaussian", nfolds = 20, alpha = 1)
    model <- glmnet(trainX, trainY, family = "gaussian", alpha = 1)
    pred <- predict.glmnet(lassoModel,newx = testX, s = optLmdTmp$lambda.min)
    mseVec[i] <- (1/n) * sum((pred - testY)^2)
  }
  return(mean(mseVec))
}

LassoMSE <- lassoKFold(20, 100, x, y)
LassoMSE
```

```
## [1] 21171.69
```