PHP 2550 HW 6

Blain Morin

December 10, 2018

In this assignment, we build upon the model for predicting glomerular filtration rate (GFR) that we constructed in homework 5. The goal of this assignment is to choose the best form of the variables. Using the variables selected by step regression, we will consider non linear transformations (polynomials, splines, step functions, and generalized additive models) for the continuous predictors.

Our strategy is to first fit polynomial, natural spline, and step transformations for each independent variable individually against the outcome. We tune each transformation using 10 fold cross validation. After determining the best form of each transformation, we compare their overall fits to the data to determine the best transformation choice. Finally, we use step generalized additive model (GAM) regression to check the significance of adding non parametric terms to our model.

These are the variables found to be predictive from homework 5:

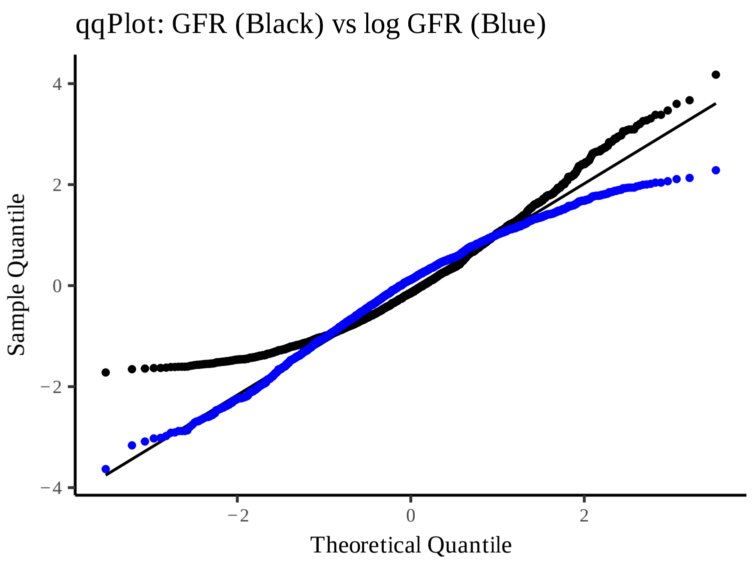
Table 1: Variable Descriptions

Variable	Description	Type
GFR	Glomerular Filtration Rate	Continuous
SUN	Serum Urea Nitrogen	Continuous
SCR	Serum Creatinine	Continuous
AGE	Age	Continuous
FEMALE	Female = 1	Binary
cys	Serum Cystatin	Continuous
Diabetes	Diabetes = 1	Binary
BMI	Body Mass Index	Continuous
BLACK	Black = 1	Binary

The model with no transformations takes the form:

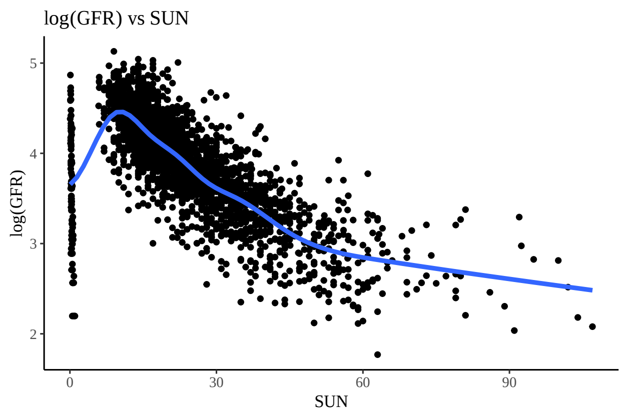
$$Model(1) : E[GFR] = \beta_0 + \beta_1 SUN + \beta_2 SCR + \beta_3 AGE + \beta_4 cys + \beta_5 BMI + \beta_6 Female + \beta_7 Black + \beta_8 Diabetes$$

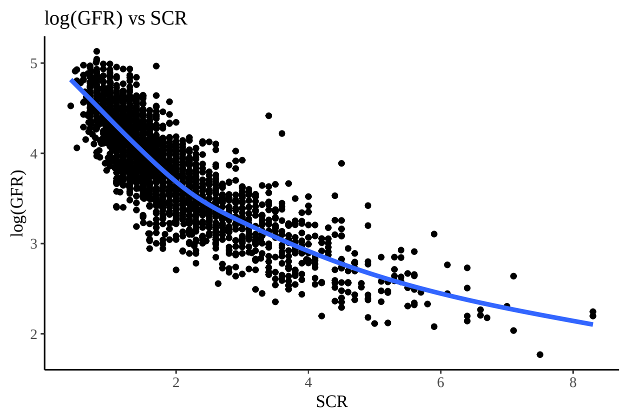
To start, we considered whether or not to transform our outcome variable, GFR, to the log scale. We would like the outcome variable to be as normally distributed as possible. We can compare normality using qqplots:

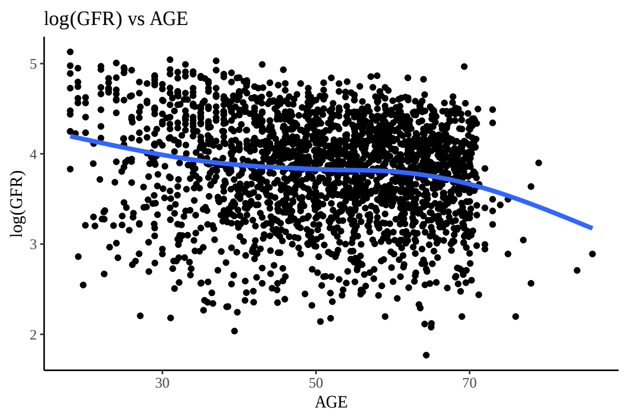


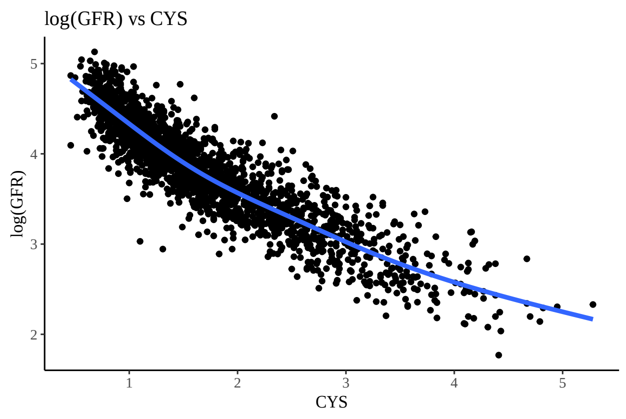
In the qqPlot, we are looking for the transformation that puts the points closest to the 45 degree line. We see that the log transformation (in blue) seems to be better, though it is not perfect. We use log GFR as the outcome variable for the rest of the analysis.

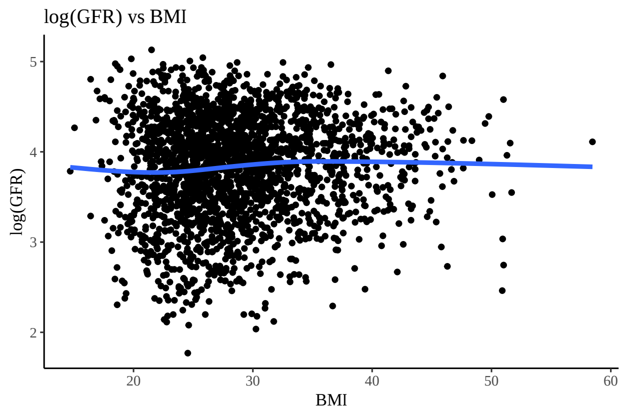
We then create scatter plots between each of the continuous variables and log GFR. We also overlay a smoothing spline on each of the scatter plots, which may clue us in on appropriate transformation choices:







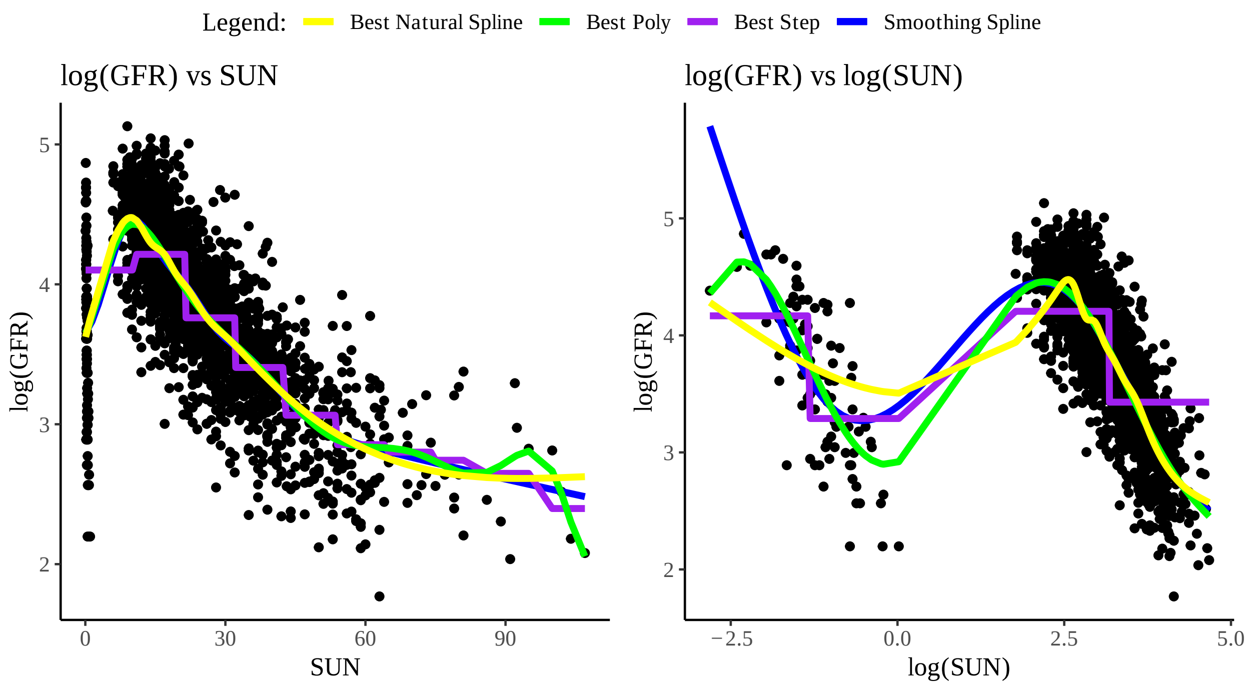




We see that the scatter plots for SUN, SCR, and CYS exhibit substantial nonlinearity (seen from the curvature in the smoothing spline). The the nonlinearity of AGE and BMI are not as visually obvious. We test for nonlinearity in the following sections.

We use 10 fold cross validation to individually choose our transformations. For polynomials, we cross validate to determine the best degree. For step-functions, we cross validate to determine the best number of steps. For natural splines, we cross validate to determine the best number of degrees of freedom. The criterion for choosing the "best" transformation is the lowest average mean squared error on the test set. We also cross validated each transformation using the log transformed independent variable.

We employ this strategy for each of the continuous variables. We start with SUN:

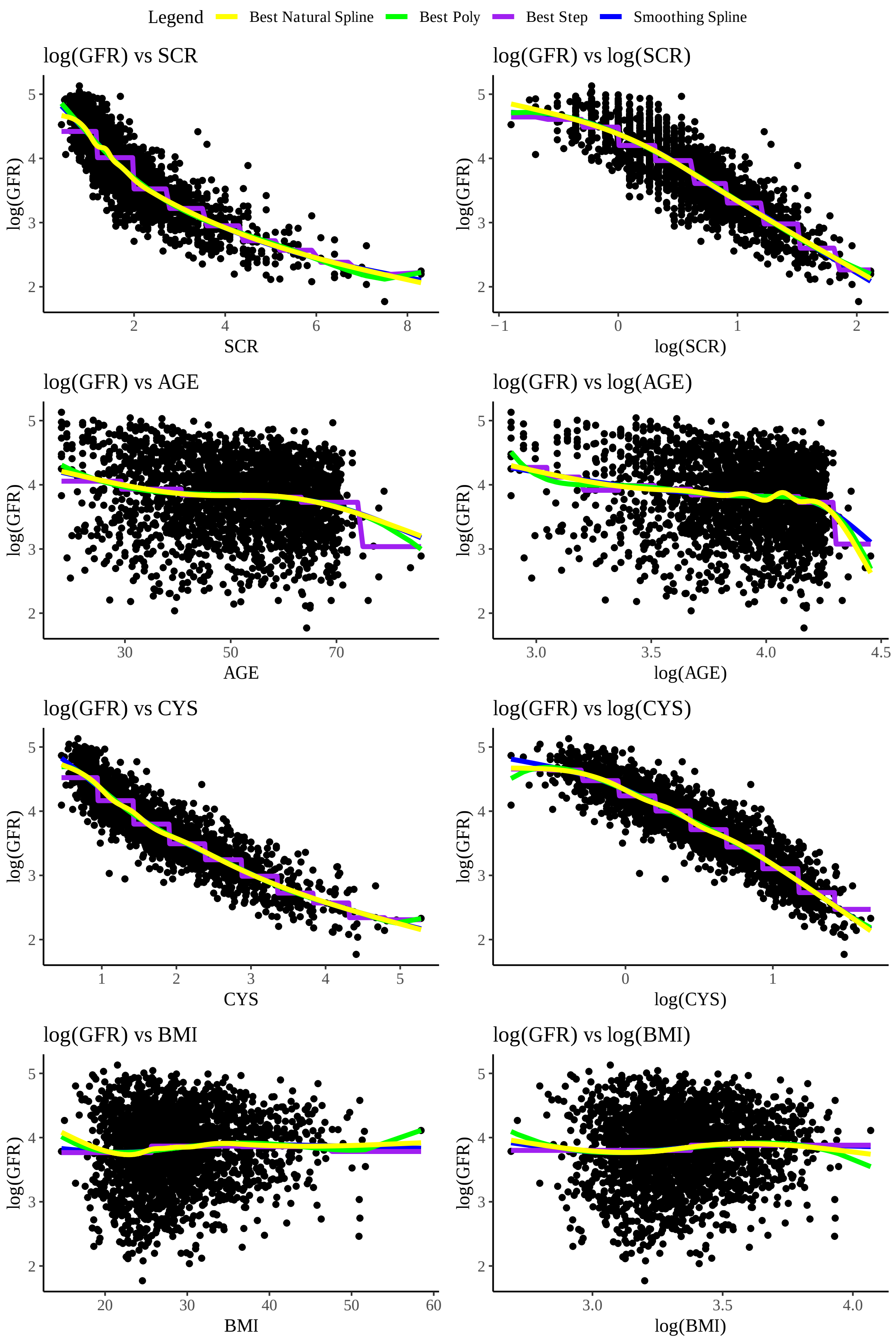


We see that the scatter plot for SUN has many values at SUN = 0. These values should be investigated further to see if there is measurement error or some systematic difference. However, for this analysis we will assume they are correct and choose the model with the best adjusted R^2. Here are the adjusted R^2 values for each of the above parametric fits:

Table 2: SUN Adjusted R2

adjRsquared	Transformation
0.640	Polynomial
0.556	Step
0.640	Natural Spline
0.673	log Polynomial
0.474	log Step
0.639	log Natural Spline

We see that our best fit (highest adjusted R^2) uses a polynomial transformation on log SUN. We repeat this process for the remaining 4 continuous variables:


Legend
Best Natural Spline
Best Poly
Best Step
Smoothing Spline
log(GFR) vs SCR
log(GFR) vs log(SCR)
log(GFR) vs AGE
log(GFR) vs log(AGE)
log(GFR) vs CYS
log(GFR) vs log(CYS)
log(GFR) vs BMI
log(GFR) vs log(BMI)
log(GFR)
SCR
log(SCR)
AGE
log(AGE)
CYS
log(CYS)
BMI
log(BMI)

For each variable and transformation, we extracted the adjusted R^2:

Table 3: Adjusted R2 for Each Variable and Transformation

SUN	SCR	AGE	CYS	BMI	Transformation
0.640	0.738	0.029	0.830	0.006	Polynomial
0.556	0.686	0.027	0.800	0.006	Step
0.640	0.739	0.028	0.831	0.005	Natural Spline
0.673	0.738	0.031	0.830	0.006	log Polynomial
0.474	0.713	0.029	0.808	0.004	log Step
0.639	0.738	0.030	0.830	0.006	log Natural Spline

The transformations with the highest adjusted R^2 for each variable are:

- SUN: log polynomial
- SCR: natural spline
- AGE: log polynomial
- CYS: natural spline
- BMI: polynomial (tie, choose polynomial for simplicity)

Using these transformations our model has the form:

$$Model(2) : E[logGFR] = \beta_0 + \beta poly(logSUN, 7) + \beta ns(SCR, 9) + \beta poly(logAGE, 5) + \beta ns(cys, 7) +$$
$$\beta poly(BMI, 4) + \beta Female + \beta Black + \beta Diabetes$$

We then perform joint significance tests for each of the terms:

Table 4: Model (2) Terms

	df	F	p-value
poly(logSUN, log.sun.best.poly)	7	6.087	0.00000
ns(SCR, scr.best.spline)	9	90.373	0
poly(logAGE, log.age.best.poly)	5	34.945	0
ns(cys, cys.best.spline)	7	96.507	0
poly(BMI, bmi.best.poly)	4	4.844	0.001
FEMALE	1	404.166	0
BLACK	1	166.288	0
Diabetes	1	1.015	0.314

In the table above, the degrees of freedom show how many beta coefficients are added to the model by each term. The F-statistic tests the joint significance of all terms. We see that the Diabetes variable is no longer significant.

Next, we run step wise selection in a GAM using our best transformations as the starting point. The scope of our step procedure allows for each term to be linear, the parametric transformation we chose above, or a non-parametric smoothing spline. Here are the steps:

Table 5: GAM Steps

From	To	Resid. Dev	AIC
	<start>	76.631	-1, 213.335
poly(logAGE, log.age.best.poly)	s(logAGE, 4)	76.607	-1, 216.051
poly(BMI, bmi.best.poly)	s(BMI, 4)	76.524	-1, 218.542
poly(logSUN, log.sun.best.poly)	s(logSUN, 4)	76.656	-1, 220.581
Diabetes		76.686	-1, 221.676
ns(cys, cys.best.spline)	s(cys, cys.best.spline)	76.678	-1, 221.916

We see from the above table that the step.Gam function chooses smoothing splines for logSUN, logAGE, cys, and BMI. The only continuous variable that was not switched to a smoothing spline was SCR. We also see that the Diabetes variable was dropped from the model.

The step function is choosing by AIC, which does not necessarily mean that the nonparameteric part is significant because some of the parametric terms from the starting model contained many coefficients. For example: the polynomial transformation for logSun adds 7 variables to the model. In contrast, the smoothing spline for logSun only contains 4 degrees of freedom. Therefore, it is not clear whether the reduction in AIC is due to reduced degrees of freedom used or is due to the non parametric part of the fit explaining significant variation. This uncertainty could be assessed with further tuning of the parametric parameters within the fully specified model.

Here are the ANOVA tests for the parametric and nonparametric terms for the model selected by step.Gam:

Table 6: Model (3) Parametric Terms

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
s(logSUN)	1	85.332	85.332	2, 516.219	0
ns(SCR, scr.best.spline)	9	417.715	46.413	1, 368.589	0
s(logAGE)	1	7.838	7.838	231.121	0
s(cys)	1	64.603	64.603	1, 904.975	0
s(BMI)	1	1.788	1.788	52.717	0
FEMALE	1	13.333	13.333	393.152	0
BLACK	1	7.824	7.824	230.695	0
Residuals	2, 266.000	76.847	0.034		

Table 7: Model (3) Non-Parametric Terms

	Npar Df	Npar F	Pr(F)
(Intercept)			
s(logSUN)	3	9.744	0.00000
ns(SCR, scr.best.spline)			
s(logAGE)	3	6.683	0.0002
s(cys)	3	23.969	0
s(BMI)	3	2.695	0.045
FEMALE			
BLACK			

We see that the joint significance test for all of the parametric parts of the model are highly significant. We also see that the nonparametric parts of the model are also significant, which means they are explaining a significant amount of the variation in logGFR.

Lastly, we compare the AIC and R2 for the three models:

Table 8: AIC and adj R2 for Each Model

Model	AIC	adj.R2
(1) No Transformations	$19,125.960$	0.682
(2) Parametric	-$1,213.335$	0.895
(3) Non Parametric	-$1,221.916$	0.895

The AIC decreases from 19,126 to -1,213 when moving from model (1) to model (2). Moreover, the adjusted R^2 increases from .682 to .895. This is a substantial improvement in fit and is strong evidence that we should use transformations.

The improvement in AIC moving from model (2) to model (3) is relatively small. There is no difference in adjusted R^2. Because the models perform similarly, we recommend using the parametric (model (2)). The parametric model is preferable because it allows us to make predictions on new data. The nonparametric model (model (3)) is dependent on the data, which makes it difficult to use in prediction.

Overall, an adjusted R^2 of .895 is relatively high. Further improvements in fit could be investigated by cross validating the smoothing spline's degrees of freedom. We could also invesigate the individual terms of the transformations to see if their significance persists when other variables are added to the model.

Appendix: R code

```r
set.seed(100)

### Load Libraries

library(knitr)
library(readr)
library(ggplot2)
library(dplyr)
library(stargazer)
library(splines)
library(gam)
library(extrafont)
library(grid)
library(gridExtra)
library(caret)
library(ggpubr)


### Load and Clean Data procedure from hw5

### Load data
```

```r
iod = read_csv("iodatadev.csv")

### Filter out columns with more that 10% missing data
iod.clean = Filter(function(x) mean(is.na(x)) < 0.1, iod)

### Get complete cases
iod.clean = iod.clean %>%
  filter(complete.cases(.))

### Remove ids, collinear terms, and unknown variables
iod.clean = iod.clean %>%
  select(-X1, -ID, -"..2", -"..2_1", -CSG, -drds, - rass1)

### Select the variables from homework 5
iod.clean = iod.clean %>%
  select(GFR, SUN, SCR, AGE, FEMALE, cys, Diabetes, BMI, BLACK)



### Create variable description table

varnames = names(iod.clean)

vardesc = c("Glomerular Filtration Rate",
            "Serum Urea Nitrogen",
            "Serum Creatinine",
            "Age",
            "Female = 1",
            "Serum Cystatin",
            "Diabetes = 1",
            "Body Mass Index",
            "Black = 1")

vartype = c("Continuous",
            "Continuous",
            "Continuous",
            "Continuous",
            "Binary",
            "Continuous",
            "Binary",
            "Continuous",
            "Binary")

table1.data = data.frame(Variable = varnames, Description = vardesc, Type = vartype)

stargazer(table1.data,
          header = FALSE,
          title = "Variable Descriptions",
          table.placement = 'H',
          summary = FALSE,
          rownames = FALSE)
```

```r
### First consider log transforms of continuous variables

### Use log GFR?

gfr.qq = iod.clean %>%
  ggplot(aes(sample = scale(GFR))) +
  stat_qq(size = .8) + stat_qq_line() +
  stat_qq(aes(sample = scale(log(GFR))), color = "blue", size = .8) +
  ylab("Sample Quantile") +
  xlab("Theoretical Quantile") +
  ggtitle("qqPlot: GFR (Black) vs log GFR (Blue)") +
  theme_classic() +
  theme(text=element_text(size=9,  family="CM Sans"))


gfr.qq


###Add log(GFR)

iod.clean = iod.clean %>%
  mutate(logGFR = log(GFR))

###Add log of all cont. variables

iod.clean = iod.clean %>%
  mutate(logSUN = log(SUN),
         logSCR = log(SCR),
         logAGE = log(AGE),
         logcys = log(cys),
         logBMI = log(BMI))


sun.smooth = iod.clean %>%
  ggplot(aes(y = logGFR, x = SUN)) +
  geom_point() +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs"), se = FALSE, size = 1.5) +
  ylab("log(GFR)") +
  xlab("SUN") +
  ggtitle("log(GFR) vs SUN") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans"))

scr.smooth = iod.clean %>%
  ggplot(aes(y = logGFR, x = SCR)) +
  geom_point() +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs"), se = FALSE, size = 1.5) +
  ylab("log(GFR)") +
  xlab("SCR") +
  ggtitle("log(GFR) vs SCR") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans"))
```

```r
age.smooth = iod.clean %>%
  ggplot(aes(y = logGFR, x = AGE)) +
  geom_point() +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs"), se = FALSE, size = 1.5) +
  ylab("log(GFR)") +
  xlab("AGE") +
  ggtitle("log(GFR) vs AGE") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans"))

cys.smooth = iod.clean %>%
  ggplot(aes(y = logGFR, x = cys)) +
  geom_point() +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs"), se = FALSE, size = 1.5) +
  ylab("log(GFR)") +
  xlab("CYS") +
  ggtitle("log(GFR) vs CYS") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans"))

bmi.smooth = iod.clean %>%
  ggplot(aes(y = logGFR, x = BMI)) +
  geom_point() +
  geom_smooth(method = "gam", formula = y ~ s(x, bs = "cs"), se = FALSE, size = 1.5) +
  ylab("log(GFR)") +
  xlab("BMI") +
  ggtitle("log(GFR) vs BMI") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans"))

grid.arrange(sun.smooth, scr.smooth,
             age.smooth, cys.smooth,
             bmi.smooth,
             nrow = 3)



### Function for best polynomial


bestpoly = function(x, y, folds, maxpoly){

  leaveout = createFolds(x, k = folds)
  error.matrix = matrix(NA, nrow = folds, ncol = maxpoly)

  for (i in 1:folds) {

    trainx = x[-leaveout[[i]]]
    trainy = y[-leaveout[[i]]]

    testx = x[leaveout[[i]]]
    testy = y[leaveout[[i]]]
    train.mat = as.data.frame(cbind(y=trainy, x = trainx))
    test.mat = as.data.frame(cbind(y = testy, x = testx))

    for (j in 1:maxpoly) {

      mod = lm(y ~ poly(x, degree = j, raw = TRUE), data = train.mat)
      preds = predict(mod, newdata = test.mat)
      error = mean((testy - preds)^2)
      error.matrix[i,j] = error

    }

  }

  ave.mse = apply(error.matrix, 2, mean)

  return(ave.mse)


}



### Function for best step (equal breaks)


beststep = function(x, y, folds, maxbreaks){

  leaveout = createFolds(x, k = folds)
  error.matrix = matrix(NA, nrow = folds, ncol = maxbreaks-1)

  for (j in 2:maxbreaks) {

    a = cut(as.numeric(x), breaks = j)

    for (i in 1:folds) {

      trainx = a[-leaveout[[i]]]
      trainy = y[-leaveout[[i]]]

      testx = a[leaveout[[i]]]
      testy = y[leaveout[[i]]]

      train.mat = as.data.frame(cbind(y=trainy, b = trainx))
      test.mat = as.data.frame(cbind(y = testy, b = testx))



      mod = lm(y ~ as.factor(b), data = train.mat)
      preds = predict(mod, newdata = test.mat)
      error = mean((testy - preds)^2)
      error.matrix[i, j-1] = error
```

```r
    }

  }

  ave.mse = apply(error.matrix, 2, mean)
  names(ave.mse) = 2:maxbreaks

  return(ave.mse)


}



### Function for best natural spline

bestspline = function(x, y, folds, dfs) {
  leaveout = createFolds(x, k = folds)
  error.matrix = matrix(NA, nrow = folds, ncol = dfs)

  for (i in 1:folds) {

    trainx = x[-leaveout[[i]]]
    trainy = y[-leaveout[[i]]]

    testx = x[leaveout[[i]]]
    testy = y[leaveout[[i]]]

    train.mat = as.data.frame(cbind(y=trainy, x = trainx))
    test.mat = as.data.frame(cbind(y = testy, x = testx))

    for (j in 1:dfs) {

      fit.nspline=lm(y~ns(x,df=j),data=train.mat)  #Fit natural spline
      preds=predict(fit.nspline,newdata=test.mat,se=T)$fit

      error = mean((testy - preds)^2)
      error.matrix[i,j] = error
    }
  }

  ave.mse = apply(error.matrix, 2, mean)

  return(ave.mse)
}


### Colors for legend

cols <- c("Smoothing Spline"="blue","Best Step"="purple","Best Poly"="green", "Best Natural Spline" = "


### SUN best transforms
### Poly
sun.best.poly = which.min(bestpoly(x = iod.clean$SUN, y = iod.clean$logGFR,
                          folds = 10, maxpoly = 10))

sun.poly.mod = lm(logGFR ~ poly(SUN, sun.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(sun.poly.preds = predict(sun.poly.mod))


### Step

sun.best.step = which.min(beststep(x = iod.clean$SUN, y = iod.clean$logGFR,
                                    folds = 10, maxbreaks = 10))

sun.step.mod = lm(logGFR ~ cut(SUN, breaks = sun.best.step + 1), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(sun.step.preds = predict(sun.step.mod))


### Spline

sun.best.spline = which.min(bestspline(x = iod.clean$SUN, y = iod.clean$logGFR,
                                    folds = 10, dfs = 10))

sun.spline.mod = lm(logGFR ~ ns(SUN, df = sun.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(sun.spline.preds = predict(sun.spline.mod))

sun.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = SUN)) +
  geom_point() +
  geom_smooth(aes(colour = "Smoothing Spline"), method = "gam", formula = y ~ s(x, bs = "cs"), se = FALS
  ylab("log(GFR)") +
  xlab("SUN") +
  ggtitle("log(GFR) vs SUN") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = sun.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = sun.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = sun.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)



### log SUN best transforms
```

```r
### Poly
log.sun.best.poly = which.min(bestpoly(x = iod.clean$logSUN, y = iod.clean$logGFR,
                         folds = 10, maxpoly = 10))

log.sun.poly.mod = lm(logGFR ~ poly(logSUN, log.sun.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(log.sun.poly.preds = predict(log.sun.poly.mod))


### Step

log.sun.best.step = which.min(beststep(x = iod.clean$logSUN, y = iod.clean$logGFR,
                                  folds = 10, maxbreaks = 5))

log.sun.step.mod = lm(logGFR ~ cut(logSUN, breaks = log.sun.best.step + 1), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(log.sun.step.preds = predict(log.sun.step.mod))


### Spline

log.sun.best.spline = which.min(bestspline(x = iod.clean$logSUN, y = iod.clean$logGFR,
                                  folds = 10, dfs = 10))

log.sun.spline.mod = lm(logGFR ~ ns(logSUN, df = log.sun.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(log.sun.spline.preds = predict(log.sun.spline.mod))

log.sun.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = logSUN)) +
  geom_point() +
  geom_smooth(aes(colour = "Smoothing Spline"), method = "gam", formula = y ~ s(x, bs = "cs"), se = FAL
  ylab("log(GFR)") +
  xlab("log(SUN)") +
  ggtitle("log(GFR) vs log(SUN)") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = log.sun.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = log.sun.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = log.sun.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)




ggarrange(sun.transforms, log.sun.transforms, common.legend = TRUE, legend = "top")
```

```r
### Function to get r2

rsquared = function(x){

  summaryx = summary(x)

  return(summaryx$adj.r.squared)

}



### Rsquared table for sun

sun.rs = c(rsquared(sun.poly.mod),
           rsquared(sun.step.mod),
           rsquared(sun.spline.mod),
           rsquared(log.sun.poly.mod),
           rsquared(log.sun.step.mod),
           rsquared(log.sun.spline.mod))

rs.name = c("Polynomial",
            "Step",
            "Natural Spline",
            "log Polynomial",
            "log Step",
            "log Natural Spline")

sun.r2.data = data.frame(adjRsquared = sun.rs, Transformation = rs.name)

stargazer(sun.r2.data,
          title = "SUN Adjusted R2",
          header = FALSE,
          summary = FALSE,
          table.placement = 'H',
          rownames = FALSE
          )


### SCR best transforms


### Poly
scr.best.poly = which.min(bestpoly(x = iod.clean$SCR, y = iod.clean$logGFR,
                         folds = 10, maxpoly = 10))

scr.poly.mod = lm(logGFR ~ poly(SCR, scr.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(scr.poly.preds = predict(scr.poly.mod))


### Step
```

```r
scr.best.step = which.min(beststep(x = iod.clean$SCR, y = iod.clean$logGFR,
                                   folds = 10, maxbreaks = 10))

scr.step.mod = lm(logGFR ~ cut(SCR, breaks = scr.best.step + 1), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(scr.step.preds = predict(scr.step.mod))


### Spline

scr.best.spline = which.min(bestspline(x = iod.clean$SCR, y = iod.clean$logGFR,
                                   folds = 10, dfs = 10))

scr.spline.mod = lm(logGFR ~ ns(SCR, df = scr.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(scr.spline.preds = predict(scr.spline.mod))

scr.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = SCR)) +
  geom_point() +
  geom_smooth(aes(colour = "Smoothing Spline"), method = "gam", formula = y ~ s(x, bs = "cs"), se = FALS
  ylab("log(GFR)") +
  xlab("SCR") +
  ggtitle("log(GFR) vs SCR") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = scr.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = scr.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = scr.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)



### AGE best transforms


### Poly
age.best.poly = which.min(bestpoly(x = iod.clean$AGE, y = iod.clean$logGFR,
                         folds = 10, maxpoly = 10))

age.poly.mod = lm(logGFR ~ poly(AGE, age.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(age.poly.preds = predict(age.poly.mod))


### Step

age.best.step = which.min(beststep(x = iod.clean$AGE, y = iod.clean$logGFR,
                                   folds = 10, maxbreaks = 10))

age.step.mod = lm(logGFR ~ cut(AGE, breaks = age.best.step + 1), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(age.step.preds = predict(age.step.mod))


### Spline

age.best.spline = which.min(bestspline(x = iod.clean$AGE, y = iod.clean$logGFR,
                                   folds = 10, dfs = 10))

age.spline.mod = lm(logGFR ~ ns(AGE, df = age.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(age.spline.preds = predict(age.spline.mod))

age.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = AGE)) +
  geom_point() +
  geom_smooth(aes(colour = "Smoothing Spline"), method = "gam", formula = y ~ s(x, bs = "cs"), se = FALS
  ylab("log(GFR)") +
  xlab("AGE") +
  ggtitle("log(GFR) vs AGE") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = age.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = age.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = age.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)



### cys best transforms


### Poly
cys.best.poly = which.min(bestpoly(x = iod.clean$cys, y = iod.clean$logGFR,
                         folds = 10, maxpoly = 10))

cys.poly.mod = lm(logGFR ~ poly(cys, cys.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(cys.poly.preds = predict(cys.poly.mod))


### Step

cys.best.step = which.min(beststep(x = iod.clean$cys, y = iod.clean$logGFR,
                                   folds = 10, maxbreaks = 10))
```

```
cys.step.mod = lm(logGFR ~ cut(cys, breaks = cys.best.step + 1), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(cys.step.preds = predict(cys.step.mod))


### Spline

cys.best.spline = which.min(bestspline(x = iod.clean$cys, y = iod.clean$logGFR,
                                   folds = 10, dfs = 10))

cys.spline.mod = lm(logGFR ~ ns(cys, df = cys.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(cys.spline.preds = predict(cys.spline.mod))

cys.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = cys)) +
  geom_point() +
  geom_smooth(aes(color = "Smoothing Spline"), method = "gam", formula = y ~ s(x, bs = "cs"), se = FALSE
  ylab("log(GFR)") +
  xlab("CYS") +
  ggtitle("log(GFR) vs CYS") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = cys.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = cys.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = cys.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)



### BMI best transforms


### Poly
bmi.best.poly = which.min(bestpoly(x = iod.clean$BMI, y = iod.clean$logGFR,
                         folds = 10, maxpoly = 10))

bmi.poly.mod = lm(logGFR ~ poly(BMI, bmi.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(bmi.poly.preds = predict(bmi.poly.mod))


### Step

bmi.best.step = which.min(beststep(x = iod.clean$BMI, y = iod.clean$logGFR,
                                   folds = 10, maxbreaks = 5))

bmi.step.mod = lm(logGFR ~ cut(BMI, breaks = bmi.best.step + 1), data = iod.clean)
```

```
iod.clean = iod.clean %>%
  mutate(bmi.step.preds = predict(bmi.step.mod))


### Spline

bmi.best.spline = which.min(bestspline(x = iod.clean$BMI, y = iod.clean$logGFR,
                                   folds = 10, dfs = 10))

bmi.spline.mod = lm(logGFR ~ ns(BMI, df = scr.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(bmi.spline.preds = predict(bmi.spline.mod))

bmi.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = BMI)) +
  geom_point() +
  geom_smooth(aes(colour = "Smoothing Spline"), method = "gam", formula = y ~ s(x, bs = "cs"), se = FAL
  ylab("log(GFR)") +
  xlab("BMI") +
  ggtitle("log(GFR) vs BMI") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = bmi.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = bmi.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = bmi.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)


### log SCR best transforms


### Poly
log.scr.best.poly = which.min(bestpoly(x = iod.clean$logSCR, y = iod.clean$logGFR,
                         folds = 10, maxpoly = 10))

log.scr.poly.mod = lm(logGFR ~ poly(logSCR, log.scr.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(log.scr.poly.preds = predict(log.scr.poly.mod))


### Step

log.scr.best.step = which.min(beststep(x = iod.clean$logSCR, y = iod.clean$logGFR,
                                   folds = 10, maxbreaks = 10))

log.scr.step.mod = lm(logGFR ~ cut(logSCR, breaks = log.scr.best.step + 1), data = iod.clean)


iod.clean = iod.clean %>%
```

```r
  mutate(log.scr.step.preds = predict(log.scr.step.mod))


### Spline

log.scr.best.spline = which.min(bestspline(x = iod.clean$logSCR, y = iod.clean$logGFR,
                                  folds = 10, dfs = 10))

log.scr.spline.mod = lm(logGFR ~ ns(logSCR, df = log.scr.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(log.scr.spline.preds = predict(log.scr.spline.mod))

log.scr.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = logSCR)) +
  geom_point() +
  geom_smooth(aes(colour = "Smoothing Spline"), method = "gam", formula = y ~ s(x, bs = "cs"), se = FALS
  ylab("log(GFR)") +
  xlab("log(SCR)") +
  ggtitle("log(GFR) vs log(SCR)") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = log.scr.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = log.scr.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = log.scr.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)



### log AGE best transforms


### Poly
log.age.best.poly = which.min(bestpoly(x = iod.clean$logAGE, y = iod.clean$logGFR,
                         folds = 10, maxpoly = 6))

log.age.poly.mod = lm(logGFR ~ poly(logAGE, log.age.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(log.age.poly.preds = predict(log.age.poly.mod))


### Step

log.age.best.step = which.min(beststep(x = iod.clean$logAGE, y = iod.clean$logGFR,
                                  folds = 10, maxbreaks = 10))

log.age.step.mod = lm(logGFR ~ cut(logAGE, breaks = log.age.best.step + 1), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(log.age.step.preds = predict(log.age.step.mod))
```

```
### Spline

log.age.best.spline = which.min(bestspline(x = iod.clean$logAGE, y = iod.clean$logGFR,
                                   folds = 10, dfs = 10))

log.age.spline.mod = lm(logGFR ~ ns(logAGE, df = log.age.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(log.age.spline.preds = predict(log.age.spline.mod))

log.age.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = logAGE)) +
  geom_point() +
  geom_smooth(aes(colour = "Smoothing Spline"), method = "gam", formula = y ~ s(x, bs = "cs"), se = FALS
  ylab("log(GFR)") +
  xlab("log(AGE)") +
  ggtitle("log(GFR) vs log(AGE)") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = log.age.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = log.age.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = log.age.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)



### log cys best transforms


### Poly
log.cys.best.poly = which.min(bestpoly(x = iod.clean$logcys, y = iod.clean$logGFR,
                         folds = 10, maxpoly = 10))

log.cys.poly.mod = lm(logGFR ~ poly(logcys, log.cys.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(log.cys.poly.preds = predict(log.cys.poly.mod))


### Step

log.cys.best.step = which.min(beststep(x = iod.clean$logcys, y = iod.clean$logGFR,
                                   folds = 10, maxbreaks = 10))

log.cys.step.mod = lm(logGFR ~ cut(logcys, breaks = log.cys.best.step + 1), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(log.cys.step.preds = predict(log.cys.step.mod))
```

```
### Spline

log.cys.best.spline = which.min(bestspline(x = iod.clean$logcys, y = iod.clean$logGFR,
                                   folds = 10, dfs = 10))

log.cys.spline.mod = lm(logGFR ~ ns(logcys, df = log.cys.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(log.cys.spline.preds = predict(log.cys.spline.mod))

log.cys.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = logcys)) +
  geom_point() +
  geom_smooth(aes(colour = "Smoothing Spline") ,method = "gam", formula = y ~ s(x, bs = "cs"), se = FAL
  ylab("log(GFR)") +
  xlab("log(CYS)") +
  ggtitle("log(GFR) vs log(CYS)") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = log.cys.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = log.cys.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = log.cys.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)



### log BMI best transforms


### Poly
log.bmi.best.poly = which.min(bestpoly(x = iod.clean$logBMI, y = iod.clean$logGFR,
                         folds = 10, maxpoly = 6))

log.bmi.poly.mod = lm(logGFR ~ poly(logBMI, log.bmi.best.poly), data = iod.clean)

iod.clean = iod.clean %>%
  mutate(log.bmi.poly.preds = predict(log.bmi.poly.mod))


### Step

log.bmi.best.step = which.min(beststep(x = iod.clean$logBMI, y = iod.clean$logGFR,
                                   folds = 10, maxbreaks = 5))

log.bmi.step.mod = lm(logGFR ~ cut(logBMI, breaks = log.bmi.best.step + 1), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(log.bmi.step.preds = predict(log.bmi.step.mod))


### Spline
```

```
log.bmi.best.spline = which.min(bestspline(x = iod.clean$logBMI, y = iod.clean$logGFR,
                                  folds = 10, dfs = 10))

log.bmi.spline.mod = lm(logGFR ~ ns(logBMI, df = log.scr.best.spline), data = iod.clean)


iod.clean = iod.clean %>%
  mutate(log.bmi.spline.preds = predict(log.bmi.spline.mod))

log.bmi.transforms = iod.clean %>%
  ggplot(aes(y = logGFR, x = logBMI)) +
  geom_point() +
  geom_smooth(aes(colour = "Smoothing Spline"), method = "gam", formula = y ~ s(x, bs = "cs"), se = FALS
  ylab("log(GFR)") +
  xlab("log(BMI)") +
  ggtitle("log(GFR) vs log(BMI)") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans")) +
  geom_line(aes(y = log.bmi.step.preds, colour = "Best Step"), size = 1.5) +
  geom_line(aes(y = log.bmi.poly.preds, colour = "Best Poly"), size = 1.5) +
  geom_line(aes(y = log.bmi.spline.preds, colour = "Best Natural Spline"), size = 1.5) +
  scale_colour_manual(name = "Legend:", values = cols)


other4 = ggarrange(scr.transforms, log.scr.transforms,
          age.transforms, log.age.transforms,
          cys.transforms, log.cys.transforms,
          bmi.transforms, log.bmi.transforms,
          ncol = 2, nrow = 4,
          common.legend = TRUE, legend = "top")

other4


### Rsquared table for other vars

scr.rs = c(rsquared(scr.poly.mod),
           rsquared(scr.step.mod),
           rsquared(scr.spline.mod),
           rsquared(log.scr.poly.mod),
           rsquared(log.scr.step.mod),
           rsquared(log.scr.spline.mod))

age.rs = c(rsquared(age.poly.mod),
           rsquared(age.step.mod),
           rsquared(age.spline.mod),
           rsquared(log.age.poly.mod),
           rsquared(log.age.step.mod),
           rsquared(log.age.spline.mod))

cys.rs = c(rsquared(cys.poly.mod),
           rsquared(cys.step.mod),
           rsquared(cys.spline.mod),
```

```r
           rsquared(log.cys.poly.mod),
           rsquared(log.cys.step.mod),
           rsquared(log.cys.spline.mod))

bmi.rs = c(rsquared(bmi.poly.mod),
           rsquared(bmi.step.mod),
           rsquared(bmi.spline.mod),
           rsquared(log.bmi.poly.mod),
           rsquared(log.bmi.step.mod),
           rsquared(log.bmi.spline.mod))

rs.name = c("Polynomial",
            "Step",
            "Natural Spline",
            "log Polynomial",
            "log Step",
            "log Natural Spline")

r2.data.4 = data.frame(SUN = sun.rs, SCR = scr.rs, AGE = age.rs,
                       CYS = cys.rs, BMI = bmi.rs,
                       Transformation = rs.name)

stargazer(r2.data.4,
          title = "Adjusted R2 for Each Variable and Transformation",
          header = FALSE,
          summary = FALSE,
          table.placement = 'H',
          rownames = FALSE
          )


model2 = mgcv::gam(log(GFR)~ poly(logSUN, log.sun.best.poly) +
                  ns(SCR, scr.best.spline) +
                  poly(logAGE, log.age.best.poly) +
                  ns(cys, cys.best.spline) +
                  poly(BMI, bmi.best.poly) +
                  FEMALE +
                  BLACK +
                  Diabetes, data=iod.clean)

sum.model2 = summary(model2)

stargazer(sum.model2$pTerms.table,
          summary = FALSE,
          header = FALSE,
          table.placement = 'H',
          title = "Model (2) Terms")


### Step GAM

gam.start = gam(log(GFR)~ poly(logSUN, log.sun.best.poly) +
                  ns(SCR, scr.best.spline) +
                 poly(logAGE, log.age.best.poly) +
                 ns(cys, cys.best.spline) +
                 poly(BMI, bmi.best.poly) +
                 FEMALE +
                 BLACK +
                 Diabetes, data=iod.clean)

gam.scope=list("logSUN" =~ 1 + logSUN + poly(logSUN, log.sun.best.poly) + s(logSUN, 4),
               "SCR" =~ 1 + SCR + ns(SCR, scr.best.spline) + s(SCR, scr.best.spline),
               "logAGE" =~ 1 + logAGE + poly(logAGE, log.age.best.poly) + s(logAGE, 4),
               "cys" =~ 1 + cys + ns(cys, cys.best.spline) + s(cys, cys.best.spline),
               "BMI" =~ 1 + BMI + poly(BMI, bmi.best.poly) + s(BMI,4),
               "Diabetes" =~ 1 + Diabetes,
               "FEMALE"=~1+FEMALE,
               "BLACK"=~1+BLACK)

mod = step.Gam(gam.start,gam.scope,direction="both",trace=F)


gam.drops = mod$anova %>%
  select(From, To, "Resid. Dev", AIC)

stargazer(gam.drops,
          header = FALSE,
          summary = FALSE,
          table.placement = 'H',
          title = "GAM Steps", rownames = FALSE)


model1 = lm(GFR ~ SUN + SCR + AGE + FEMALE +
                   cys + Diabetes, data = iod.clean)

model3 = mgcv::gam(log(GFR)~ s(logSUN) +
                 ns(SCR, scr.best.spline) +
                 s(logAGE) +
                 s(cys) +
                 s(BMI) +
                 FEMALE +
                 BLACK,
                 data=iod.clean)

model3a = gam(log(GFR)~ s(logSUN) +
                 ns(SCR, scr.best.spline) +
                 s(logAGE) +
                 s(cys) +
                 s(BMI) +
                 FEMALE +
                 BLACK,
                 data=iod.clean)

sum.model3a = summary.Gam(model3a)

stargazer(sum.model3a$parametric.anova,
```

```r
          summary = FALSE,
          header = FALSE,
          table.placement = 'H',
          title = "Model (3) Parametric Terms")

stargazer(sum.model3a$anova,
          header = FALSE,
          summary = FALSE,
          table.placement = 'H',
          title = "Model (3) Non-Parametric Terms")


sum.model1 = summary(model1)

final.names = c("(1) No Transformations",
                "(2) Parametric",
                "(3) Non Parametric")

final.aic = c(AIC(model1), AIC(model2), -1221.916)

final.r2 = c(sum.model1$adj.r.squared, summary(model2)$r.sq, summary(model3)$r.sq)

final.table = data.frame(Model = final.names, AIC = final.aic, adj.R2 = final.r2)

stargazer(final.table,
          header = FALSE,
          table.placement = 'H',
          rownames = FALSE,
          title = "AIC and adj R2 for Each Model",
          summary = FALSE)
```