PDA Assignment #5

Anthony Sisti

12/4/2019

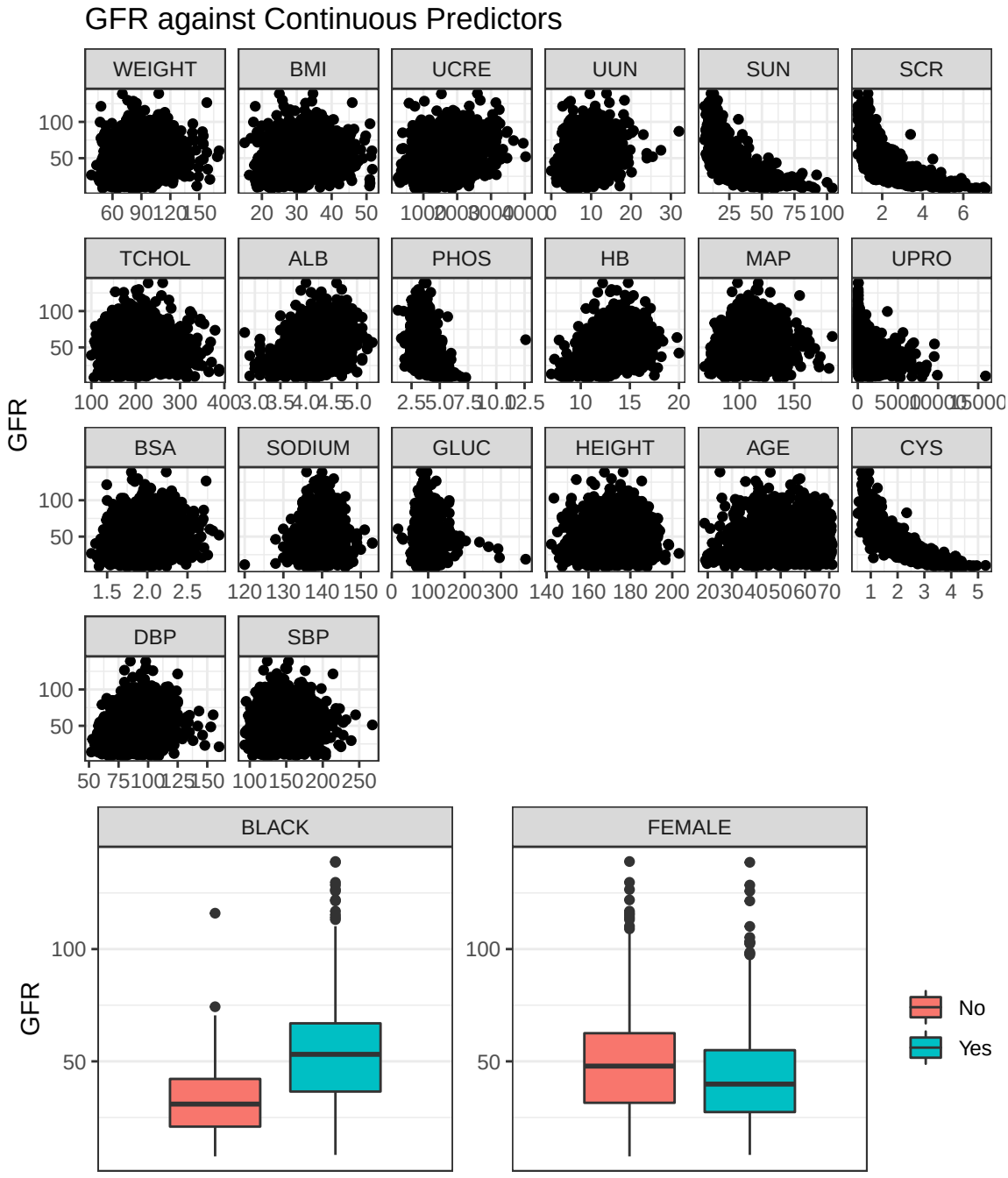
Problem 1)

Use the data from the iod.csv file you analyzed in the previous homework to create a model for GFR that considers nonlinear transformations (polynomials, step functions, splines, generalized additive models) for continuous predictors. Try each type of nonlinearfunction and give statistical support (e.g, by using a hypothesis test) to your final choice. If the function is nonparametric or hard to interpret, try to represent it using a functionthat can be interpreted straightforwardly.

Referencing the preliminary analysis conducted in Assignment 5, we only consider the complete cases from the IOD data set containing the following predictors.

- WEIGHT: (kg)
- BMI: body mass index
- GFR: glomerular filtration rate
- UCRE: urine creatinine
- UUN: urine urea nitrogen
- SUN: serum urea nitrogen
- SCR: serum creatinine
- TCHOL: total cholesterol
- ALB: albumin
- PHOS: serum phosphorus
- HB: hemoglobin
- MAP: mean arterial pressure
- UPRO: urine protein
- BSA: body surface area
- SODIUM: sodium
- GLUC: glucose
- BLACK: black race
- HEIGHT: height (cm)
- AGE: age
- FEMALE: female
- CYS: serum cystatin
- DBP: diastolic blood pressure

Below, we plot GFR, the response variable in each of our models, against all of the predictor variables. We assess the relationships presented in the visuals.



The scatter plot above shows apparent strong relationships between `GFR` and `SUN`, `SCR` and `CYS`. These relationships are certainly non-linear and will have to be examined further in model building. The side by side box-plot shows separation in GFR values for those who are black and those who are not, and also those who are female and those who are not. The relationship between `BLACK` and `GFR` appears to be stronger, with those individuals who are black having higher GFR values on average than those who are not. Both of these binary variables could be useful in model building. It also clear that there is an extreme outlier in the `PHOS` variable. Due to its egregiously high value, and the fact that we have 1561 other observations, we remove it. We now assess the correlation between predictors to see if we can select a useful subset.

The table above shows variables in the data set with a correlation above 0.7 or below -0.7. In this case, they all happen to be positive. The strongest correlation is between `WEIGHT` and `BSA`, while the fourth strongest is between `WEIGHT` and `BMI`. Due to this, we will remove `BSA` and `BMI` as predictors and keep `WEIGHT`.

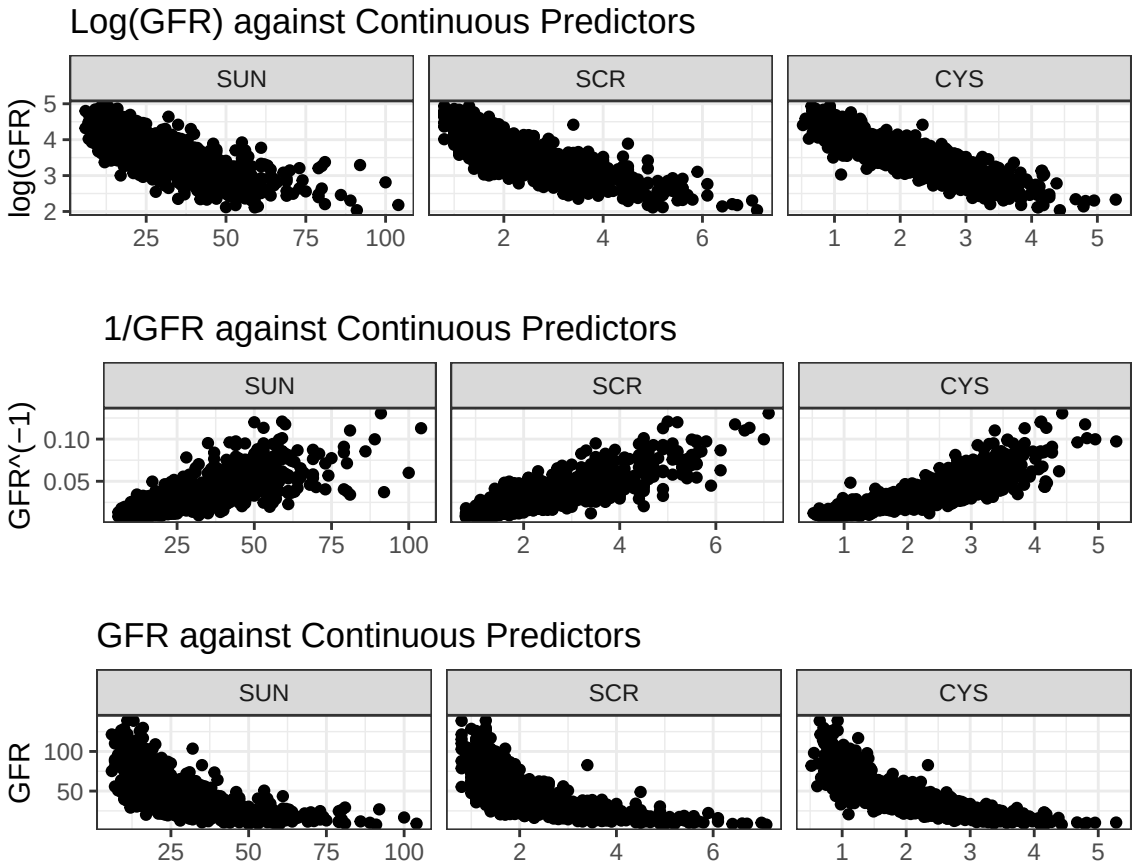
Table 1: Predictor Correlations above 0.7

	Var1	Var2	Correlation
1	WEIGHT	BSA	0.943
2	MAP	DBP	0.937
3	SBP	MAP	0.901
4	BMI	WEIGHT	0.859
5	SCR	CYS	0.845
6	SUN	CYS	0.783
7	HEIGHT	BSA	0.706

Additionally, `MAP` has a very strong correlation with both `DBP` and `SBP`. We keep `MAP` and drop `DBP` and `SBP` as predictors. Although SCR, SUN and CYS have high correlations, we leave them in the data set because they appear to have the strongest relationship with `GFR`. We leave the section between these predictors for the model building process. This leaves us 1562 observations of 19 variables.

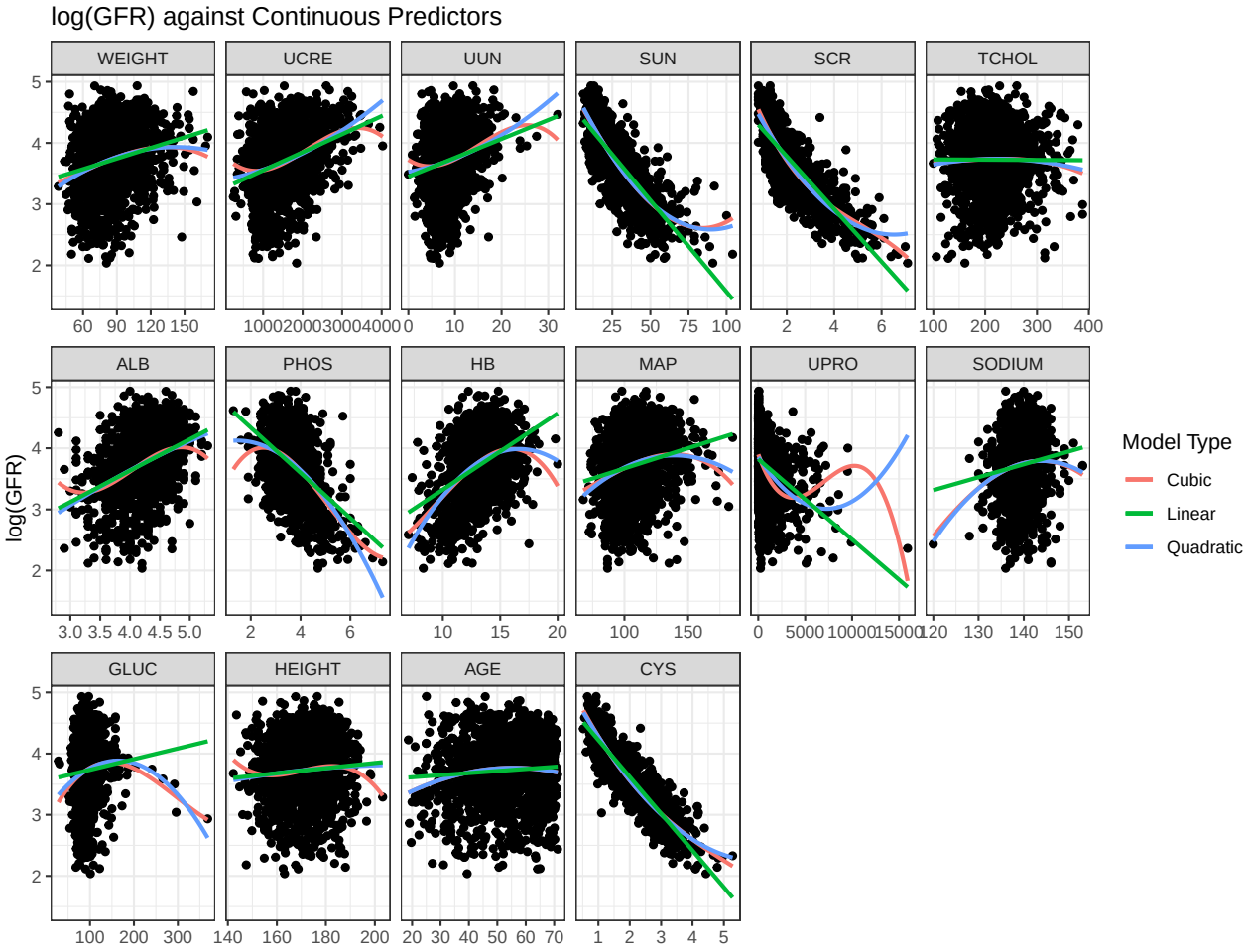
Polynomial Models

Since the relationship between `GFR` and `SUN`, `SCR` and `CYS` appear to be the strongest, but non-linear, we assess scatter plots of different polynomial and log transformations of `GFR` against each.

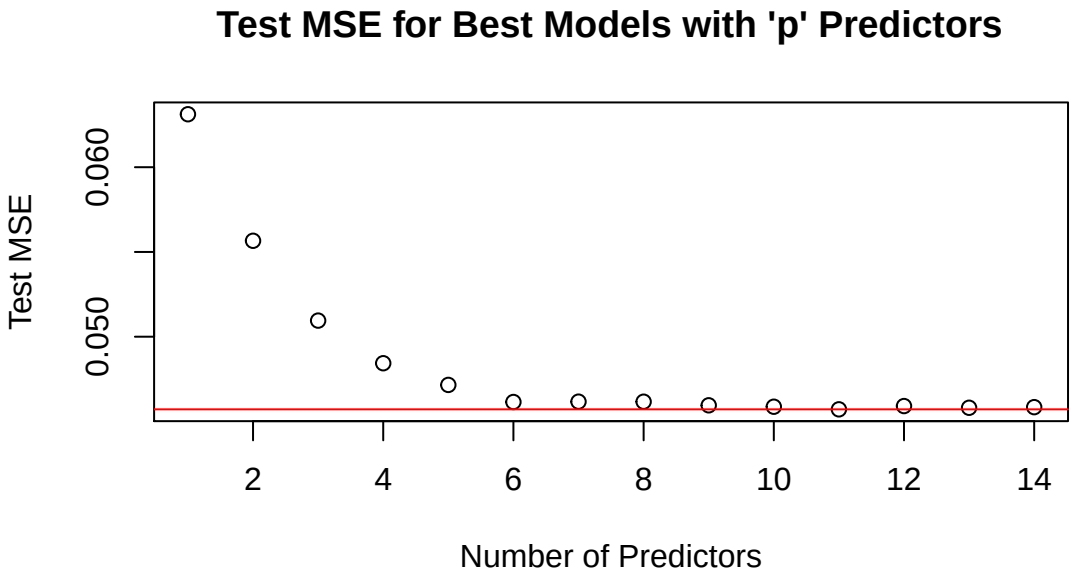


Taking the log of `GFR` appears to substantially correct the non-linearity in its apparent strongest relationships, so we use this transformation for model building. The inverse appears effective as well, but the plot of that relationship shows fanning in the data, so we would have to take the log correct this anyway. We continue

with log(GFR) as our response variable and plot it against all of the other predictors, with linear, cubic and quadratic fit lines for each.



Based on the above plots (that exclude the binary predictors), the candidates for higher order fits are `SCR`, `PHOS`, `MAP` and `HB`. For these predictors we will only consider quadratic fits. The rest of the predictors only display non-linearity in areas of sparse data, or are highly influenced by few points, causing the cubic and quadratic fits to experience high variance. We begin by dividing our data into a test and training set, and fitting a model with the predictors that are not being considered for higher order terms. From here, we will select an optimal set of variables and add the remaining predictors with their potential higher order coefficients. To select the first set of variables, we use step wise regression, fitting the best model of p predictors (p ranging from 1 to 14, AIC as selection criterion). We do this on the training set. We then select the best model based on its performance on the test set. We plot the result of this procedure below.



Based on the above graph, the model with six predictors shows significant improvement over other models of smaller size, while being close in Test MSE to all models with a greater number of predictors. Due to this, we select the model with six predictors to continue to build with. The formula for this model is

```
logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS
```

We now add the predictors with potential higher order terms to this model, and determine if they enhance the current state of the model on the training set. We show the results of regressions when adding these predictors below.

Table 2: Model Results when Predictors are Added to the Model Individually

Added Predictor	p value	Adj R^2
NONE	NA	0.850
PHOS	0.665	0.850
$PHOS^2$	0.971	0.849
MAP	0.372	0.850
MAP^2	0.559	0.850
HB	0.000	0.852
HB^2	0.688	0.852
SCR	0.000	0.870
SCR^2	0.000	0.878

The above table displays the new R^2 and coefficient p-value when each new predictor or their quadratic term is added to the base model (formula displayed earlier). While `HB` has a significant coefficient, it does not reduce the model R^2 very much. The only predictor that shows a noticeable improvement in the model's R^2 is `SCR`. The quadratic term increases R^2 even further and it's coefficient is significantly different from zero at the 0.05 level. Based on this, we decide to build onto the base model by adding the quadratic `SCR` term. The model now has the formula

```
logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + SCR +SCR^2
```

From here, we test the interactions between `FEMALE` and `BLACK`, and the the interaction between `SUN` and `CYS` which were shown to be potentially related in the first analysis we did on `GFR`. We build models with these interactions and determine if they enhance the model at all.

Table 3: Model Results with Interaction

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.878	0.079	49.309	0.000
UUN	0.012	0.002	7.283	0.000
SUN	-0.004	0.001	-5.577	0.000
ALB	0.069	0.016	4.386	0.000
BLACK	0.165	0.016	10.003	0.000
FEMALE	-0.140	0.020	-6.901	0.000
CYS	-0.274	0.016	-17.059	0.000
poly(SCR, 2)1	-6.633	0.412	-16.101	0.000
poly(SCR, 2)2	1.704	0.196	8.681	0.000
BLACK:FEMALE	0.005	0.024	0.193	0.847

Table 4: Model Results with Interaction

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	4.061	0.087	46.669	0
UUN	0.013	0.002	7.742	0
SUN	-0.010	0.002	-6.697	0
CYS	-0.360	0.024	-14.754	0
ALB	0.068	0.016	4.357	0
BLACK	0.156	0.014	11.256	0
FEMALE	-0.128	0.013	-9.739	0
poly(SCR, 2)1	-6.405	0.411	-15.579	0
poly(SCR, 2)2	1.078	0.236	4.559	0
SUN:CYS	0.003	0.001	4.660	0

The above table shows that the `BLACK:FEMALE` interaction is statistically insignificant at the 5% level, while the interaction between `SUN:CYS` is. The R^2 does not change very much, so we run an ANOVA to determine if we should keep this interaction.

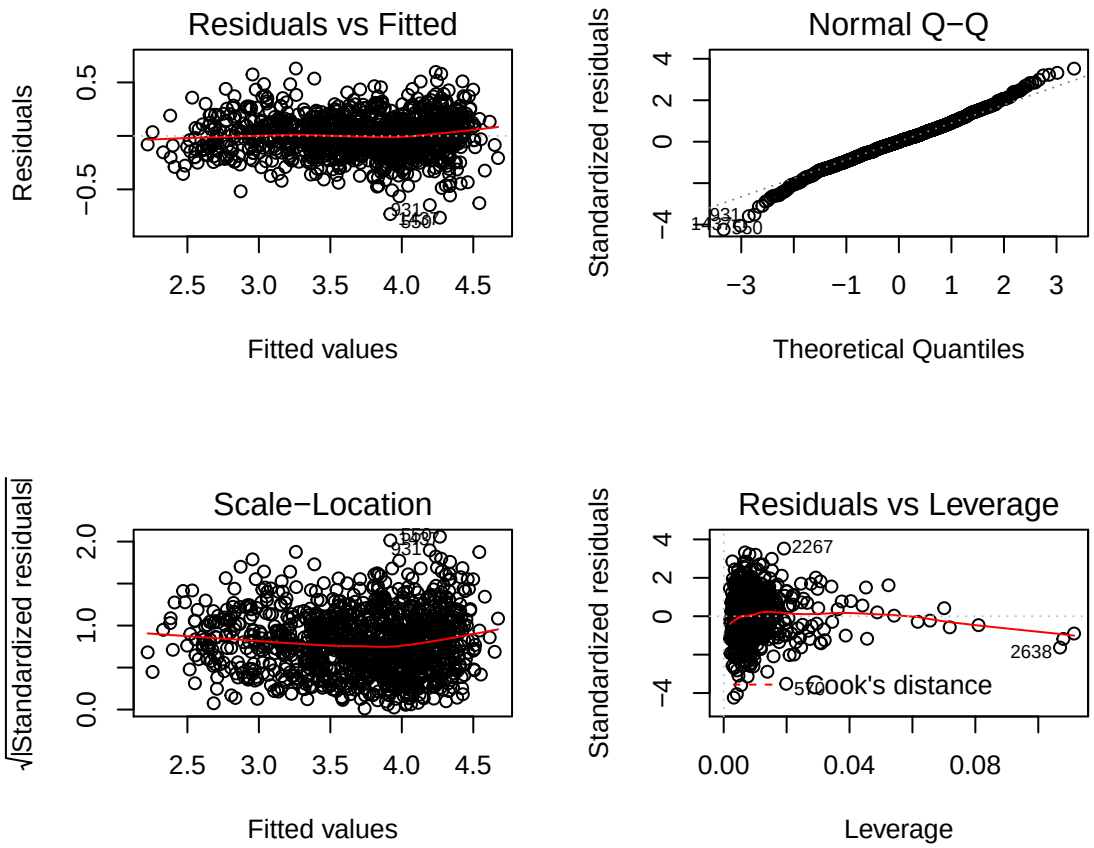
Table 5: Model Comparison with Interaction

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1161	38.497				
1160	37.79	1	0.707	21.716	0

Since the ANOVA table shows that the model with interaction is significantly better at the 5% level, we make this as our final model. The formula for this model is displayed below:

```
logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + SCR +SCR^2 + SUN:CYS
```

We now examine the residuals of this model fit.



The residuals are approximately normal and the variance is approximately constant across fitted values. There are not any particularly influential points. We decide to move forward with this as our final model for the polynomial fit. We will compare this to other models in our concluding remarks.

Step Fucntion

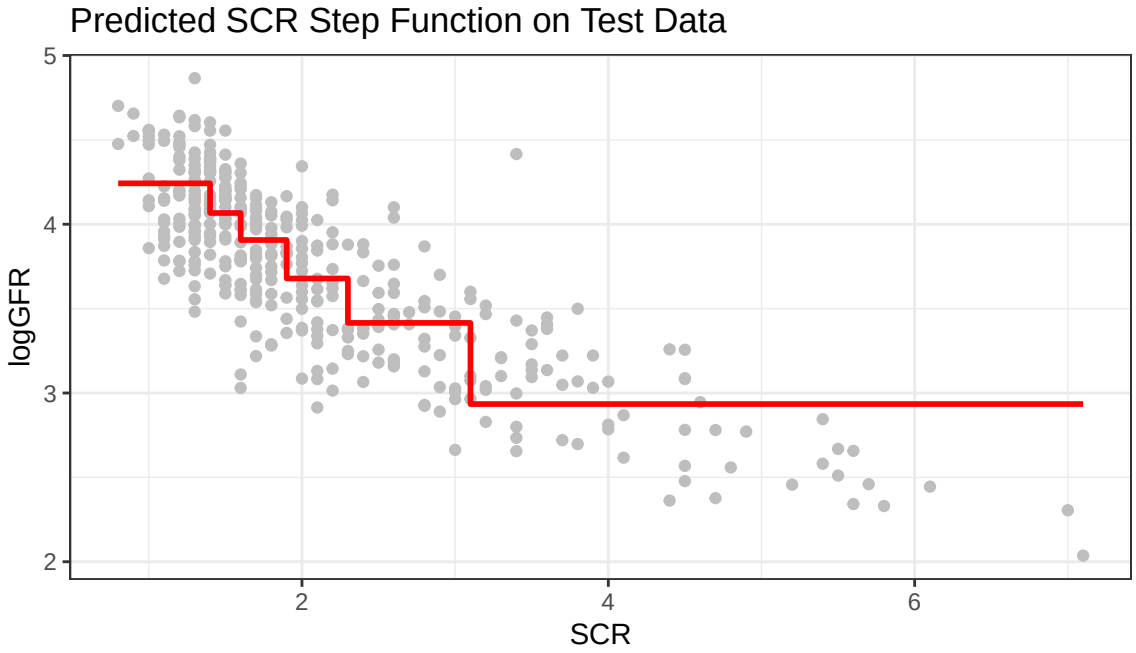
For fitting step functions, we only consider predictors that were determined to be important predictors of `logGFR` in the polynomial regression model. This allows us to focus on a smaller subset of predictors that we know can provide information on `logGFR` individually. The continuous predictors that we will consider are `UUN`, `SUN`, `ALB`, `CYS` and `SCR`. We begin by fitting a step function to predict `GFR` with `SCR`. We break the `SCR` training data into six intervals, ensuring that each of them has approximately the same number of data points. The coefficients associated with each interval are shown below.

Table 6: SCR Step Function Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
SCR_Interval[0.8,1.4)	4.242	0.022	195.663	0
SCR_Interval[1.4,1.6)	4.067	0.023	176.390	0
SCR_Interval[1.6,1.9)	3.907	0.020	200.328	0
SCR_Interval[1.9,2.3)	3.679	0.023	163.144	0
SCR_Interval[2.3,3.1)	3.415	0.022	154.328	0
SCR_Interval[3.1,7.1]	2.934	0.023	126.548	0

All of the coefficients for the step function are statistically significant at the 5% level. The model captures

the negative association between SCR and logGFR, as each interval's coefficient progressively decreases as SCR increases. To better understand the effectiveness of this model, we plot the predicted step function on top of the test data.

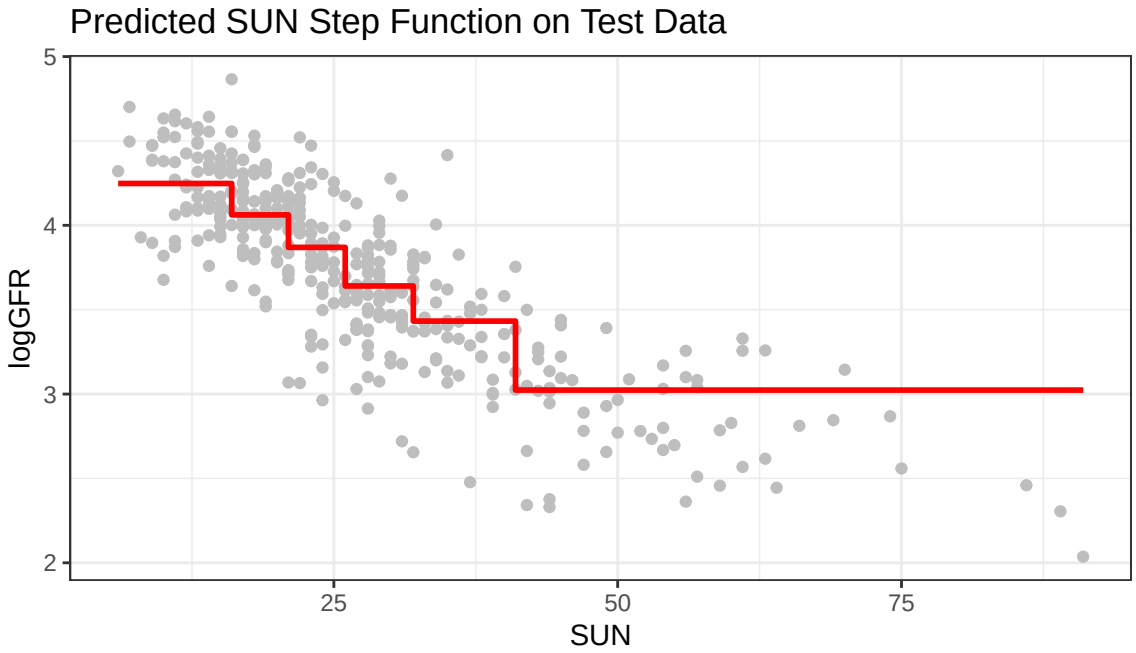


The line appears to be useful for capturing the general relationship between logGFR and SCR when the values of SCR are small. Within the interval that SCR is the largest, the model appears to greatly overestimate the logGFR values. The residuals across the latter half of this interval are entirely negative, which indicates a flaw in this prediction method. A step function would not be appropriate in this case. We now build a step function for the predictor SUN. The coefficients for this model are shown below.

Table 7: SUN Step Function Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
SUN_Interval[6, 16)	4.248	0.023	182.502	0
SUN_Interval[16, 21)	4.062	0.022	183.943	0
SUN_Interval[21, 26)	3.869	0.024	163.692	0
SUN_Interval[26, 32)	3.641	0.024	150.778	0
SUN_Interval[32, 41)	3.432	0.024	143.305	0
SUN_Interval[41,104]	3.024	0.024	126.912	0

The model results seen above are very similar to those from the SCR step function. The coefficients capture the negative associate between SUN and logGFR, albeit less drastic than what we saw between logGFR and SCR. To evaluate this fit, we plot the step function over our test data.

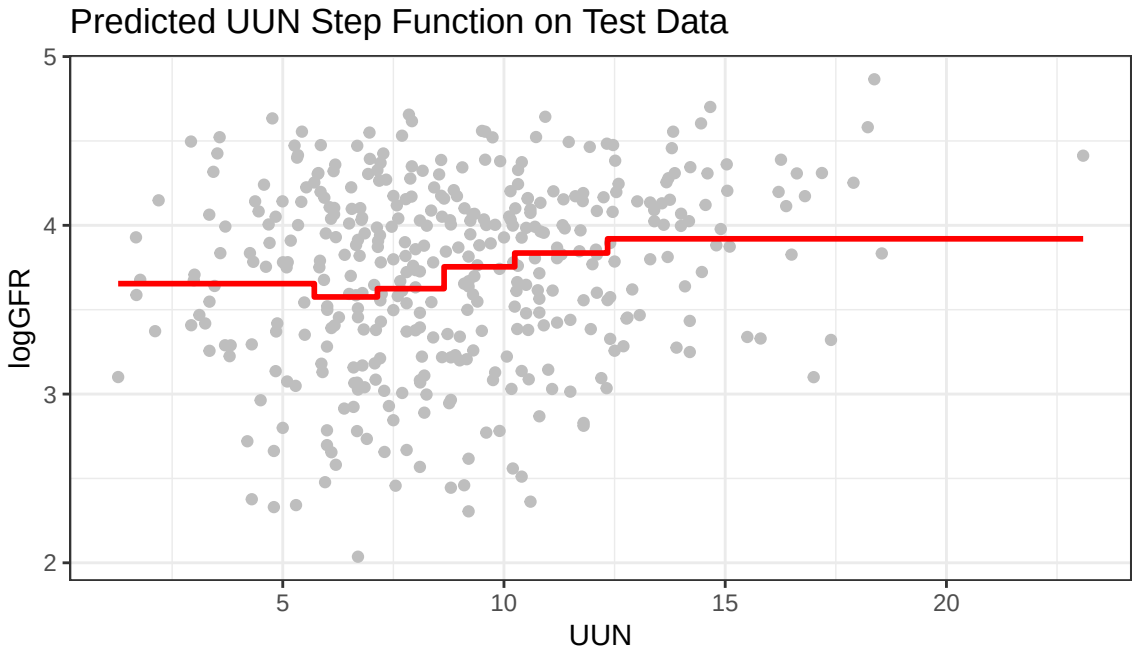


This model approximates `logGFR` values well for small values of `SUN`, but as the intervals get wider due to a lack of data, the model tends to overestimate, a similar result to what we saw for the `SCR` step function. Again, the residuals corresponding to the largest interval of `SUN` are mostly negative. We now build a step function model for `UUN`. The coefficients of this fit are shown below.

Table 8: UUN Step Function Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
UUN_Interval[0.00, 5.63)	3.654	0.036	101.626	0
UUN_Interval[5.63, 7.14)	3.575	0.037	97.928	0
UUN_Interval[7.14, 8.60)	3.625	0.037	99.273	0
UUN_Interval[8.60,10.24)	3.754	0.037	102.820	0
UUN_Interval[10.24,12.34)	3.836	0.037	104.513	0
UUN_Interval[12.34,32.03]	3.920	0.036	107.916	0

The this model fits shows less clear separation between interval coefficients than we saw with the previous predictors. The range of predicted `logGFR` across each level remains between 3.575 and 3.85, which doesn't showcase too much separation between intervals. Also, the overall trend of `logGFR` as it relates to `UUN` is less clear than it was for the previous predictors. We visualize this model over the test data below.

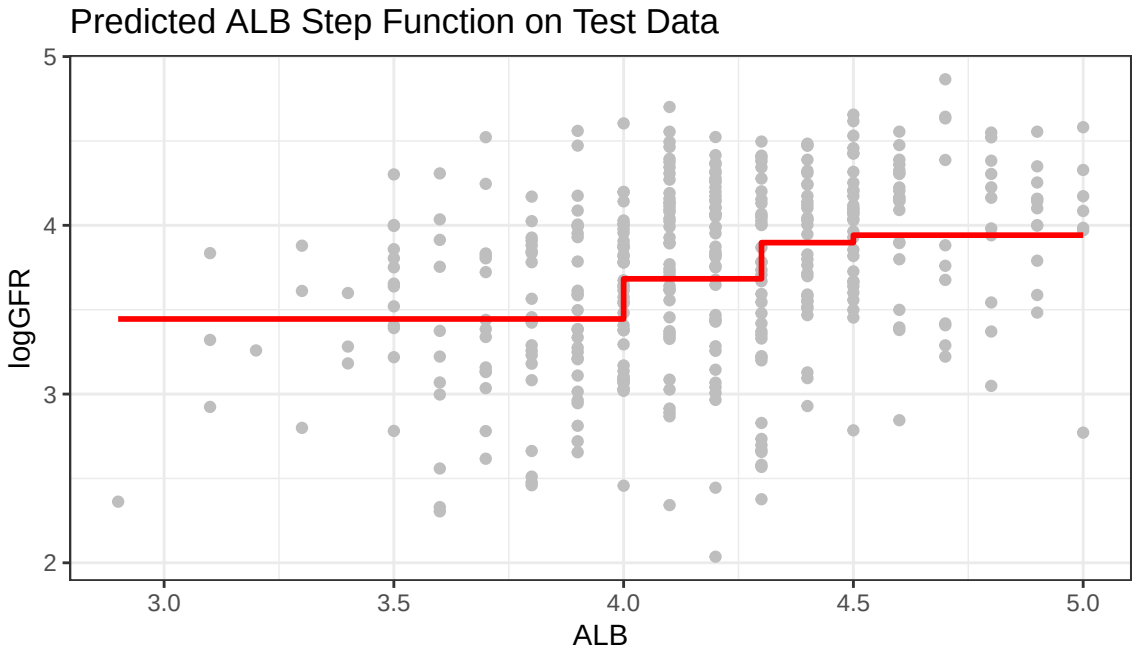


This step function captures a different trend than seen with the previous models. The relationship between `logGFR` and `UUN` is slightly positive after the first interval, and the residuals are more evenly dispersed, even for the larger values of `UUN`. Despite this, the levels of the step function do not change that drastically from step to step, and therefore do not offer that much more information than just fitting a grand mean. Next, we fit a step function for `ALB`. We use four intervals for this model since `ALB` is only documented to the tenths place, and grouping the data evenly in many intervals becomes more difficult. The coefficients are displayed below.

Table 9: ALB Step Function Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
ALB_Interval[2.8,4.0)	3.445	0.028	121.658	0
ALB_Interval[4.0,4.3)	3.683	0.026	144.322	0
ALB_Interval[4.3,4.5)	3.897	0.030	128.432	0
ALB_Interval[4.5,5.3]	3.942	0.030	130.397	0

This model shows a consistent positive association between `ALB` and `logGFR`. While the increases might not be drastic, (Coefficient Values ranging from 3.445 to 3.942), each interval has a larger coefficient than the last. To visualize this model, we plot it over the test data.

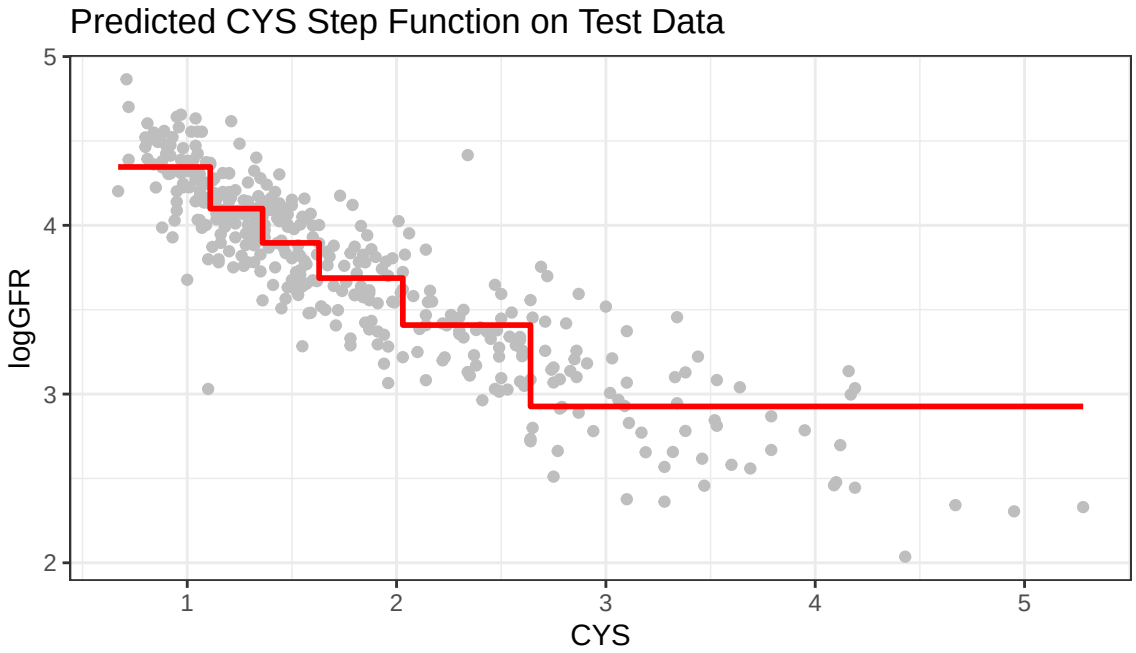


This model captures more variation across levels than the one build for `UUN`. Residuals on the test set are approximately evenly dispersed around the predictions, and the differences in predicted `logGFR` values between the first three levels are noticeably getting larger. Finally, we fit the step function to `CYS`. The coefficients are displayed below.

Table 10: CYS Step Function Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
CYS_Interval[0.52,1.11)	4.346	0.018	244.947	0
CYS_Interval[1.11,1.36)	4.099	0.018	227.428	0
CYS_Interval[1.36,1.63)	3.896	0.018	221.277	0
CYS_Interval[1.63,2.03)	3.687	0.017	211.025	0
CYS_Interval[2.03,2.64)	3.409	0.017	196.540	0
CYS_Interval[2.64,5.28]	2.927	0.018	162.383	0

This model shows a similar pattern to the `SCR` step function. There are steady decreases in predicted `logGFR` as `CYS` increases. The change in coefficients over the intervals is substantial compared to the previous few models we have considered, ranging from 4.346 to 2.927. We plot this step function over our test data below.

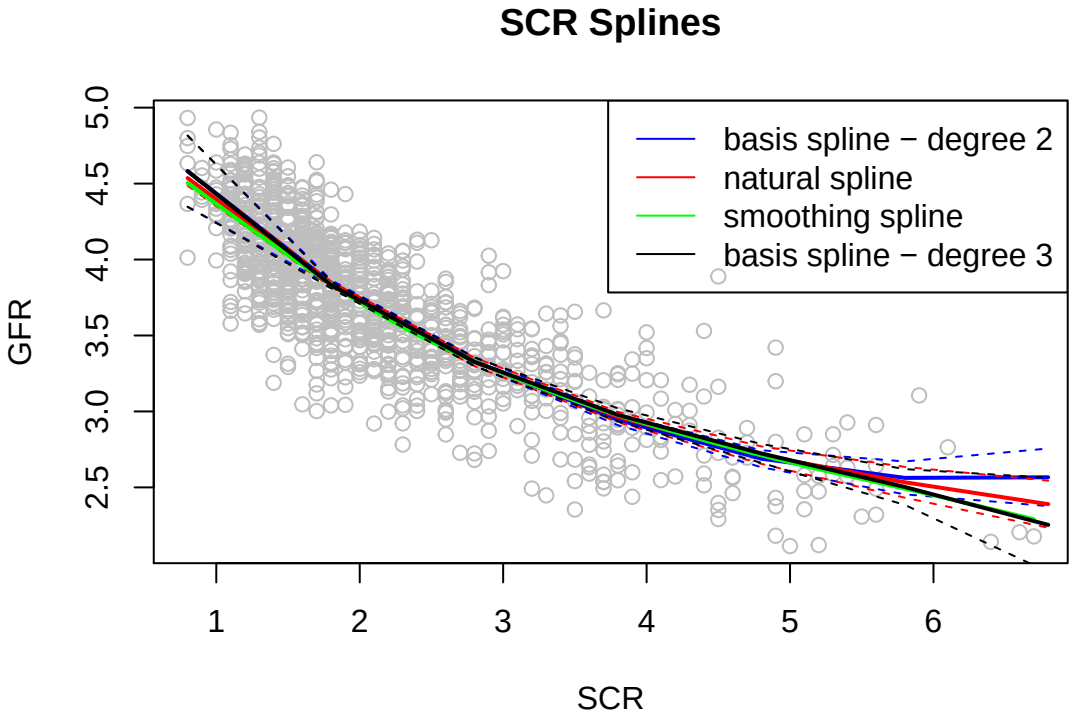


Just like the `SCR` step model, this model predicts `logGFR` better for smaller values of `CYS`. In the latter part of final interval, the residuals are almost exclusively negative, which shows this model's limitations. If we wanted to remedy this, we would have to add more intervals which could lead to overfitting.

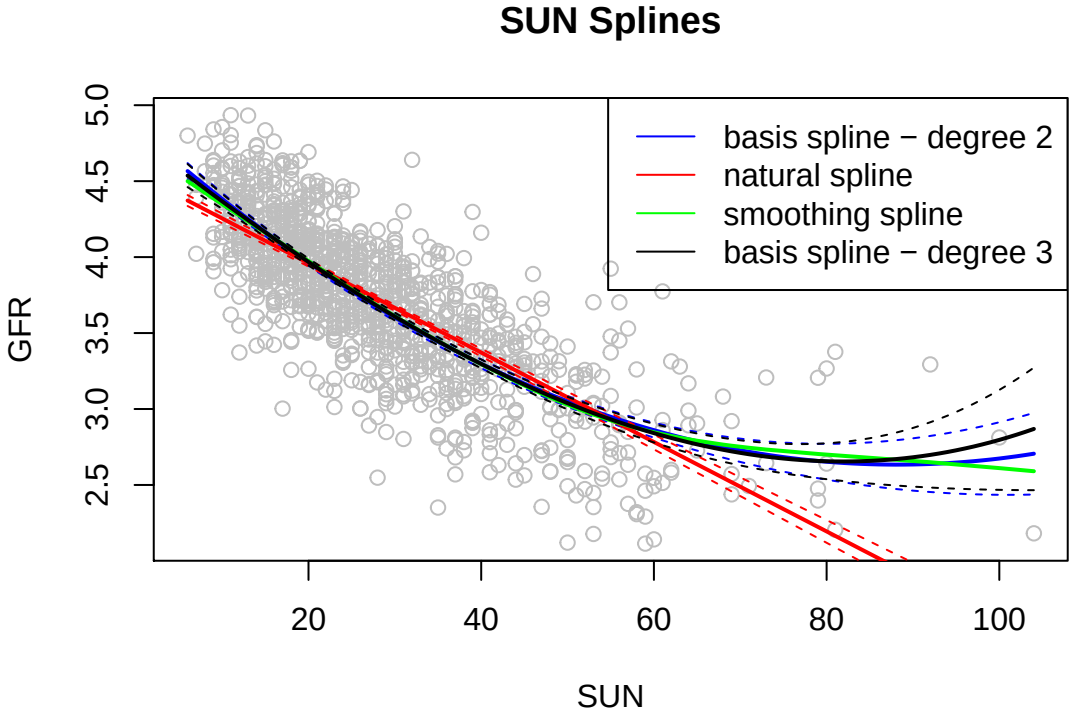
Overall, for the predictors with the strongest relationships with `logGFR`, it does not appear that step functions can capture the full pattern, overestimating `logGFR` for the highest predictor values. Due to this fact, we investigate the use of splines to capture these relationships.

Spline Fucntion

For building splines we consider possible non-linear predictors from the group selected for the polynomial regression. In that model, we only found that `SCR` was quadratic, however, we will also consider `SUN` as a potential candidate for a spline. We begin by fitting a variety of splines over the `SCR` data in order to predict `logGFR`. The resulting fits are plotted below. They include a degree 2 and 3 basis spline, a smoothed spline fit with cross validation and a natural spline fit with cross validation. Models were built on the training set, and knots in the basis spines and natural spline are were set to the 10th, 50th and 90th percentile values.



The fits of each of the splines are nearly identical, and it does appear that there is a non-linear trend in the data. Since they are so close, we adopt the natural spline for the sake of simplicity. We now fit the same set of models to the `SUN` variable, predicting `logGFR`.



The above plot shows similar results for each of the splines except for the natural spline. The natural spline for this data is just a linear fit. This aligns with our findings from the polynomial regression, as `SUN` was considered as a linear predictor only. The plot also shows that the other set of splines are reacting to fit on the sparsely spread data for large values of `SUN`. To avoid this, we decide that `SUN` will be looked at as a linear predictor. In our building of GAMs in the next section, we will consider adding the natural spline that appeared effective for capturing the relationship between `SCR` and `logGFR`.

GAMs

The variable `SCR` has consistently been more useful when considered to be a non-linear predictor of `logGFR`. Therefore, we build a GAM that fits a smooth function to `SCR` and a GAM that fits a natural spline to `SCR` (as seen above) in order to predict `logGFR`. We then compare the model fits and decide which has the best fit. The model formulas that were used for fitting are shown below.

GAM with smooth function of `SCR`:

```
logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + SUN:CYS + s(SCR)
```

GAM with natural spline of `SCR`:

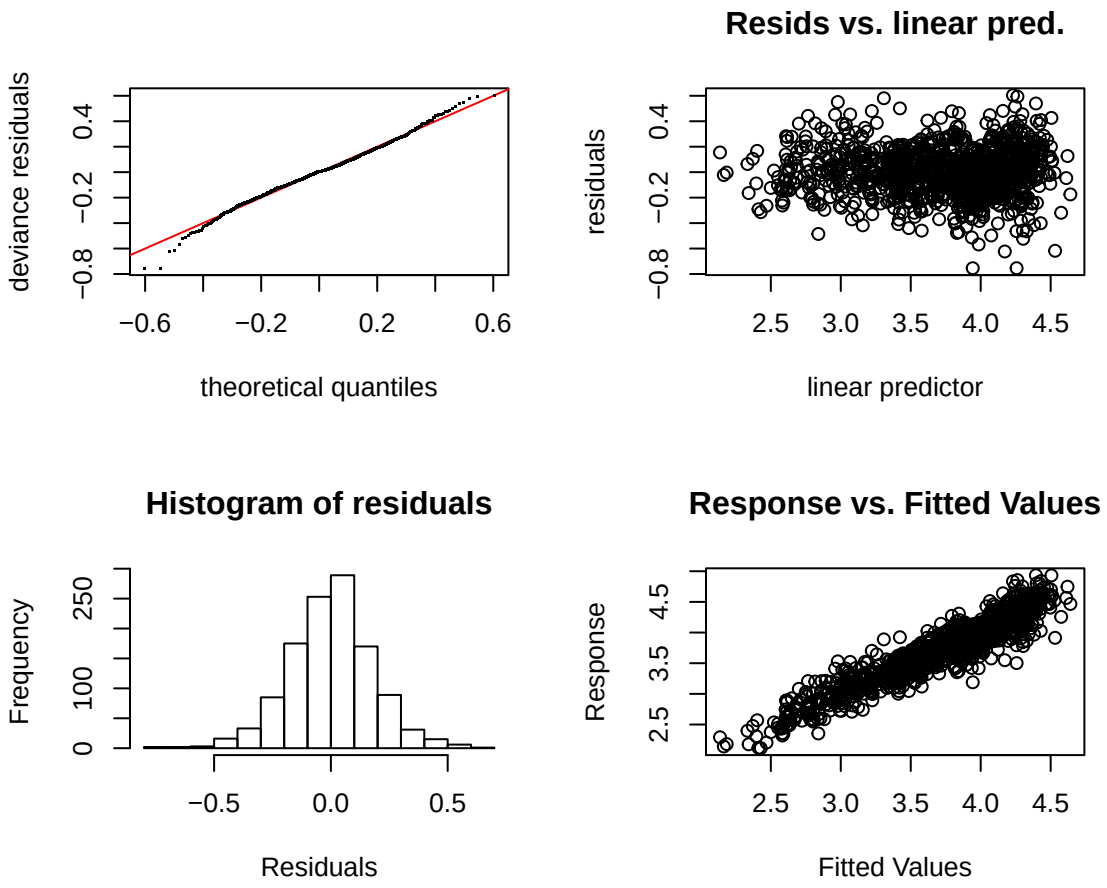
```
logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + SUN:CYS + ns(SCR, knots = c(0.1, 0.5,
0.9))
```

The BIC of these fits on the training data is show in the following table:

Table 11: GAM Comparison

GAM	BIC
Smooth SCR	-603.656
Spliine SCR	-616.649

Based on the value of BIC between the two models, the one which includes the natural spline of `SCR` appears to fit the data better. We analyze the residuals of this model below.



The residuals appear to be normally distributed with constant variance, outside of a few points, across the linear predicted values. Since this model satisfies the regression assumptions, we move forward with it as a final model.

We also examine a model that is developed via step wise selection, allowing all of the variables in the model to possibly be smoothed, or be fit with a natural spline, if it makes the model have a lower AIC. While this does allow us to test all model combinations, it removes our intuition about which predictors should just be linear effects based on the previous data exploration from the model building process. We again consider only the predictors that were chosen for our polynomial fit, and allow them to be possibly included as smoothed predictors in the GAM. The model is fit with the following formula:

```
logGFR ~ s(SCR) + s(CYS) + s(SUN) + s(UUN) + ALB + FEMALE + BLACK
```

The Analysis of Variance and AIC progression as predictors were changed from linear effects to smoothed is shown below.

Table 12: Stepwise Selection GAM ANOVA

From	To	Df	Deviance	Resid. Df	Resid. Dev	AIC
	<start>			1162	41.016	-582.113
SCR	s(SCR)	-3	-3.614	1159	37.402	-684.033
CYS	s(CYS)	-3	-0.881	1156	36.521	-705.927
UUN	s(UUN)	-3	-0.306	1153	36.215	-709.758
SUN	s(SUN)	-3	-0.211	1150	36.005	-710.586

The results show that changing the effect of `SCR` , `CYS`, `SUN` and `UUN` to smooth functions decreases the AIC

of the GAM on the training data. This is a departure from the previous construction of the GAM, where we found the best model included `SCR` as a natural spline, and the rest of the variables as linear effects. We will compare this final model to the manually fit GAM and Polynomial Regression.

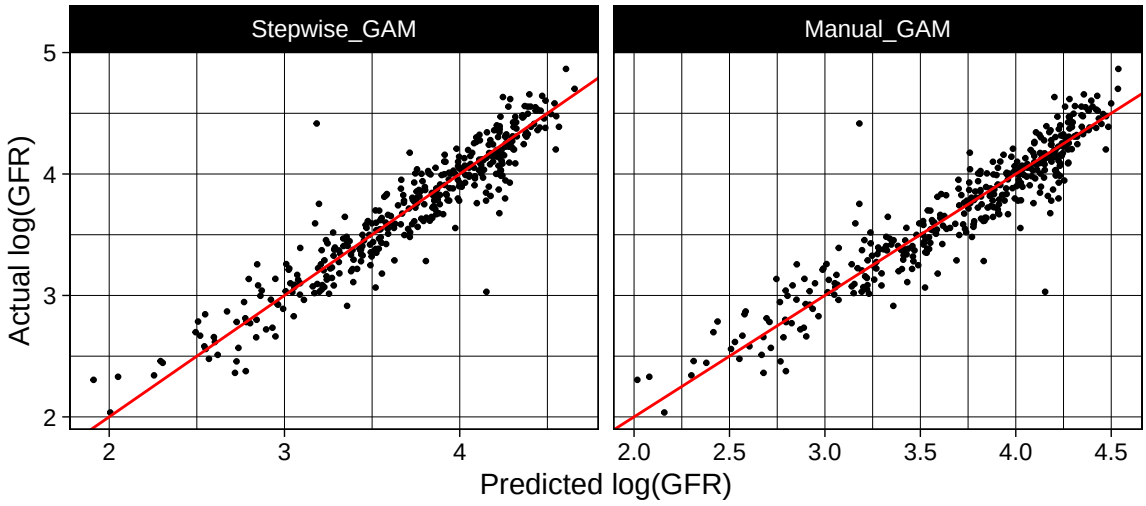
Model Comparison

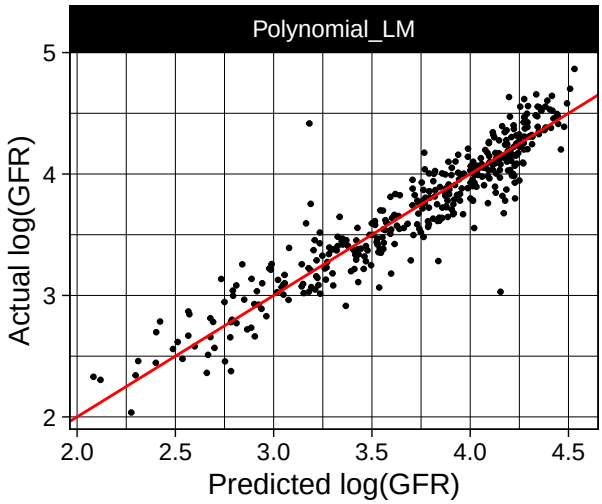
In the previous sections, we have developed a polynomial regression model, a manually fit GAM and step wise fit GAM. We compare these models based on their performance on the test set, and training set, in order to choose the best one. Below, we display a table summarizing some performance metrics.

Table 13: Model Comparison

Model	Training BIC	Test MSE	Test R^2
Stepwise GAM	-665.003	0.0341	0.883
Manual GAM	-616.649	0.0355	0.878
Polynomial Regression	-596.594	0.0360	0.877

The model comparison above shows that the step wise GAM performed much better on the training set than the other models. This is to be expected, as the step wise GAM was optimized using AIC on the training set as a metric. It also performed slightly better on the test set when using test MSE and test R^2 as metrics. Overall, the polynomial fit performed the worst in each of the measurements we used, but was much closer to equivalent performance on the test set than on the training set. We now examine the fitted against actual values for each model





As the close test R^2 and test MSE values suggested, there are no glaring differences between the residual structures of any of the three models. Each performs well, and offers a lot of information about the variability in the test set. Although the step wise GAM performs much better on the training data, we lose a lot of interpretability when we include four smooth functions of predictors in the same model. Since it was only marginally better than the other two models on the test data, we rule it out from our final model selection. For all of our metrics, the manually fit GAM performed better than the polynomial fit. While less interpretable than the polynomial fit, it only includes one smooth function of `SCR` and leaves us with enough interpretability to justify taking the gain in performance on the training and test data. Because of this, we decide that the manually fit GAM is the 'best' model of the three, and we would move forward with it as our final selection.

PhD Problems

Problem 2)

Let

$$f(x) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \theta_1 (X - \xi_1)_+^3 + \theta_2 (X - \xi_2)_+^3$$

We have to show that $f(x)$ is continuous with continuous first and second derivatives. Consider the case when $X < \xi_1$. Then

$$
\begin{aligned}
f(x) &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 \\
f'(x) &= \beta_1 + 2\beta_2 X + 3\beta_3 X^2 \\
f''(x) &= 2\beta_1 + 6\beta_3 X
\end{aligned}
$$

Clearly reach of these functions is continuous. Now consider the case when $\xi_1 < X < \xi_2$. Then

$$
\begin{aligned}
f(x) &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \theta_1 (X - \xi_1)^3 \\
f'(x) &= \beta_1 + 2\beta_2 X + 3\beta_3 X^2 + 3\theta_1 (X - \xi_1)^2 \\
f''(x) &= 2\beta_2 + 6\beta_3 X + 6\theta (X - \xi_1)
\end{aligned}
$$

Again, these functions are clearly continuous. Finally consider $\xi_2 < X$.

$$
\begin{aligned}
f(x) &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \theta_1 (X - \xi_1)^3 + \theta_2 (X - \xi_2)^3 \\
f'(x) &= \beta_1 + 2\beta_2 X + 3\beta_3 X^2 + 3\theta_1 (X - \xi_1)^2 + 3\theta_1 (X - \xi_2)^2 \\
f''(x) &= 2\beta_2 + 6\beta_3 X + 6\theta (X - \xi_1) + +6\theta (X - \xi_2)
\end{aligned}
$$

These functions are clearly continuous. We must now consider the potential points of discontinuity: $X = \xi_1$, $X = \xi_2$. For the value of the function at $X = \xi_1$, and for the limit of these functions as $X \to \xi_1$ from the left and from the right,

$$
\begin{aligned}
f(\xi_1) &= \beta_0 + \beta_1 \xi_1 + \beta_2 \xi_1^2 + \beta_3 \xi_1^3 \\
f'(\xi_1) &= \beta_1 + 2\beta_2 \xi_1 + 3\beta_3 \xi_1^2 \\
f''(\xi_1) &= 2\beta_1 + 6\beta_3 \xi_1
\end{aligned}
$$

Since the limits from both sides and the functions themselves all take the same value at the point $X = \xi_1$, the functions are all continuous at $X = \xi_1$. Similarly, considering the limits from the left and the right and the function values at $X = \xi_2$ we have

$$
\begin{aligned}
f(\xi_2) &= \beta_0 + \beta_1 \xi_2 + \beta_2 \xi_2^2 + \beta_3 \xi_2^3 + \theta_1 (\xi_2 - \xi_1)^3 \\
f'(\xi_2) &= \beta_1 + 2\beta_2 \xi_2 + 3\beta_3 \xi_2^2 + 3\theta_1 (\xi_2 - \xi_1)^2 \\
f''(\xi_2) &= 2\beta_2 + 6\beta_3 \xi_2 + 6\theta (\xi_2 - \xi_1)
\end{aligned}
$$

Again, since the limit from both the left and the right, and the functions themselves take the same value at $X = \xi_2$, the functions are all continuous at $X = \xi_2$. We have shown that the piece-wise cubic polynomials are continuous and have continuous first and second derivatives as desired.

Problem 3)

We first deal with showing (5.71). Since ξ_1 is the smallest knot, when $X < \xi_1$ we are only dealing with the first sum of $f(x)$. But on this region, $f(x)$ should be linear, so within the sum $\sum_{j=0}^{3} \beta_j X^j$, we know that $\beta_2 = \beta_3 = 0$. We take a similar approach when considering $X > \xi_K$ the largest knot value. In this case, we have all X terms of $f(x)$ taking non-zero value, so we consider $\sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} (X - \xi_k)^3$. We know this

region must be linear as well. Looking at the second sum, we have that $(X-\xi_k)^3 = X^3 - 3\xi_k X^2 + 3\xi_k^2 X - \xi_k^3$. In order to maintain linearity we need $\beta_2 = \sum_{k=1}^{K} 3\xi_k\theta_k$ and we need $\beta_3 = -\sum_{k=1}^{K}\theta_k$. Thus, we know that $\sum_{k=1}^{K} 3\xi_k\theta_k = 0$ and $\sum_{k=1}^{K}\theta_k = 0$ as desired.

We now must show the the definition of the cubic splines by truncated power series representation and natural boundary conditions is equivalent to the definition by the basis functions given in the question. We will derive the basis functions using the truncated power series representation and natural boundary conditions. Assuming this definition, The coefficients of $N_1(X)$ and $N_2(X)$ are β_0 and β_1 respectively. The coefficient for $N_{k+2}(X)$ is $(\xi_K - \xi_k)\theta_k$ for $K-2 \geq k$. Considering only the coefficients of $N_{k+2}(X)$ we have

$$\sum_{k=1}^{K-2}(\xi_K-\xi_k)\theta_k\left[\frac{(X-\xi_k)_+^3-(X-\xi_K)_+^3}{\xi_K-\xi_k}-\frac{(X-\xi_{K-1})_+^3-(X-\xi_K)_+^3}{\xi_K-\xi_{K-1}}\right] =$$
$$\sum_{k=1}^{K-2}\theta_k(X-\xi_k)_+^3-\sum_{k=1}^{K-2}\theta_k(X-\xi_K)_+^3-$$
$$\frac{1}{\xi_K-\xi_{K-1}}\left[\sum_{k=1}^{K-2}\xi_K\theta_k-\sum_{k=1}^{K-2}\xi_k\theta_k\right]\left[(X-\xi_{K-1})_+^3-(X-\xi_K)_+^3\right]$$

Using the boundary conditions we derived in the first part, we can show

$$\sum_{k=1}^{K-2}\theta_k = -\theta_{K-1}-\theta_K \quad \text{and}$$

$$\sum_{k=1}^{K-2}\xi_k\theta_k = -\xi_{K-1}\theta_{K-1}-\xi_K\theta_K$$

Using these facts, we have

$$\sum_{k=1}^{K-2}\theta_K(X-\xi_K)_+^3+(\theta_{K-1}+\theta_K)(X-\xi_K)_+^3-\frac{1}{\xi_K-\xi_{k-1}}*$$
$$[\xi_{K-1}\theta_{K-1}+\xi_K\theta_K-\xi_K\theta_{K-1}-\xi_K\theta_K]\left[(X-\xi_{K-1})_+^3-(X-\xi_K)_+^3\right] =$$
$$\sum_{k=1}^{K-2}\theta_k(X-\xi_k)_+^3+(\theta_K+\theta_{K-1})(X-\xi_K)_+^3-\frac{\xi_{K-1}\theta_{K-1}-\xi_K\theta_{K-1}}{\xi_K-\xi_{K-1}}\left[(X-\xi_{K-1})_+^3-(X-\xi_K)_+^3\right] =$$
$$\sum_{k=1}^{K-2}\theta_k(X-\xi_k)_+^3+\theta_K(X-\xi_K)_+^3+\theta_{K-1}(X-\xi_K)_+^3-\theta_{K-1}(X-\xi_K)_+^3+\theta_{K-1}(X-\xi_{K-1})_+^3 =$$
$$\sum_{k=1}^{K}\theta_k(X-\xi_K)_+^3$$

Where the last step is valid because the middle terms cancel and the remaining terms outside of the sum can be included by changing the upper summation limit to K. This completes the desired derivation.

For reference when stuck I consulted:

Elements of Statistical Learning. Weathermax, J., Epstein, D. 28 October 2019. *weatherworksmath.com*

Code Appendix

```
knitr::opts_chunk$set(echo = F)
knitr::opts_chunk$set(message = F)
knitr::opts_chunk$set(warning = F)
knitr::opts_chunk$set(fig.height = 3.5)
knitr::opts_chunk$set(fig.width = 6)
knitr::opts_chunk$set(fig.align="center")
`%notin%` <- Negate(`%in%`)
library(tidyverse)
library(caret)
library(leaps)

IOV = read.csv('iodatadev.csv')[,-1]

#Rename cols to match those in assignment
IOV = IOV %>% dplyr::rename(UPRO = upro, DIAB = Diabetes , CYS = cys,
                            HBPSTATUS = hbpstatus)

#select columns specified in assignment
ModelCols = c('WEIGHT', 'BMI', 'GFR', 'UCRE', 'UUN', 'UPHO', 'SUN','SCR',
              'TCHOL', 'ALB', 'HBA1C', 'PHOS', 'TRIG', 'LDL', 'HDL', 'HB',
              'MAP', 'UPRO', 'BSA', 'SODIUM', 'GLUC', 'BLACK', 'HEIGHT', 'AGE',
              'FEMALE', 'CYS', 'DBP', 'SBP', 'CRP', 'DIAB', 'HBPSTATUS')

IOV = IOV %>% dplyr::select(ModelCols)


library(fBasics)
library(kableExtra)
#Check how many complete cases we have
#sum(complete.cases(IOV))/nrow(IOV)

#Look at particularly sparse columns
CheckNA = IOV[, c("UPHO", "HBA1C", "TRIG", "LDL", "HDL", "CRP", "HBPSTATUS")]

#See how many NAs they have
# summary(CheckNA)

NAsummary = round(basicStats(CheckNA)[c("Mean", "Stdev",  "Minimum", "1. Quartile", "Median", "3. Quarti

NAsummary[8,] = as.character(as.integer(NAsummary[8,]))

rownames(NAsummary)[c(4,6)] = c("Q1", "Q3")

kable_styling(kable(NAsummary, escape = F, format = "latex", booktabs= TRUE, align = 'c'
                    ), latex_options = "HOLD_position")

#Pick columns that arent in the sparse set
# GoodCols= which(colnames(IOV) %notin% c(colnames(CheckNA), "SBP"))
GoodCols = which(colnames(IOV) %notin% c(colnames(CheckNA)))

#sum(complete.cases(IOV[,GoodCols]))/nrow(IOV)
```

```
ReducedIOV = IOV[,GoodCols]
# sum(!complete.cases(ReducedIOV))

CompleteCases = which(complete.cases(ReducedIOV))
# print("Removed Cases")
# summary(ReducedIOV[-CompleteCases,c("UUN", "UCRE", "DIAB")])
# print("----------------------------------------")
# print("Remaining Cases")
# summary(IOV[CompleteCases,c("UUN", "UCRE", "DIAB")])

RC = ReducedIOV[-CompleteCases,c("UUN", "UCRE", "DIAB")]
CC = ReducedIOV[CompleteCases,c("UUN", "UCRE", "DIAB")]

completeSummary = round(basicStats(CC)[c("Mean", "Stdev", "Median", "Minimum", "1. Quartile", "3. Quarti

rownames(completeSummary)[c(5,6)] = c("Q1", "Q3")

completeSummary[8,] = as.character(as.integer(completeSummary[8,]))

kable_styling(kable(completeSummary, escape = F, format = "latex", booktabs= TRUE, align = 'l', caption
                    ), latex_options = "HOLD_position")

removedSummary = round(basicStats(RC)[c("Mean", "Stdev", "Median", "Minimum", "1. Quartile", "3. Quartil

rownames(removedSummary)[c(5,6)] = c("Q1", "Q3")
removedSummary[8,] = as.character(as.integer(removedSummary[8,]))

kable_styling(kable(removedSummary, escape = F, format = "latex", booktabs= TRUE, align = 'l', caption =
                    ), latex_options = "HOLD_position")


NewIOV = ReducedIOV[CompleteCases, -which(colnames(ReducedIOV)=="DIAB")]

library(reshape2)
ScatterIOV <- melt(NewIOV, "GFR")
ScatterIOVcont = ScatterIOV[which(ScatterIOV$variable %notin% c("BLACK", "FEMALE")),]
ScatterIOVbinary = ScatterIOV[which(ScatterIOV$variable %in% c("BLACK", "FEMALE")),]
library(ggplot2)
# ScatterIOV$Check = factor(ifelse(ScatterIOV$GFR == ScatterIOV$GFR[530], 1, 0))
p1 <- ggplot(ScatterIOVcont, aes(value, GFR)) +  geom_point() +
  facet_wrap(.~variable, ncol = 6, nrow = 4, scales = "free_x") + theme_bw() +
  ggtitle("GFR against Continuous Predictors")+ xlab("")

p1
p2 <- ggplot(ScatterIOVbinary, mapping = aes(x = value, y = GFR, group = factor(value), fill = factor(va
  geom_boxplot()+ facet_wrap(~variable, scales = 'free') +
  theme_bw() + scale_x_discrete() + labs(fill='') + xlab("")
p2
library(data.table)

PolyIOV = NewIOV
PolyIOV$GFR = 1/PolyIOV$GFR
```

```
Polycor = round(cor(PolyIOV[,-which(colnames(PolyIOV) == "GFR")]), 3)

MaxCors = setDT(melt(Polycor))[Var1 != Var2, .SD[which.max(abs(value))], keyby=Var1]
colnames(MaxCors)[3] <- c("Correlation")



MaxCors = MaxCors[order(abs(MaxCors$Correlation), decreasing = T),]
MaxCors = MaxCors[which(abs(MaxCors$Correlation)>0.7)]
MaxCors = MaxCors[c(1,3,5,6,7,9,10),]

kable_styling(kable(MaxCors, escape = F, format = "latex", booktabs= TRUE, align = 'c',
                    caption = "Predictor Correlations above 0.7", row.names = T))

# which(NewIOV$PHOS >12)
NewIOV = NewIOV[-which(NewIOV$PHOS >12),-which(colnames(NewIOV) %in% c("BSA","BMI", "DBP", "SBP") )]

ScatterIOVGFR_trans = ScatterIOV[which(ScatterIOV$variable %in% c("SUN","SCR", "CYS")),]

p3 <- ggplot(ScatterIOVGFR_trans, aes(value, log(GFR))) +  geom_point() +
  facet_wrap(.~variable, ncol = 6, nrow = 4, scales = "free_x") + theme_bw() +
  ggtitle("Log(GFR) against Continuous Predictors")+ xlab("")

p4 <- ggplot(ScatterIOVGFR_trans, aes(value, GFR^(-1))) +  geom_point() +
  facet_wrap(.~variable, ncol = 6, nrow = 4, scales = "free_x") + theme_bw() +
  ggtitle("1/GFR against Continuous Predictors")+ xlab("")

p5 <- ggplot(ScatterIOVGFR_trans, aes(value, GFR)) +  geom_point() +
  facet_wrap(.~variable, ncol = 6, nrow = 4, scales = "free_x") + theme_bw() +
  ggtitle("GFR against Continuous Predictors")+ xlab("")

p6 <- ggplot(ScatterIOVGFR_trans, aes(value, GFR^2)) +  geom_point() +
  facet_wrap(.~variable, ncol = 6, nrow = 4, scales = "free_x") + theme_bw() +
  ggtitle("GFR^2 against Continuous Predictors")+ xlab("")

p7 <- ggplot(ScatterIOVGFR_trans, aes(value, GFR^3)) +  geom_point() +
  facet_wrap(.~variable, ncol = 6, nrow = 4, scales = "free_x") + theme_bw() +
  ggtitle("GFR^3 against Continuous Predictors")+ xlab("")

gridExtra::grid.arrange(p3,p4,p5, nrow = 5, ncol = 1)
ScatterIOV <- melt(NewIOV, "GFR")
ScatterIOVcont = ScatterIOV[which(ScatterIOV$variable %notin% c("BLACK", "FEMALE")),]
ScatterIOVbinary = ScatterIOV[which(ScatterIOV$variable %in% c("BLACK", "FEMALE")),]

# ggplot(ScatterIOVcont, aes(value, log(1/GFR))) +  geom_point() +
#   facet_wrap(.~variable, ncol = 6, nrow = 4, scales = "free_x") + theme_bw() +
#   ggtitle("log(1/GFR) against Continuous Predictors")+ xlab("") + geom_smooth(method = "lm")+
#   geom_smooth(method = 'lm', formula = y ~ poly(x,2), color = "red" ) +
#   geom_smooth(method = 'lm', formula = y ~ poly(x,3), color = "green" )

ggplot(ScatterIOVcont, aes(value, log(GFR))) +  geom_point() +
  facet_wrap(.~variable, ncol = 6, nrow = 3, scales = "free_x") + theme_bw() +
  ggtitle("log(GFR) against Continuous Predictors")+ xlab("") +
```

```
  geom_smooth(method = 'lm', formula = y ~ poly(x,3), se = F , aes(color = "Cubic")) +
    geom_smooth(method = 'lm', formula = y ~ poly(x,2), se = F , aes(color = "Quadratic")) +
  geom_smooth(method = "lm", se = F , aes(color = "Linear")) + labs(color = "Model Type")


NewIOVLinear = NewIOV[, -which(colnames(NewIOV) %in% c("SCR", "PHOS", "MAP", "HB"))]
NewIOVLinear$GFR = log(NewIOVLinear$GFR)
colnames(NewIOVLinear)[which(colnames(NewIOVLinear)=="GFR")] = "logGFR"


set.seed(1234)
Step_Train = sample(1:nrow(NewIOVLinear), 0.75*nrow(NewIOVLinear))

library(leaps)
bestmodels = regsubsets(logGFR~., data = NewIOVLinear[Step_Train,], nvmax = 14)
summ_bestmodels = summary(bestmodels)

get_model_formula <- function(id, object, outcome){
  models <- summary(object)$which[id,-1]
  predictors <- names(which(models == TRUE))
  predictors <- paste(predictors, collapse = "+")
  as.formula(paste0(outcome, "~", predictors))
}
# function taken from a reference page posted by Alboukadel Kassambara, PhD
# website http://www.sthda.com/

MSEs = rep(NA, 14)

for(i in 1:14){
  model = get_model_formula(i, bestmodels, "logGFR")
  fit = lm(model, data = NewIOVLinear[Step_Train,])
  y_resp = NewIOVLinear[-Step_Train, "logGFR"]
  y_preds = predict(fit, newdata = NewIOVLinear[-Step_Train,])
  error = mean((y_resp - y_preds)^2)
  MSEs[i] = error
}

plot(1:14, MSEs, ylab= "Test MSE", xlab = "Number of Predictors", main = "Test MSE for Best Models with
abline(h=min(MSEs), col = "red")

# get_model_formula(6, bestmodels,"logGFR")
NewIOV$GFR = log(NewIOV$GFR)

colnames(NewIOV)[which(colnames(NewIOV)=="GFR")] = "logGFR"

Base = summary(lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS, data = NewIOV[Step_Train, ]))
PhosFirst = summary(lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + PHOS, data = NewIOV[Step_Train
PhosSecond = summary(lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + poly(PHOS,2), data = NewIOV[S
MapFirst = summary(lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + MAP, data = NewIOV[Step_Train,
MapSecond = summary(lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + poly(MAP,2), data = NewIOV[Ste
HBFirst = summary(lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + HB, data = NewIOV[Step_Train, ]))
HBSecond = summary(lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + poly(HB,2), data = NewIOV[Step_
ScrFirst = summary(lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + SCR, data = NewIOV[Step_Train,
```

```r
ScrSecond = summary(lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + poly(SCR,2), data = NewIOV[Ste

SecondOrderChecks = data.frame("Added_Predictor" = c("NONE", "PHOS", "$\\text{PHOS}^2$", "MAP", "$\\tex
                               "p_value" = rep('',9),
                               "R2" = rep('',9 ))


R2 = c(Base$r.squared, PhosFirst$adj.r.squared, PhosSecond$adj.r.squared, MapFirst$adj.r.squared, MapSe
       HBFirst$adj.r.squared, HBSecond$adj.r.squared, ScrFirst$adj.r.squared, ScrSecond$adj.r.squared)

SecondOrderChecks$R2 = R2

ps = c(0, PhosFirst$coefficients[nrow(PhosFirst$coefficients), 4],
       PhosSecond$coefficients[nrow(PhosSecond$coefficients), 4],
MapFirst$coefficients[nrow(MapFirst$coefficients), 4],
MapSecond$coefficients[nrow(MapSecond$coefficients), 4],
HBFirst$coefficients[nrow(HBFirst$coefficients), 4],
HBSecond$coefficients[nrow(HBSecond$coefficients), 4],
ScrFirst$coefficients[nrow(ScrFirst$coefficients), 4],
ScrSecond$coefficients[nrow(ScrSecond$coefficients), 4]
)

SecondOrderChecks$p_value = ps


SecondOrderChecks[,c(2,3)] = round(SecondOrderChecks[, c(2,3)], 3)

SecondOrderChecks[1, "p_value"] = NA


colnames(SecondOrderChecks)[1] <- "Added Predictor"
colnames(SecondOrderChecks)[2] <- "p value"
colnames(SecondOrderChecks)[3] <- "Adj $R^2$"

kable_styling(kable(SecondOrderChecks, escape = F, format = "latex", booktabs= TRUE, align = 'c'
                    , caption = "Model Results when Predictors are Added to the Model Individually"), la
FEBLintFit <- lm(logGFR ~ UUN + SUN + ALB + BLACK* FEMALE + CYS + poly(SCR,2), data = NewIOV[Step_Train
# summary(FEBLintFit)

kable_styling(kable(round(summary(FEBLintFit)$coefficients,3), escape = F, format = "latex", booktabs= T
                    , caption = "Model Results with Interaction"), latex_options = "HOLD_position")


SUCYintFit <- lm(logGFR ~ UUN + SUN*CYS + ALB + BLACK + FEMALE  + poly(SCR,2), data = NewIOV[Step_Train
# summary(SUCYintFit)

kable_styling(kable(round(summary(SUCYintFit)$coefficients,3), escape = F, format = "latex", booktabs= T
                    , caption = "Model Results with Interaction"), latex_options = "HOLD_position")

ComparedModel = lm(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + poly(SCR,2), data = NewIOV[Step_Tra

checkInteraction = round(anova(ComparedModel, SUCYintFit),3)
```

```
checkInteraction[1, is.na(checkInteraction[1,])] = ''
checkInteraction[1,] = as.character(checkInteraction[1,])

kable_styling(kable(checkInteraction, escape = F, format = "latex", booktabs= TRUE, align = 'c'
                    , caption = "Model Comparison with Interaction"), latex_options = "HOLD_position")

FinalPoly = lm(logGFR ~ UUN +  ALB + BLACK + FEMALE + SUN*CYS + poly(SCR,2), data = NewIOV[Step_Train, ]

par(mfrow = c(2,2))
plot(FinalPoly)
library(Hmisc)
StepSCR = data.frame(logGFR= NewIOV$logGFR,
                     SCR = NewIOV$SCR,
                     SCR_Interval = cut2(NewIOV$SCR, g = 6))


fit.stepSCR=lm(logGFR~ -1 + SCR_Interval,data=StepSCR[Step_Train,])

SCRStepCoefs = round(summary(fit.stepSCR)$coefficients,3)

kable_styling(kable(SCRStepCoefs, format = "latex", booktabs= TRUE, align = 'c'
                    , caption = "SCR Step Function Coefficients"), latex_options = "HOLD_position")

StepSCRpreds = predict(fit.stepSCR, newdata = StepSCR[-Step_Train,], se=T)

dfPlotSCRstep = cbind(StepSCR[-Step_Train,], StepSCRpreds$fit,
                      (StepSCRpreds$fit+2*StepSCRpreds$se.fit), (StepSCRpreds$fit-2*StepSCRpreds$se.fit)

colnames(dfPlotSCRstep)[4:6] = c("StepSCRpreds", "Upper", "Lower")

ggplot(data = dfPlotSCRstep, aes(x= SCR, y = logGFR)) +geom_point( color = "grey") + geom_step(aes(x = S

StepSUN = data.frame(logGFR= NewIOV$logGFR,
                     SUN = NewIOV$SUN,
                     SUN_Interval = cut2(NewIOV$SUN, g = 6))

fit.stepSUN=lm(logGFR~ -1 + SUN_Interval,data=StepSUN[Step_Train,])

SUNStepCoefs = round(summary(fit.stepSUN)$coefficients,3)

kable_styling(kable(SUNStepCoefs, format = "latex", booktabs= TRUE, align = 'c'
                    , caption = "SUN Step Function Coefficients"), latex_options = "HOLD_position")

StepSUNpreds = predict(fit.stepSUN, newdata = StepSUN[-Step_Train,], se=T)

dfPlotSUNstep = cbind(StepSUN[-Step_Train,], StepSUNpreds$fit,
                      (StepSUNpreds$fit+2*StepSUNpreds$se.fit), (StepSUNpreds$fit-2*StepSUNpreds$se.fit)

colnames(dfPlotSUNstep)[4:6] = c("StepSUNpreds", "Upper", "Lower")

ggplot(data = dfPlotSUNstep, aes(x= SUN, y = logGFR)) +geom_point( color = "grey") + geom_step(aes(x = S

StepUUN = data.frame(logGFR= NewIOV$logGFR,
```

```
                      UUN = NewIOV$UUN,
                      UUN_Interval = cut2(NewIOV$UUN, g = 6))

fit.stepUUN=lm(logGFR~ -1 + UUN_Interval,data=StepUUN[Step_Train,])

UUNStepCoefs = round(summary(fit.stepUUN)$coefficients,3)

kable_styling(kable(UUNStepCoefs, format = "latex", booktabs= TRUE, align = 'c'
                    , caption = "UUN Step Function Coefficients"), latex_options = "HOLD_position")

StepUUNpreds = predict(fit.stepUUN, newdata = StepUUN[-Step_Train,], se=T)

dfPlotUUNstep = cbind(StepUUN[-Step_Train,], StepUUNpreds$fit,
                      (StepUUNpreds$fit+2*StepUUNpreds$se.fit), (StepUUNpreds$fit-2*StepUUNpreds$se.fit)

colnames(dfPlotUUNstep)[4:6] = c("StepUUNpreds", "Upper", "Lower")

ggplot(data = dfPlotUUNstep, aes(x= UUN, y = logGFR)) +geom_point( color = "grey") + geom_step(aes(x = U

StepALB = data.frame(logGFR= NewIOV$logGFR,
                      ALB = NewIOV$ALB,
                      ALB_Interval = cut2(NewIOV$ALB, g = 4))

fit.stepALB=lm(logGFR~ -1 + ALB_Interval,data=StepALB[Step_Train,])

ALBStepCoefs = round(summary(fit.stepALB)$coefficients,3)

kable_styling(kable(ALBStepCoefs, format = "latex", booktabs= TRUE, align = 'c'
                    , caption = "ALB Step Function Coefficients"), latex_options = "HOLD_position")

StepALBpreds = predict(fit.stepALB, newdata = StepALB[-Step_Train,], se=T)

dfPlotALBstep = cbind(StepALB[-Step_Train,], StepALBpreds$fit,
                      (StepALBpreds$fit+2*StepALBpreds$se.fit), (StepALBpreds$fit-2*StepALBpreds$se.fit)

colnames(dfPlotALBstep)[4:6] = c("StepALBpreds", "Upper", "Lower")

ggplot(data = dfPlotALBstep, aes(x= ALB, y = logGFR)) +geom_point( color = "grey") + geom_step(aes(x = A

StepCYS = data.frame(logGFR= NewIOV$logGFR,
                      CYS = NewIOV$CYS,
                      CYS_Interval = cut2(NewIOV$CYS, g = 6))

fit.stepCYS=lm(logGFR~ -1 + CYS_Interval,data=StepCYS[Step_Train,])

CYSStepCoefs = round(summary(fit.stepCYS)$coefficients,3)

kable_styling(kable(CYSStepCoefs, format = "latex", booktabs= TRUE, align = 'c'
                    , caption = "CYS Step Function Coefficients"), latex_options = "HOLD_position")

StepCYSpreds = predict(fit.stepCYS, newdata = StepCYS[-Step_Train,], se=T)

dfPlotCYSstep = cbind(StepCYS[-Step_Train,], StepCYSpreds$fit,
```

```
                        (StepCYSpreds$fit+2*StepCYSpreds$se.fit), (StepCYSpreds$fit-2*StepCYSpreds$se.fit)

colnames(dfPlotCYSstep)[4:6] = c("StepCYSpreds", "Upper", "Lower")

ggplot(data = dfPlotCYSstep, aes(x= CYS, y = logGFR)) +geom_point( color = "grey") + geom_step(aes(x = C

NewIOVTrain = NewIOV[Step_Train,]
library(splines)
Deg2SCRBSpline <- lm(logGFR ~ bs(SCR, knots = c(0.1, 0.5, 0.9), degree = 2), data = NewIOVTrain)
scrlims <- range(NewIOVTrain$SCR)
scr.grid=seq(from=scrlims[1],to=scrlims[2]+0.9)

pred.bspline=predict(Deg2SCRBSpline, newdata=list(SCR=scr.grid),se=T)
plot(NewIOVTrain$SCR, NewIOVTrain$logGFR,col="gray", xlab = "SCR", ylab = "GFR", main = "SCR Splines")
lines(scr.grid,pred.bspline$fit,lwd=2, col = "blue")
lines(scr.grid,pred.bspline$fit+2*pred.bspline$se,lty="dashed", col = "blue")
lines(scr.grid,pred.bspline$fit-2*pred.bspline$se,lty="dashed", col = "blue")

fit.nspline <- lm(logGFR~ns(SCR, knots = c(0.1, 0.5, 0.9)), data=NewIOVTrain)
pred.nspline <- predict(fit.nspline, newdata=list(SCR=scr.grid), se=T)
lines(scr.grid, pred.nspline$fit,col="red",lwd=2)
lines(scr.grid,pred.nspline$fit+2*pred.nspline$se,lty="dashed", col = "red")
lines(scr.grid,pred.nspline$fit-2*pred.nspline$se,lty="dashed", col = "red")

fit.smoothspline=smooth.spline(NewIOVTrain$SCR,NewIOVTrain$logGFR, cv = TRUE)
lines(fit.smoothspline,col="green",lwd=2)


Deg2SCRBSpline <- lm(logGFR ~ bs(SCR, knots = c(0.1, 0.5, 0.9), degree = 3), data = NewIOVTrain)
scrlims <- range(NewIOVTrain$SCR)
scr.grid=seq(from=scrlims[1],to=scrlims[2]+0.9)

pred.bspline=predict(Deg2SCRBSpline, newdata=list(SCR=scr.grid),se=T)
lines(scr.grid,pred.bspline$fit,lwd=2, col = "black")
lines(scr.grid,pred.bspline$fit+2*pred.bspline$se,lty="dashed", col = "black")
lines(scr.grid,pred.bspline$fit-2*pred.bspline$se,lty="dashed", col = "black")


legend("topright", legend = c("basis spline - degree 2", "natural spline", "smoothing spline", "basis sp
NewIOVTrain = NewIOV[Step_Train,]
library(splines)
Deg2SUNBSpline <- lm(logGFR ~ bs(SUN, knots = c(0.1, 0.5, 0.9), degree = 2), data = NewIOVTrain)
SUNlims <- range(NewIOVTrain$SUN)
SUN.grid=seq(from=SUNlims[1],to=SUNlims[2]+0.9)

pred.bspline=predict(Deg2SUNBSpline, newdata=list(SUN=SUN.grid),se=T)
plot(NewIOVTrain$SUN, NewIOVTrain$logGFR,col="gray", xlab = "SUN", ylab = "GFR", main = "SUN Splines")
lines(SUN.grid,pred.bspline$fit,lwd=2, col = "blue")
lines(SUN.grid,pred.bspline$fit+2*pred.bspline$se,lty="dashed", col = "blue")
lines(SUN.grid,pred.bspline$fit-2*pred.bspline$se,lty="dashed", col = "blue")

fit.nspline <- lm(logGFR~ns(SUN, knots = c(0.25, 0.5, 0.75)), data=NewIOVTrain)
pred.nspline <- predict(fit.nspline, newdata=list(SUN=SUN.grid), se=T)
```

```r
lines(SUN.grid, pred.nspline$fit,col="red",lwd=2)
lines(SUN.grid,pred.nspline$fit+2*pred.nspline$se,lty="dashed", col = "red")
lines(SUN.grid,pred.nspline$fit-2*pred.nspline$se,lty="dashed", col = "red")

fit.smoothspline=smooth.spline(NewIOVTrain$SUN,NewIOVTrain$logGFR, cv = TRUE)
lines(fit.smoothspline,col="green",lwd=2)


Deg2SUNBSpline <- lm(logGFR ~ bs(SUN, knots = c(0.1, 0.5, 0.9), degree = 3), data = NewIOVTrain)
SUNlims <- range(NewIOVTrain$SUN)
SUN.grid=seq(from=SUNlims[1],to=SUNlims[2]+0.9)

pred.bspline=predict(Deg2SUNBSpline, newdata=list(SUN=SUN.grid),se=T)
lines(SUN.grid,pred.bspline$fit,lwd=2, col = "black")
lines(SUN.grid,pred.bspline$fit+2*pred.bspline$se,lty="dashed", col = "black")
lines(SUN.grid,pred.bspline$fit-2*pred.bspline$se,lty="dashed", col = "black")


legend("topright", legend = c("basis spline - degree 2", "natural spline", "smoothing spline", "basis sp
library(mgcv)

GAM_SCRSmooth <- gam(logGFR ~ UUN  + ALB + BLACK + FEMALE + SUN + CYS + s(SCR), data = NewIOVTrain)

GAM_SCRSpline <- gam(logGFR ~ UUN + ALB + BLACK + FEMALE +  SUN + CYS + ns(SCR, knots = c(0.1, 0.5, 0.9

BICsmoothgam = round(BIC(GAM_SCRSmooth),3)
BICnaturalgam = round(BIC(GAM_SCRSpline),3)

GAM_comparison <- data.frame(GAM = c("Smooth SCR", "Spliine SCR"),
                             BIC = c(BICsmoothgam,BICnaturalgam))

kable_styling(kable(GAM_comparison, format = "latex", booktabs= TRUE, align = 'c'
                    , caption = "GAM Comparison"), latex_options = "HOLD_position")

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


library(gam)
gam.start = gam(logGFR ~ UUN  + ALB + BLACK + FEMALE + SUN + CYS + SCR, data=NewIOVTrain)

gam.scope=list("SCR"= ~1+SCR+s(SCR) + ns(SCR, knots = c(0.1, 0.5, 0.9)),
               "CYS"=~1+CYS+s(CYS)+ ns(CYS, knots = c(0.1, 0.5, 0.9)),
               "SUN"=~1+SUN+s(SUN)+ ns(SUN, knots = c(0.1, 0.5, 0.9)),
               "UUN"= ~1+UUN+s(UUN)+ ns(UUN, knots = c(0.1, 0.5, 0.9)),
               "ALB"=~1+ALB+s(ALB)+ ns(ALB, knots = c(0.1, 0.5, 0.9)),
               "FEMALE"=~1+FEMALE,
               "BLACK"=~1+BLACK)

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


GamStepAOV = as.data.frame(mod$anova)
```

```r
GamStepAOV[2:5,3:7] = round(GamStepAOV[2:5,3:7],3)
GamStepAOV$AIC = round(GamStepAOV$AIC,3)
GamStepAOV$`Resid. Dev` = round(GamStepAOV$`Resid. Dev`,3)
GamStepAOV[is.na(GamStepAOV)] = ''
kable_styling(kable(GamStepAOV, format = "latex", booktabs= TRUE, align = 'c'
                    , caption = "Stepwise Selection GAM ANOVA"),
              latex_options = "HOLD_position")

Stepwise_GAM = gam(mod$formula, data = NewIOVTrain)
Manual_GAM = gam(logGFR ~ UUN + SUN + ALB + BLACK + FEMALE + CYS + ns(SCR, knots = c(0.1, 0.5, 0.9)), da
Poly_fit = lm(logGFR ~ UUN + SUN + ALB + BLACK* FEMALE + CYS + poly(SCR,2), data = NewIOV[Step_Train, ])

library("MLmetrics")

logGFRresponse = NewIOV[-Step_Train, 'logGFR']

StepGAMpred = predict(Stepwise_GAM, newdata = NewIOV[-Step_Train,])
ManualGAMpred = predict(Manual_GAM, newdata = NewIOV[-Step_Train,])
Polypred = predict(Poly_fit, newdata = NewIOV[-Step_Train,])

StepGamMSE = round(MSE(StepGAMpred, logGFRresponse),4)
ManualGamMSE = round(MSE(ManualGAMpred, logGFRresponse),4)
PolyFitMSE = round(MSE(Polypred, logGFRresponse),4)

RsquareTest = function(test, model, data = NewIOV){
  testPreds = predict(model, newdata = data[test,])
  testResponse = data[test, ]$logGFR

  SS.Total = sum((testResponse-mean(testResponse))^2)
  #SS.Regression = sum((testPreds - mean(testResponse))^2)
  SS.Residual = sum((testResponse - testPreds)^2)

  # rsq = SS.Regression/SS.Total
  rsq = 1 - SS.Residual/SS.Total
  return(round(rsq, 3))
}

StepGamTestR2 = RsquareTest(-Step_Train, Stepwise_GAM)
ManualGamTestR2 = RsquareTest(-Step_Train, Manual_GAM)
PolyFitTestR2 = RsquareTest(-Step_Train, Poly_fit)

StepGamTrainBIC = round(BIC(Stepwise_GAM),3)
ManualGamTrainBIC = round(BIC(Manual_GAM),3)
PolyFitTrainBIC = round(BIC(Poly_fit),3)


ComparingModels = data.frame(Model = c("Stepwise GAM", "Manual GAM", "Polynomial Regression"),
                             "Training BIC" = c(StepGamTrainBIC, ManualGamTrainBIC, PolyFitTrainBIC),
                             "Test MSE" = c(StepGamMSE, ManualGamMSE, PolyFitMSE),
                             TestR2 = c(StepGamTestR2, ManualGamTestR2, PolyFitTestR2))

colnames(ComparingModels)[4] = "Test $R^2$"
colnames(ComparingModels)[2] = "Training BIC"
```

```r
colnames(ComparingModels)[3] = "Test MSE"

kable_styling(kable(ComparingModels, format = "latex", escape = F, booktabs= TRUE, align = 'c'
                    , caption = "Model Comparison"),
              latex_options = "HOLD_position")

ModelPredplot = data.frame(logGFR = logGFRresponse,
                           Stepwise_GAM = StepGAMpred,
                           Manual_GAM = ManualGAMpred,
                           Polynomial_LM = Polypred)

ModelPredplot = melt(ModelPredplot,"logGFR")


p = ggplot(data = ModelPredplot, aes(x=value, y = logGFR)) +geom_point(size=0.5)+ facet_wrap(.~variable

p1 = p %+% subset(ModelPredplot, variable %in% c("Stepwise_GAM", "Manual_GAM"))
p2 = p %+% subset(ModelPredplot,variable %in% c("Polynomial_LM"))


gridExtra::grid.arrange(grobs = lapply(
  list(p1, p2),
  egg::set_panel_size,
  width = unit(7, "cm"),
  height = unit(5, "cm")
))
```