STA 5363, Homework 4

Carson Slater *Baylor University*

Consider your own 'NHANES' data created in HW2 and use the following variables as covariates: `Age`, `Poverty`, `Weight`, `Height`, `Pulse`, `BPSysAve`, `BPDiaAve`, `DirectChol`, `TotChol`.

1. Is there evidence for non-linear relationships between BMI and the covariates? Create some informative plots to justify your answer.

Figure 1 is a pairs plot visualizing pairwise relationships between one another and also `BMI`. Many relationships seem to follow nonlinear trends. Among the variables possessing seemingly nonlinear relationships with `BMI` are `Height`, `DirectChol`, and potentially `Age`. The rest seem to have either a linear, or no clear trend.

To be slightly more thorough, we fit a series of simple linear regression models, each predicting BMI using one of the other variables in the `nhanes` dataset. For each model, we plot the residuals (the differences between observed and predicted BMI values) against the fitted values in Figure 2. These residual plots help assess the adequacy of the linear model. Ideally, residuals should be randomly scattered around zero with no discernible pattern. If the plot shows a curved or structured pattern (e.g., U-shaped), it suggests a nonlinear relationship between the predictor and BMI, indicating that a simple linear model may not be appropriate for that variable. This visual diagnostic tool is useful for identifying variables that may benefit from transformations or more flexible modeling techniques.

Figure 2 shows the residual plots for the simple linear regression models for each of the predictors on `BMI`. In addition to `Height`, `DirectChol`, and `Age`, it appears that `BPSysAve` and `BPDiaAve`.

2. Perform polynomial regression to predict 'BMI' using each covariate selected as having non-linear relationship with 'BMI.' Use cross-validation to select the optimal degree for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting fit to the data.

In this analysis, we aimed to model the nonlinear relationship between `BMI` and several covariates from the `nhanes` dataset using **polynomial regression**, selecting the optimal degree using **cross-validation** and comparing it to a classical **ANOVA-based model selection** strategy. Our goal was both predictive accuracy and interpretability.

Step 1: Function Design

We created a reusable function, `analyze_polynomial_relationship()`, which accepts a dataset, a response variable, a predictor, and ranges of polynomial degrees for both cross-validation (`degrees_cv`) and hypothesis testing (`max_degree_f`). The function outputs a plot of the fitted curves and a collection of model selection diagnostics. This modular design allowed us to reuse the function across multiple predictors and isolate the logic for fitting, testing, and plotting. See the code appendix to see this function.

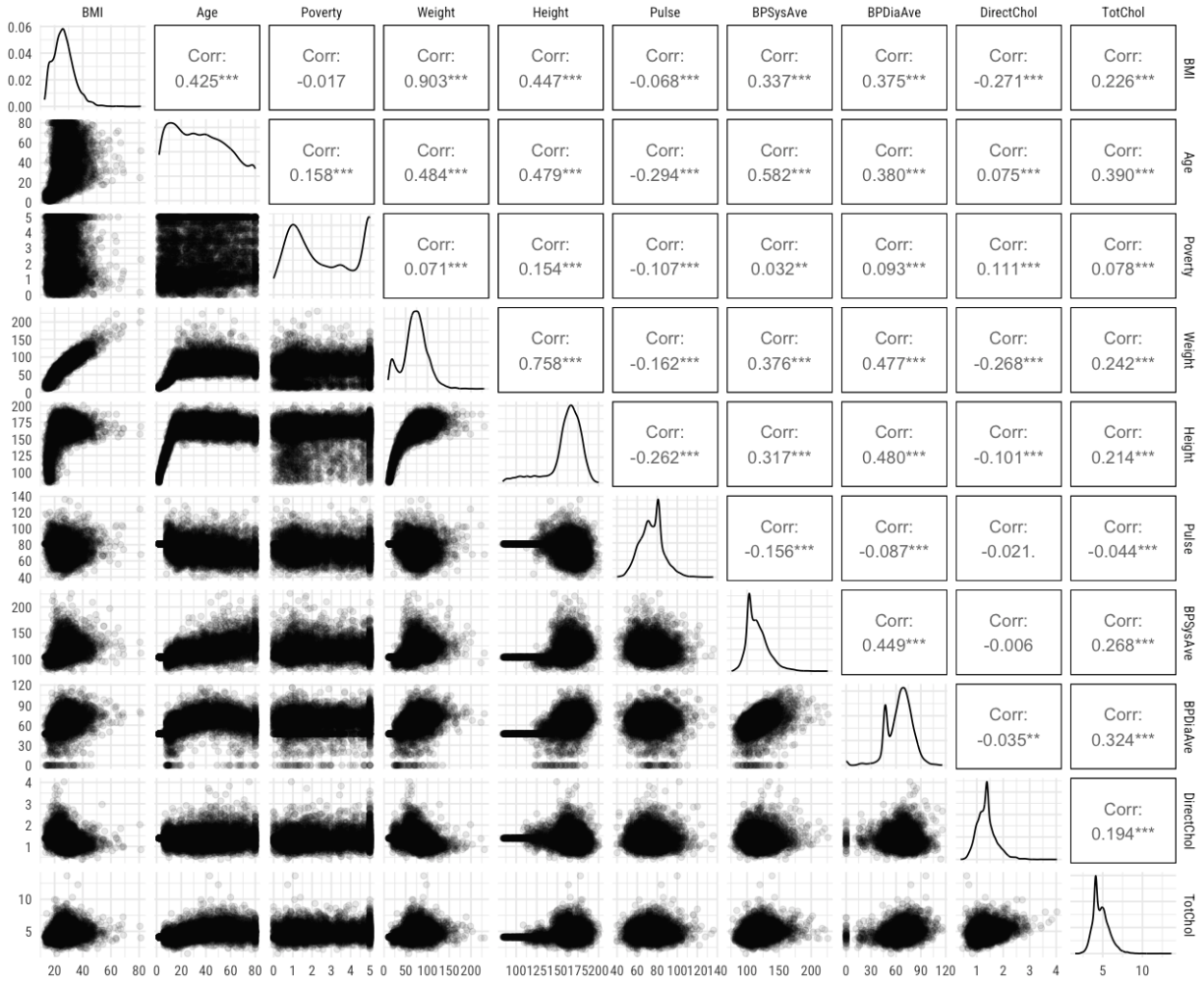

Figure 1: Pairs plot for filtered and imputed NHANES data.

Step 2: Cross-Validation Approach

We performed **10-fold cross-validation** for polynomial degrees ranging from 1 to 10 to identify the degree that minimized prediction error. We used the `caret::train()` function to streamline the CV workflow, evaluating model performance using the RMSE metric. This approach guards against overfitting by balancing model complexity with generalization performance. The degree with the **lowest average RMSE** was chosen as the optimal degree under this criterion.

Step 3: Model Comparison via Partial F-Tests

In parallel, we **use sequential ANOVA with partial F-tests** to evaluate whether increasing the degree of the polynomial significantly improves model fit. Starting from a linear model (degree 1), we sequentially test higher-degree models up to `max_degree_f = 10`, stopping when the increase in complexity is no longer statistically justified at the $\alpha = 0.05$ level. This method follows traditional statistical inference and provides a principled way for us to detect the point of diminishing returns in model complexity.

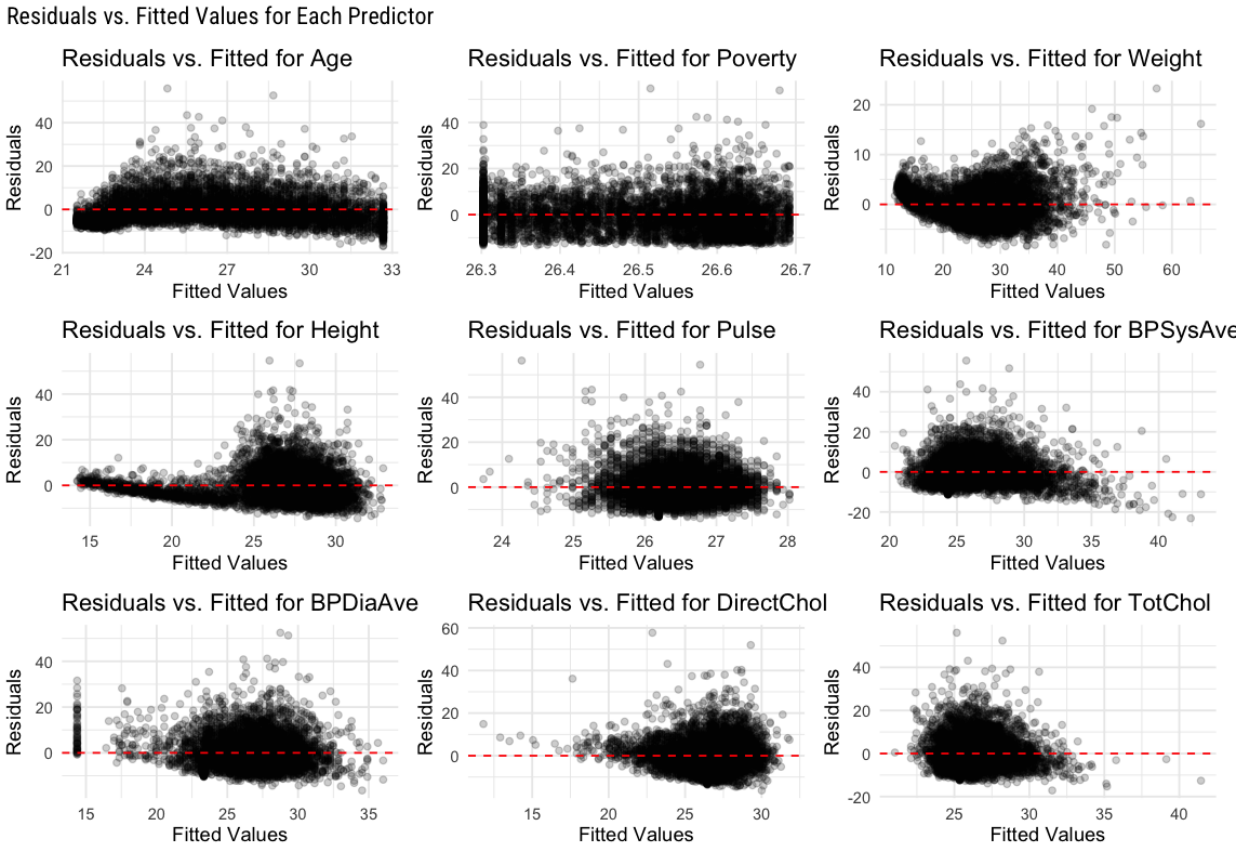

Figure 2: Residual plots for simple linear regression models fit on each predicor.

Step 4: Visualization of Fitted Models

To examine the selected models, we **create a grid of plots** using `ggplot2` that shows the raw data alongside the fitted curves from both the **CV-selected model** and the **F-test-selected model**. We use color and a shared legend to clearly distinguish the model types. Although the homework specifies a focus on cross-validation, we visualize both to support our comparison.

Step 5: Summary Plot of CV Performance

To compare how RMSE varies with degree across all selected predictors, we **extract the RMSE results**. We then **visualize RMSE across polynomial degrees for each variable**, which helps us assess whether certain variables benefit from higher-order terms, as shown in Figure 3.

Justification for Our Approach

- **Using both cross-validation and hypothesis testing** gives us a richer understanding of model performance from both predictive and inferential perspectives. CV emphasizes prediction error on unseen data, while the F-test focuses on statistical significance and parsimony.

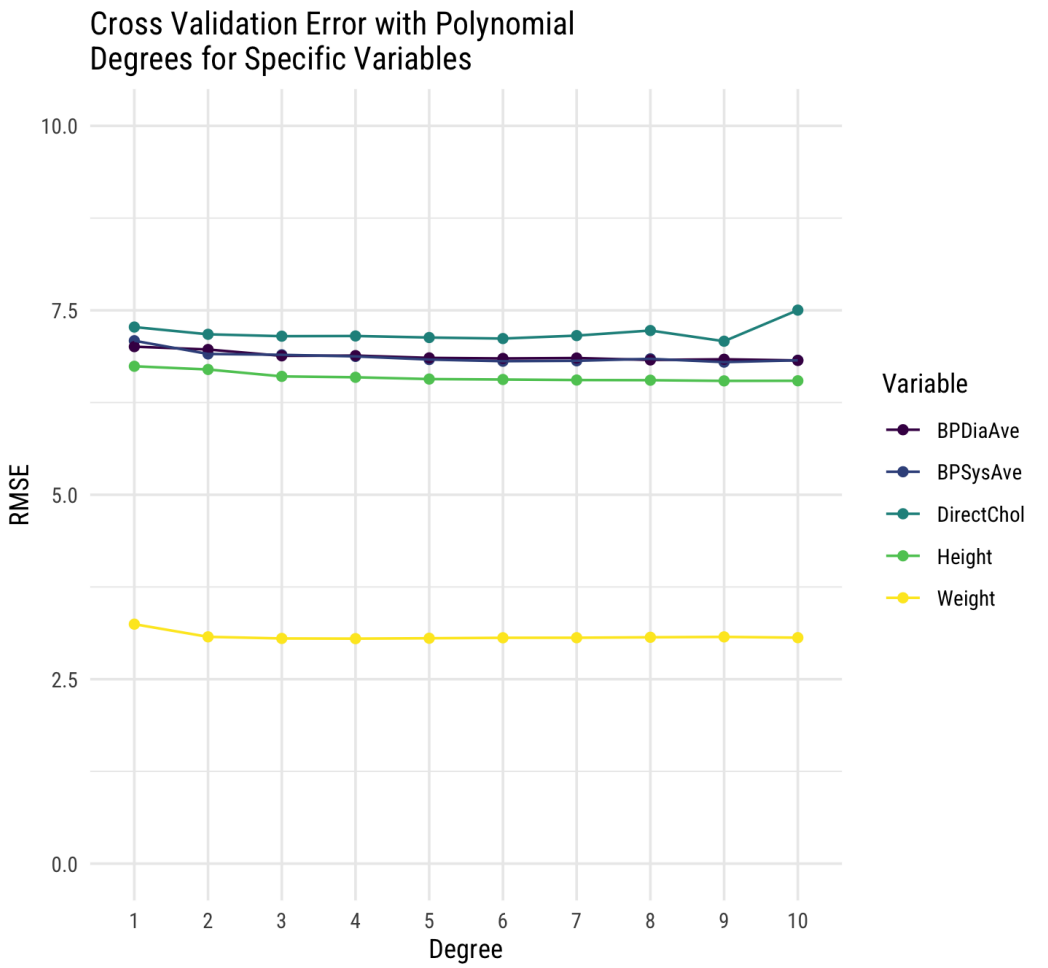

Figure 3: Average cross validation RMSE for each predictor across different polynomial degree fits.

- We **automate the analysis across multiple predictors** by mapping the function over a list of variables that exhibit signs of nonlinearity in preliminary diagnostic plots.
- **Visualizing both the model fit and cross-validation errors** allows us to communicate our modeling choices clearly and evaluate the tradeoffs between complexity and performance.
- Our code design prioritizes **clarity, reproducibility, and scalability**, making it easy for us to extend the analysis to other datasets or predictors.

A Note

This problem requires us to choose polynomial degrees using both cross-validation and the partial-F test. Our results in Figures 3 and 4 indicate that these methods often identify higher-degree polynomial fits as 'best'—even when a simpler model may perform just as well. In practice, we find it best to use the lowest-degree polynomial possible for model simplicity. Figure 3 shows that in nearly all cases, increasing model complexity yields only marginal gains in performance. In some instances—such as with the blood pressure and cholesterol variables—the partial-F test recognizes this and selects a much lower-degree polynomial than cross-validation does.

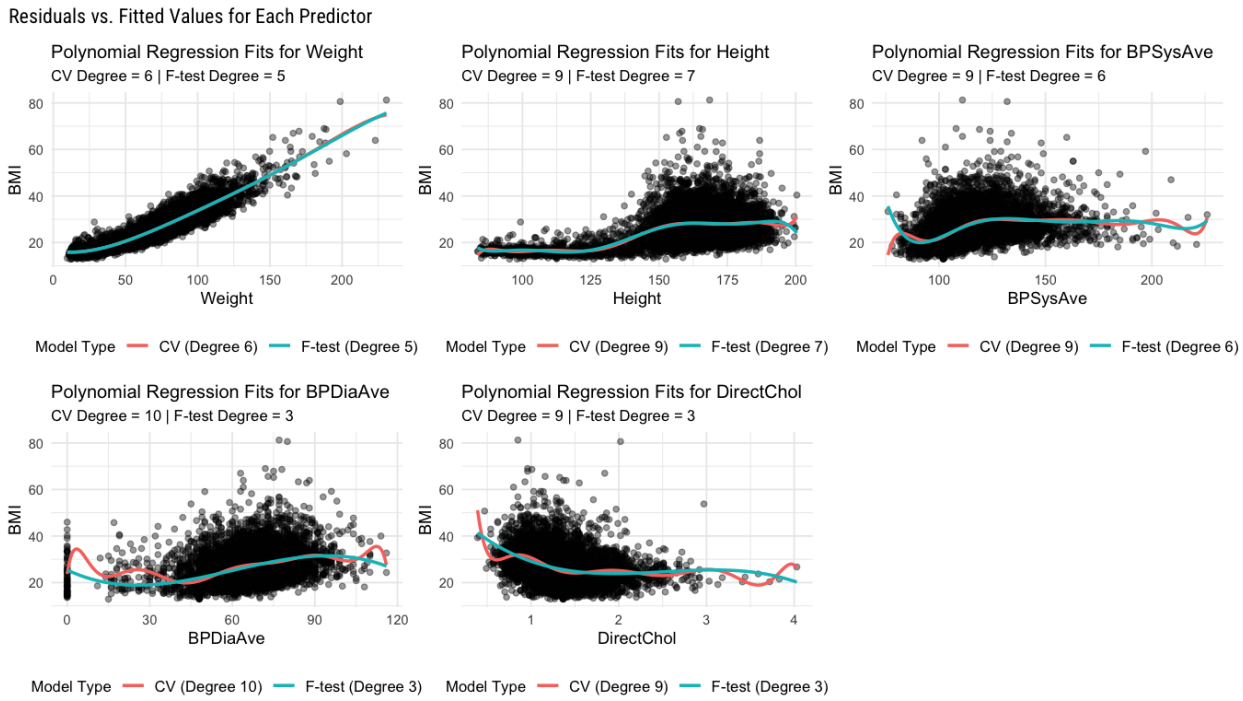

Figure 4: Polynomial CV and partial-F test chosen fits for each of the variables suspected to have nonlinear relationship with `BMI`.

3. Use the `bs()` function to fit a regression spline to predict 'BMI' using each covariate selected as having nonlinear relationship with 'BMI.' Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

We fit regression spline models with 4 degrees of freedom for each of the predictors: `Weight`, `Height`, `BPSysAve`, `BPDiaAve`, and `DirectChol`. With the `bs()` function, knots must be specified manually. Table 1 shows strategically chosen knots based off of domain knowledge. We elect to compare this to quantile-based knots. We place knots at the quantiles (25%, 50%, 75%) of each predictor's distribution. This ensures that the model adapts to the data density, providing more flexibility where data are concentrated.

We consider quantile-based knots because they are robust to skewed distributions, they avoid placing knots in areas with few observations, and they provide a balanced compromise between interpretability and flexibility when the number of degrees of freedom is fixed.

Table 1: Suggested knot locations and rationales for each predictor using domain intuition.

Predictor	**Suggested Knots**	**Rationale**
`Weight`	60, 80, 100	Common adult weight ranges
`Height`	150, 170, 190	Common adult height cutoffs
`BPSysAve`	110, 130, 150	Normal / Elevated / Hypertensive
`BPDiaAve`	70, 80, 90	Similar rationale for diastolic BP
`DirectChol`	30, 50, 70	Common thresholds for direct cholesterol

In Figure 5, we observe that our chosen knots perform essentially the same as the quantile-based knots.

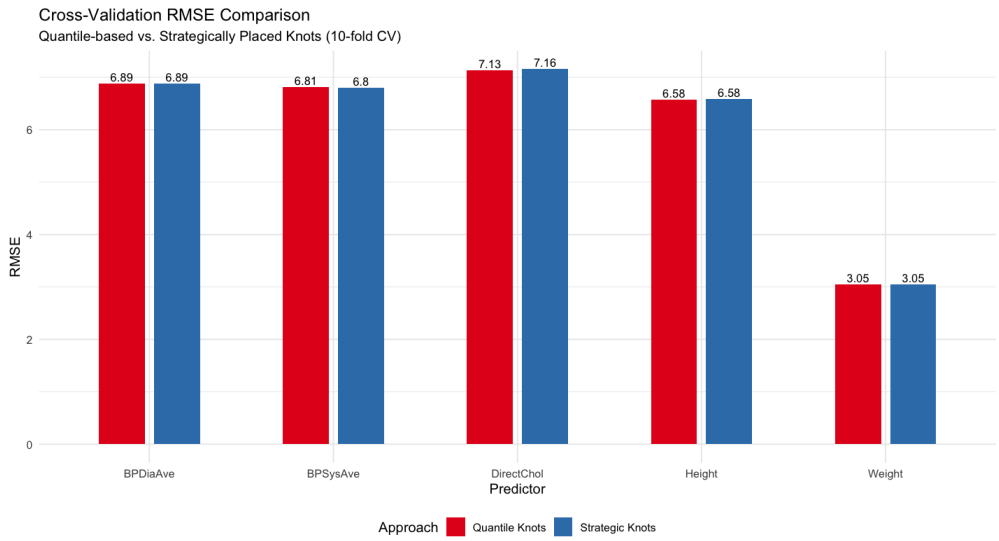

Figure 5: Average RMSE for 10-fold cross validation for spline fits on all variables that do not have a nonlinear relationship with BMI.

All of the performance was marginal, so we elect to use the quantile-based knots. We fit all of the splines and plotted them. Figure 6 shows these fitted splines against the data.

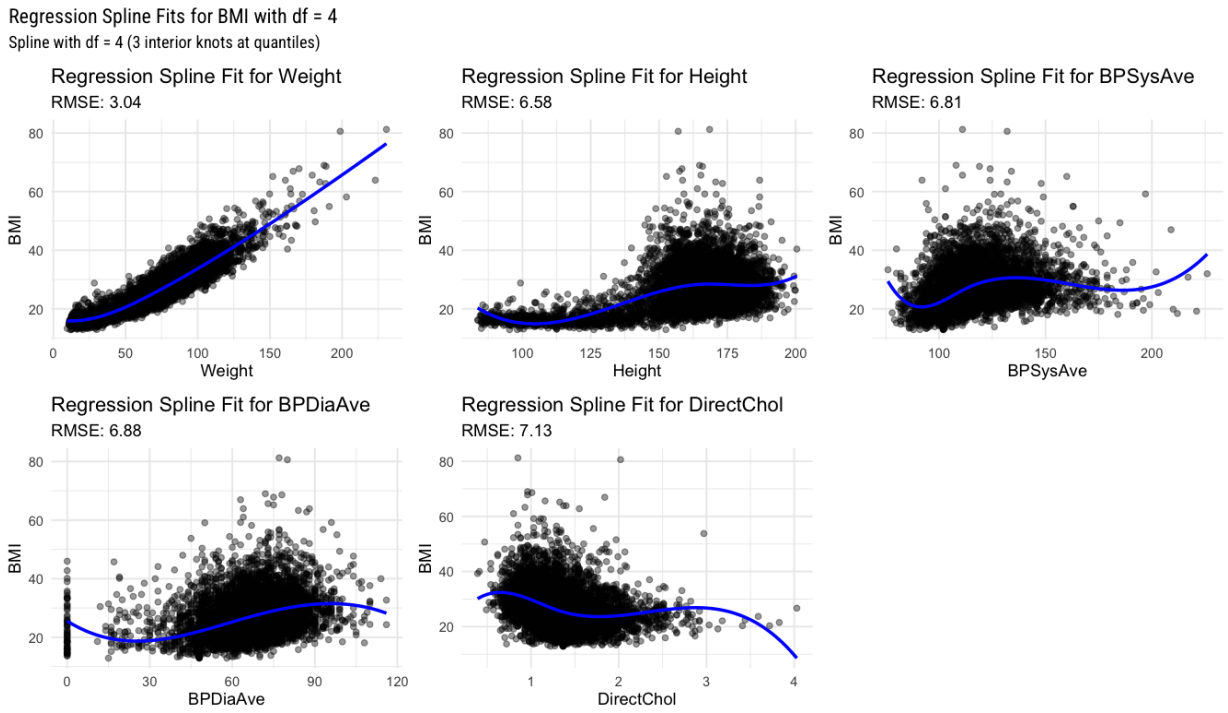

Figure 6: Spline fits with four degrees of freedom for each of the variables suspected to have nonlinear relationship with BMI.

Table 2: Linear model estimates for BMI

Characteristic	Beta	95% CI	p-value
Weight_over_Height_sq	1.0	1.0, 1.0	<0.001

Abbreviation: CI = Confidence Interval

4. Construct a GAM for the data. Identify the covariates that should be incorporated into the model. For significant covariates, define a suitable functional form for each variable, such as linear, polynomial, or regression spline.

To build a GAM, we elect not to include `Weight` and `Height`, as BMI $= \frac{\text{weight}}{\text{height}^2}$. Including both `Height` and `Weight` as predictors in a GAM for `BMI` requires caution, as BMI is itself defined as weight divided by height squared. First, such a model essentially attempts to regress a variable on its own mathematical components, which may offer little new insight. Second, a linear model with untransformed height and weight may misrepresent the true nonlinear relationship, potentially yielding misleading coefficients, such as a negative association with height. Third, including both variables without accounting for their formulaic link to BMI may obscure the model's interpretability and introduce multicollinearity. Thus, while not technically incorrect, such modeling must be approached with careful justification and awareness of the underlying definitions. Essentially, the true model is

$$\text{BMI} = \frac{\text{weight}}{\text{height}^2}.$$

Fitting a GAM when we not only *know* the true relationship, but also *observe* the predictors of which the response variable is comprised, is poor practice. Essentially, we would be favoring an incorrect model (e.g. BMI $= f(\text{Height}) + g(\text{Weight}) + h(\text{BPSysAve}) + \ldots + \epsilon$) knowing the true one is BMI $= \frac{\text{weight}}{\text{height}^2}$. To demonstrate this we create a column in our data called `Weight_over_Height_sq`, and fit a simple linear regression model,

$$\text{BMI} = \beta_0 + \beta_1 \text{Weight\_over\_Height\_sq} + \epsilon, \quad \epsilon \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2).$$

Table 2 shows the coefficient estimate for the fitted model. For this model, the $R^2 = 1$. Note how $\hat{\beta}_1 = 1$, and its $\text{SE}(\hat{\beta}) = 0$. Also, $\hat{\beta}_0 = 0$. This is because the model we just fit was,

$$\text{BMI} = \beta_0 + \beta_1 \frac{\text{weight}}{\text{height}^2} + \epsilon, \quad \epsilon \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2).$$

When $\beta_0 = 0$ and $\beta_1 = 1$, then the model becomes

$$\text{BMI} = \frac{\text{weight}}{\text{height}^2} + \epsilon, \quad \epsilon \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2).$$

But this does not make sense, since we know the truth is BMI $= \frac{\text{weight}}{\text{height}^2}$. So, essentially we are forced to guess the purpose of this homework problem. We must determine if the problem is to get us practice building GAMs, or if we are supposed to see this fact and abuse the information to build the best predictive model. Instead of electing to leverage this knowledge, and essentially build a tautological model that will have excellent prediction, we elect to exclude `Height` and `Weight` from the model moving forward.

Table 3: RMSE Values for Different Fit Methods.

Variable	Spline	CV Polynomial	Partial-F Test Polynomial
BPSysAve	6.813	6.810	6.809
BPDiaAve	6.883	6.820	6.883
DirectChol	7.128	7.084	7.155

For this particular problem, based off our findings in Figure 2, we elect to consider 3 of the seven remaining predictors (not `Weight` and `Height`) as having linear relationships with `BMI`, and hence we will include them in the GAM with linear components initially.

We also need to decide how we will consider the other predictors that have a percieved nonlinear relationship with `BMI`. To do this, we first consider the information in Table 3. Initially, it might seem better to represent the nonlinear relationships using the methods that yielded the lowest RMSE, but note that the 10-fold cross validation polynomial chose very high degree polynomials, trading model complexity for marginal performance. Because the performance bump is marginal we elect to use simpler lower degree polynomials for these three variables. For `BPSysAve`, we choose $d = 6$. For `BPDiaAve`, we choose $d = 3$ and for `DirectChol`, we initially try $d = 3$.

Table 4: Coefficient estimates and standard errors for the **initial** GAM.

Characteristic	**Beta**	**95% CI**	**p-value**
Age	0.11	0.10, 0.11	<0.001
Poverty	-0.28	-0.37, -0.19	<0.001
Pulse	0.03	0.02, 0.05	<0.001
TotChol	0.46	0.30, 0.63	<0.001
BPSysAve			
BPSysAve	32.76	16.00, 49.52	<0.001
BPSysAve^2	-64.39	-76.98, -51.79	<0.001
BPSysAve^3	28.84	16.58, 41.09	<0.001
BPSysAve^4	19.49	7.43, 31.56	0.002
BPSysAve^5	-31.85	-43.89, -19.80	<0.001
BPSysAve^6	21.11	9.21, 33.02	<0.001
BPDiaAve			
BPDiaAve	104.33	89.99, 118.68	<0.001
BPDiaAve^2	27.20	14.40, 40.01	<0.001
BPDiaAve^3	-28.74	-41.53, -15.95	<0.001
DirectChol			
DirectChol	-166.15	-178.53, -153.76	<0.001
DirectChol^2	38.20	26.06, 50.33	<0.001
DirectChol^3	-9.56	-21.54, 2.42	0.12

Abbreviation: CI = Confidence Interval

Table 4 displays the coefficient estimates for the original model. Note how the coefficient for the third

degree polynomial term for `DirectChol` is not significant in the presence of all other predictors. We elect to remove this term in our final GAM. Table 5 displays the coefficient estimates and standard errors for the final model and Figure 7 shows the partial effects for each variable in the model. The final model is written in (1):

$$\begin{aligned}\text{BMI} = \beta_0 &+ \beta_1 \text{Age} + \beta_2 \text{Poverty} + \beta_3 \text{Pulse} + \beta_4 \text{TotChol} \\ &+ \beta_5 \text{BPSysAve} + \beta_6 \text{BPSysAve}^2 + \beta_7 \text{BPSysAve}^3 + \beta_8 \text{BPSysAve}^4 + \beta_9 \text{BPSysAve}^5 + \beta_{10} \text{BPSysAve}^6 \\ &+ \beta_{11} \text{BPDiaAve} + \beta_{12} \text{BPDiaAve}^2 + \beta_{13} \text{BPDiaAve}^3 \\ &+ \beta_{14} \text{DirectChol} + \beta_{15} \text{DirectChol}^2 + \epsilon. \end{aligned} \tag{1}$$

Table 5: Coefficient estimates and standard errors for the **updated** GAM.

Characteristic	**Beta**	**95% CI**	**p-value**
Age	0.11	0.10, 0.12	<0.001
Poverty	-0.28	-0.37, -0.19	<0.001
Pulse	0.03	0.02, 0.05	<0.001
TotChol	0.47	0.30, 0.63	<0.001
BPSysAve			
BPSysAve	32.39	15.64, 49.15	<0.001
BPSysAve^2	-64.42	-77.01, -51.82	<0.001
BPSysAve^3	28.97	16.71, 41.22	<0.001
BPSysAve^4	19.50	7.43, 31.57	0.002
BPSysAve^5	-31.91	-43.96, -19.86	<0.001
BPSysAve^6	21.17	9.26, 33.08	<0.001
BPDiaAve			
BPDiaAve	104.56	90.22, 118.90	<0.001
BPDiaAve^2	27.35	14.55, 40.16	<0.001
BPDiaAve^3	-28.77	-41.56, -15.98	<0.001
DirectChol			
DirectChol	-166.28	-178.66, -153.90	<0.001
DirectChol^2	38.01	25.87, 50.14	<0.001

Abbreviation: CI = Confidence Interval

5. Split the data into a training set and a test set. Fit the GAM in (4) on the training data, plot the results, and explain your findings.

We split the model into training and testing data, reserving 80% for the training data set and 20% for the testing data set. Coefficient estimates for this model can be seen in Table 6 and partial effects cam be observed in Figure 8. While some coefficient estimates are more sensitive than others to the data reduction, they all reflect similar patterns when compare to the coefficients of the model trained on the full dataset. This does not give rise to any caution from a modeling standpoint.

Table 6: Coefficient estimates and standard errors for the **training** GAM.

Characteristic	**Beta**	**95% CI**	**p-value**
Age	0.10	0.09, 0.11	<0.001
Poverty	-0.29	-0.39, -0.19	<0.001
Pulse	0.03	0.01, 0.04	<0.001
TotChol	0.50	0.31, 0.69	<0.001
BPSysAve			
BPSysAve	29.48	12.57, 46.39	<0.001
BPSysAve^2	-55.65	-68.33, -42.96	<0.001
BPSysAve^3	26.46	14.15, 38.77	<0.001
BPSysAve^4	20.70	8.54, 32.86	<0.001
BPSysAve^5	-26.56	-38.70, -14.41	<0.001
BPSysAve^6	18.84	6.83, 30.84	0.002
BPDiaAve			
BPDiaAve	92.37	78.00, 106.74	<0.001
BPDiaAve^2	23.64	10.73, 36.55	<0.001
BPDiaAve^3	-24.84	-37.72, -11.95	<0.001
DirectChol			
DirectChol	-150.31	-162.77, -137.86	<0.001
DirectChol^2	35.37	23.13, 47.61	<0.001

Abbreviation: CI = Confidence Interval

6. Evaluate the model obtained on the test set, and explain the results obtained.

On the test set, we obtained an RMSE of 5.878 units. This is reflective of an okay model. Intuitively there could be smaller RMSE with perhaps a simpler model.

7. Fit a linear regression model on the training data and evaluate the model with the test set.

We fit a multiple linear regression model on the training data, represented by (2) and obtained a test RMSE of 6.028.

$$\begin{aligned} \text{BMI} = \beta_0 + \beta_1 \text{Age} + \beta_2 \text{Poverty} + \beta_3 \text{Pulse} + \beta_4 \text{TotChol} + \beta_5 \text{BPSysAve} \\ + \beta_6 \text{BPDiaAve} + \beta_7 \text{DirectChol} + \epsilon, \end{aligned} \tag{2}$$

8. Which model is better, linear or nonlinear? Explain your findings.

Based on our results, the nonlinear model outperforms the linear model, though the improvement is modest.

After splitting the data into training and test sets, we fit both a generalized additive model (GAM) incorporating nonlinear terms (specifically, polynomial terms for `BPSysAve`, `BPDiaAve`, and `DirectChol`), as well as a multiple linear regression model. On the test set, the GAM achieved an RMSE of **5.878**, while the linear model had a slightly higher RMSE of **6.028**.

This suggests that the nonlinear model provides a better fit by capturing curvature in the relationships between BMI and several predictors. As seen in the partial effect plots (Figures 7 and \ref{fig:pet), certain variables exhibit clear nonlinear behavior—behavior that a strictly linear model fails to capture. By allowing for this flexibility, the GAM offers improved predictive performance.

That said, we emphasize that the gain in predictive accuracy is relatively small. In practice, such improvements must be weighed against the increased complexity and reduced interpretability that come with higher-degree polynomial terms. Our approach balances model fit with parsimony by only applying nonlinear transformations where justified by diagnostics and model comparison metrics. Thus, while both models perform reasonably well, we conclude that the nonlinear model is preferable for this problem.

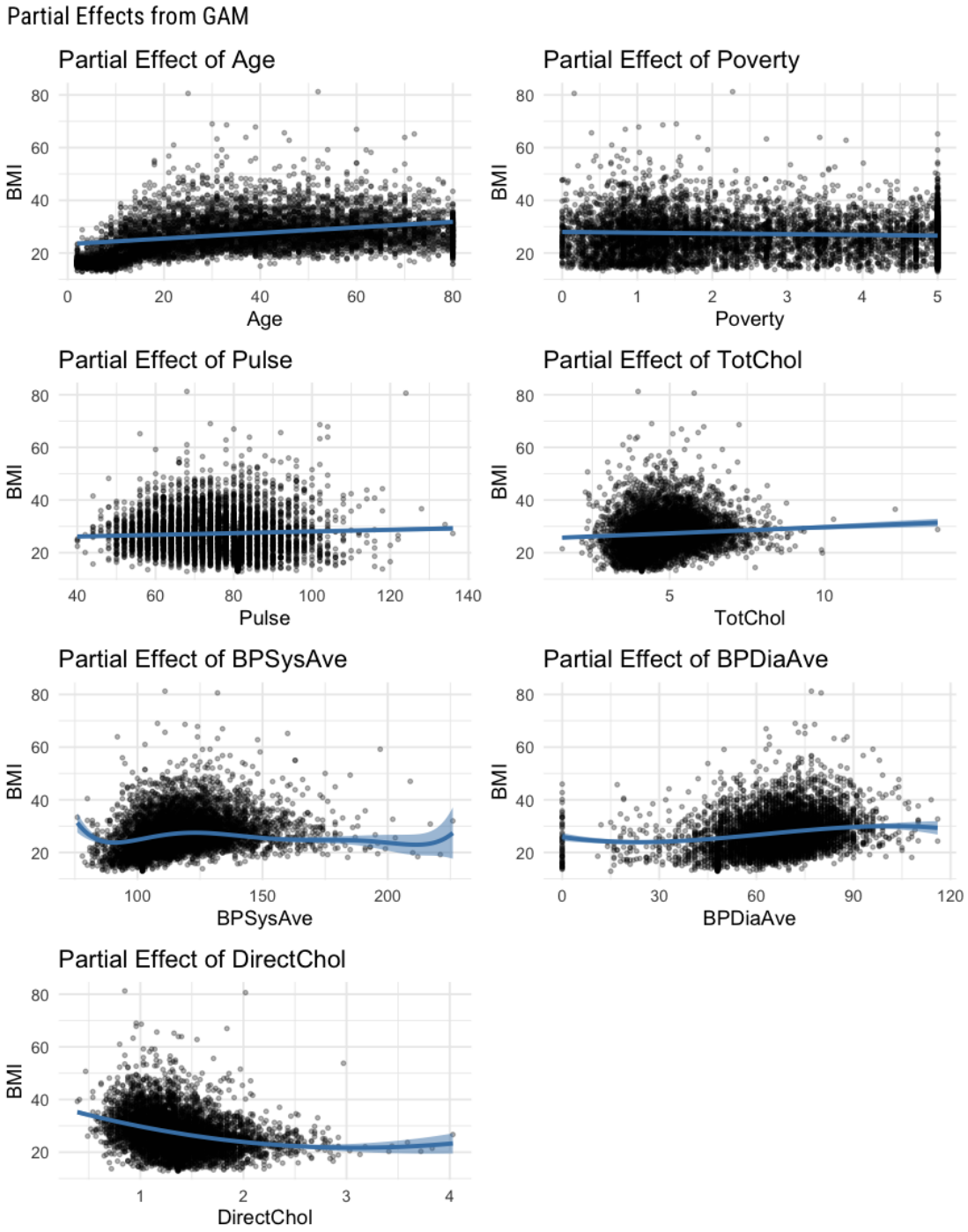

Figure 7: Partial effects for each component on the GAM.

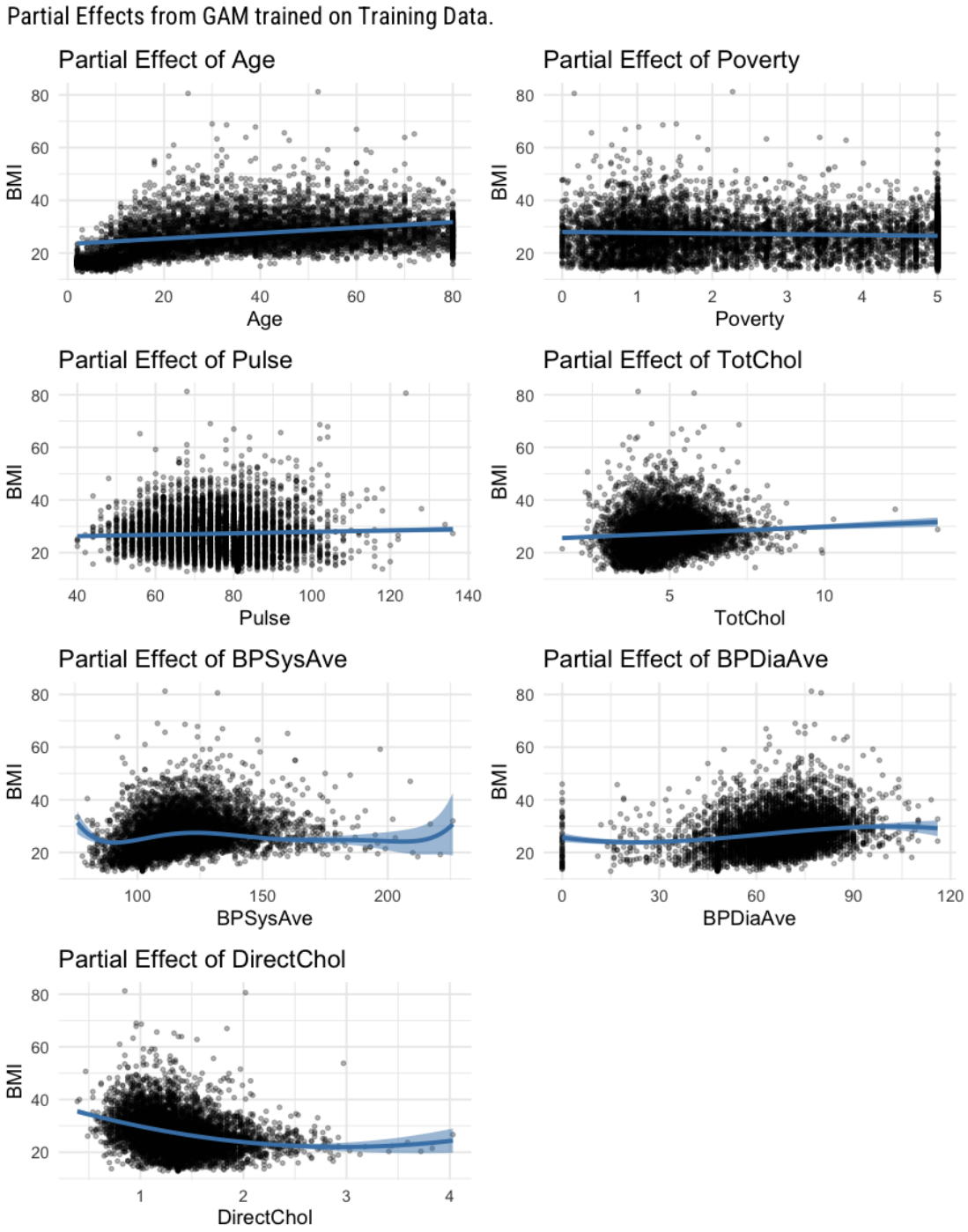

Figure 8: Partial effects for each component on the GAM fitted with the training data.

Code Appendix

```
knitr::opts_chunk$set(dev = "cairo_pdf",
                      fig.width = 5,
                      fig.height = 3.25,
                      fig.align = 'center',
                      echo = FALSE,
                      message = FALSE,
                      warning = FALSE,
                      error = FALSE,
                      cache = TRUE)
library("tidymodels")
library("patchwork")
library("glue")
library("scales")
library("extrafont")
library("tinytex")
library("patchwork")
library("gridExtra")
library("tidyr")
library("latex2exp")
library("GGally")
library("caret")
library("broom")
library("gt")
library("knitr")
library("gam")
library("splines")
library("rprojroot")
theme_set(theme_minimal(base_family = "Roboto Condensed"))

conflicted::conflicts_prefer(
  readr::col_factor(),
  purrr::discard(),
  rstan::extract(),
  dplyr::lag(),
  rstan::traceplot(),
  viridis::viridis_pal(),
  readr::parse_date
)
nhanes <- read_rds(here::here("HW4", "nhanes_impute.rds"))

nhanes <- nhanes |>
  select(
    BMI,
    Age,
```

```
    Poverty,
    Weight,
    Height,
    Pulse,
    BPSysAve,
    BPDiaAve,
    DirectChol,
    TotChol
  )
# Problem 1 ------------------------------------------------------------------
nhanes |>
  GGally::ggpairs(
    lower = list(continuous = wrap("points", alpha = 0.1))
  )
# Create a list of predictor variables (everything except BMI)
predictors <- setdiff(names(nhanes), "BMI")

# Generate residual vs. fitted plots
resid_plots <- map(predictors, function(var) {
  # Create formula for regression
  formula <- as.formula(paste("BMI ~", var))

  # Fit model
  fit <- lm(formula, data = nhanes)

  # Create tibble of fitted and residual values
  plot_data <- tibble(
    Fitted = fitted(fit),
    Residuals = resid(fit)
  )

  # Generate ggplot
  ggplot(plot_data, aes(x = Fitted, y = Residuals)) +
    geom_point(alpha = 0.2) +
    geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
    labs(
      title = paste("Residuals vs. Fitted for", var),
      x = "Fitted Values",
      y = "Residuals"
    ) +
    theme_minimal()
})

# Combine all plots into a grid using patchwork
wrap_plots(resid_plots) +
  plot_annotation(title = "Residuals vs. Fitted Values for Each Predictor")
```

```r
# Problem 2 ----------------------------------------------------------------
analyze_polynomial_relationship <- function(data, response, predictor, degree

  #  Cross-Validation Procedure
  cv_results <- list()
  for (degree in degrees_cv) {
    formula <- as.formula(paste(response, "~ poly(", predictor, ", degree = "
    control <- trainControl(method = "cv", number = 10)
    model <- train(formula, data = data, method = "lm", trControl = control)
    cv_results[[as.character(degree)]] <- model
  }

  cv_rmses <- sapply(cv_results, function(x) x$results$RMSE)
  best_cv_degree <- degrees_cv[which.min(cv_rmses)]
  best_cv_model <- lm(as.formula(paste(response, "~ poly(", predictor, ",", b

  #  Partial F-Test Procedure
  f_models <- list()
  f_models[[1]] <- lm(as.formula(paste(response, "~ poly(", predictor, ", 1)"
  for (d in 2:max_degree_f) {
    model_new <- lm(as.formula(paste(response, "~ poly(", predictor, ",", d,
    test <- anova(f_models[[d - 1]], model_new)
    f_models[[d]] <- model_new
    p_val <- test$`Pr(>F)`[2]
    if (is.na(p_val) || p_val > alpha) {
      best_f_degree <- d - 1
      best_f_model <- f_models[[d - 1]]
      last_p_val <- p_val
      break
    } else {
      best_f_degree <- d
      best_f_model <- model_new
      last_p_val <- p_val
    }
  }

  #  Prediction Data for Plotting
  height_seq <- seq(min(data[[predictor]], na.rm = TRUE),
                    max(data[[predictor]], na.rm = TRUE),
                    length.out = 200)
  pred_df <- tibble(!!predictor := height_seq)
  pred_cv <- predict(best_cv_model, newdata = pred_df)
  pred_f <- predict(best_f_model, newdata = pred_df)

  plot_data <- bind_rows(
    pred_df |> mutate(BMI = pred_cv, Method = paste0("CV (Degree ", best_cv_d
    pred_df |> mutate(BMI = pred_f, Method = paste0("F-test (Degree ", best_f_
```

```
  )

  #  Visualization
  #  Updated Visualization
  plot <- ggplot(data, aes_string(x = predictor, y = response)) +
    geom_point(alpha = 0.4) +
    geom_line(
      data = plot_data,
      aes_string(x = predictor, y = "BMI", color = "Method"),
      linewidth = 1
    ) +
    labs(
      title = paste("Polynomial Regression Fits for", predictor),
      subtitle = paste("CV Degree =", best_cv_degree, "| F-test Degree =", bes
      x = predictor,
      y = response,
      color = "Model Type"
    ) +
    theme_minimal() +
    theme(
      legend.position = "bottom",
      legend.title = element_text(size = 10),
      legend.text = element_text(size = 10),
      plot.title = element_text(size = 12),
      plot.subtitle = element_text(size = 10)
    )

  #  Output
  list(
    plot = plot,
    best_cv_degree = best_cv_degree,
    best_f_degree = best_f_degree,
    best_f_model = best_f_model,
    cv_rmse = cv_rmses,
    last_f_test_p_value = last_p_val
  )
}

nonlinear_preds <- c("Weight", "Height", "BPSysAve", "BPDiaAve", "DirectChol"

results <- map(nonlinear_preds, ~ {
  analyze_polynomial_relationship(nhanes, response = "BMI", predictor = paste
  }) |>
  set_names(nonlinear_preds)
map(results, ~ .x$plot) |>
  wrap_plots() +
  plot_annotation(title = "Residuals vs. Fitted Values for Each Predictor")
```

```r
map(results, ~ .x$cv_rmse) |>
  bind_cols() |>
  bind_cols(tibble("Degree" = 1:10)) |>
  pivot_longer(
    cols = -Degree,
    names_to = "Variable",
    values_to = "RMSE"
  ) |>
  mutate(Degree = factor(Degree, levels = paste0(1:10))) |>
  ggplot(aes(Degree, RMSE, color = Variable)) +
  geom_point() +
  geom_line(aes(group = Variable)) +
  ylim(c(0, 10)) +
  labs(title = "Cross Validation Error with Polynomial\nDegrees for Specific V
  scale_colour_viridis_d() +
  coord_equal()


# Problem 3 ------------------------------------------------------------------
#--- Strategic knots ----
strategic_knots <- list(
  Weight = c(60, 80, 100),
  Height = c(150, 170, 190),
  BPSysAve = c(110, 130, 150),
  BPDiaAve = c(70, 80, 90),
  DirectChol = c(30, 50, 70)
)

#--- CV wrapper function ----
cv_compare_splines <- function(data, response, predictor, manual_knots, df_qua
  set.seed(613)  # for reproducibility

  control <- trainControl(method = "cv", number = 10)

  # Quantile spline model using bs() with df
  quant_formula <- as.formula(glue("{response} ~ bs({predictor}, df = {df_quan
  quant_model <- train(quant_formula, data = data, method = "lm", trControl =
  quant_rmse <- quant_model$results$RMSE

  # Manual knot spline model
  manual_formula <- as.formula(
    glue("{response} ~ bs({predictor}, knots = c({paste(manual_knots, collapse
  )
  manual_model <- train(manual_formula, data = data, method = "lm", trControl
  manual_rmse <- manual_model$results$RMSE

  tibble(
```

```
    Predictor = predictor,
    Approach = c("Quantile Knots", "Strategic Knots"),
    RMSE = c(quant_rmse, manual_rmse)
  )
}

#--- Run across all nonlinear predictors ----
cv_comparisons <- map_dfr(nonlinear_preds, ~
  cv_compare_splines(nhanes, "BMI", .x, manual_knots = strategic_knots[[.x]])
)

#--- Plot comparison ----
ggplot(cv_comparisons, aes(x = Predictor, y = RMSE, fill = Approach)) +
  geom_col(position = position_dodge(width = 0.6), width = 0.5) +
  geom_text(aes(label = round(RMSE, 2)), vjust = -0.3, position = position_do
  labs(
    title = "Cross-Validation RMSE Comparison",
    subtitle = "Quantile-based vs. Strategically Placed Knots (10-fold CV)",
    y = "RMSE",
    x = "Predictor"
  ) +
  scale_fill_brewer(palette = "Set1") +
  theme_minimal(base_size = 12) +
  theme(legend.position = "bottom")
# Function to fit regression splines and create plots
fit_regression_spline <- function(data, response, predictor, df = 4) {

  # Construct formula
  formula <- as.formula(paste(response, "~ bs(", predictor, ", df =", df, ")"

  # Fit the model
  spline_model <- lm(formula, data = data)

  # Compute RMSE on training data
  fitted_vals <- predict(spline_model, newdata = data)
  rmse <- sqrt(mean((data[[response]] - fitted_vals)2, na.rm = TRUE))

  # Create a sequence of predictor values for smooth prediction
  pred_vals <- tibble(!!predictor := seq(min(data[[predictor]], na.rm = TRUE)
                                         max(data[[predictor]], na.rm = TRUE)
                                         length.out = 200))

  # Predict fitted values for plotting
  pred_vals[[response]] <- predict(spline_model, newdata = pred_vals)

  # Plot
  p <- ggplot(data, aes_string(x = predictor, y = response)) +
```

```
    geom_point(alpha = 0.4, color = "black") +
    geom_line(data = pred_vals, aes_string(x = predictor, y = response), colo
    labs(
      title = glue("Regression Spline Fit for {predictor}"),
      subtitle = glue("RMSE: {round(rmse, 2)}"),
      x = predictor,
      y = response
    ) +
    theme_minimal()

  # Return both plot and RMSE
  list(plot = p, rmse = rmse)
}

# Apply to all selected nonlinear predictors
splines <- map(nonlinear_preds, ~ fit_regression_spline(nhanes, response = "BM
spline_plots <- map(splines, ~ .x$p)




# Combine plots
# wrap_plots(spline_plots) +
#   plot_annotation(
#     title = "Regression Spline Fits for BMI with df = 4",
#     subtitle = glue("Spline with df = 4 (3 interior knots at quantiles)")
#   )




# Problem 4 ---------------------------------------------------------------
nhanes_new <- nhanes |> mutate(Weight_over_Height_sq = Weight/((Height/100)2)
ex_lm <- lm(BMI ~ Weight_over_Height_sq, data = nhanes_new) |> summary()
# gtsummary::tbl_regression(ex_lm)

# base_lm <- lm(BMI ~ ., data = nhanes)
nhanes_rm <- nhanes |> select(-c("Height", "Weight"))
# lm_rm1 <- lm(BMI ~ ., data = nhanes_rm)
# lm_rm1 |> summary()
# car::vif(lm_rm1)

spline_rmse <- map_vec(splines, ~ .x$rmse) |> set_names(nonlinear_preds)
poly_rmse <- map_vec(results, ~ {
  rmse_vec <- .x$cv_rmse
  names(rmse_vec) <- as.character(1:10)
  rmse_vec[which.min(rmse_vec)]
```

```
})
f_test_rmse <- map_vec(results, ~ sqrt(mean(.x$best_f_model$residuals2))) |>
  set_names(nonlinear_preds)

rmse_tbl <- tibble(
  Variable = nonlinear_preds,
  Spline = spline_rmse,
  Polynomial = poly_rmse,
  "Partial-F Test" = f_test_rmse
) |>
  rowid_to_column("row")

# Make gt table
gt_tbl <- rmse_tbl |>
  select(-row) |>
  gt() |>
  fmt_number(
    columns = c(Spline, Polynomial, `Partial-F Test`),
    decimals = 3
  ) |>
  tab_header(
    title = "RMSE Values for Different Fit Methods"
  )

# Add red color to minimum RMSE per row
for (i in seq_len(nrow(rmse_tbl))) {
  row_vals <- rmse_tbl[i, c("Spline", "Polynomial", "Partial-F Test")]
  min_col <- names(row_vals)[which.min(row_vals)]

  gt_tbl <- gt_tbl |>
    tab_style(
      style = cell_text(color = "red"),
      locations = cells_body(
        columns = all_of(min_col),
        rows = i
      )
    )
}


# gt_tbl

gam_mod <- gam(
  BMI ~ Age + Poverty + Pulse + TotChol +poly(BPSysAve, 6) + poly(BPDiaAve, 3
  data = nhanes_rm
)
# plot.Gam(gam_mod, se=TRUE)
```

```r
gam_lm1 <- lm(
  BMI ~ Age + Poverty + Pulse + TotChol +poly(BPSysAve, 6) + poly(BPDiaAve, 3
  data = nhanes_rm
)

gam_lm <- lm(
  BMI ~ Age + Poverty + Pulse + TotChol +poly(BPSysAve, 6) + poly(BPDiaAve, 3
  data = nhanes_rm
)

gtsummary::tbl_regression(
  gam_lm1,
  estimate_fun = ~ gtsummary::style_number(.x, digits = 2)
)

gtsummary::tbl_regression(
  gam_lm,
  estimate_fun = ~ gtsummary::style_number(.x, digits = 2)
)
library("mgcv")

# Extract names of all predictors
predictor_vars <- all.vars(formula(gam_mod))[-1]  # drop response var

# Get mean values for all predictors (used for conditioning)
mean_vals <- nhanes_rm |>
  summarise(across(all_of(predictor_vars), ~ mean(.x, na.rm = TRUE)))

# Function to make partial effect plot for a given variable
make_partial_plot <- function(var) {
  # Generate a sequence for the variable
  grid_vals <- tibble(!!sym(var) := seq(
    min(nhanes_rm[[var]], na.rm = TRUE),
    max(nhanes_rm[[var]], na.rm = TRUE),
    length.out = 200
  ))

  # Hold other variables at their means
  other_vals <- mean_vals |> select(-all_of(var))
  newdata <- crossing(grid_vals, other_vals)

  # Get predictions and standard errors
  pred <- predict(gam_mod, newdata = newdata, se.fit = TRUE, type = "response

  # Assemble into data frame
```

```r
  plot_df <- newdata |>
    mutate(
      fit = pred$fit,
      se = pred$se.fit,
      lower = fit - 2 * se,
      upper = fit + 2 * se
    )

  scatter_df <- nhanes_rm |> select(all_of(var), BMI)

  # Plot
  ggplot() +
    geom_point(data = scatter_df, aes_string(x = var, y = "BMI"),
               alpha = 0.3, size = 0.8, color = "black") +
    geom_line(data = plot_df, aes_string(x = var, y = "fit"), color = "steelb
    geom_ribbon(data = plot_df, aes_string(x = var, ymin = "lower", ymax = "u
                alpha = 0.5, fill = "steelblue") +
    labs(
      title = paste("Partial Effect of", var),
      x = var,
      y = "BMI"
    ) +
    theme_minimal()
}

# Generate plots for all variables
plots <- map(predictor_vars, make_partial_plot)

# Combine with patchwork
# wrap_plots(plots, ncol = 2) + plot_annotation(title = "Partial Effects from



# Problem 5 ------------------------------------------------------------------
set.seed(613)
split <- initial_split(nhanes_rm, prop = .8)
nhanes_rm_train <- training(split)
nhanes_rm_test <- testing(split)
gam_lm_train <- lm(
  BMI ~ Age + Poverty + Pulse + TotChol +poly(BPSysAve, 6) + poly(BPDiaAve, 3
  data = nhanes_rm_train
)
gam_train <- gam::gam(
  BMI ~ Age + Poverty + Pulse + TotChol +poly(BPSysAve, 6) + poly(BPDiaAve, 3
  data = nhanes_rm_train
)
```

```r
# Function to make partial effect plot for a given variable
make_partial_plot <- function(var) {
  # Generate a sequence for the variable
  grid_vals <- tibble(!!sym(var) := seq(
    min(nhanes_rm[[var]], na.rm = TRUE),
    max(nhanes_rm[[var]], na.rm = TRUE),
    length.out = 200
  ))

  # Hold other variables at their means
  other_vals <- mean_vals |> select(-all_of(var))
  newdata <- crossing(grid_vals, other_vals)

  # Get predictions and standard errors
  pred <- predict(gam_train, newdata = newdata, se.fit = TRUE, type = "respons

  # Assemble into data frame
  plot_df <- newdata |>
    mutate(
      fit = pred$fit,
      se = pred$se.fit,
      lower = fit - 2 * se,
      upper = fit + 2 * se
    )

  scatter_df <- nhanes_rm |> select(all_of(var), BMI)

  # Plot
  ggplot() +
    geom_point(data = scatter_df, aes_string(x = var, y = "BMI"),
               alpha = 0.3, size = 0.8, color = "black") +
    geom_line(data = plot_df, aes_string(x = var, y = "fit"), color = "steelb
    geom_ribbon(data = plot_df, aes_string(x = var, ymin = "lower", ymax = "u
                alpha = 0.5, fill = "steelblue") +
    labs(
      title = paste("Partial Effect of", var),
      x = var,
      y = "BMI"
    ) +
    theme_minimal()
}

# Generate plots for all variables
plots <- map(predictor_vars, make_partial_plot)

# Combine with patchwork
# wrap_plots(plots, ncol = 2) + plot_annotation(title = "Partial Effects from
```

```r
gtsummary::tbl_regression(
  gam_lm_train,
  estimate_fun = ~ gtsummary::style_number(.x, digits = 2)
)


# Problem 6 ------------------------------------------------------------------
sqrt(mean((predict.lm(gam_lm_train, newdata = nhanes_rm_test) - nhanes_rm_tes

# Problem 7 ------------------------------------------------------------------
lm_train <- lm(BMI ~ ., data = nhanes_rm_train)
sqrt(mean((predict.lm(lm_train, newdata = nhanes_rm_test) - nhanes_rm_test$BM
```