STA 5363, Homework 2

Carson Slater *Baylor University*

1. Create your own data set to analyze the NHANES data using some filtering (e.g handling missing values and dropping observations and variables). Consider the selected variables below instead of all variables in the original data set. Justify your filtering procedure.

The initial step in processing the NHANES data involved loading the data set and selecting a specific subset of 36 variables, a decision made to focus the analysis on key demographic, socioeconomic, health, and lifestyle indicators while excluding less relevant columns. To ensure data integrity, duplicate rows were removed using `distinct()`. However, it was discovered that some individuals (identified by `ID`) had multiple, conflicting records. A decision was made to resolve this by retaining only the single record for each person that contained the fewest missing (NA) values, a strategy that prioritizes the most complete observation for each participant. Following this, a data quality threshold was established: any variable with more than 20% of its values missing was removed from the data set. This decision ensures that the subsequent analysis is built upon variables with a high degree of completeness. For the remaining missing data, an imputation strategy was implemented using a recipe. The `ID` column was explicitly excluded from being a predictor. A sophisticated machine learning approach was chosen for imputation: bagged decision trees were used for numeric variables and the k-nearest neighbors (KNN) algorithm was used for categorical variables.

```
> prep(impute_rec, new_data = indiv_train)
--- Recipe ----------------------------------------------------------

--- Inputs
Number of variables by role
outcome:    1
predictor: 16
ID:         1

--- Training information
Training data contained 6779 data points and 2329 incomplete rows.

--- Operations
• Bagged tree imputation for: Age and HHIncomeMid, ... | Trained
• K-nearest neighbor imputation for: Gender, ... | Trained
> indiv_train_bake <- bake(impute_rec, new_data = indiv_train)
```

These methods were selected over simpler techniques like mean/median imputation because they can capture complex, non-linear relationships in the data, leading to more accurate and plausible imputed values. The final bake step applies this trained imputation logic, resulting in a clean and complete dataset ready for modeling.

```
> colMeans(is.na(indiv_train_bake)) |> round(2)
          ID       Gender          Age        Race1 HHIncomeMid
        0.00         0.00         0.00         0.00         0.00
     Poverty    HomeRooms     HomeOwn       Weight       Height
        0.00         0.00         0.00         0.00         0.00
       Pulse    BPSysAve     BPDiaAve  DirectChol      TotChol
        0.00         0.00         0.00         0.00         0.00
    Diabetes  PhysActive          BMI
        0.00         0.00         0.00
```

Per the code output in the verbatim environment above, all of the columns selected have no missing data.

2. EDA

(a) Produce a scatterplot matrix which includes all of the variables in your own data set. Report your findings. (b) Compute the matrix of correlations between variables after excluding qualitative variables. Report your findings.

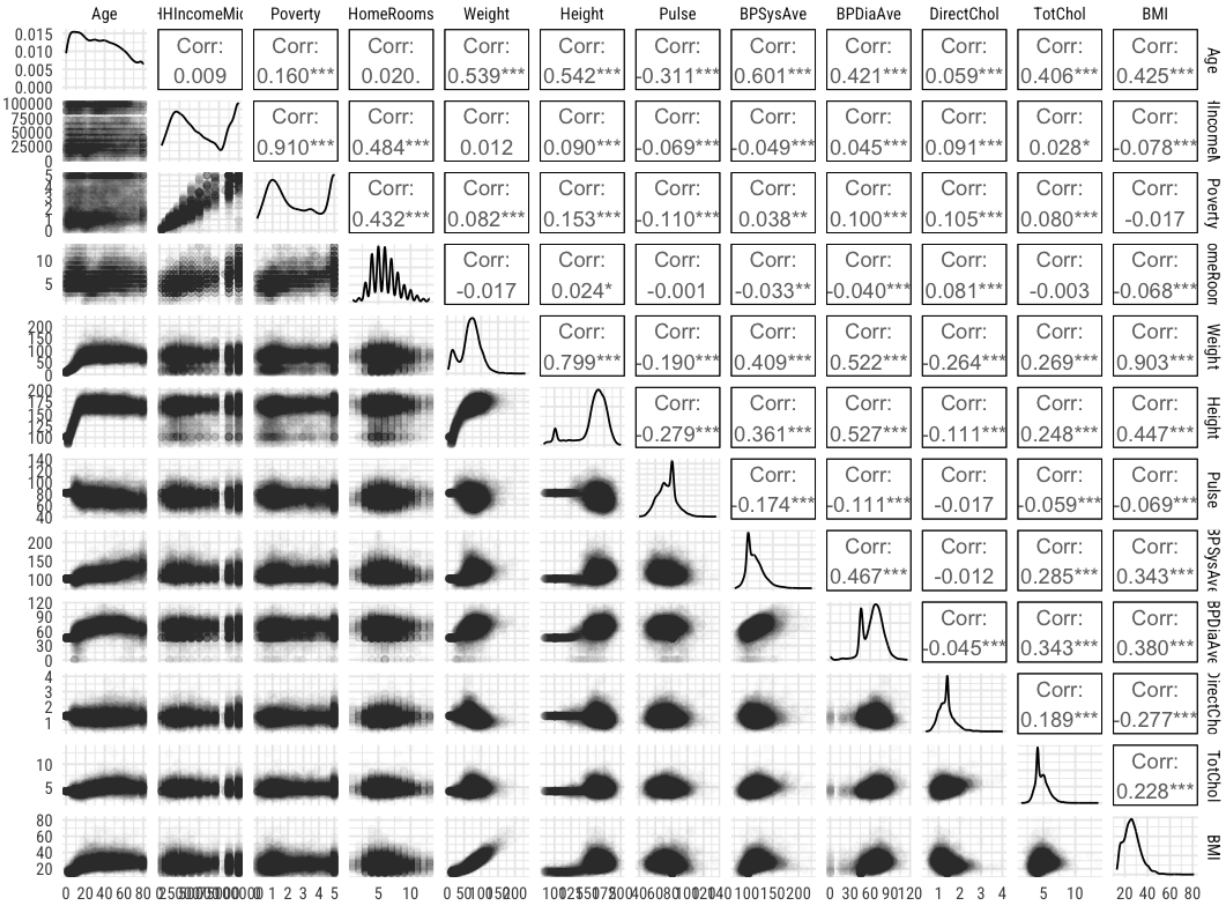

Figure 1: Pairs plot for filtered and imputed NHANES data.

The last row of the pairs plot shows the relationship between each variable and BMI. We can group the potential predictors by the strength and significance of their correlation with BMI.

- **Primary Predictors:** The relationship between BMI and its component measures, **Weight** and **Height**, is, as expected, very strong.
  - **Weight:** Shows a very strong, positive, and linear relationship with BMI ($r = 0.903^{***}$). This is the most dominant predictor.
  - **Height:** Shows a moderate positive correlation ($r = 0.447^{***}$).

  It is critical to note that BMI is calculated directly from Weight and Height ($BMI = \frac{\text{weight}}{\text{height}^2}$). Including them as predictors in a regression model for BMI is not appropriate since BMI is a function of Weight and Height.
- **Moderate Predictors:** Several other variables show statistically significant and moderately strong correlations with BMI. These would be the primary candidates for a model that excludes Weight and Height.
  - **Age:** A moderate positive correlation ($r = 0.425^{***}$).
  - **BPDiaAve** (Average Diastolic Blood Pressure): A moderate positive correlation ($r = 0.380^{***}$).
  - **BPSysAve** (Average Systolic Blood Pressure): A moderate positive correlation ($r = 0.343^{***}$).
  - **DirectChol** (Direct Cholesterol): A moderate negative correlation ($r = -0.277^{***}$).
  - **TotChol** (Total Cholesterol): A weak-to-moderate positive correlation ($r = 0.228^{***}$).
- **Weak or Insignificant Predictors:** The following variables have very weak correlations with BMI and are likely poor candidates for inclusion in the model.
  - **HHIncomeMid**, **Poverty**, **HomeRooms**, and **Pulse**. All have correlation coefficients with an absolute value less than 0.1.

\subsubsection**{Multicollinearity Among Predictors}

When building a multiple regression model, it is important to check for high correlations between predictor variables, as this can lead to unstable coefficient estimates.

- **Weight and Height:** These variables exhibit a very high positive correlation ($r = 0.799^{***}$). If one were to build a model including them, this high degree of multicollinearity would be a significant issue.
- **Blood Pressure Variables:** Average Systolic (BPSysAve) and Average Diastolic (BPDiaAve) blood pressures are moderately correlated ($r = 0.467^{***}$). Including both in a model might introduce multicollinearity. It may be preferable to select one (e.g., BPDiaAve due to its slightly higher correlation with BMI) or use a combined measure.
- **Age and Other Variables:** Age is moderately to strongly correlated with several other potential predictors, including BPSysAve ($r = 0.601^{***}$), Weight ($r = 0.539^{***}$), Height ($r = 0.542^{***}$), and TotChol ($r = 0.406^{***}$). The inclusion of Age will likely influence the coefficients of these other variables.

3. Split your data into training (80%) and test (20%) data sets.

```
# Problem 3 ----------------------------------------------------------

split <- initial_split(indiv_train_bake, prop = 0.8, strata = Race1)
indiv_train <- training(split)
indiv_test <- testing(split)
```

4. Fit a multiple regression model to training, provide an interpretation of each coefficient in the model with statistical significance. Compute test MSE by predicting on the test data.

Using information from our graphic in Figure 1, we elect to exclude the following predictors to avoid misuse of variables or multicollinearity: `Weight`, `Height`, `BPSysAve`, and `DirectChol`. Again, the ID column is also not a predictor, but is useful in the data cleaning process. We use the rest and do not perform any more formal variable selection.

Characteristic	**Beta**	**95% CI**[1]	**p-value**
Gender			
female	—	—	
male	-0.64	-1.0, -0.28	<0.001
Age	0.09	0.08, 0.10	<0.001
Race1			
Black	—	—	
Hispanic	-0.99	-1.8, -0.21	0.012
Mexican	-0.25	-0.93, 0.43	0.5
White	-1.3	-1.8, -0.73	<0.001
Other	-3.1	-3.8, -2.3	<0.001
BPDiaAve	0.13	0.11, 0.14	<0.001
TotChol	0.27	0.07, 0.46	0.007
HomeRooms	-0.04	-0.14, 0.06	0.4
HomeOwn			
Own	—	—	
Rent	0.43	-0.02, 0.88	0.063
Other	0.33	-0.78, 1.4	0.6
HHIncomeMid	0.00	0.00, 0.00	0.8
Poverty	-0.13	-0.41, 0.14	0.3
Pulse	0.01	0.00, 0.03	0.077
Diabetes			
No	—	—	
Yes	3.1	2.4, 3.8	<0.001
PhysActive			
No	—	—	
Yes	-1.3	-1.7, -0.87	<0.001

[1]CI = Confidence Interval

Computing the MSE for this initial model, we have:

```
mean((predict(mlr, newdata = indiv_test) - indiv_test$BMI)2, na.rm = TRUE)
```

```
## [1] 45.50236
```

5. Perform variable selection in order to choose the best model with training. Compute test MSE by predicting on the test data.

We performed an exhaustive all subsets regression analysis and chose the model with the lowest BIC.

```
> subset_summary$outmat[best_bic_model, ]
   Gendermale            Age Race1Hispanic  Race1Mexican
          "*"            "*"           "*"           " "
   Race1White    Race1Other      BPDiaAve       TotChol
          "*"            "*"           "*"           " "
    HomeRooms   HomeOwnRent  HomeOwnOther   HHIncomeMid
          " "            " "           " "           "*"
      Poverty          Pulse   DiabetesYes PhysActiveYes
          " "            " "           "*"           "*"
```

Characteristic	Beta	95% CI[1]	p-value
Gender			
female	—	—	
male	-0.72	-1.1, -0.37	<0.001
Age	0.09	0.08, 0.10	<0.001
Race1			
Black	—	—	
Hispanic	-0.86	-1.6, -0.09	0.029
Mexican	-0.20	-0.87, 0.48	0.6
White	-1.3	-1.8, -0.76	<0.001
Other	-3.0	-3.8, -2.3	<0.001
BPDiaAve	0.13	0.12, 0.15	<0.001
HHIncomeMid	0.00	0.00, 0.00	0.007
Diabetes			
No	—	—	
Yes	3.1	2.4, 3.7	<0.001
PhysActive			
No	—	—	
Yes	-1.3	-1.7, -0.94	<0.001

[1]CI = Confidence Interval

Males had a significantly lower BMI than females by 0.72 units ($p < 0.001$). Age was positively associated with BMI, increasing by 0.09 units per year ($p < 0.001$). Compared to Black individuals, those identifying as Hispanic, White, and Other races had significantly lower BMIs, with the largest reduction seen in the "Other" category ($\hat{\beta} = -3.0$, $p < 0.001$). Mexican identity was not associated with a significant BMI difference ($p = 0.6$). Higher diastolic blood pressure was associated with increased BMI ($\hat{\beta} = 0.13$, $p < 0.001$), and mid-level household income had a statistically significant but negligible effect ($\hat{\beta} = 0.00$, $p = 0.007$). Individuals with diabetes had BMIs that were 3.1 units higher than those without ($p < 0.001$), and physically active individuals had BMIs 1.3 units lower than inactive individuals ($p < 0.001$). These findings highlight the persistent influence of demographic, health, and lifestyle factors on BMI.

```
mean((predict(mlr_selection, newdata = indiv_test) - indiv_test$BMI)2, na.rm
```

```
## [1] 42.30766
```

After variable selection, the MSE was slightly lower, and after removing variables that did not seem to be predictors, the model is simpler than before is at less risk for inflated standard errors, and multicollinearity.

6. Fit a ridge regression on training, with λ chosen by cross-validation (CV). Report test data MSE.

Table 3: Coefficient estimates from the ridge regression model fit on the training data.

Predictor	Estimate
(Intercept)	26.444
Age	1.946
BPDiaAve	1.833
TotChol	0.310
HomeRooms	-0.092
HHIncomeMid	-0.056
Poverty	-0.118
Pulse	0.130
Gender_male	-0.308
Race1_Hispanic	-0.213
Race1_Mexican	-0.039
Race1_White	-0.527
Race1_Other	-0.800
HomeOwn_Rent	0.180
HomeOwn_Other	0.051
Diabetes_Yes	0.853
PhysActive_Yes	-0.637

We fit a ridge regression model and chose $\lambda = 0.0000000001$ using 10-fold cross validation. All computations were performed using the `tidymodels` metapackage.

```
test_mse
```

```
## [1] 42.37768
```

We were able to achieve an MSE of 42.378 with ridge regression.

7. Fit a lasso regression on training, with λ chosen by CV. Report test data MSE, along with non-zero predictors.

Table 4: Coefficient estimates from the LASSO model fit on the training data.

Predictor	Estimate
(Intercept)	26.444
Age	2.034
BPDiaAve	1.885
TotChol	0.256
HomeRooms	-0.071
Poverty	-0.167
Pulse	0.136
Gender_male	-0.304
Race1_Hispanic	-0.186
Race1_White	-0.518
Race1_Other	-0.798
HomeOwn_Rent	0.185
HomeOwn_Other	0.029
Diabetes_Yes	0.843
PhysActive_Yes	-0.616

We fit a ridge regression model and chose $\lambda = 0.023$ using 10-fold cross validation. All computations were performed using the `tidymodels` metapackage.

```
lasso_test_mse
```

```
## [1] 42.37506
```

We achieve a marginally lower MSE with LASSO than the ridge regression model. We were able to achieve an MSE of 42.375.

8. Fit a PCR on training, with M chosen by CV. Report test data MSE, along with the value of M selected by CV.

We first manually create dummy variables for the `pls::pcr()` function, and choose the number of components to be the minimum number that explains at least 85% of the total variance. We chose twelve components after using cross validation.

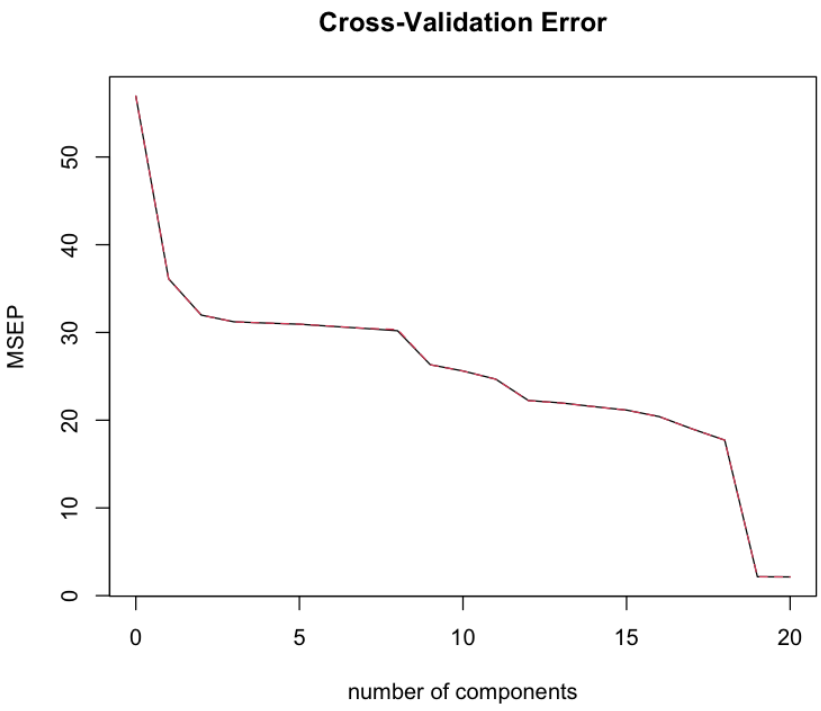

Figure 2: Cross-validation error plot with the number of principle components used in PCR.

```
> # Model summary
> summary(pcr_final)
Data:   X dimension: 6475 20
        Y dimension: 6475 1
Fit method: svdpc
Number of components considered: 12
TRAINING: % variance explained
     1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X      17.54    31.29    38.97    44.90    50.75    56.33
BMI    36.60    43.91    45.28    45.52    45.81    46.29
     7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
X      61.76    66.93    72.00     76.23     80.07     83.57
BMI    46.76    47.13    53.99     55.26     56.94     61.13
```

The final model coefficients are shown in the following above. We also compute the test set MSE.

Data: X dimension: 6475 20 Y dimension: 6475 1 Fit method: svdpc Number of components considered: 12 TRAINING: % variance explained 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps X 17.55 31.29 38.97 44.91 50.76 56.33 61.76 66.93 BMI 36.75 43.97 45.32 45.53 45.82 46.29 46.76 47.11 9 comps 10 comps 11 comps 12 comps X 72.00 76.23 80.07 83.57 BMI 54.01 55.29 56.91 61.15

Table 5: Principal Component Regression Coefficients (12 Components)

Variable	Coefficient
(Intercept)	6.358
Age	0.764
HHIncomeMid	-0.306
Poverty	-0.168
HomeRooms	-0.083
Weight	2.494
Height	1.914
Pulse	0.568
BPSysAve	-0.014
BPDiaAve	1.121
DirectChol	-1.507
TotChol	-0.311
Gender_male	-1.512
Race_Black	-0.076
Race_Hispanic	-0.294
Race_Mexican	-0.103
Race_Other	-0.753
HomeOwn_Rent	0.060
HomeOwn_Other	-0.106
Diabetes_Yes	0.723
PhysActive_Yes	-0.477

```
# Predict on test set
test_predictions <- predict(
  pcr_final,
  newdata = pcr_test_data,
  ncomp = n_comp
)
(pcr_test_mse <- mean((pcr_test_data$BMI - test_predictions)2))
```

[1] 22.2219

From PCR with 12 components we were able to achieve an MSE of 22.22 units.

9. Fit a PLS on training, with M chosen by CV. Report test data MSE, along with the value of M selected by CV.

We first manually create dummy variables for the `pls::plsr()` function, and choose the number of components to be the number minimizes the training variance. We chose twelve components after using cross validation.

```
> # Model summary
```

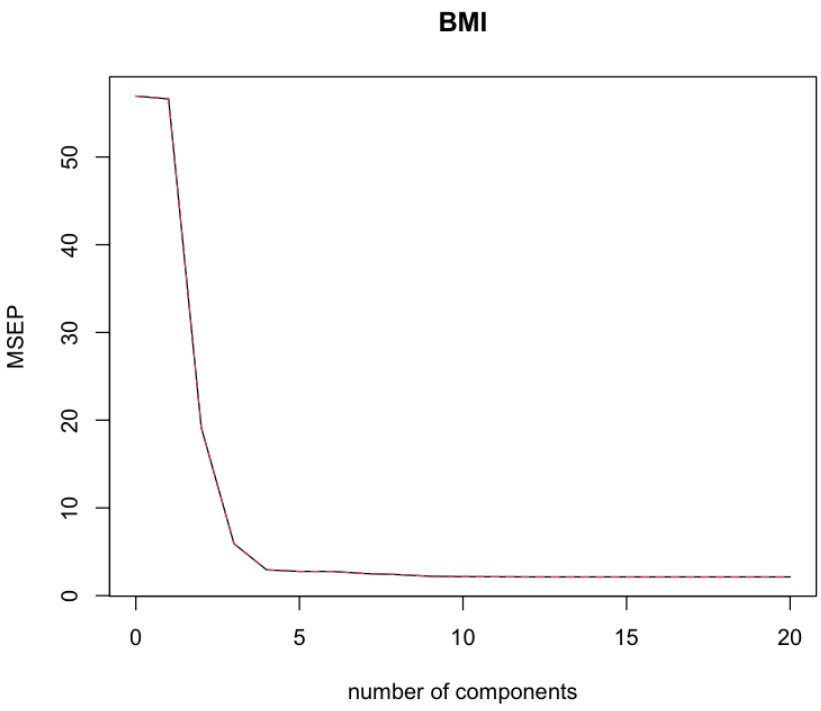

Figure 3: Cross-validation error plot with the number of components used in PLS.

```
> summary(final_pls)
Data:   X dimension: 6475 20
        Y dimension: 6475 1
Fit method: kernelpls
Number of components considered: 15
TRAINING: % variance explained
     1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X    99.9998   100.00   100.00   100.00   100.00   100.00
BMI   0.6092    66.41    89.63    94.86    95.16    95.17
     7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
X     100.00   100.00   100.00    100.00    100.00    100.00
BMI    95.62    95.82    96.15     96.22     96.25     96.27
     13 comps  14 comps  15 comps
X      100.00    100.00    100.00
BMI     96.28     96.28     96.28
```

Table 6: Partial Least Squares Coefficients (15 Components)

Variable	Coefficient
(Intercept)	32.010
Age	0.019
HHIncomeMid	0.000
Poverty	-0.127
HomeRooms	0.001
Weight	0.347
Height	-0.187
Pulse	0.000
BPSysAve	0.000
BPDiaAve	-0.001
DirectChol	-0.079
TotChol	0.022
Gender_male	-1.652
Race_Black	0.142
Race_Hispanic	0.553
Race_Mexican	0.701
Race_Other	0.284
HomeOwn_Rent	0.028
HomeOwn_Other	0.062
Diabetes_Yes	0.131
PhysActive_Yes	0.023

After fitting the model, we chose the optimal number of components to be 15 from cross validation. The estimated coefficients for the final model are in the table above. With this model we were able to obtain a testing set MSE of 2.211.

```
pls_test_mse
```

```
[1] 2.210811
```

10. Summarize your findings. Which model is the best in terms of test MSE? How did the dimension techniques work? Which variables are important in predicting `BMI`?

Model Performance Comparison

Based on the test MSE results across all fitted models, the performance ranking from best to worst is as follows:

The Partial Least Squares (PLS) model achieved the best performance with a test MSE of 2.211, representing a dramatic improvement over all other approaches. This is followed by PCR with an MSE of 22.222, while the remaining models (LASSO, Ridge, variable selection, and standard MLR) all performed similarly with MSEs in the 42-45 range.

Model	Test MSE
Partial Least Squares (PLS)	2.211
Principal Component Regression (PCR)	22.222
LASSO Regression	42.375
Ridge Regression	42.378
Variable Selection (BIC)	42.308
Multiple Linear Regression	45.502

Table 7: Test MSE comparison across all models

How Dimension Reduction Techniques Work

The dimension reduction techniques employed in this analysis work through fundamentally different approaches:

Principal Component Regression (PCR)

PCR reduces dimensionality by finding linear combinations of predictors that maximize variance in the predictor space, regardless of their relationship to the response variable. In our analysis, PCR with 12 components was able to explain 83.57% of the variance in the predictors and 61.13% of the variance in BMI. The method creates orthogonal components that eliminate multicollinearity issues observed in the original predictors.

Partial Least Squares (PLS)

PLS takes a supervised approach by finding linear combinations that simultaneously maximize variance in predictors while maintaining strong correlation with the response variable. Our PLS model with 15 components explained nearly 100% of predictor variance while achieving 96.28% explained variance in `BMI`. This supervised dimension reduction explains why PLS dramatically outperformed PCR and all other methods—it specifically optimizes for prediction of the response variable rather than just reducing predictor dimensionality.

The superior performance of PLS suggests that the relationship between `BMI` and the predictors benefits significantly from this supervised approach to dimension reduction, as PLS components are constructed to be maximally predictive of `BMI` rather than just capturing general variation in the predictor space.

Consistently Important Predictors

The following variables showed consistent importance across multiple models:

- **Age**: Consistently significant across all models with positive associations (e.g., $\hat{\beta} = 0.09$ in the variable selection model, $p < 0.001$)

- **Gender**: Males consistently had lower BMI than females (e.g., $\hat{\beta} = -0.72$ in variable selection model, $p < 0.001$)
- **Diabetes Status**: Individuals with diabetes had substantially higher BMI across all models (e.g., $\hat{\beta} = 3.1$ in variable selection model, $p < 0.001$)
- **Physical Activity**: Physically active individuals consistently showed lower BMI (e.g., $\hat{\beta} = -1.3$ in variable selection model, $p < 0.001$)
- **Diastolic Blood Pressure (BPDiaAve)**: Showed consistent positive association with BMI (e.g., $\hat{\beta} = 0.13$ in variable selection model, $p < 0.001$)

Race/Ethnicity Effects

Compared to Black individuals (reference category), all other racial/ethnic groups showed significantly lower BMI values, with the "Other" category showing the largest reduction ($\hat{\beta} = -3.0$, $p < 0.001$). Hispanic and White individuals also showed significant reductions in section* compared to Black individuals.

Variables with Inconsistent or Weak Effects

Several variables showed weak or inconsistent relationships with BMI:

- Household income, poverty status, and number of home rooms had very weak correlations with BMI ($|r| < 0.1$)
- Total cholesterol showed some significance in certain models but was eliminated in variable selection
- Home ownership status showed minimal predictive value

Excluded Variables

As expected from the EDA, Weight and Height were excluded from most models due to their definitional relationship with BMI ($BMI = \frac{weight}{height^2}$). Including these would create a tautological relationship rather than meaningful prediction. Additionally, systolic blood pressure was excluded due to multicollinearity with diastolic blood pressure.

Conclusion

The analysis demonstrates that while traditional regression approaches provide interpretable results, dimension reduction techniques—particularly PLS—can achieve dramatically superior predictive performance when the goal is minimizing prediction error. The success of PLS in this context suggests that BMI prediction benefits from supervised dimension reduction that specifically targets the response variable rather than unsupervised approaches that focus solely on predictor variance. The consistently important predictors (age, gender, diabetes status, physical activity, and blood pressure) align with established medical knowledge about BMI determinants, providing confidence in the model's validity.

Code Appendix

```
knitr::opts_chunk$set(dev = "cairo_pdf",
                      fig.width = 5,
                      fig.height = 3.25,
                      fig.align = 'center',
                      echo = FALSE,
                      message = FALSE,
                      warning = TRUE,
                      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("leaps")
library("broom")
library("pls")
library("knitr")
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
)
# Problem 1 ----------------------------------------------------------

library("NHANES")
data("NHANES")
# ?NHANES
new_data <- NHANES |>
  select(
    ID, Gender, Age, Race1, Education, MaritalStatus, HHIncomeMid,
```

```
    Poverty, HomeRooms, `HomeOwn`, Weight, Height, BMI, Pulse,
    BPSysAve, BPDiaAve, DirectChol, TotChol, Diabetes, HealthGen,
    DaysPhysHlthBad, DaysMentHlthBad, Depressed, SleepHrsNight,
    PhysActive,PhysActiveDays, TVHrsDay, CompHrsDay, Alcohol12PlusYr,
    AlcoholDay, SmokeNow, Smoke100, Marijuana, HardDrugs, SexEver,
    SexNumPartnLife
 ) |> distinct()

# Found multiple people with duplicate rows and conflicting information.
# Selected row for each person with least amount of NA values
indiv <- new_data |>
  group_by(ID) |>
  group_split() |>
  map_dfr(~ .x[which.min(rowSums(is.na(.x))), ])

# Selected variables with at least 80% of the data
na_props <- colMeans(is.na(indiv))
fltr_cols <- na_props[which(na_props <= 0.20)] |> names()

indiv <- indiv |> select(all_of(fltr_cols))

impute_rec <- recipe(BMI ~ ., data = indiv) |>
  # Assign "ID" as the role for the ID column so it's not used as a predictor
  update_role(ID, new_role = "ID") |>

  # Impute numeric variables using all other valid predictors.
  # Omitting `impute_with` uses the default, which is `all_predictors()`.
  step_impute_bag(all_numeric_predictors()) |>

  # Impute factor variables using all other valid predictors.
  step_impute_knn(all_factor_predictors())

# Prep the recipe and save the trained object
impute_rec_trained <- prep(impute_rec, training = indiv)

# Now, use the trained recipe object to bake the data
indiv_train_bake <- bake(impute_rec_trained, new_data = indiv) |>
  drop_na(BMI)

# View the imputed data
# head(indiv_train_bake)

# indiv_train_bake |>
#   select(-c("BMI", "ID")) |>  # Don't check the outcome variable
#   map_df(~sum(is.na(.))) |>
#   glimpse()
```

```r
# colMeans(is.na(indiv_train_bake)) |> round(2)


# Problem 2a & b ----------------------------------------------------

indiv_train_bake |>
  select(where(is.numeric)) |>
  select(-ID) |>
  GGally::ggpairs(
     lower = list(continuous = wrap("points", alpha = 0.01))
  )


# Problem 3 ---------------------------------------------------------

split <- initial_split(indiv_train_bake, prop = 0.8, strata = Race1)
indiv_train <- training(split)
indiv_test <- testing(split)


# Problem 4 ---------------------------------------------------------

indiv_train <- indiv_train |>
  select(
    BMI,
    Gender,
    Age,
    Race1,
    BPDiaAve,
    TotChol,
    HomeRooms,
    HomeOwn,
    HHIncomeMid,
    Poverty,
    Pulse,
    Diabetes,
    PhysActive
  )

mlr <- lm(BMI ~ ., data = indiv_train)

gtsummary::tbl_regression(mlr)
mean((predict(mlr, newdata = indiv_test) - indiv_test$BMI)2, na.rm = TRUE)


# Problem 5 ---------------------------------------------------------
```

```r
# Perform all subsets regression
subset_fit <- regsubsets(BMI ~ ., data = indiv_train, nvmax = NULL, method = 
# Summary of the results
subset_summary <- summary(subset_fit)
# Find best model by Bayesian Information Criterion
best_bic_model <- which.min(subset_summary$bic)

subset_summary$outmat[best_bic_model, ]
mlr_selection <- lm(BMI ~ Gender + Age + Race1 + BPDiaAve + HHIncomeMid + Dia

gtsummary::tbl_regression(mlr_selection)
mean((predict(mlr_selection, newdata = indiv_test) - indiv_test$BMI)2, na.rm =


# Problem 6 ----------------------------------------------------------

# 1. Define the ridge model with tunable penalty (lambda)
ridge_spec <- linear_reg(penalty = tune(), mixture = 0) |>  # mixture = 0 for
  set_engine("glmnet")

# 2. Set up 10-fold cross-validation
set.seed(123)
cv_folds <- vfold_cv(indiv_train, v = 10)

# 3. Create recipe
ridge_recipe <- recipe(BMI ~ ., data = indiv_train) |>
  step_corr(threshold = 0.8) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_predictors())


# 4. Create workflow
ridge_workflow <- workflow() |>
  add_model(ridge_spec) |>
  add_recipe(ridge_recipe)

# 5. Define grid of penalty values
lambda_grid <- grid_regular(penalty(), levels = 50)

# 6. Tune the model using CV
ridge_tuned <- tune_grid(
  ridge_workflow,
  resamples = cv_folds,
  grid = lambda_grid,
  metrics = metric_set(rmse)
)
```

```r
# 7. Select best lambda
best_ridge <- select_best(ridge_tuned, metric = "rmse")

# 8. Finalize workflow with best lambda
final_ridge_workflow <- finalize_workflow(ridge_workflow, best_ridge)

# 9. Fit final model on training and evaluate on test set
final_ridge_fit <- fit(final_ridge_workflow, data = indiv_train)

final_ridge_model <- extract_fit_parsnip(final_ridge_fit)

# 10. Predict and compute test MSE
ridge_preds <- predict(final_ridge_fit, indiv_test) |>
  bind_cols(indiv_test)

# Compute test MSE
test_mse <- ridge_preds |>
  metrics(truth = BMI, estimate = .pred) |>
  filter(.metric == "rmse") |>
  mutate(mse = .estimate2) |>
  pull(mse)
tidy(final_ridge_model) |>
  select(term, estimate) |>
  `colnames<-`(c("Predictor", "Estimate")) |>
  knitr::kable(digits = 3)
test_mse


# Problem 7 ------------------------------------------------------------

# 1. Define the lasso model with tunable penalty (lambda)
lasso_spec <- linear_reg(penalty = tune(), mixture = 1) |>  # mixture = 1 for
  set_engine("glmnet")

# 2. Set up 10-fold cross-validation
set.seed(123)
cv_folds <- vfold_cv(indiv_train, v = 10)

# 3. Create recipe
lasso_recipe <- recipe(BMI ~ ., data = indiv_train) |>
  step_corr(threshold = 0.8) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_predictors())


# 4. Create workflow
lasso_workflow <- workflow() |>
```

```r
  add_model(lasso_spec) |>
  add_recipe(lasso_recipe)

# 5. Define grid of penalty values
lambda_grid <- grid_regular(penalty(), levels = 50)

# 6. Tune the model using CV
lasso_tuned <- tune_grid(
  lasso_workflow,
  resamples = cv_folds,
  grid = lambda_grid,
  metrics = metric_set(rmse)
)

# 7. Select best lambda
best_lasso <- select_best(lasso_tuned, metric = "rmse")

# 8. Finalize workflow with best lambda
final_lasso_workflow <- finalize_workflow(lasso_workflow, best_lasso)

# 9. Fit final model on training and evaluate on test set
final_lasso_fit <- fit(final_lasso_workflow, data = indiv_train)

final_lasso_model <- extract_fit_parsnip(final_lasso_fit)

# 10. Predict and compute test MSE
lasso_preds <- predict(final_lasso_fit, indiv_test) |>
  bind_cols(indiv_test)

# Compute test MSE
lasso_test_mse <- lasso_preds |>
  metrics(truth = BMI, estimate = .pred) |>
  filter(.metric == "rmse") |>
  mutate(mse = .estimate2) |>
  pull(mse)
tidy(final_lasso_model) |>
  select(term, estimate) |>
  filter(!near(estimate, 0, tol = 0.001)) |>
  `colnames<-`(c("Predictor", "Estimate")) |>
  knitr::kable(digits = 3)
lasso_test_mse


# Problem 8 ----------------------------------------------------------

# Prepare data for PCR - remove ID and convert factors to dummy variables
pcr_data <- indiv_train_bake |>
```

```r
  select(-ID) |>
  mutate(
    Gender_male = as.numeric(Gender == "male"),
    Race_Black = as.numeric(Race1 == "Black"),
    Race_Hispanic = as.numeric(Race1 == "Hispanic"),
    Race_Mexican = as.numeric(Race1 == "Mexican"),
    Race_Other = as.numeric(Race1 == "Other"),
    HomeOwn_Rent = as.numeric(HomeOwn == "Rent"),
    HomeOwn_Other = as.numeric(HomeOwn == "Other"),
    Diabetes_Yes = as.numeric(Diabetes == "Yes"),
    PhysActive_Yes = as.numeric(PhysActive == "Yes")
  ) |>
  select(-Gender, -Race1, -HomeOwn, -Diabetes, -PhysActive)

# Fit PCR model with all components for scree plot analysis
pcr_full <- pls::pcr(BMI ~ ., data = pcr_data, scale = TRUE, validation = "CV
# validationplot(pcr_full, val.type = "MSEP", main = "Cross-Validation Error",
# Extract variance explained
var_explained <- explvar(pcr_full)
cumvar_explained <- cumsum(var_explained)

# 85% of the variance explained is the threshold I want to use
n_comp <- length(cumvar_explained[which(cumvar_explained <= 85)])

# Fit final PCR model with optimal number of components
pcr_final <- pcr(BMI ~ ., data = pcr_data, ncomp = n_comp, scale = TRUE)

# Model summary
summary(pcr_final)

# Convert the 3D array slice into a tidy tibble
coef_original <- coef(pcr_final, ncomp = n_comp, intercept = TRUE)
coef_values <- coef_original[, , "12 comps"]
variable_names <- dimnames(coef_original)[[1]]

# Build tibble with names
coef_tbl <- tibble(
  Variable = variable_names,
  Coefficient = as.vector(coef_values)
)

# Create kable table
kable(coef_tbl, digits = 3, caption = "Principal Component Regression Coeffici
pcr_test_data <- indiv_test |>
  select(-ID) |>
  mutate(
    Gender_male = as.numeric(Gender == "male"),
```

```
    Race_Black = as.numeric(Race1 == "Black"),
    Race_Hispanic = as.numeric(Race1 == "Hispanic"),
    Race_Mexican = as.numeric(Race1 == "Mexican"),
    Race_Other = as.numeric(Race1 == "Other"),
    HomeOwn_Rent = as.numeric(HomeOwn == "Rent"),
    HomeOwn_Other = as.numeric(HomeOwn == "Other"),
    Diabetes_Yes = as.numeric(Diabetes == "Yes"),
    PhysActive_Yes = as.numeric(PhysActive == "Yes")
  ) |>
  select(-Gender, -Race1, -HomeOwn, -Diabetes, -PhysActive)
# Predict on test set
test_predictions <- predict(
  pcr_final,
  newdata = pcr_test_data,
  ncomp = n_comp
)
(pcr_test_mse <- mean((pcr_test_data$BMI - test_predictions)2))



# Problem 9 ----------------------------------------------------------
pls_fit <- plsr(BMI ~ ., data = pcr_data, validation = "CV")

# Plot cross-validation results
# validationplot(pls_fit, val.type = "MSEP", main = "Validation Plot for PLS")

# Extract optimal number of components (minimum CV error)
cv_results <- pls_fit$validation$PRESS
optimal_comps <- which.min(cv_results)

# Refit with optimal components
final_pls <- plsr(BMI ~ ., data = pcr_data, ncomp = optimal_comps)

# Model summary
# summary(final_pls)

# Predictions and R²
pls_pred <- predict(final_pls, newdata = pcr_test_data, ncomp = optimal_comps)
# Convert the 3D array slice into a tidy tibble
coef_original_pls <- coef(final_pls, ncomp = optimal_comps, intercept = TRUE)
coef_values_pls <- coef_original_pls[, , "15 comps"]
# variable_names <- dimnames(coef_original_pls)[[1]]

# Build tibble with names
coef_tbl_pls <- tibble(
  Variable = variable_names,
  Coefficient = as.vector(coef_values_pls)
```

```
)

# Create kable table
kable(coef_tbl_pls, digits = 3, caption = "Partial Least Squares Coefficients
pls_test_mse <- mean((pcr_test_data$BMI - pls_pred)2)
pls_test_mse
```