# Predicting County-Level Heart Disease Mortality in the United States

*Charlotte Abrams, Laura Cosgrove, Alyssa Vanderbeek*

*7 April 2019*

## Introduction

### Background

Heart disease remains one of the leading causes of death in adults in the US. Understanding risk factors for diseases of this kind is an important task in working towards reducing the number of lives lost. There is obvious benefit in doing this at an individual level, examining personal lifestyle, genetic profile, and family history. But there may be important environmental components that have predictive value in assessing risk for heart disease. We examine economic, health, and demographic data for thousands of counties across the US. This data is synthesized from several sources, including the USDA Economic Research Service, Bureau of Labor Statistics, US Census, Behavioral Risk Factor Surveillance System, the CDC, and others. Our goal in this project is to most effectively predict the county-level heart disease mortality rate per 100,000 persons. We build 6 predictive models (stepwise linear regression, Lasso, Ridge, PCR, GAM, and MARS), and compare them on their predictive capacity quantified by the root mean squared error (RMSE).

### Exploratory Data Analysis (Laura)

## Data Methods

**note: transform test data by centering and scaling with TRAINING means and standard deviations. Does predict do this automatically when using `preProcess` in caret? – done

can consider imputation: say 10%, one possible solution. use the training set to build. Look at the preProcess function, can use knn. – done. Still dropped two columns because they had 90% missing data; imputation not reliable

Check/detect near-zero variance predictors: to decrease computational time and complexity. – done. A few of the categories for urban influence came back as having near-zero variance. These were examined as dummy variables; 2000 rows divided across 12 levels means that these are likely to be "rare". We determined that we ought to leave them in the model since they are part of the larger variable "urban_influence".

## Linear Models

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   metro = col_character(),
##   urban_influence = col_character(),
##   economic_typology = col_character(),
##   pure_population = col_character(),
##   metro_adjacency = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

### Stepwise Selection (Alyssa)

We first fit a stepwise linear regression model.

### Lasso (Charlotte)

### Ridge (Laura)

### PCR (Charlotte)

## Nonlinear Models

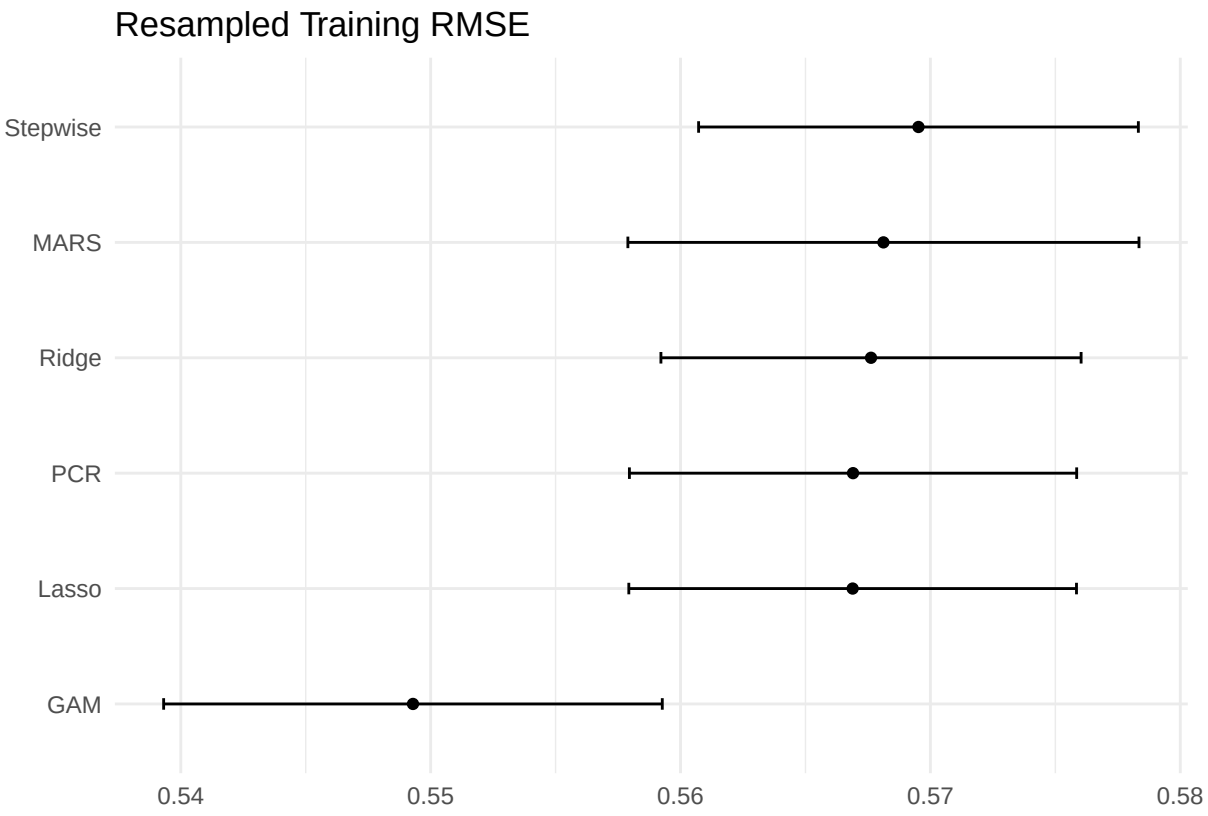
### GAM (Alyssa)

### MARS (Laura)

Note that we chose the most parsimonious model.

## Model Comparison

### Training RMSE




The more flexible model

**Test RMSE (Alyssa)**

Table 1: Predictive RMSE on training and test data

| | Train RMSE | Test RMSE |
|---|---|---|
| Stepwise | 0.5695231 | 0.5774001 |
| Ridge | 0.5676237 | 0.5799894 |
| Lasso | 0.5668887 | 0.5772404 |
| PCR | 0.5669038 | 0.5777914 |
| GAM | 0.5492921 | 0.5493805 |
| MARS | 0.5681207 | 0.5469220 |

Table 1 presents the RMSE of the predicted heart disease mortality rate for all models on both the training and testing datasets. Although GAM appeared to have the lowest RMSE when using the training data, MARS has a slight advantage (a difference of 0.003) on the test data.

**Interpretations (Laura)**

*Coefficient Shrinkage: Lasso and Ridge*

*Investigation of MARS*

The minimum generalized cross-validation error was achieved

*Investigation of GAM*

## Discussion

## Appendix

### Data prep

```
predictors <- read_csv("./data/Training_values.csv")
response <- read_csv("./data/Training_labels.csv")


## Manipulation
data <- response %>%
  full_join(predictors, by = "row_id") %>%
  separate(col = area__rucc, into = c('metro', 'population'), sep = ' - ') %>%
  rename(urban_influence = area__urban_influence,
         economic_typology = econ__economic_typology) %>%
  mutate(pure_population = fct_collapse(as.factor(population),
                                        "more_than_1mil" = "Counties in metro areas of 1 million popula
                                        "250k_to_1mil" = "Counties in metro areas of 250,000 to 1 milli
                                        "less_than_250k" = "Counties in metro areas of fewer than 250,0
                                        "more_than_20k" = c("Urban population of 20,000 or more, adjace
                                                            "Urban population of 20,000 or more, not
                                        "2500_to_20k" = c("Urban population of 2,500 to 19,999, adjacen
                                                          "Urban population of 2,500 to 19,999, not a
                                        "less_than_2500" = c("Completely rural or less than 2,500 urban
                                                             "Completely rural or less than 2,500 urban popul
         economic_typology = as.factor(recode(economic_typology,
                                              "Nonspecialized" = "Nonspecialized",
                                              "Manufacturing-dependent" = "Manufacturing",
                                              "Farm-dependent" = "Farming",
                                              "Federal/State government-dependent" = "Government",
                                              "Mining-dependent" = "Mining",
                                              "Recreation" = "Recreation")),
         metro = factor(metro,
                        levels = c("Metro", "Nonmetro")),
         urban_influence = str_replace_all(urban_influence, " |/|-", "_"), # replace problematic charact
         urban_influence = str_replace_all(urban_influence, ",", ""), # replace problematic characters
         demo__pct_nonwhite = demo__pct_hispanic + demo__pct_asian + demo__pct_american_indian_or_alaskan
         urban_influence = fct_rev(urban_influence),
         metro_adjacency = fct_collapse(population,
                                        metro = c("Counties in metro areas of 1 million population or mo
                                                  "Counties in metro areas of 250,000 to 1 million popula
                                                  "Counties in metro areas of fewer than 250,000 populati
                                        adjacent = c("Urban population of 20,000 or more, adjacent to a
                                                     "Urban population of 2,500 to 19,999, adjacent to a
                                                     "Completely rural or less than 2,500 urban populatio
                                        nonadjacent = c("Urban population of 20,000 or more, not adjacen
                                                        "Urban population of 2,500 to 19,999, not adjacen
                                                        "Completely rural or less than 2,500 urban popula
  dplyr::select(-demo__pct_hispanic,
                -demo__pct_asian,
                -demo__pct_american_indian_or_alaskan_native,
                -demo__pct_non_hispanic_african_american,
                -demo__pct_non_hispanic_white,
                -health__homicides_per_100k, # >90% missing
```

```
                -health__pct_excessive_drinking, # >90% missing
                -yr,
                -population)
```

Training and testing data split. Imputation on missing data.

```
## training/test data
set.seed(1)
train_ind <- sample(seq_len(nrow(data)), size = 2/3*nrow(data)) # select rows in 2:1 ratio

train <- data[train_ind, ] # training dataset
test <- data[-train_ind, ] # testing dataset

# Imputation for missing values with caret, based on training data
training_preproc = caret::preProcess(train,
                                     method = "knnImpute", # automatically centers and scales data
                                     pcaComp = 10,
                                     na.remove = TRUE,
                                     k = 5,
                                     knnSummary = mean,
                                     outcome = NULL,
                                     fudge = .2,
                                     numUnique = 3,
                                     verbose = TRUE)

# Impute training imputation on both training and testing datasets
train_imputed = predict(training_preproc, train)
test_imputed = predict(training_preproc, test)

x <- model.matrix(heart_disease_mortality_per_100k ~ ., data = heart)[,-1]
y <- heart$heart_disease_mortality_per_100k
```

*Linear Models*

Stepwise regression:

```
ctrl = trainControl(method = "repeatedcv", number = 10, repeats = 5)
set.seed(2)
step.fit = caret::train(x, y,
                        method = 'glmStepAIC',
                        metric = "RMSE",
                        trControl = ctrl)
```

Lasso:

```
set.seed(2)
lasso_fit <- caret::train(x, y,
                          method = "glmnet",
                          metric = "RMSE",
                          tuneGrid = expand.grid(alpha = 1,
                                                 lambda = exp(seq(-10, 0, length = 200))),
                          preProcess = c("zv"),
                          trControl = ctrl)
```

Ridge:

PCR:

```
set.seed(2)
pcr_fit <- caret::train(x, y,
                        method = "pcr",
                        trControl = ctrl,
                        metric = "RMSE",
                        tuneLength = 200)
```

*Non-linear models*

GAM:

```
set.seed(2)
gam_fit <- caret::train(x, y,
                        method = "gam",
                        metric = 'RMSE',
                        tuneGrid = data.frame(method = "GCV.Cp", select = c(TRUE, FALSE)),
                        trControl = ctrl)
summary(gam_fit)
saveRDS(gam_fit, "gam_fit_imputed.rds")
```

MARS:

```
mars_grid <- expand.grid(degree = 1:3, # degree: 1 vs 2 vs 3, no interaction vs. interaction;
                         nprune = 24:40) # nprune is number of coef

set.seed(2)

mars_fit <- caret::train(x, y,
                         method = "earth",
                         tuneGrid = mars_grid,
                         trControl = ctrl)

#based on initial results, choose parsimonious version
mars_grid_refined <- expand.grid(degree = 2, nprune = 25:30)

set.seed(2)

mars_fit_refined <- caret::train(x, y,
                                 method = "earth",
                                 tuneGrid = mars_grid_refined,
                                 trControl = ctrl)
```