Arjun Narang Final Project Modeling

Arjun Narang.37

2025-11-19

Load Packages and Data

```
# Packages
if (!require("splines")) {
  install.packages("splines")
  library(splines)
}
```

```
## Loading required package: splines
```

```
if (!require("ISLR2")) {
  install.packages("ISLR2")
  library(ISLR2)
}
```

```
## Loading required package: ISLR2

## Warning: package 'ISLR2' was built under R version 4.4.3
```

```
if (!require("dplyr")) {
  install.packages("dplyr")
  library(dplyr)
}
```

```
## Loading required package: dplyr

## Warning: package 'dplyr' was built under R version 4.4.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
if (!require("corrplot")) {
  install.packages("corrplot")
  library(corrplot)
}
```

```
## Loading required package: corrplot
```

```
## Warning: package 'corrplot' was built under R version 4.4.3
```

```
## corrplot 0.95 loaded
```

```r
if (!require("ggplot2")) {
  install.packages("ggplot2")
  library(ggplot2)
}
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```r
if (!require("patchwork")) {
  install.packages("patchwork")
  library(patchwork)
}
```

```
## Loading required package: patchwork
```

```r
if (!require("MASS")) {
  install.packages("MASS")
  library(MASS)
}
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:patchwork':
##
##     area
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```r
if (!require("tibble")) {
  install.packages("tibble")
  library(tibble)
}
```

```
## Loading required package: tibble
```

```r
if (!require("boot")) {
  install.packages("boot")
  library(boot)
}
```

```
## Loading required package: boot
```

```r
if (!require("mgcv")) {
  install.packages("mgcv")
  library(mgcv)
}
```

```
## Loading required package: mgcv

## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```r
if (!require("car")) {
  install.packages("car")
  library(car)
}
```

```
## Loading required package: car

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:boot':
##
##     logit

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
if (!require("tree")) {
  install.packages("tree")
  library(tree)
}
```

```
## Loading required package: tree
```

```
## Warning: package 'tree' was built under R version 4.4.3
```

```r
if (!require("randomForest")) {  # for bagging + RF
  install.packages("randomForest")
  library(randomForest)
}
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 4.4.3
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
if (!require("gbm")) {           # for boosting
  install.packages("gbm")
  library(gbm)
}
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 4.4.3
```

```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

```r
load("Wage_Stat4620_2023.RData")
data <- Wage_Stat4620
```

Clean Data

```r
# Get rid of all the N/A response variables
data_clean <- data %>% filter(!is.na(Resp))
data <- data_clean
dim(data)
```

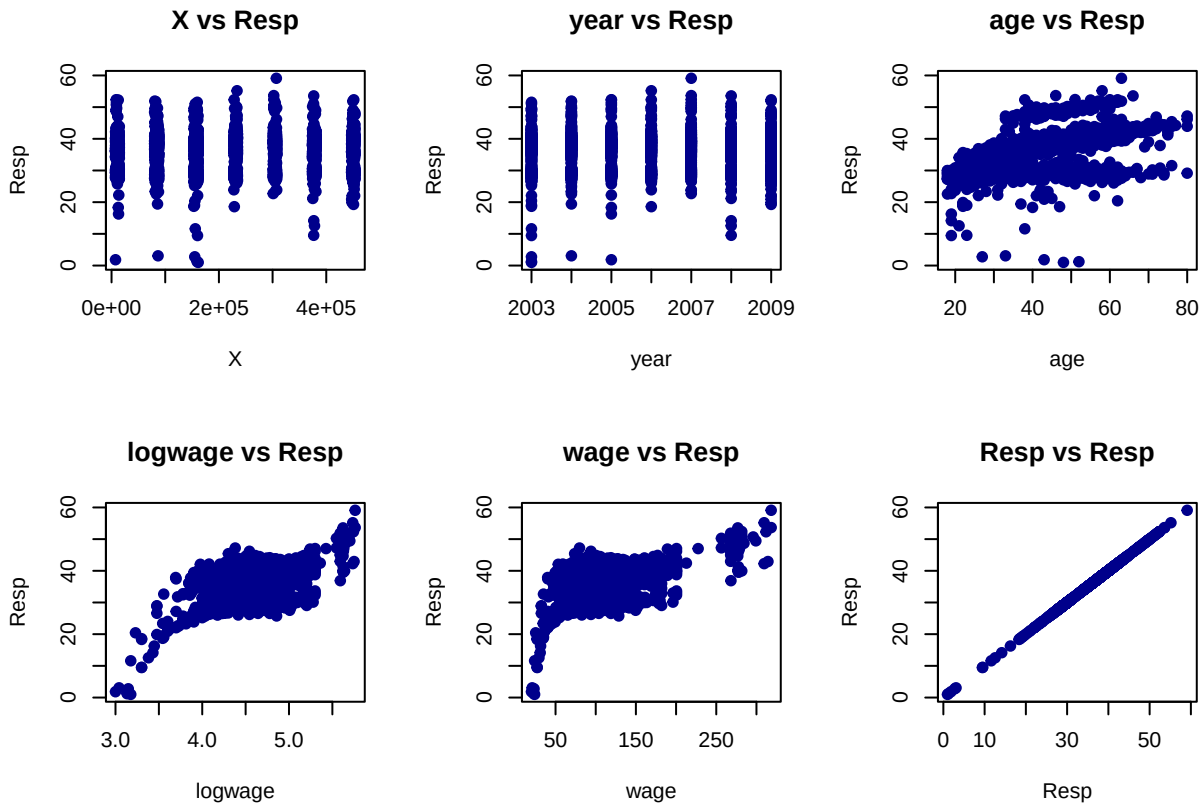
```
## [1] 2940   13
```

```r
# Change marital status to married vs not married
data$maritl <- ifelse(data$maritl == "2. Married",
                      "Married",
                      "Not Married")

data$maritl <- factor(data$maritl,
                      levels = c("Not Married", "Married"))
```

We aggregate some of the values for maritl since we see a difference in married vs all the other values in the response variable. This will be observed below.

EDA Views

```r
numeric_vars <- names(data)[sapply(data, is.numeric)]
par(mfrow=c(2,3))
for(v in numeric_vars){
  plot(data[[v]], data$Resp, pch=19, col="darkblue",
       main=paste(v, "vs Resp"), xlab=v, ylab="Resp")
}
```



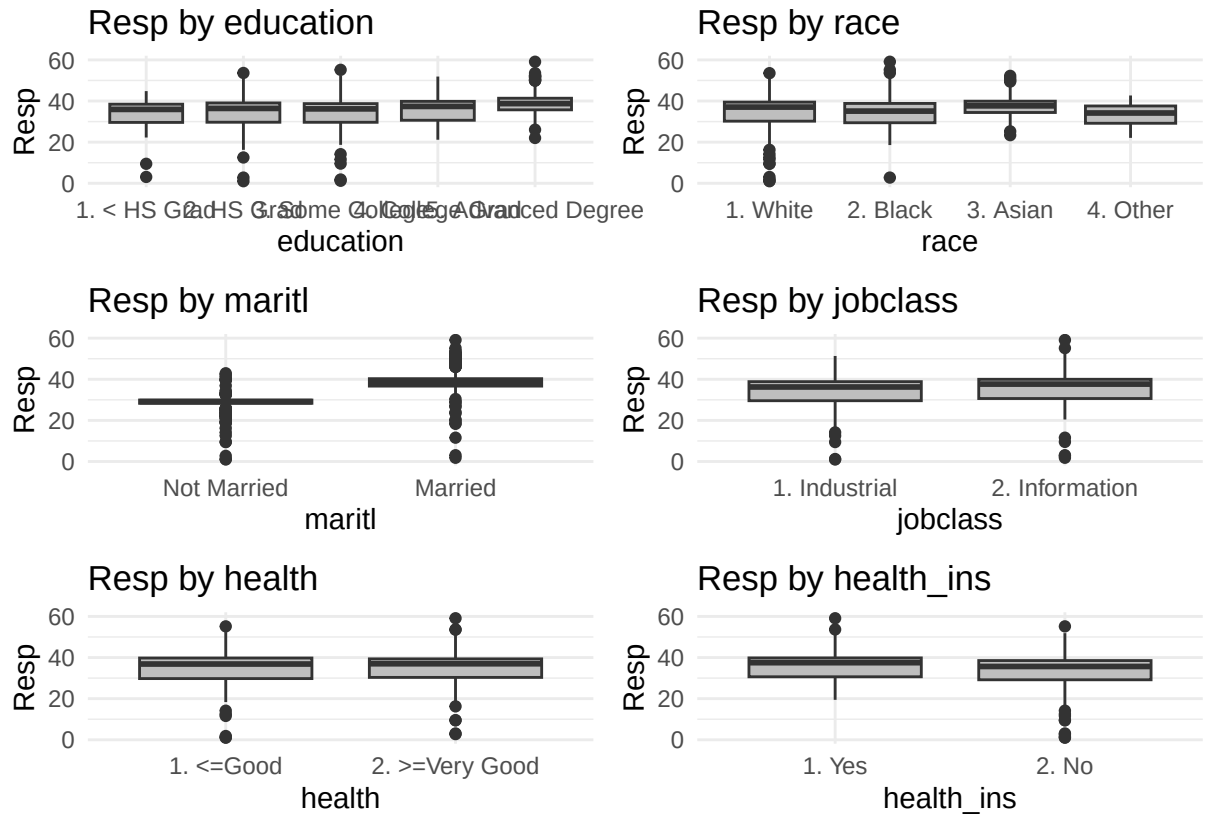

```
par(mfrow=c(1,1))
```

With some preliminary EDA, we can see that the response variable has a bi-modal distribution. Additionally it is slightly skewed to the right. This lack of normality in the distribution would violate most linear models. This means our modeling will be focusing in on models that allow interaction effects like polynomial model, GAMs, and possibly trees.

```
# Categorical Predictors vs Response
cat_vars <- c("education", "race", "maritl", "jobclass", "health", "health_ins")

plot_list <- lapply(cat_vars, function(v) {
  ggplot(data, aes_string(x = v, y = "Resp")) +
    geom_boxplot(fill = "grey") +
    labs(title = paste("Resp by", v), x = v, y = "Resp") +
    theme_minimal()
})
```

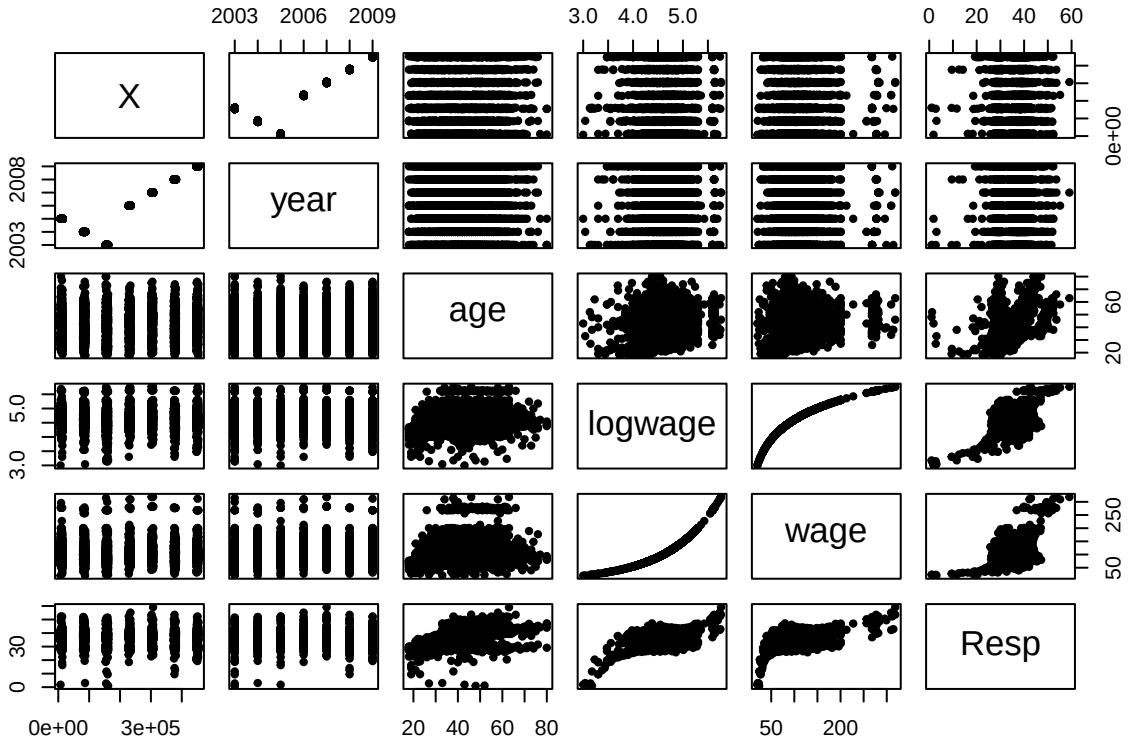
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
combined_plot <- plot_list[[1]] + plot_list[[2]] + plot_list[[3]] +
                 plot_list[[4]] + plot_list[[5]] + plot_list[[6]] +
                 plot_layout(nrow = 3, ncol = 2)
combined_plot
```



It seems that the response does vary a bit amongst education, race, and marital status. These will likely be useful in helping us model the response variable.

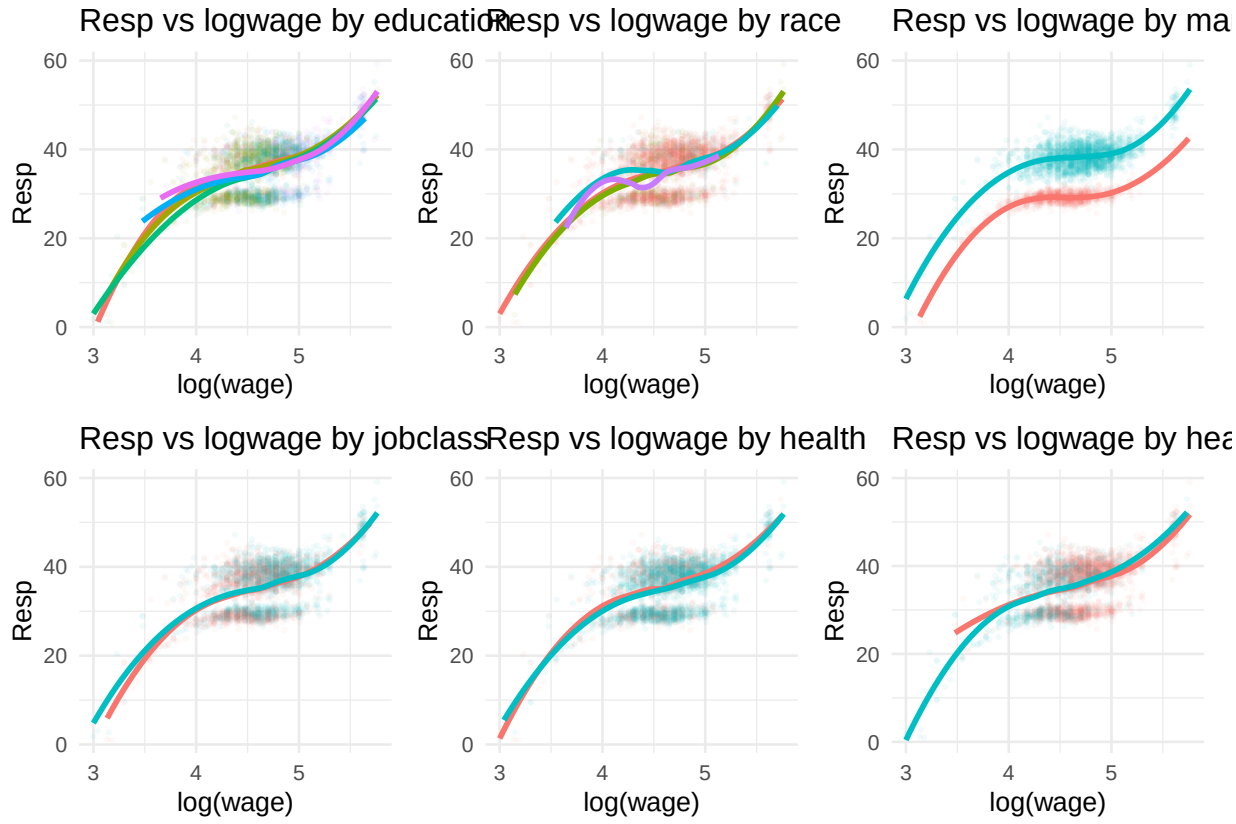
```
pairs(data[, numeric_vars], pch=20)
```



Based off of the this we see that logwage and and age seem to be good predictors let's try to break them down further.

```
plot_cat <- function(var) {
  ggplot(data, aes(x = logwage, y = Resp, color = .data[[var]])) +
    geom_point(alpha = 0.05, size = 0.4) +
    geom_smooth(method = "loess", se = FALSE, linewidth = 1) +
    labs(title = paste("Resp vs logwage by", var),
         x = "log(wage)", y = "Resp") +
    theme_minimal(base_size = 10) +
    theme(legend.position = "none")  # hides legends to shrink figure
}

p1 <- plot_cat("education")
p2 <- plot_cat("race")
p3 <- plot_cat("maritl")
p4 <- plot_cat("jobclass")
p5 <- plot_cat("health")
p6 <- plot_cat("health_ins")

# Combine into a 3x2 grid
final_fig <- (p1 | p2 | p3) /
             (p4 | p5 | p6)

final_fig
```

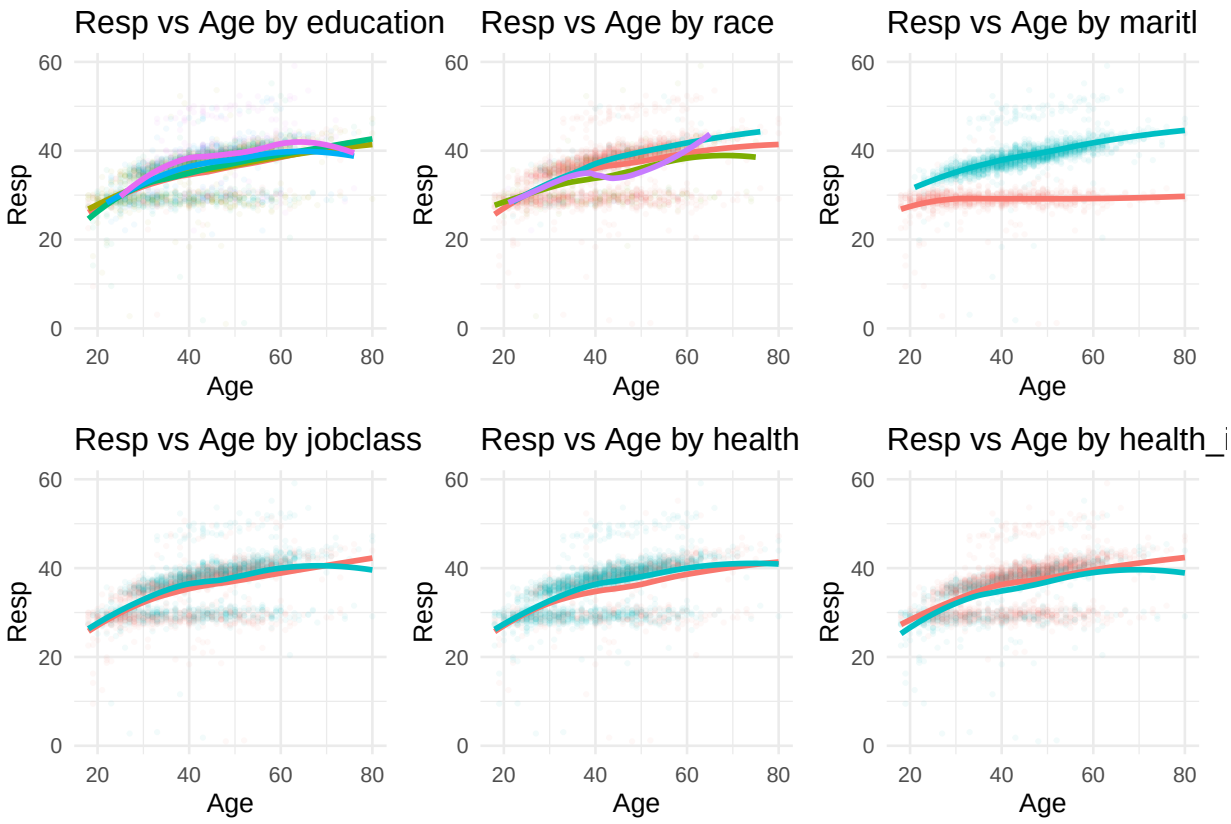
```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```




```r
# Create each plot
plot_age_cat <- function(var) {
  ggplot(data, aes(x = age, y = Resp, color = .data[[var]])) +
    geom_point(alpha = 0.05, size = 0.4) +
    geom_smooth(method = "loess", se = FALSE, linewidth = 1) +
    labs(title = paste("Resp vs Age by", var),
         x = "Age", y = "Resp") +
    theme_minimal(base_size = 10) +
    theme(legend.position = "none")
}

# Create each individual plot
p1 <- plot_age_cat("education")
p2 <- plot_age_cat("race")
p3 <- plot_age_cat("maritl")
p4 <- plot_age_cat("jobclass")
p5 <- plot_age_cat("health")
p6 <- plot_age_cat("health_ins")
```

```
# Combine into 3x2 patchwork layout
age_fig <- (p1 | p2 | p3) /
           (p4 | p5 | p6)

age_fig
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



Out of all the categorical variables, marital status seems to be the only clear interaction with logWage and age. When modeling we will definitely include that as an interaction and perhaps include education and race as a main effect and see how it performs with the test MSE.

Test/Train Split

```
set.seed(4620)

n <- nrow(data)
train_size <- round(n * 0.8) # Create an 80/20 split
```

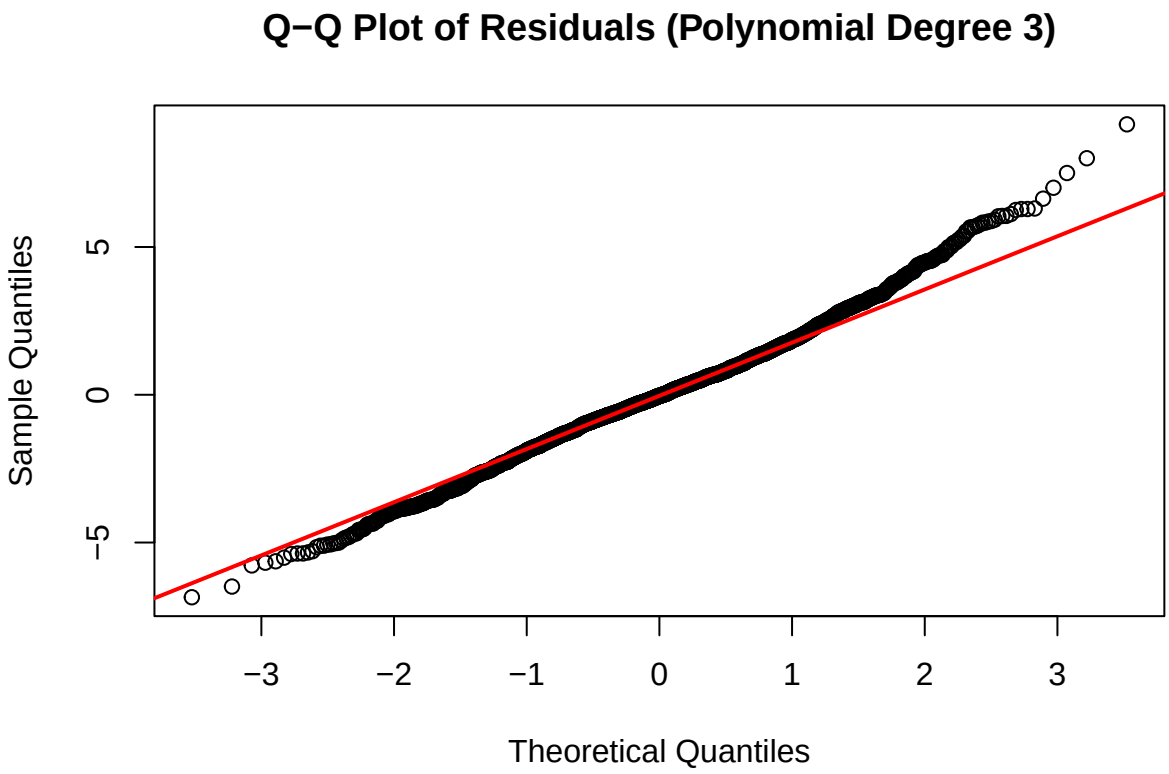
```r
train <- sample(1:n,train_size, replace = FALSE)
test <- -train

Wage.train <- data[train, ]
Wage.test  <- data[test, ]
```

Modeling logWage against Response

```r
# Fit 3rd degree polynomial using basic, interaction, and GAM
fit_poly_3 <- lm(Resp ~ poly(logwage, 3), data = Wage.train)
fit_poly3_int <- lm(Resp ~ poly(logwage, 3) * maritl, data = Wage.train)
fit_gam_3 <- gam(Resp ~ s(logwage, by = maritl) + maritl, data = Wage.train)

summary(fit_poly3_int)
```

```
##
## Call:
## lm(formula = Resp ~ poly(logwage, 3) * maritl, data = Wage.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.8513 -1.2495 -0.0392  1.1782  9.1501
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      29.41863    0.08402 350.157   <2e-16 ***
## poly(logwage, 3)1                94.75084    4.49044  21.101   <2e-16 ***
## poly(logwage, 3)2               -11.09254    4.36200  -2.543   0.0111 *
## poly(logwage, 3)3                84.95076    3.98302  21.328   <2e-16 ***
## maritlMarried                     8.97307    0.09856  91.043   <2e-16 ***
## poly(logwage, 3)1:maritlMarried   3.67814    5.18162   0.710   0.4779
## poly(logwage, 3)2:maritlMarried   3.55129    5.06834   0.701   0.4836
## poly(logwage, 3)3:maritlMarried  -0.63384    4.70719  -0.135   0.8929
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.027 on 2344 degrees of freedom
## Multiple R-squared:  0.8706, Adjusted R-squared:  0.8702
## F-statistic:  2252 on 7 and 2344 DF,  p-value: < 2.2e-16
```

```r
res <- residuals(fit_poly3_int)

qqnorm(res, main = "Q-Q Plot of Residuals (Polynomial Degree 3)")
qqline(res, col = "red", lwd = 2)
```



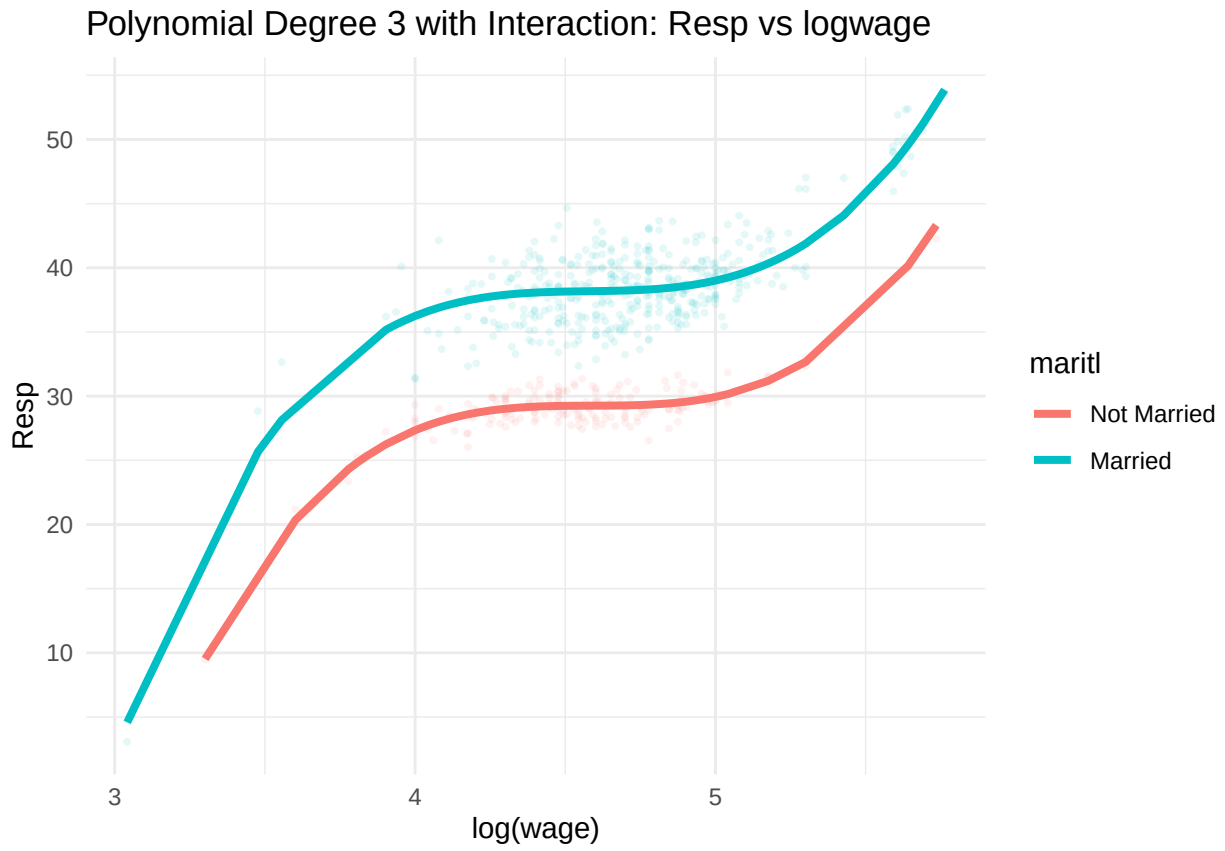
We see the response being modeled quite well by the ploynomial, a bit of variability towards the ends of the plot, but we can work to imrpove that as we continue to evaluate variables.

```
# Degree 3 predictions + MSE
pred3 <- predict(fit_poly_3, newdata = data[test, ])
pred3_int <- predict(fit_poly3_int, newdata = data[test, ])
pred_gam_3 <- predict(fit_gam_3, newdata = data[test, ])

# Calculate the MSE prayer
mse3  <- mean((pred3 - data[test, ]$Resp)^2)
mse3_int <- mean((pred3_int - data[test, ]$Resp)^2)
mse3_gam <- mean((pred_gam_3 - data[test, ]$Resp)^2)

mse_results <- data.frame(
  Model = c("Polynomial (Degree 3)", "Polynomial (Degree 3) with Marriage Interaction", " GAM"),
  Test_MSE = c(mse3, mse3_int, mse3_gam)
)

mse_results
```

```
##                                          Model  Test_MSE
## 1                        Polynomial (Degree 3) 19.075852
## 2 Polynomial (Degree 3) with Marriage Interaction  4.043070
## 3                                          GAM  4.068914
```

We see that polynomial (degree 3) with interaction effects has the lowest test MSE.

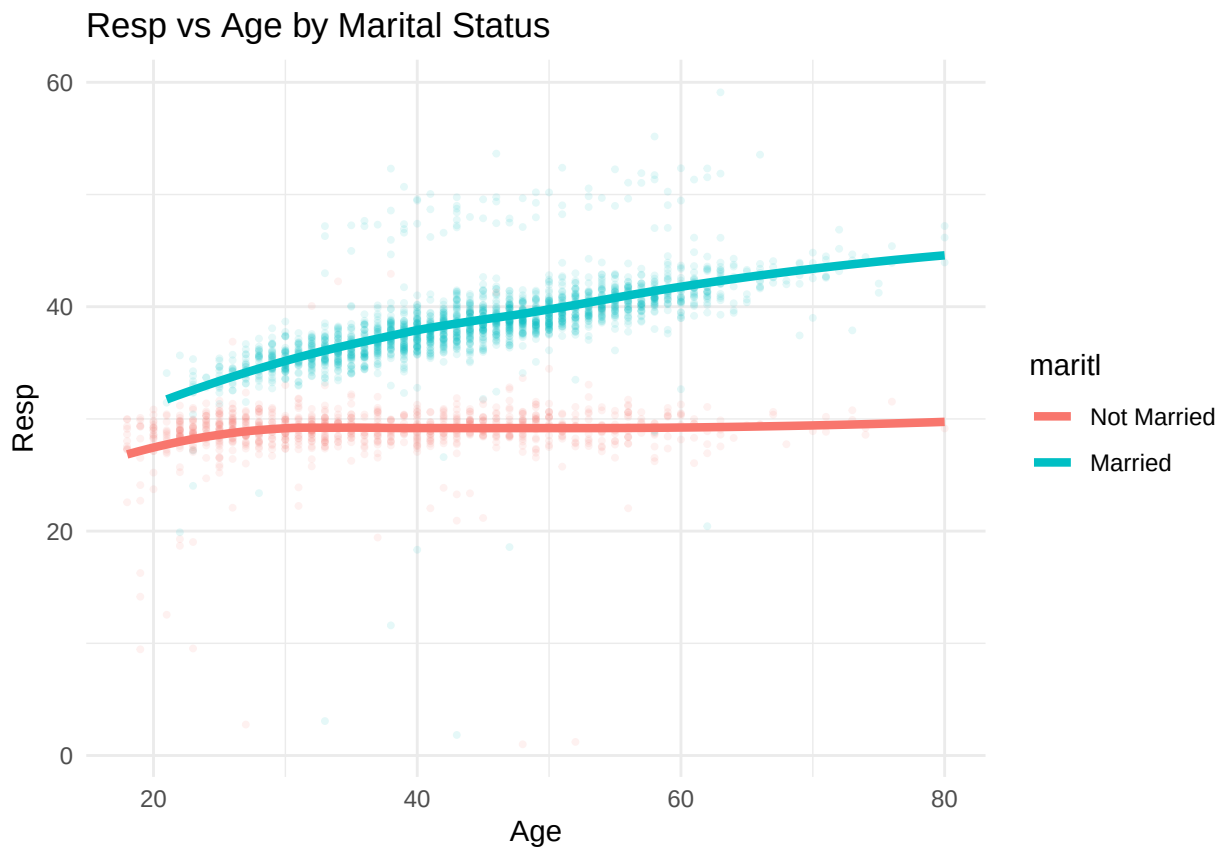
```r
# Add predictions to test set
Wage.test$pred_poly3_int <- pred3_int

ggplot(Wage.test, aes(x = logwage, y = Resp, color = maritl)) +
  geom_point(alpha = 0.1, size = 0.7) +
  geom_line(aes(y = pred_poly3_int),
            linewidth = 1.4) +
  labs(title = "Polynomial Degree 3 with Interaction: Resp vs logwage",
       x = "log(wage)", y = "Resp") +
  theme_minimal()
```



Modeling Age vs Resp

```r
ggplot(data, aes(x = age, y = Resp, color = maritl)) +
  geom_point(alpha = 0.1, size = 0.7) +
  geom_smooth(method = "loess", se = FALSE, linewidth = 1.5) +
  labs(title = "Resp vs Age by Marital Status",
       x = "Age", y = "Resp") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

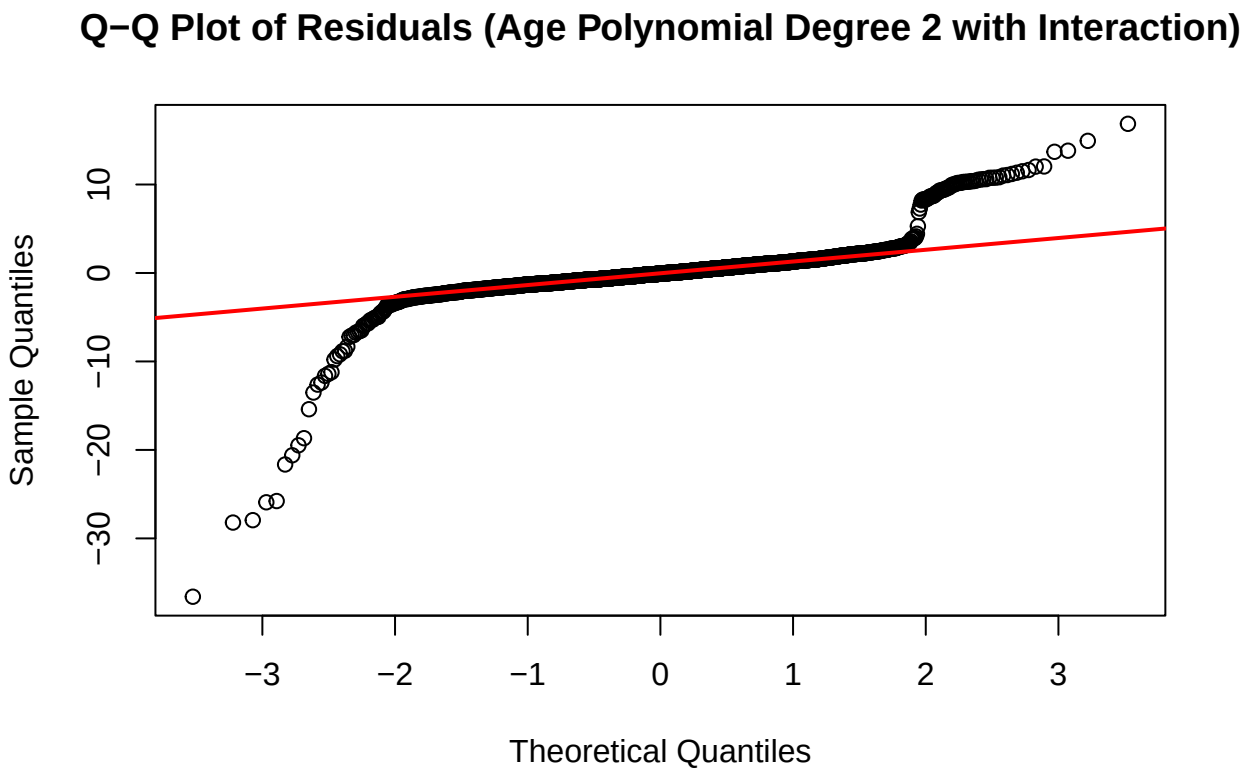



```
# Fit 2nd-degree polynomial using basic, interaction, and GAM for age
fit_age_poly2     <- lm(Resp ~ poly(age, 2), data = Wage.train)
fit_age_poly2_int <- lm(Resp ~ poly(age, 2) * maritl, data = Wage.train)
fit_age_gam       <- gam(Resp ~ s(age, by = maritl) + maritl, data = Wage.train)

summary(fit_age_poly2_int)
```

```
##
## Call:
## lm(formula = Resp ~ poly(age, 2) * maritl, data = Wage.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.573  -0.936  -0.072   0.854  16.944
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    29.0671     0.1145 253.899  < 2e-16 ***
## poly(age, 2)1                   8.8495     5.2221   1.695  0.09028 .
## poly(age, 2)2                 -14.4792     4.9586  -2.920  0.00353 **
## maritlMarried                   8.9739     0.1365  65.756  < 2e-16 ***
## poly(age, 2)1:maritlMarried   115.1084     6.6468  17.318  < 2e-16 ***
## poly(age, 2)2:maritlMarried     0.6592     6.3978   0.103  0.91795
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 2.789 on 2346 degrees of freedom
## Multiple R-squared:  0.7548, Adjusted R-squared:  0.7542
## F-statistic:  1444 on 5 and 2346 DF,  p-value: < 2.2e-16
```

```r
res_age2 <- residuals(fit_age_gam)

qqnorm(res_age2, main = "Q-Q Plot of Residuals (Age Polynomial Degree 2 with Interaction)")
qqline(res_age2, col = "red", lwd = 2)
```



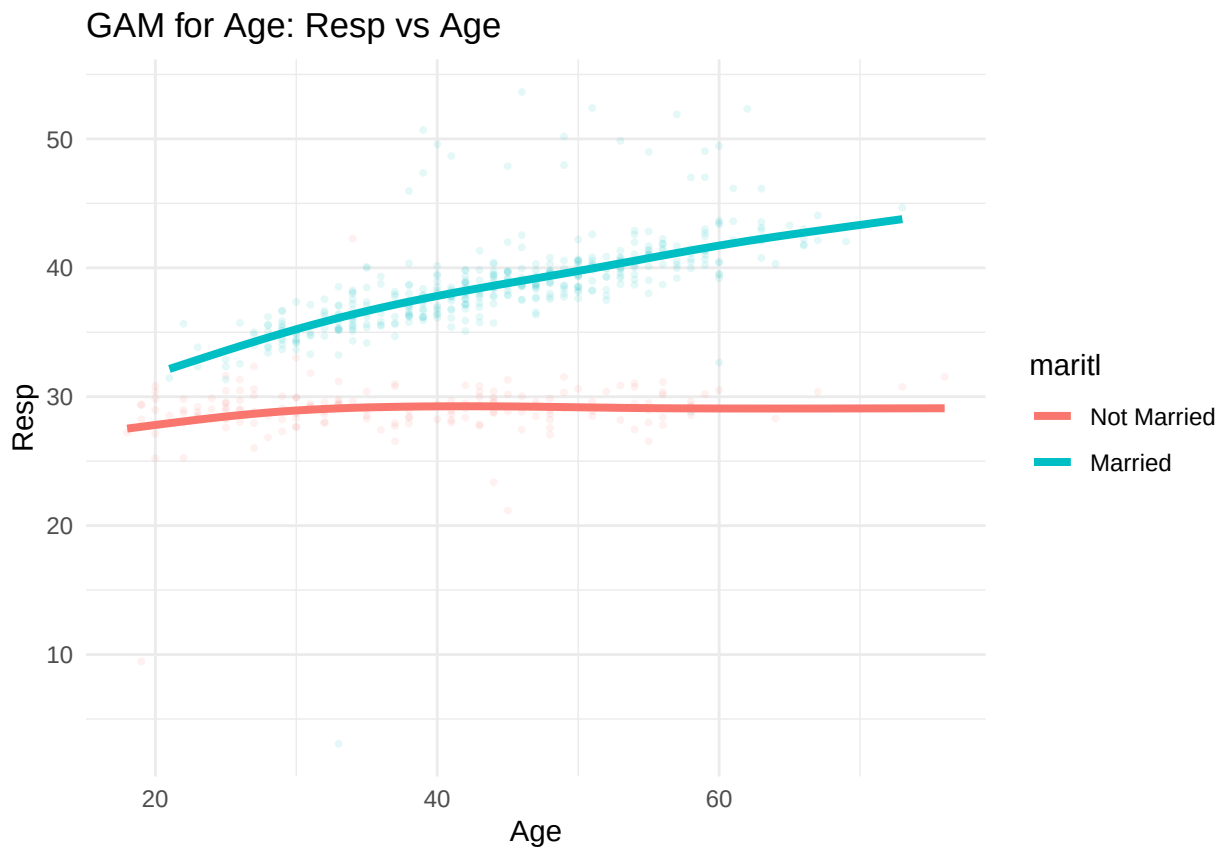
We see the ends of the Q-Q plot being modeled very poorly. This is likely due to the upper elements in Age around 40-50 being underestimated.

```r
# Predictions + MSE for age-based models
pred_age2      <- predict(fit_age_poly2,     newdata = Wage.test)
pred_age2_int  <- predict(fit_age_poly2_int, newdata = Wage.test)
pred_age_gam   <- predict(fit_age_gam,       newdata = Wage.test)

mse_age2      <- mean((pred_age2     - Wage.test$Resp)^2)
mse_age2_int  <- mean((pred_age2_int - Wage.test$Resp)^2)
mse_age2_gam  <- mean((pred_age_gam  - Wage.test$Resp)^2)

mse_age_results <- data.frame(
  Model    = c("Age Polynomial (Degree 2)",
               "Age Polynomial (Degree 2) with Marriage Interaction",
               "Age GAM"),
  Test_MSE = c(mse_age2, mse_age2_int, mse_age2_gam)
```

```
)

mse_age_results
```

```
##                                                 Model  Test_MSE
## 1                           Age Polynomial (Degree 2) 23.124524
## 2 Age Polynomial (Degree 2) with Marriage Interaction  8.068780
## 3                                             Age GAM  8.035405
```

```
Wage.test$pred_age_gam <- pred_age_gam

ggplot(Wage.test, aes(x = age, y = Resp, color = maritl)) +
  geom_point(alpha = 0.1, size = 0.7) +
  geom_line(aes(y = pred_age_gam), linewidth = 1.4) +
  labs(title = "GAM for Age: Resp vs Age",
       x = "Age", y = "Resp") +
  theme_minimal()
```



Final GAM Model

Now that we modeled both logWage and age, we must combine them. The best strategy for this would be to use a GAM that allows for different variables/interactions to explain parts of the variability.

```r
# BASE GAM (no age interaction)
gam_base <- gam(
  Resp ~
    s(logwage, by = maritl) +
    maritl +
    s(age),
  data = Wage.train
)

# BASE GAM (age interaction)
gam_full_base <- gam(
  Resp ~
    s(logwage, by = maritl) +
    maritl +
    s(age, by = maritl),
  data = Wage.train
)

# BASE GAM (no Maritl)
gam_full_base_test <- gam(
  Resp ~
    s(logwage, by = maritl) +
    s(age, by = maritl),
  data = Wage.train
)

# Add education
gam_full_edu_ageint <- gam(
  Resp ~ s(logwage, by = maritl) + maritl +
    s(age, by = maritl) +
    education,
  data = Wage.train
)

# Add race
gam_full_race_ageint <- gam(
  Resp ~ s(logwage, by = maritl) + maritl +
    s(age, by = maritl) +
    race,
  data = Wage.train
)

# Add jobclass
gam_full_job_ageint <- gam(
  Resp ~ s(logwage, by = maritl) + maritl +
    s(age, by = maritl) +
    jobclass,
  data = Wage.train
)

# Add education + race
gam_full_edu_race_ageint <- gam(
  Resp ~ s(logwage, by = maritl) + maritl +
```

```
    s(age, by = maritl) +
    education + race,
  data = Wage.train
)

# Add education + jobclass
gam_full_edu_job_ageint <- gam(
  Resp ~ s(logwage, by = maritl) + maritl +
    s(age, by = maritl) +
    education + jobclass,
  data = Wage.train
)

# Add race + jobclass
gam_full_race_job_ageint <- gam(
  Resp ~ s(logwage, by = maritl) + maritl +
    s(age, by = maritl) +
    race + jobclass,
  data = Wage.train
)

# FULL GAM with age interaction + all categoricals
gam_full_age_interaction <- gam(
  Resp ~ s(logwage, by = maritl) + maritl +
    s(age, by = maritl) +
    education + race + jobclass,
  data = Wage.train
)
```

```
# Put all GAMs in a named list (with age interaction)
gam_age_int_models <- list(
  "GAM base (no age interaction)" = gam_base,
  "GAM base" = gam_full_base,
  "GAM base (no maritl)" = gam_full_base_test,
  "GAM + education"  = gam_full_edu_ageint,
  "GAM + race"       = gam_full_race_ageint,
  "GAM + jobclass"   = gam_full_job_ageint,
  "GAM + edu + race" = gam_full_edu_race_ageint,
  "GAM + edu + job"  = gam_full_edu_job_ageint,
  "GAM + race + job" = gam_full_race_job_ageint,
  "GAM FULL"           = gam_full_age_interaction
)

# Compute MSE
gam_age_int_mse <- sapply(gam_age_int_models, function(mod) {
  pred <- predict(mod, newdata = Wage.test)
  mean((pred - Wage.test$Resp)^2)
})

gam_age_int_mse_df <- data.frame(
  Model    = names(gam_age_int_mse),
  Test_MSE = as.numeric(gam_age_int_mse),
  row.names = NULL
```

```
)

gam_age_int_mse_df
```

```
##                              Model   Test_MSE
## 1  GAM base (no age interaction)  1.9030994
## 2                         GAM base  1.0434639
## 3           GAM base (no maritl) 13.6248555
## 4                GAM + education  1.0437867
## 5                      GAM + race  1.0421953
## 6                 GAM + jobclass  0.9942502
## 7               GAM + edu + race  1.0437511
## 8                GAM + edu + job  0.9905290
## 9               GAM + race + job  0.9934535
## 10                       GAM FULL  0.9895433
```

We see the best Test MSE comes from using all the variables and interactions for marital status and logWage and age.

Cross Validation

Now lets use cross validation to ensure that the best model CV MSE is selected.

```
gam_cv <- function(formula, data, k = 10) {
  folds <- sample(rep(1:k, length.out = nrow(data)))
  cv_errors <- numeric(k)

  for (i in 1:k) {
    train_data <- data[folds != i, ]
    test_data  <- data[folds == i, ]

    model <- gam(formula, data = train_data)
    preds <- predict(model, newdata = test_data)
    cv_errors[i] <- mean((test_data$Resp - preds)^2)
  }

  mean(cv_errors)
}
```

```
cv_base        <- gam_cv(Resp ~ s(logwage, by = maritl) + maritl + s(age, by = maritl), data)
cv_job         <- gam_cv(Resp ~ s(logwage, by = maritl) + maritl + s(age, by = maritl) + jobclass, data
cv_edu_job     <- gam_cv(Resp ~ s(logwage, by = maritl) + maritl + s(age, by = maritl) + education + jo
cv_full        <- gam_cv(Resp ~ s(logwage, by = maritl) + maritl + s(age, by = maritl) + education + ra
```

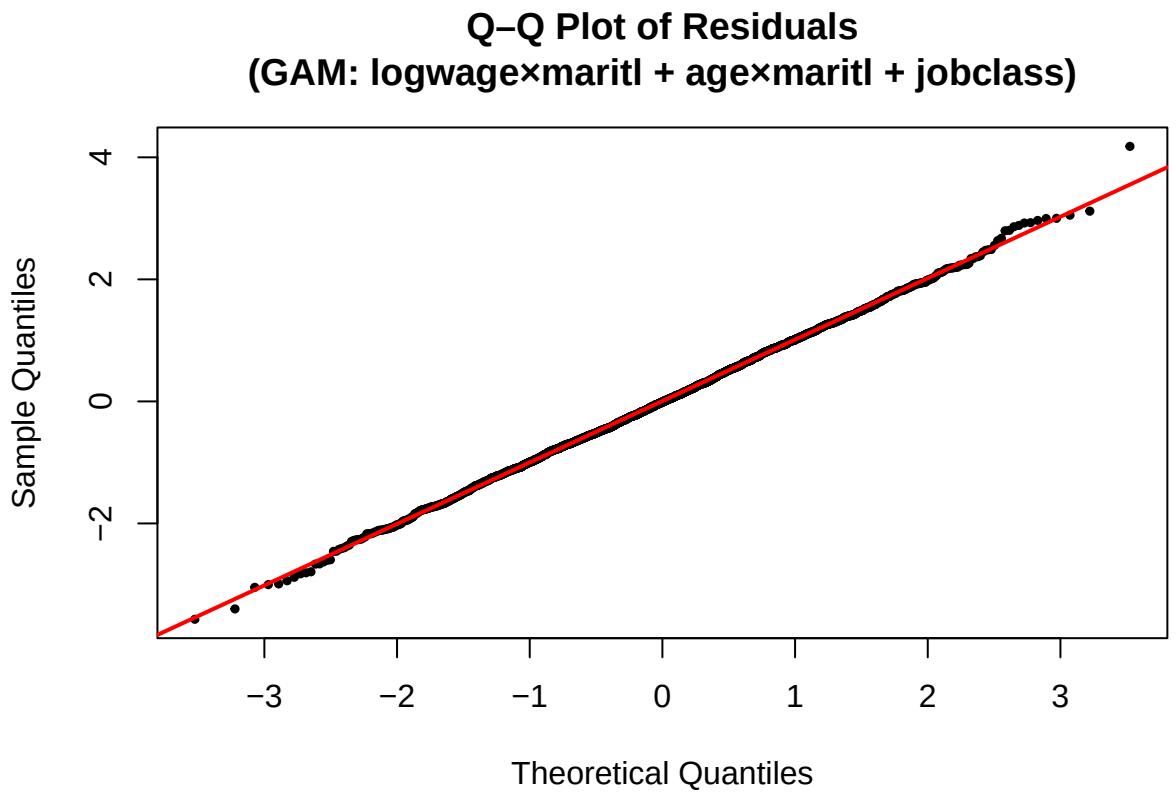
```
cv_results <- data.frame(
  Model = c("GAM base",
            "GAM + jobclass",
            "GAM + edu + jobclass",
            "GAM FULL"),
  CV_MSE = c(cv_base, cv_job, cv_edu_job, cv_full)
)
```

```
cv_results
```

```
##                   Model   CV_MSE
## 1             GAM base 1.081722
## 2       GAM + jobclass 1.024693
## 3 GAM + edu + jobclass 1.026178
## 4             GAM FULL 1.028967
```

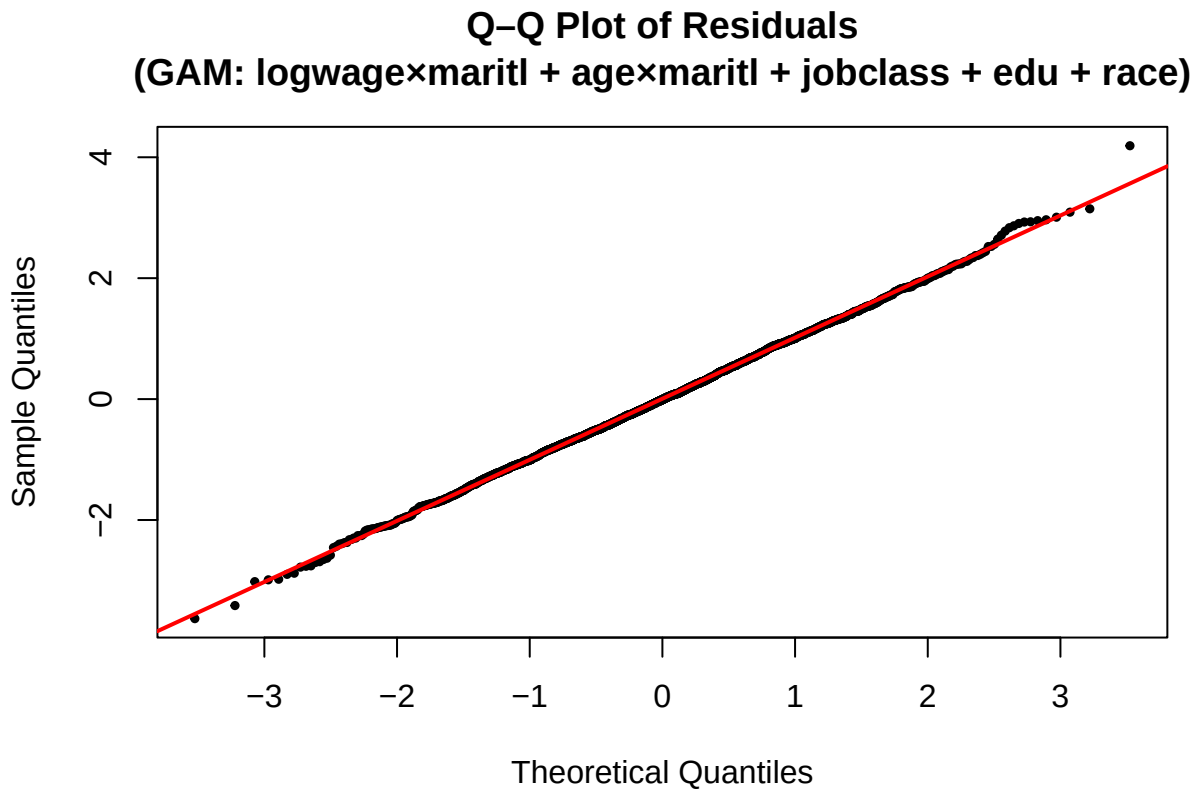
We see through 10-fold cross validation that the Cross Validation MSE is actually lowest when considering interactions with age/logWage with marital status and only including job class.

```
# Extract residuals from the GAM
res_gam <- residuals(gam_full_job_ageint)

# Q-Q Plot
qqnorm(res_gam,
       main = "Q-Q Plot of Residuals\n(GAM: logwage×maritl + age×maritl + jobclass)",
       pch = 19, cex = 0.5)
qqline(res_gam, col = "red", lwd = 2)
```



That being said, we will still likely go with Test MSE as our metric of choice since CV MSE would be more useful in a hyperparameter tuning scenario, which this isn't.

```r
# Extract residuals from the GAM
res_gam <- residuals(gam_full_age_interaction)

# Q-Q Plot
qqnorm(res_gam,
       main = "Q-Q Plot of Residuals\n(GAM: logwage×maritl + age×maritl + jobclass + edu + race)",
       pch = 19, cex = 0.5)
qqline(res_gam, col = "red", lwd = 2)
```



Trees, Random Forest, Bagging, and Boosting

```r

Wage.train <- Wage.train %>%
  mutate(across(where(is.character), factor))

Wage.test <- Wage.test %>%
  mutate(across(where(is.character), factor))
```

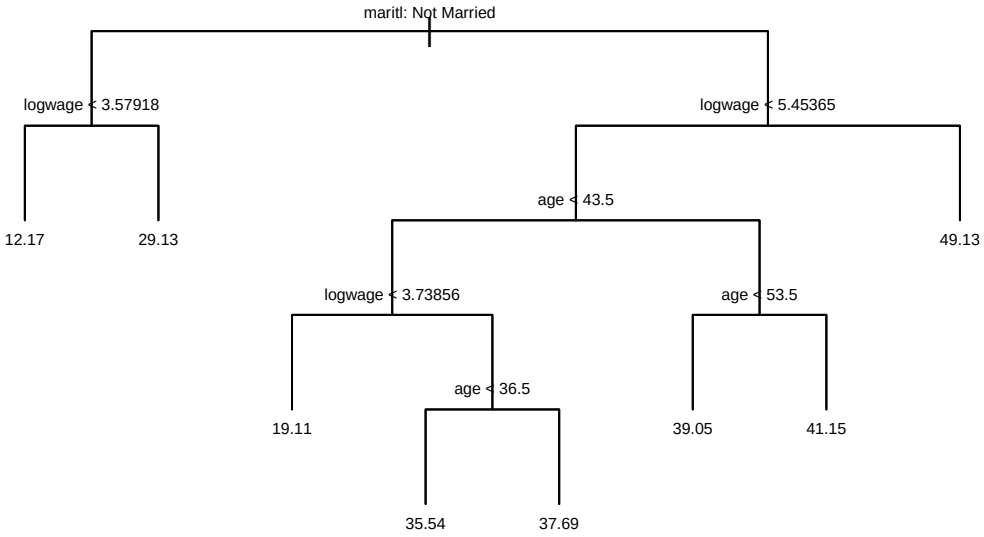
Variable alteration for tree use

```r
# Fit full regression tree on training data
tree_fit <- tree(Resp ~ . -wage, data = Wage.train)
summary(tree_fit)
```

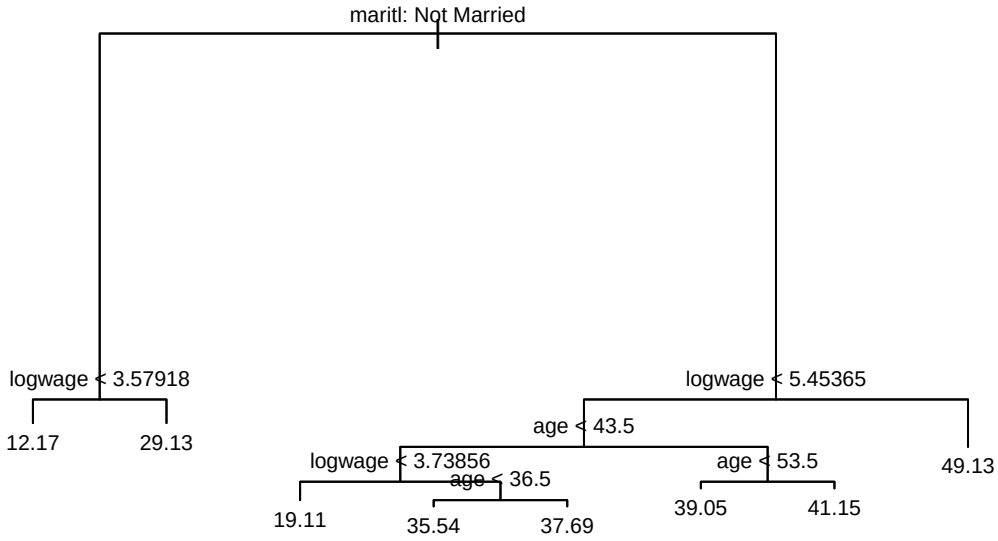
CART

```
##
## Regression tree:
## tree(formula = Resp ~ . - wage, data = Wage.train)
## Variables actually used in tree construction:
## [1] "maritl"  "logwage" "age"
## Number of terminal nodes:  8
## Residual mean deviance:  3.438 = 8059 / 2344
## Distribution of residuals:
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -20.720000  -0.879500   0.000788   0.000000   0.893000  13.800000
```

```r
# Plot full tree
plot(tree_fit, type = "uniform")
text(tree_fit, pretty = 0, cex = 0.5)
```



```r
# Cross-validation to choose tree size
cv_tree <- cv.tree(tree_fit)     # uses deviance
best_size <- cv_tree$size[which.min(cv_tree$dev)]

# Prune to optimal size
prune_fit <- prune.tree(tree_fit, best = best_size)

plot(prune_fit)
text(prune_fit, pretty = 0, cex = 0.7)
```



Prune Tree

```r
# Test MSE for pruned tree
tree_pred <- predict(prune_fit, newdata = Wage.test)
tree_mse  <- mean((tree_pred - Wage.test$Resp)^2)
tree_mse
```

```
## [1] 3.527937
```

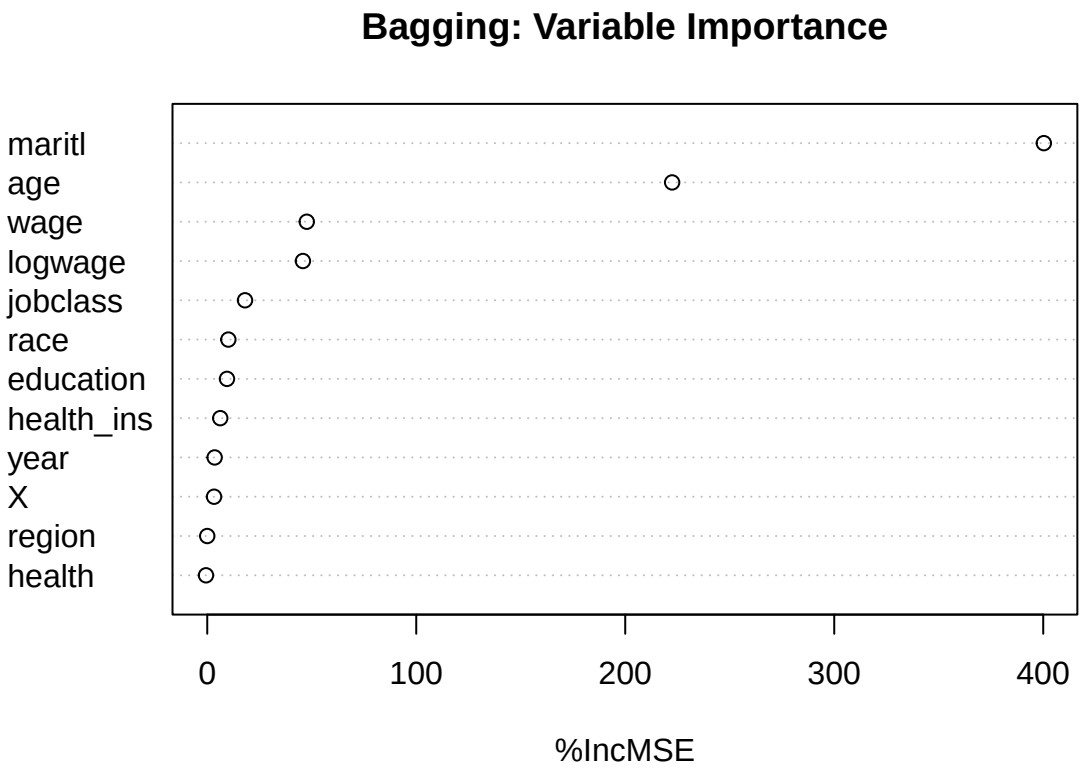
```r
p <- ncol(Wage.train) - 1  # number of predictors (exclude Resp)

bag_fit <- randomForest(
```

```
  Resp ~ .,
  data       = Wage.train,
  subset     = NULL,
  mtry       = p,          # bagging: use ALL predictors at each split
  importance = TRUE,
  ntree      = 500         # like Boston example
)

bag_pred <- predict(bag_fit, newdata = Wage.test)
bag_mse  <- mean((bag_pred - Wage.test$Resp)^2)
bag_mse
```

Bagging

```
## [1] 1.47186
```

```
# Variable importance
varImpPlot(bag_fit, type = 1, main = "Bagging: Variable Importance")
```



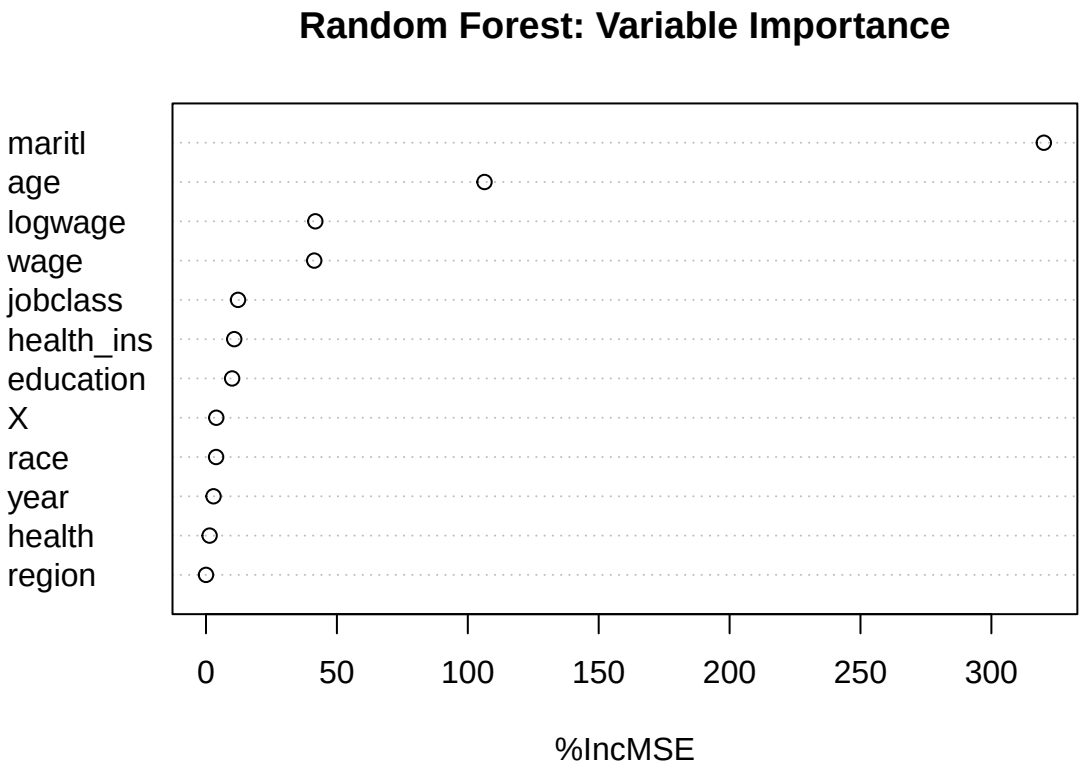

```
rf_fit <- randomForest(
  Resp ~ .,
  data       = Wage.train,
```

```
  importance = TRUE          # so we can plot importance
  # mtry left as default: p/3 for regression
)

rf_pred <- predict(rf_fit, newdata = Wage.test)
rf_mse  <- mean((rf_pred - Wage.test$Resp)^2)
rf_mse
```

Random Forest

```
## [1] 1.603544
```

```
# Variable importance plot (like notes)
par(mfrow = c(1,1))
varImpPlot(rf_fit, type = 1, main = "Random Forest: Variable Importance")
```



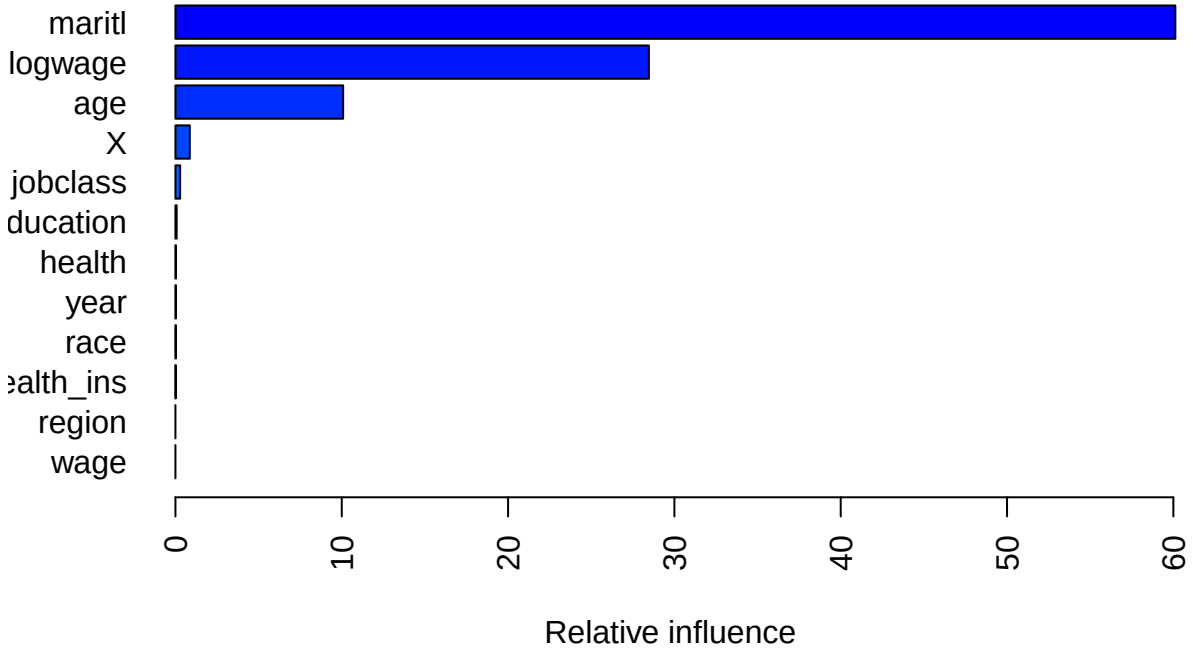

```
boost_fit <- gbm(
  Resp ~ .,
  data              = Wage.train,
  distribution      = "gaussian",  # regression
  n.trees           = 500,         # same as Boston example
  interaction.depth = 2            # tree depth
```

```
  # you could also add cv.folds = 5 to pick n.trees by CV
)
```

Boosting

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 7: region has no variation.
```

```
# Relative influence plot and table
summary(boost_fit, las = 2)
```




```
##                   var     rel.inf
## maritl         maritl 60.12095479
## logwage       logwage 28.46674638
## age               age 10.08269512
## X                   X  0.86256584
## jobclass     jobclass  0.28326186
## education   education  0.07176701
## health         health  0.03039291
## year             year  0.03006926
## race             race  0.02825698
## health_ins health_ins  0.02328985
## region         region  0.00000000
## wage             wage  0.00000000
```

```r
# Test MSE (using 500 trees)
boost_pred <- predict(boost_fit, newdata = Wage.test, n.trees = 500)
boost_mse  <- mean((boost_pred - Wage.test$Resp)^2)
boost_mse
```

```
## [1] 1.418969
```

```r
tree_results <- data.frame(
  Model    = c("Pruned Tree", "Bagging", "Random Forest", "Boosting"),
  Test_MSE = c(tree_mse, bag_mse, rf_mse, boost_mse)
)

tree_results[order(tree_results$Test_MSE), ]
```

```
##           Model Test_MSE
## 4      Boosting 1.418969
## 2       Bagging 1.471860
## 3 Random Forest 1.603544
## 1   Pruned Tree 3.527937
```

We can see that none of these tree based methods come close in approaching the GAM's Test MSE.