R Code for Part 1:

```
#INSTALL THE GGPLOT2 PACKAGE (MUST BE DONE ONCE)
install.packages('ggplot2')

#LOAD THE GGPLOT2 LIBRARY (MUST BE DONE EVERY TIME)
library(ggplot2)

##BUILDING OUR VISUALS USING GGPLOT2

##FIRST VISUAL - SCATTERPLOT SHOWING HOURLY MEAN WAGE
AND EMPLOYMENT PER 1,000 JOBS)
ggplot(df, aes(x = emp, y = Hourlywage)) +
  geom_point() +
  xlab('Employment per 1,000 Jobs') +
  ylab('Hourly Mean Wage') +
  geom_smooth() +
  ggtitle('Scatterplot of Hourly Mean Wage and Employment per 1,000
Jobs') +
  theme(plot.title = element_text(hjust=.5))



#SECOND VISUAL - REGION AND HOME VALUE VIOLIN PLOT
ggplot(df, aes(Region, homeval)) +
  geom_violin() +
  ylab('Typical $ Value of 3 Bed Homes') +
  ylim (0,1000000) +
  ggtitle('Violin Plot of Region and Typical Dollar Value of Three Bedroom
Homes') +
  theme(plot.title = element_text(hjust=.5))
```

```
#THIRD - geom bar with region and employment
ggplot(df, aes(Employment, Region, color = Region)) +
  geom_point() +
  geom_smooth(method = lm, size = 1) +
  ggtitle('Geom Plot of Employment by Region') +
  theme(plot.title = element_text(hjust=.5))



#FOURTH- GEOM TEXT WITH HOURLY WAGE AND HOME VALUE
ggplot(df, aes(Hourlywage, homeval, color=state)) +
  geom_text(aes(label = state), check_overlap = TRUE) +
  xlab('Annual Hourly Wage') +
  ylab('Typical $ Value of 3 Bedroom Home') +
  ggtitle('Scatterplot of Annual Hourly Wage and Typical Value of 3 Bedroom
Home Based on State') +
  theme(plot.title = element_text(hjust=.5))


##FIFTH- GEOM BOXPLOT WITH REGION AND LOCATION QUOTIENT
ggplot(df, aes(Region, LocationQuotient)) +
geom_boxplot(aes(fill=Region))+ ylab("Location Quotient") +
  ggtitle('Concentration of Accountant and Auditor jobs')
+scale_y_continuous(breaks = c(.5,.6,.7,.8, .9,1,
1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9,2))+
  theme(plot.title = element_text(hjust =.5))


###SIXTH- GEOM REGION AND MEAN SALARY TO HOME PRICE
RATIO
ggplot(df, aes(Region, MeanSalaryHomePriceRatio)) +
geom_hex(aes(color= Region)) + xlab("Region")+ ylab("Salary and House
Price Ratio")+ggtitle("House Price to Income Ratio")+ theme(plot.title =
element_text(hjust =.5))+scale_y_continuous(breaks =
c(.5,.1,.15,.2,.25,.3,.35,.4,.45,.5,.55,.6,.65,.7,.75,.8,.85,.9,.95,1))
```

R Code for Part 2:

Christine’s R Code

```
#Christine's pt. 2 model 1#

##SETUP##


#INSTALL THE GGPLOT2 PACKAGE (MUST BE DONE ONCE)
install.packages('ggplot2')

#LOAD THE GGPLOT2 LIBRARY (MUST BE DONE EVERY TIME)
library(ggplot2)

#TRYING TO MODEL THE RELATIONSHIP BETWEEN ANNUAL HOURLY WAGE AND TYPICAL $ VALUE OF 3 BEDROOM HOMES
#INCORPORATING NONLINEAR (POLYNOMIAL) TRANSFORMATIONS OF homeval
acct$homeval2<-acct$homeval^2 #QUADRATIC TRANSFORMATION (2nd ORDER)
acct$homeval3<-acct$homeval^3
acct$homeval4<-acct$homeval^4
```

```
acct$homeval5<-acct$homeval^5
acct$ln_homeval<-log(acct$homeval)
acct$emp2<-acct$emp^2
acct$emp3<-acct$emp^3
acct$emp4<-acct$emp^4
acct$emp5<-acct$emp^5
acct$ln_emp<-log(acct$emp)

#fraction of sample to be used for training
p<-.7 #use 70% of the data to train/build the model

#number of observations (rows) in the dataframe
obs_count<-dim(acct)[1]

#number of observations to be selected for the training
partition
#the floor() function rounds down to the nearest integer
training_size <- floor(p * obs_count)
training_size

#set the seed to make your partition reproducible
set.seed(1234)

#create a vector with the shuffled row numbers of the
original dataset
train_ind <- sample(obs_count, size = training_size)
```

Training <- acct[train_ind,] #pulls random rows for training
Testing <- acct[-train_ind,] #pulls random rows for testing

##CHRISTINE'S MODEL 1##
#BUILDING THE QUADRATIC MODEL FROM THE TRAINING DATA
M1 <- lm(Hourlywage ~ homeval + homeval2, Training)
summary(M1) #generates summary diagnostic output

#GENERATING PREDICTIONS ON THE TRAINING DATA
PRED_1_IN <- predict(M1, Training) #generate predictions on the (in-sample) training data

#GENERATING PREDICTIONS ON THE TEST DATA FOR BENCHMARKING
PRED_1_OUT <- predict(M1, Testing) #generate predictions on the (out-of-sample) testing data

#COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN SQUARED ERROR
RMSE_1_IN<-sqrt(sum((PRED_1_IN-Training$Hourlywage)^2)/length(PRED_1_IN)) #computes in-sample error

```
RMSE_1_OUT<-sqrt(sum((PRED_1_OUT-Testing$Hourly
wage)^2)/length(PRED_1_OUT)) #computes
out-of-sample

RMSE_1_IN #IN-SAMPLE ERROR
RMSE_1_OUT #OUT-OF-SAMPLE ERROR

#PLOTTING THE MODEL IN 2D AGAINST BOTH DATA
PARTITIONS
x_grid <- seq(0,1600000,1) #CREATES GRID OF X-AXIS
VALUES
predictions <- predict(M1, list(homeval=x_grid,
homeval2=x_grid^2))
plot(Training$Hourlywage ~ Training$homeval, xlim=
c(95000,1000000), main= "Mean Hourly Wage vs. Typical
Home Value", sub=
      "Quadratic Model", xlab= "Typical Home Value", ylab=
"Mean Hourly Wage", col='blue')
lines(x_grid, predictions, col='green', lwd=3)
points(Testing$Hourlywage ~ Testing$homeval, col='red',
pch=3)
legend("bottomright", legend = c("Testing Data
Points","Training Data","Model"), fill=c("red", "blue",
"green"), bg="orange", title="Legend")

#Christine's Model #2
```

```
M2 <- lm(Hourlywage ~ emp4 + emp3 + emp2 + emp +
homeval2 + homeval , Training)
summary(M2)
PRED_2_IN <- predict(M2, Training)
PRED_2_OUT <- predict(M2, Testing)
RMSE_2_IN<-sqrt(sum((PRED_2_IN-Training$Hourlywag
e)^2)/length(PRED_2_IN))
RMSE_2_OUT<-sqrt(sum((PRED_2_OUT-Testing$Hourly
wage)^2)/length(PRED_2_OUT))
RMSE_2_IN
RMSE_2_OUT

#can't plot multivariate model# - don't need plots for
multivariate cases
```

Flavio's R Code

```
#IMPORT THE EXCEL FILE

########PLACE GITHUB PATH HERE########



#A LOGARITHMIC TRANSFORMATION OF HOMEVAL

df$ln_homeval<-log(df$homeval)
```

```
#CREATING DUMMY VARIBALES IN THE DATA SET FOR STATE

df$AL <- ifelse(df$state == 'AL', 1, 0)

df$AR <- ifelse(df$state == 'AR', 1, 0)

df$AZ <- ifelse(df$state == 'AZ', 1, 0)

df$CA <- ifelse(df$state == 'CA', 1, 0)

df$CO <- ifelse(df$state == 'CO', 1, 0)

df$CT <- ifelse(df$state == 'CT', 1, 0)

df$DE <- ifelse(df$state == 'DE', 1, 0)

df$FL <- ifelse(df$state == 'FL', 1, 0)

df$GA <- ifelse(df$state == 'GA', 1, 0)

df$IA <- ifelse(df$state == 'IA', 1, 0)

df$ID <- ifelse(df$state == 'ID', 1, 0)

df$IL <- ifelse(df$state == 'IL', 1, 0)

df$IN <- ifelse(df$state == 'IN', 1, 0)
```

```
df$KS <- ifelse(df$state == 'KS', 1, 0)

df$KY <- ifelse(df$state == 'KY', 1, 0)

df$LA <- ifelse(df$state == 'LA', 1, 0)

df$MA <- ifelse(df$state == 'MA', 1, 0)

df$MD <- ifelse(df$state == 'MD', 1, 0)

df$ME <- ifelse(df$state == 'ME', 1, 0)

df$MI <- ifelse(df$state == 'MI', 1, 0)

df$MN <- ifelse(df$state == 'MN', 1, 0)

df$MO <- ifelse(df$state == 'MO', 1, 0)

df$MS <- ifelse(df$state == 'MS', 1, 0)

df$MT <- ifelse(df$state == 'MT', 1, 0)

df$NC <- ifelse(df$state == 'NC', 1, 0)

df$ND <- ifelse(df$state == 'ND', 1, 0)

df$NE <- ifelse(df$state == 'NE', 1, 0)

df$NH <- ifelse(df$state == 'NH', 1, 0)

df$NJ <- ifelse(df$state == 'NJ', 1, 0)
```

```
df$NM <- ifelse(df$state == 'NM', 1, 0)

df$NV <- ifelse(df$state == 'NV', 1, 0)

df$NY <- ifelse(df$state == 'NY', 1, 0)

df$OH <- ifelse(df$state == 'OH', 1, 0)

df$OK <- ifelse(df$state == 'OK', 1, 0)

df$OR <- ifelse(df$state == 'OR', 1, 0)

df$PA <- ifelse(df$state == 'PA', 1, 0)

df$SC <- ifelse(df$state == 'SC', 1, 0)

df$SD <- ifelse(df$state == 'SD', 1, 0)

df$TN <- ifelse(df$state == 'TN', 1, 0)

df$TX <- ifelse(df$state == 'TX', 1, 0)

df$UT <- ifelse(df$state == 'UT', 1, 0)

df$VA <- ifelse(df$state == 'VA', 1, 0)

df$VT <- ifelse(df$state == 'VT', 1, 0)

df$WA <- ifelse(df$state == 'WA', 1, 0)

df$WI <- ifelse(df$state == 'WI', 1, 0)
```

```
df$WV <- ifelse(df$state == 'WV', 1, 0)

df$WY <- ifelse(df$state == 'WY', 1, 0)



#fraction of sample to be used for training

p<-.7 #use 70% of the data to train/build the model


#number of observations (rows) in the dataframe

obs_count<-dim(df)[1]


#Setting the training data

#floor=round down

#p=.7

#obs_count=# of observations

training_size <- floor(p * obs_count)
```

```
#set the seed to make your partition reproducible

set.seed(1234)

#create a vector with the shuffled row numbers of the original dataset

train_ind <- sample(obs_count, size = training_size)



Training <- df[train_ind, ] #pulls random rows for training

Testing <- df[-train_ind, ] #pulls random rows for testing



#PLOTTING THE TRAINING AND TESTING PARTITIONS

plot(df$Hourlywage ~ df$homeval, df, xlim=c(50000,700000),
ylim=c(0,75)) #PLOT ENTIRE DATASET

plot(df$Hourlywage ~ df$homeval, Training, xlim=c(50000,700000),
ylim=c(0,75), col ='blue') #PLOTS THE IN-SAMPLE TRAINING
PARTITION

plot(df$Hourlywage ~ df$homeval, Testing, xlim=c(50000,700000),
ylim=c(0,75),  col ='red', pch=3) #PLOTS THE OUT-OF-SAMPLE
TESTING PARTITION

points(Training$homeval, Training$Hourlywage, col='blue') #PLOTS
THE OUT-OF-SAMPLE Training PARTITION
```

```
points(Testing$homeval, Testing$Hourlywage, col='red', pch=3) #PLOTS THE OUT-OF-SAMPLE TESTING PARTITION



#BUILDING THE MODEL FROM THE TRAINING DATA

M1 <- lm(Hourlywage ~ homeval, Training)

summary(M1) #generates summary diagnostic output



#Generating Predictions for the training data

PRED_1_IN <- predict(M1, Training) #generate predictions on the (in-sample) training data

View(PRED_1_IN)



#GENERATING PREDICTIONS ON THE TEST DATA FOR BENCHMARKING

PRED_1_OUT <- predict(M1, Testing) #generate predictions on the (out-of-sample) testing data

View (PRED_1_OUT)
```

```
#COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN
SQUARED ERROR

RMSE_1_IN<-sqrt(sum((PRED_1_IN-Training$Hourlywage)^2)/length
(PRED_1_IN))  #computes in-sample error

RMSE_1_OUT<-sqrt(sum((PRED_1_OUT-Testing$Hourlywage)^2)/le
ngth(PRED_1_OUT)) #computes out-of-sample

RMSE_1_IN #IN-SAMPLE ERROR

RMSE_1_OUT #OUT-OF-SAMPLE ERROR

#PLOTTING THE MODEL IN 2D AGAINST BOTH DATA
PARTITIONS

x_grid <- seq(0,7500000,100000) #CREATES GRID OF X-AXIS
VALUES

predictions <- predict(M1, list(homeval=x_grid))

plot(Training$Hourlywage ~ Training$homeval, col='blue')
```

lines(x_grid, predictions, col='green', lwd=3)

points(Testing$Hourlywage ~ Testing$homeval, col='red', pch=3)

#BUILDING THE LOGARITHMIC MODEL FROM THE TRAINING DATA

M4 <- lm(Hourlywage ~ ln_homeval , Training)

summary(M4) #generates summary diagnostic output

#GENERATING PREDICTIONS ON THE TRAINING DATA

PRED_4_IN <- predict(M4, Training) #generate predictions on the (in-sample) training data

View(PRED_4_IN)

View(M4$fitted.values) #these are the same as the fitted values

#PLOTTING RESIDUALS

```
residuals<-resid(M4)

plot(M4$fitted.values,residuals, main= "Plot: Heteroscedasticity of Residuals")

abline(0,0, col=5, lwd=3)

abline(7,0, col="red", lwd=.5)

abline(-7,0, col="red", lwd=.5)


#Plotting Linearity for Expalantory Variable

plot(residuals ~ Training$ln_homeval, main= "Testing for Linearity")

abline(0,0, col=5, lwd=3)

abline(7,0, col="red", lwd=.5)

abline(-7,0, col="red", lwd=.5)


#RESIDUAL ANALYSIS FOR NORMALITY

hist(residuals)
```

```
#GENERATING PREDICTIONS ON THE TEST DATA FOR
BENCHMARKING

PRED_4_OUT <- predict(M4, Testing) #generate predictions on the
(out-of-sample) testing data



#COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN
SQUARED ERROR

RMSE_4_IN<-sqrt(sum((PRED_4_IN-Training$Hourlywage)^2)/length
(PRED_4_IN))  #computes in-sample error

RMSE_4_OUT<-sqrt(sum((PRED_4_OUT-Testing$Hourlywage)^2)/le
ngth(PRED_4_OUT)) #computes out-of-sample



RMSE_4_IN #IN-SAMPLE ERROR

RMSE_4_OUT #OUT-OF-SAMPLE ERROR



#PLOTTING THE MODEL IN 2D AGAINST BOTH DATA
PARTITIONS
```

```
x_grid <- seq(0,7500000,100000) #CREATES GRID OF X-AXIS VALUES

predictions <- predict(M4,list(ln_homeval=log(x_grid)))

plot(Training$Hourlywage ~ Training$homeval, xlim= c(95000,1000000), main= "Mean Hourly Wage vs. Typical Home Value", sub=

      "Natrual Log Model", xlab= "Typical Home Value", ylab= "Mean Hourly Wage", col='blue')

lines(x_grid, predictions, col='green', lwd=3)

points(Testing$Hourlywage ~ Testing$homeval, col='red', pch=3)



#ADDING A LEGEND

legend ("bottomright", legend= c("Testing Data Points","Training Data Points","Model"), fill= c("red", "blue", "green"), bg= "orange", title= "Legend")




#Curious about correlations

cor(df$Hourlywage,df$LocationQuotient)
```

```
cor(df$Hourlywage,df$emp)

cov(df)

cov(df[,c(5,21,20,19,18,17)])

cor(df[,c(5,21,20,19,18,17)])



#THE MULTIVARIATE MODEL

M5 <- lm (Hourlywage ~ homeval + LocationQuotient + emp + AL +
AR + AZ + CA + CO + CT

        + DE + FL + GA + IA + ID + IL + IN + KS  + KY + LA + MA + MD
+ ME + MI + MN

        + MO + MS + MT + NC + ND + NE + NH + NJ + NM + NV + NY
+ OH + OK + OR + PA +

          SC + SD + TN + TX + UT + VA + VT + WA  + WI + WV + WY,
Training)

summary(M5)



#GENERATING PREDICTIONS ON THE TRAINING DATA

PRED_5_IN <- predict(M5, Training) #generate predictions on the
(in-sample) training data
```

```
View(PRED_5_IN)

View(M4$fitted.values) #these are the same as the fitted values



#GENERATING PREDICTIONS ON THE TEST DATA FOR BENCHMARKING

PRED_5_OUT <- predict(M5, Testing) #generate predictions on the (out-of-sample) testing data



#COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN SQUARED ERROR

RMSE_5_IN<-sqrt(sum((PRED_5_IN-Training$Hourlywage)^2)/length(PRED_5_IN))  #computes in-sample error

RMSE_5_OUT<-sqrt(sum((PRED_5_OUT-Testing$Hourlywage)^2)/length(PRED_5_OUT)) #computes out-of-sample



RMSE_5_IN #IN-SAMPLE ERROR

RMSE_5_OUT #OUT-OF-SAMPLE ERROR
```

```
#PLOTTING RESIDUALS

residuals5<-resid(M5)

plot(M5$fitted.values,residuals, main= "Plot: Heteroscedasticity of Residuals")



#Plotting Linearity for emp Variable

plot(residuals ~ Training$emp, main= "Testing for Linearity")

abline(0,0, col=5, lwd=3)

abline(7,0, col="red", lwd=.5)

abline(-7,0, col="red", lwd=.5)



#Plotting Linearity for LocationQuotient Variable

plot(residuals ~ Training$LocationQuotient, main= "Testing for Linearity")

abline(0,0, col=5, lwd=3)

abline(7,0, col="red", lwd=.5)

abline(-7,0, col="red", lwd=.5)
```

```
#RESIDUAL ANALYSIS FOR NORMALITY

hist(residuals5)

#MULTIVARIATE MODEL W/O CATEGORICAL VARIABLES INCLUDED

#THE MULTIVARIATE MODEL

M6 <- lm (Hourlywage ~ homeval + LocationQuotient + emp, Training)

summary(M6)

#GENERATING PREDICTIONS ON THE TRAINING DATA

PRED_6_IN <- predict(M6, Training) #generate predictions on the (in-sample) training data

View(PRED_6_IN)

View(M6$fitted.values) #these are the same as the fitted values
```

#GENERATING PREDICTIONS ON THE TEST DATA FOR BENCHMARKING

PRED_6_OUT <- predict(M6, Testing) #generate predictions on the (out-of-sample) testing data

#COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN SQUARED ERROR

RMSE_6_IN<-sqrt(sum((PRED_6_IN-Training$Hourlywage)^2)/length (PRED_6_IN)) #computes in-sample error

RMSE_6_OUT<-sqrt(sum((PRED_6_OUT-Testing$Hourlywage)^2)/length(PRED_6_OUT)) #computes out-of-sample

RMSE_5_IN #IN-SAMPLE ERROR

RMSE_5_OUT #OUT-OF-SAMPLE ERROR

<u>Alejandra's R Code</u>

####Plotting a linear model-Alejandra####

library(ggplot2)

```
##import dataset##

#fraction of sample to be used for training
p<-.7 #use 70% of the data to train/build the model#########

#number of observations (rows) in the dataframe
obs_count<-dim(df)[1]

#number of observations to be selected for the training partition
#the floor() function rounds down to the nearest integer
training_size <- floor(p * obs_count)
training_size
#set the seed to make your partition reproducible
set.seed(1234)
#create a vector with the shuffled row numbers of the original dataset
train_ind <- sample(obs_count, size = training_size)

Training <- df[train_ind, ] #pulls random rows for training
Testing <- df[-train_ind, ] #pulls random rows for testing

#CHECKING THE DIMENSIONS OF THE PARTITIONED DATA
dim(Training)
dim(Testing)

#PLOTS THE IN-SAMPLE TRAINING PARTITION

plot(Hourlywage ~ homeval, Testing, xlim=c(10000,500000),
ylim=c(10,70),  col ='red', pch=3) #PLOTS THE OUT-OF-SAMPLE
TESTING PARTITION
points(Training$homeval, Training$Hourlywage, col='blue') #PLOTS
THE OUT-OF-SAMPLE TESTING PARTITION
```

```
points(Testing$homeval, Testing$Hourlywage, col='red', pch=3)
#PLOTS THE OUT-OF-SAMPLE TESTING PARTITION


#BUILDING THE MODEL FROM THE TRAINING DATA
M1 <- lm(Hourlywage ~ homeval, Training)
summary(M1) #generates summary diagnostic output

#GENERATING PREDICTIONS ON THE TRAINING DATA
PRED_1_IN <- predict(M1, Training) #generate predictions on the
(in-sample) training data
View(PRED_1_IN)
View(M1$fitted.values) #these are the same as the fitted values

#GENERATING PREDICTIONS ON THE TEST DATA FOR
BENCHMARKING
PRED_1_OUT <- predict(M1, Testing) #generate predictions on the
(out-of-sample) testing data (red points)

#COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN
SQUARED ERROR
RMSE_1_IN<-sqrt(sum((PRED_1_IN-Training$Hourlywage)^2)/length
(PRED_1_IN))  #computes in-sample error
RMSE_1_OUT<-sqrt(sum((PRED_1_OUT-Testing$Hourlywage)^2)/le
ngth(PRED_1_OUT)) #computes out-of-sample

RMSE_1_IN #IN-SAMPLE ERROR
RMSE_1_OUT #OUT-OF-SAMPLE ERROR

#PLOTTING THE MODEL IN 2D AGAINST BOTH DATA
PARTITIONS
```

```
x_grid <- seq(0,15000000,1)  #CREATES GRID OF X-AXIS VALUES
predictions <- predict(M1, list(homeval=x_grid))
plot(Training$Hourlywage ~ Training$homeval, col='blue', xlab="Home
Value
  Simple Linear Model", ylab="Hourly Wage", main= "Mean Hourly
Wage vs. Typical Home Value")
lines(x_grid, predictions, col='green', lwd=3)
points(Testing$Hourlywage ~ Testing$homeval, col='red', pch=3)
legend ("bottomright", legend= c("Testing Data Points","Training Data
Points","Model"), fill= c("red", "blue", "green"), bg= "orange", title=
"Legend")


##################################################
#####Plotting with more than one x variable#####
##################################################

library(ggplot2)

##import dataset##

#fraction of sample to be used for training
p<-.7 #use 70% of the data to train/build the model#########

#number of observations (rows) in the dataframe
obs_count<-dim(df)[1]

#number of observations to be selected for the training partition
#the floor() function rounds down to the nearest integer
training_size <- floor(p * obs_count)
training_size
#set the seed to make your partition reproducible
```

```
set.seed(1234)
#create a vector with the shuffled row numbers of the original dataset
train_ind <- sample(obs_count, size = training_size)

Training <- df[train_ind, ] #pulls random rows for training
Testing <- df[-train_ind, ] #pulls random rows for testing

#CHECKING THE DIMENSIONS OF THE PARTITIONED DATA
dim(Training)
dim(Testing)


#BUILDING THE MODEL FROM THE TRAINING DATA
M2 <- lm(Hourlywage ~ homeval+ Employment, Training)
summary(M2) #generates summary diagnostic output

#GENERATING PREDICTIONS ON THE TRAINING DATA
PRED_1_IN_2 <- predict(M2, Training) #generate predictions on the
(in-sample) training data
View(PRED_1_IN_2)
View(M2$fitted.values) #these are the same as the fitted values

#GENERATING PREDICTIONS ON THE TEST DATA FOR
BENCHMARKING
PRED_1_OUT_2 <- predict(M2, Testing) #generate predictions on the
(out-of-sample) testing data (red points)

#COMPUTING IN-SAMPLE AND OUT-OF-SAMPLE ROOT MEAN
SQUARED ERROR
RMSE_1_IN_2<-sqrt(sum((PRED_1_IN_2-Training$Hourlywage)^2)/le
ngth(PRED_1_IN_2))  #computes in-sample error
```

```
RMSE_1_OUT_2<-sqrt(sum((PRED_1_OUT_2-Testing$Hourlywage)^
2)/length(PRED_1_OUT_2)) #computes out-of-sample

RMSE_1_IN_2 #IN-SAMPLE ERROR
RMSE_1_OUT_2 #OUT-OF-SAMPLE ERROR
```