Solution 1

```
library(mlbench)
library(MASS)

data('BostonHousing')

# (a)
hist(BostonHousing$medv, breaks = 20)
# It is essential to plot the data. Here it turns out that houses costing more than 50 were capped at 50.
# If there were a lot of values at 50, you could consider how to downweight them, to limit their effect on the fit.

pairs(BostonHousing)
# gives a general overview of data.  Some variables take only two values, some nonlinear relations with the response are visible.

# (b)
model1 <- lm(formula = medv ~ ., data = BostonHousing)
summary(model1)
# R-squared value suggests that this is a decent fit. Except for "indus" and "age", all other variables seem to be quite significant.
# 95% confidence interval is Estimate +/- t_quantile * SE
# Do the signs of the estimates make sense? (e.g., crim has a negative sign, so high crim is associated with lower prices)
# Which of the variables are most significant?  Why might that be?

# (c)
par(mfrow=c(2,2)) # 2x2 matrix of plots
plot(model1)
# Line above already plots residuals vs. fitted values, showing clear pattern and asymmetric residuals.
# Next line shows fitted vales and standardized residuals.
plot(predict(model1, newdata=BostonHousing[, -14]), rstandard(model1), xlab = 'Fitted values', ylab = 'Standardized residuals')
# Seems like variance of the response depends on its mean.
# The QQ plots suggest that the standardized residuals are right-skewed.
# Influential points may have large leverage and/or large Cook's distance (see plots). The plots label points 369, 372, and 373 as outliers.

# (d)
par(mfrow=c(3,4))
resstan <- rstandard(model1)
for (i in colnames(BostonHousing)[-14]) {
    plot(BostonHousing[[i]], resstan, xlab = i, ylab = 'Standardized residuals')
}
# mean-variance relationship is due to rm and/or lstat?

# Try transforming to squared
BostonHousing_transformed <- transform(BostonHousing, rm = rm^2)
BostonHousing_transformed <- transform(BostonHousing_transformed, lstat = lstat^2)
model_transformed <- lm(formula = medv ~ ., data = BostonHousing_transformed)
summary(model_transformed)
plot(model_transformed)

resstan_transformed <- rstandard(model_transformed)
for (i in colnames(BostonHousing_transformed)[-14]) {
    plot(BostonHousing_transformed[[i]], resstan_transformed, xlab = i, ylab = 'Standardized residuals')
}
# Seems like there is quadratic relation for standardized residuals vs. "rm" and "lstat".
# plot for rm suggests that the model underpredicts prices when rm is lower or higher than the range 5 to 7.5.
# plot for lstat suggests that the model underpredicts prices when lstat < 10 or lstat > 30.
# Above quadratic transformation for "rm" and "lstat" does not seem to improve the fit (R-squared decreased).

# (e)
boxcox_lambdas <- boxcox(model1)
lambda <- boxcox_lambdas$x[which.max(boxcox_lambdas$y)] # Get the best lambda
# plot suggests that a log transformation (i.e., relative prices) will be much better, even if optimal transformation has lambda = 0.1
BostonHousing_transformed_boxcox <- transform(BostonHousing, medv = (medv^lambda - 1)/lambda) # Optimal transform
model_boxcox <- lm(formula = medv ~ ., data = BostonHousing_transformed_boxcox) # Fit
summary(model_boxcox)
# Box-Cox transformation increases R-squared quite a bit.  Better fit?

# Try log transformation and find almost no difference, so log is preferred as easy to interpret.
model_log <- lm(formula = log(medv) ~ ., data = BostonHousing) # Fit
summary(model_log)
plot(model_log)
# the diagnostic plots are better; still some underprediction for the lowest prices.  The cap at 50 is very obvious.
# residuals now more symmetric, even if they are still heavy-tailed relative to the normal.

# (f)  Carry on with log prices
LogBostonHousing <- BostonHousing
LogBostonHousing[,14] <- log(BostonHousing[,14])
nullmodel <- lm(medv ~ 1, data = LogBostonHousing)
fullmodel <- lm(medv ~ ., data = LogBostonHousing)

# Forward selection
model.step.f <- step(nullmodel, scope = list(lower = nullmodel, upper = fullmodel), direction = 'forward')

# Backward elimination
model.step.b <- step(fullmodel, direction = 'backward')

# Stepwise selection
model.step <- step(nullmodel, scope = list(lower = nullmodel, upper = fullmodel), direction = 'both')

# AICs and BICs
AIC(model.step.f)
BIC(model.step.f)
AIC(model.step.b)
BIC(model.step.b)
```

```
AIC(model.step)
BIC(model.step)
# The models do not differ (not always the case). The models have "indus" and "age" removed, consistent with (b).

# (g)
iterations <- 500
prediction_errors <- c()
for (it in 1:iterations) {
    train_indices <- sample(1:nrow(LogBostonHousing), 0.7*nrow(LogBostonHousing))
    train_boston_housing <- LogBostonHousing[train_indices, ]
    test_boston_housing <- LogBostonHousing[-train_indices, ]
    model2 <- lm(formula = medv ~ ., data = train_boston_housing)
    predictions <- predict(model2, newdata=test_boston_housing[-14], interval='confidence')
    prediction_error <- mean((predictions[, 1] - test_boston_housing$medv)^2)
    prediction_errors <- c(prediction_errors, prediction_error)
}
hist(prediction_errors, breaks = 20)
# The histogram gives an idea of the stability of the prediction errors over 500 random splits.
```