① $P=0: \; Y_i = \beta_0^{(0)} + \beta_1^{(0)} X_i + \text{noise}, \; n_0$

$P=1: \; Y_i = \beta_0^{(1)} + \beta_1^{(1)} X_i + \text{noise}, \; n_1$

② $Y_i = \beta_0 + \beta_1 X_i + \beta_2 P_i + \beta_3 (X_i \cdot P_i) + \text{noise}$

$\Rightarrow$ when $P=0$: $Y_i | P_i = 0 = \beta_0 + \beta_1 X_i$

when $P=1$: $Y_i | P_i = 1 = \beta_0 + \beta_1 X_i + \beta_2 + \beta_3 X_i$

$\Rightarrow Y_i | P_i = 1 = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) X_i$

a) In both linear regression options, we are assuming that $E[Y|X,P]$ is linear in the covariates and that $E[\text{noise}] = 0$. Specifically, the options are the same because both condition on P. So, we set the options equal for when $P=0$ and $P=1$ to see the relationship between the coefficients:

- $P=0$: $\beta_0^{(0)} = \beta_0$

  $\beta_1^{(0)} = \beta_1$
- $P=1$: $\beta_0^{(1)} = \beta_0 + \beta_2$

  $\beta_1^{(1)} = \beta_1 + \beta_3$

b) Combined dataset: $n_0 + n_1 = n$

$Y = Y_1 \dots Y_n$ (full)

$X = X_1 \dots X_n$ (full)

$P = P_1, \dots, P_n$ (1 or 0)

Note that the combined linear regression includes an interaction term between X and P. So,

$$Var[Y_i] = Var[\beta_0 + \beta_1 X_i + \beta_2 P_i + \beta_3 X_i \cdot P_i + \text{noise}]$$

$= Var[\text{noise}] = \sigma^2$ since X, P fixed, β_i's constants.

However, in the combined model,

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} \hat{\epsilon}_i^2}{n-p} = \frac{\sum_{i=1}^{n} \hat{\epsilon}_i^2}{n_0+n_1-4} = \frac{RSS_n}{n_0+n_1-4}$$

While in each separate model,

$$\hat{\sigma}_0^2 = \frac{\sum_{i=1}^{n_0} \hat{\epsilon}_i^2}{n_0-2} = \frac{RSS_0}{n_0-2}, \quad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^{n_1} \hat{\epsilon}_i^2}{n_1-2} = \frac{RSS_1}{n_1-2}$$

So we actually assume different $Var[Y|X,P]$ between the two model options since in option 1 we are getting two separate estimates for $P=0,1$ based on smaller n_0 and n_1, compared to an estimate $\hat{\sigma}^2$ based on n and more parameters, including an interaction term as well that affects RSS calculation.

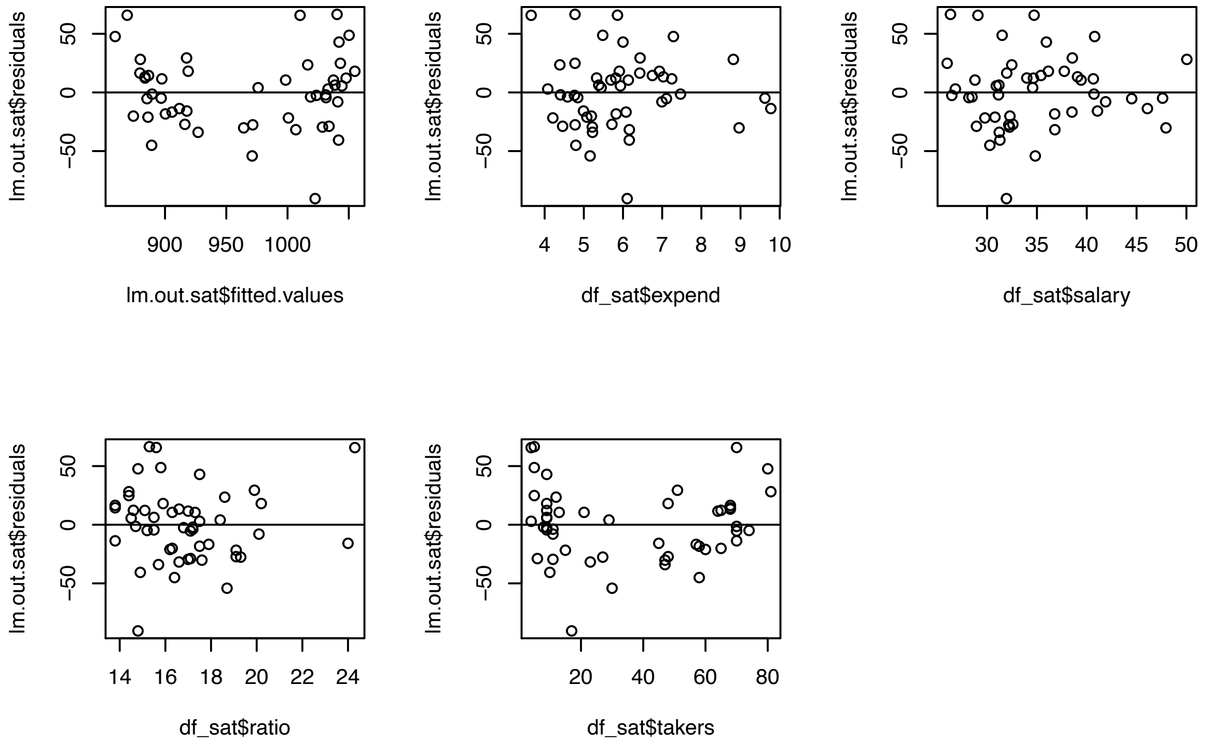
2.

```r
library(faraway)
df_sat <- data.frame(sat)
lm.out.sat <- lm(total ~ expend + salary + ratio + takers, data = df_sat)
lm.out.sat1 <- lm(total ~ expend + salary + ratio + log(takers), data = df_sat)
```

a)

Check constant variance assumption for the errors

```r
par(mfrow = c(2,3))
plot(y = lm.out.sat$residuals, x = lm.out.sat$fitted.values)
abline(h=0)
plot(y = lm.out.sat$residuals, x = df_sat$expend)
abline(h=0)
plot(y = lm.out.sat$residuals, x = df_sat$salary)
abline(h=0)
plot(y = lm.out.sat$residuals, x = df_sat$ratio)
abline(h=0)
plot(y = lm.out.sat$residuals, x = df_sat$takers)
abline(h=0)
```

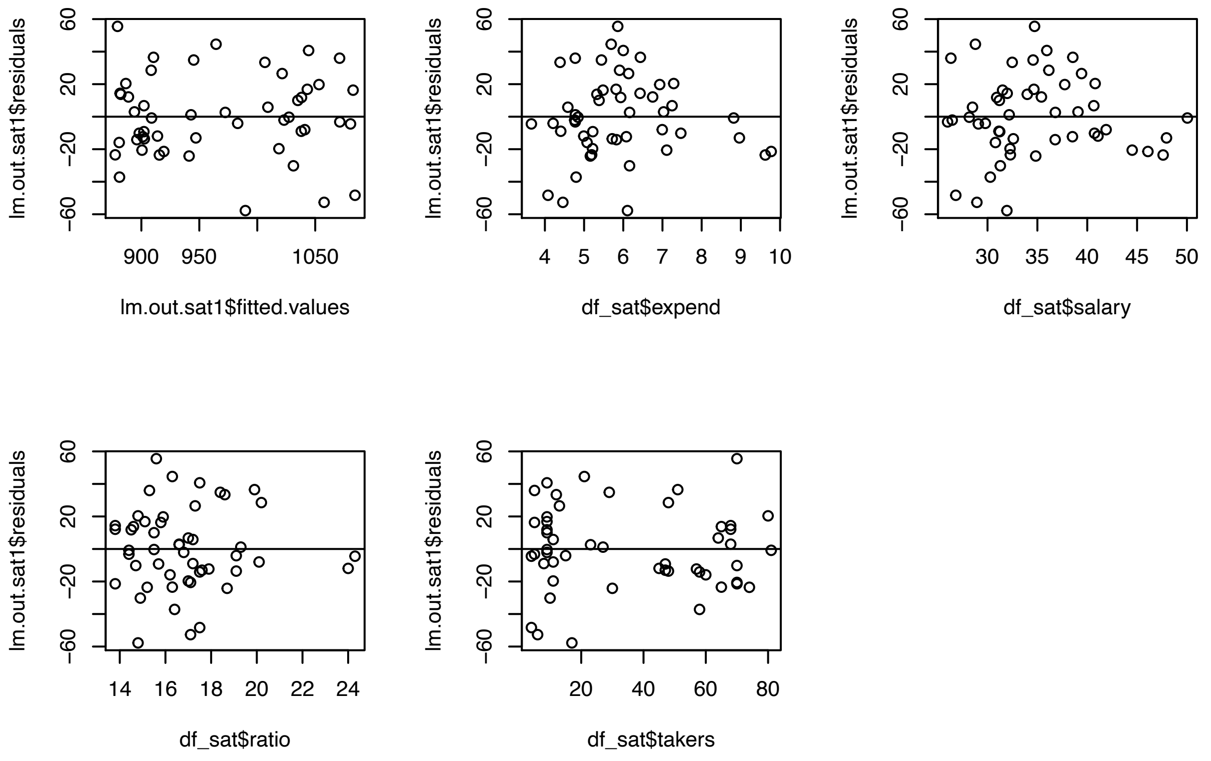


First we check the plot of model residuals vs. fitted values and see a pattern in the residuals along with nonconstant variance. Specifically, there appears to be a negative linear trend and then a positive linear trend.

When we examine the plots of the model residuals vs. each covariate, we generally see homoskedasticity in the plots for `expend`, `salary`, and `ratio` with the exception of a few outliers. However, in the plot of residuals vs. `takers`, we see a very similar trend to what we saw in the residuals vs. fitted values plot of a negative linear trend and then positive linear trend. This suggests that our model may not be the most appropriate and we should consider a log transformation of the covariate `takers`.

The following are the same residual plots but for the linear model rerun with `log(takers)`. Clearly, we see that the pattern in the residuals vs. fitted values plot and the residuals vs. `log(takers)` plot is now gone, and there does not seem to be any trend at all. Additionally, the constant variance assumption appears to be much more valid after this transformation. We will proceed with this new model for further model diagnostics.

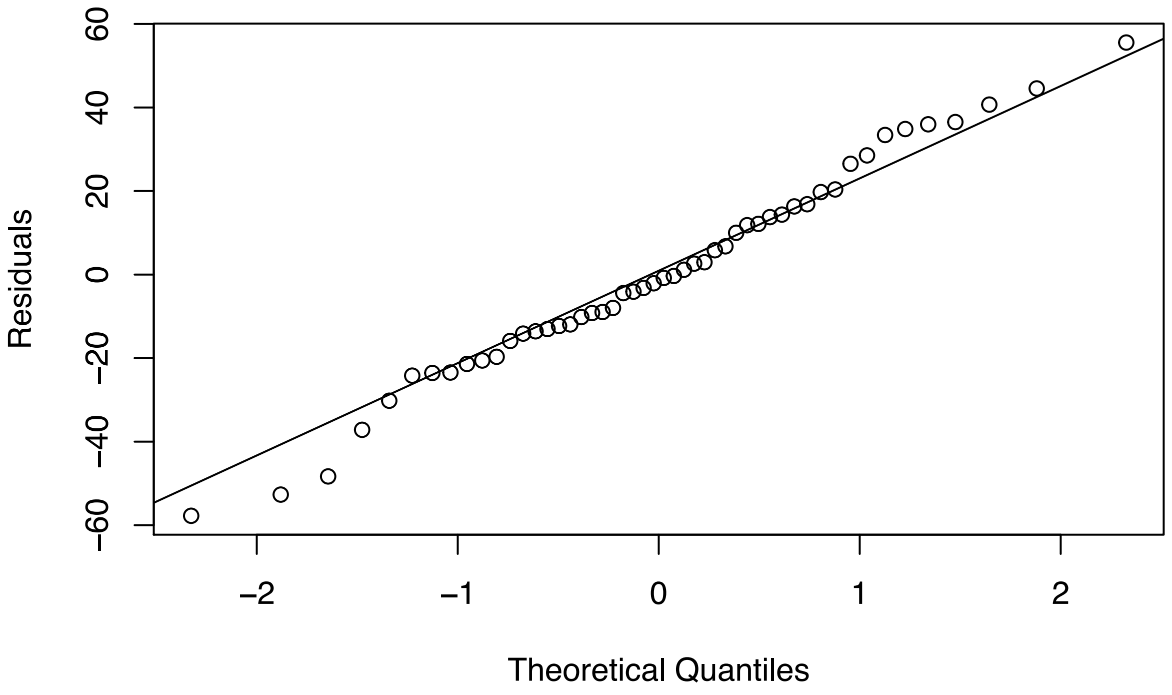
```r
par(mfrow = c(2,3))
plot(y = lm.out.sat1$residuals, x = lm.out.sat1$fitted.values)
abline(h=0)
plot(y = lm.out.sat1$residuals, x = df_sat$expend)
abline(h=0)
plot(y = lm.out.sat1$residuals, x = df_sat$salary)
abline(h=0)
plot(y = lm.out.sat1$residuals, x = df_sat$ratio)
abline(h=0)
plot(y = lm.out.sat1$residuals, x = df_sat$takers)
abline(h=0)
```



b)

Check the normality assumption

```r
qqnorm(lm.out.sat1$residuals, ylab = "Residuals", main = "")
qqline(lm.out.sat1$residuals)
```



```
shapiro.test(lm.out.sat1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lm.out.sat1$residuals
## W = 0.98809, p-value = 0.8918
```

The normal Q-Q plot of the residuals of the original linear model looks quite good. There is a bit of deviation from the line on each tail due to short-tailed errors, but it is not very serious. Formally, after conducting a Shapiro-Wilk test for the null hypothesis that the residuals are normal, we get a large p-value of `0.8918` so we fail to reject this null hypothesis. There is not enough evidence to conclude that the residuals are not normally distributed. This is consistent with the conclusions we drew from the Q-Q plot alone. We can proceed with having satisfied the normality assumptions.

c)

Check for large leverage points

```
hatv <- hatvalues(lm.out.sat1)
head(hatv) # first 6 points' hat values
```

```
##    Alabama     Alaska    Arizona   Arkansas California   Colorado
## 0.09802278 0.16906517 0.05815975 0.06583836 0.28287373 0.03336440
```

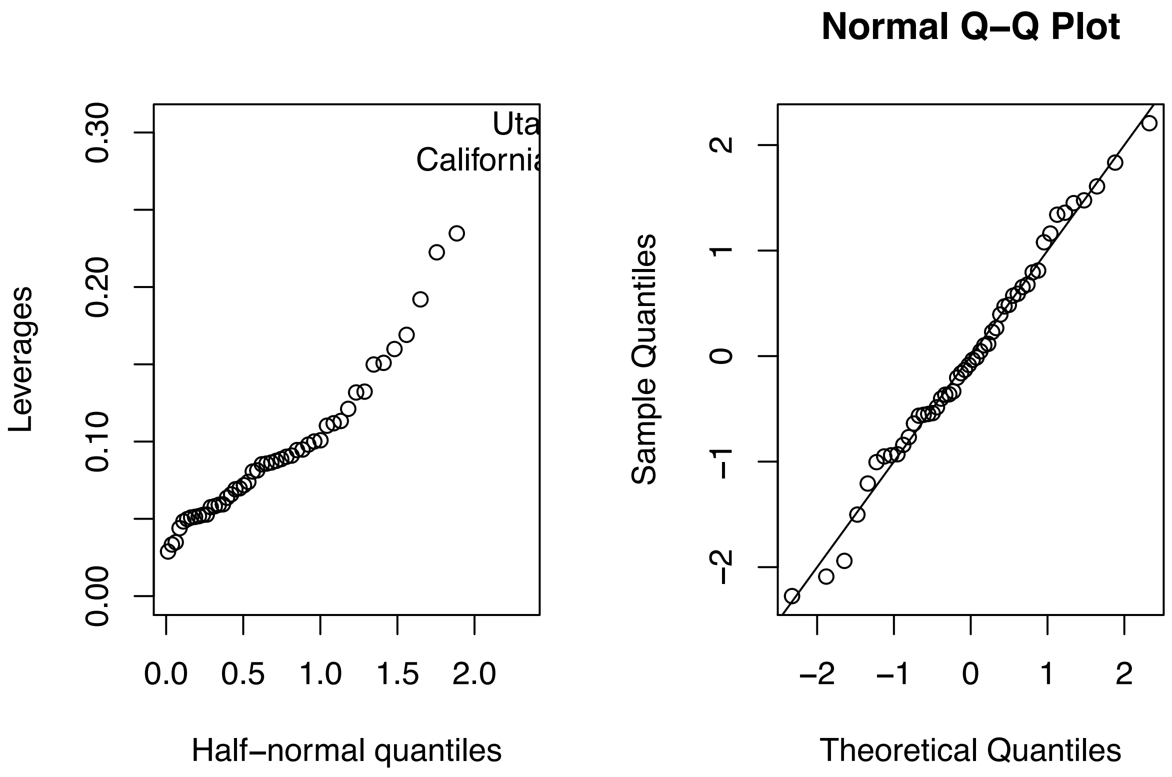
```
sum(hatv) # sum of hat values = # parameters (good)
```

```
## [1] 5
```

We will use a half-normal plot to identify unusually large values of the leverage. We plot the leverages vs. half-normal quantiles as well as a normal Q-Q plot for the standardized residuals below. In the half-normal plot, the two highest leverage points are labeled; they are California and Utah in this data set. They are large leverage values as they diverge most heavily from the rest of the data.

In the Q-Q plot of the standardized residuals, we see that all the points follow the `y=x` line quite closely. This plot looks very similar, if not better than, the previous Q-Q plot of the original residuals (not standardized). This suggests that the large leverage points we identify in the half-normal plot are not affecting the shape of the plot very much.

```r
par(mfrow = c(1,2))
states <- row.names(df_sat)
halfnorm(hatv, labs = states, ylab = "Leverages")
qqnorm(rstandard(lm.out.sat1))
abline(0,1)
```



d)

To check if the apparently large leverage points we previously saw are true statistical outliers, we will examine the studentized residuals from leave-one-out regressions for each data point and compare them to the α/n quantile value for the Bonferroni correction.

```r
stud <- rstudent(lm.out.sat1)
stud[which.max(abs(stud))]
```

```
## West Virginia
##     -2.390615
```

```r
# sort(abs(stud), decreasing = TRUE)
```

The largest residual of -2.39 is associated with West Virginia. On a standard normal scale, this is fairly large since it is nearly a critical score of 3, but now we will formally test if this point is an outlier at a level of $\alpha = 0.05$:

```r
nn <- nrow(df_sat)
p <- 5
```

```
dof <- nn - p - 1
qt(0.05 / nn, 44)
```

```
## [1] -3.286072
```

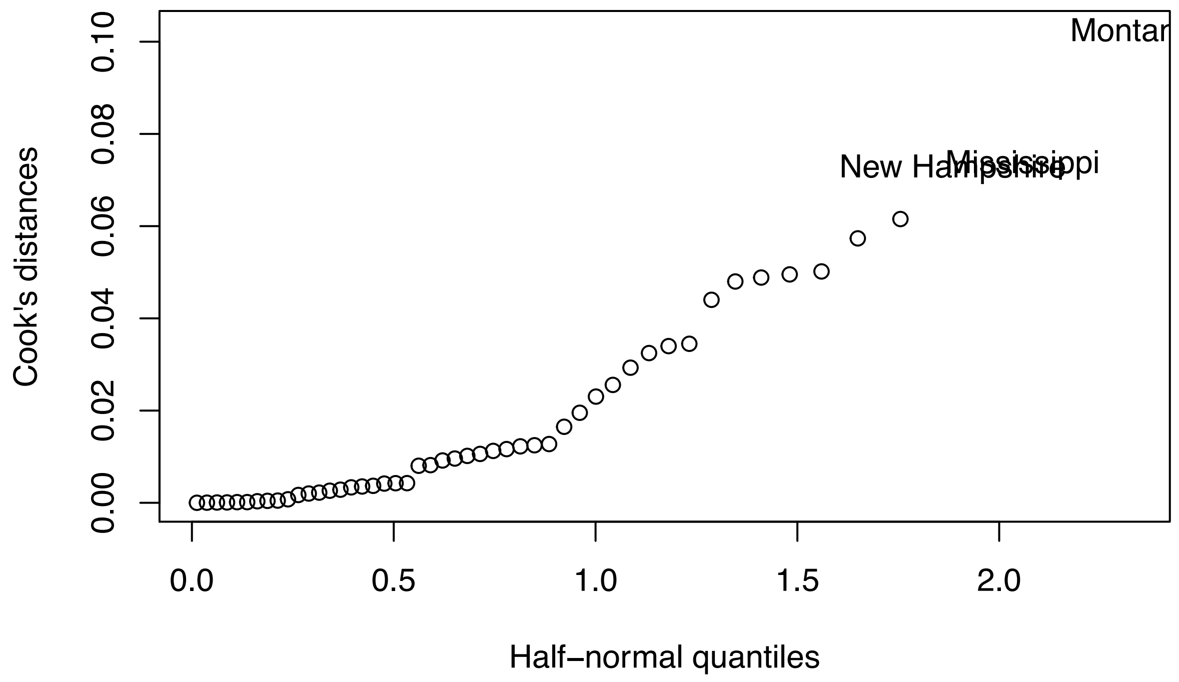
Since the studentized residual for West Virginia (-2.39) is less than the critical value of -3.28, we conclude that the point is not an outlier. Since this point held the largest magnitude studentized residual and it still isn't an outlier, we can assume none of the other points are outliers either.

e)

Check for influential points

We will use Cook statistics to check if any of our data points are influential, meaning that removing them from the dataset would significantly change the model fit. These points are not necessarily outliers.

```
cook <- cooks.distance(lm.out.sat1)
halfnorm(cook, nlab = 3, labs = states, ylab = "Cook's distances")
```



We calculate the Cook statistics for the data and plot them on a half-normal plot again. Note that the largest value is associated with Montana and is quite far from the other points. We will now refit the model without this data point and see how the estimates are affected:

```
lm.out.sat2 <- lm(total ~ expend + salary + ratio + log(takers),
                  data = df_sat, subset = (cook < max(cook)))

sumary(lm.out.sat2)
```

```
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 1148.56495   41.29654  27.8126   <2e-16
## expend         0.37146    8.55586   0.0434   0.9656
## salary         2.85957    1.95433   1.4632   0.1505
## ratio         -1.79573    2.51091  -0.7152   0.4783
## log(takers)  -80.82527    4.72413 -17.1090   <2e-16
```

```
##
## n = 49, p = 5, Residual SE = 25.37445, R-Squared = 0.9
```

```
sumary(lm.out.sat1)
```

```
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 1144.32105   42.38907  26.9957   <2e-16
## expend         5.11816    8.40580   0.6089   0.5457
## salary         1.62570    1.89300   0.8588   0.3950
## ratio         -0.79339    2.52265  -0.3145   0.7546
## log(takers)  -79.77452    4.82242 -16.5424   <2e-16
##
## n = 50, p = 5, Residual SE = 26.08454, R-Squared = 0.89
```

The most notable change when leaving out the Montana data point is the coefficient on `expend` reducing from 5.12 to 0.37 (about 14x smaller now) along with the coefficients for `salary` and `ratio` each increasing by 1.

Working with the original model using all data points, we are interested in the change in each beta coefficient when a case is left out. For each coefficient, we calculate this change for each case and output the cases for which change is highest for each coefficient.

```
head(sort(abs(dfbeta(lm.out.sat)[,2]), decreasing = TRUE)) # Utah
```

```
##          Utah West Virginia       Florida   Connecticut       Wyoming
##      5.405334      2.896706      2.660732      2.552167      2.249555
##        Alaska
##      2.149511
```

```
head(sort(abs(dfbeta(lm.out.sat)[,3]), decreasing = TRUE)) # none
```

```
##          Utah   Connecticut       Florida  North Dakota West Virginia
##     1.4585122     0.8560963     0.6935990     0.5562298     0.5503854
##       Wyoming
##     0.4278283
```

```
head(sort(abs(dfbeta(lm.out.sat)[,4]), decreasing = TRUE)) # Utah
```

```
##          Utah       Florida   Connecticut        Oregon    Washington
##     4.0149120     0.9843776     0.9467891     0.8257784     0.5606705
## Massachusetts
##     0.5500516
```

```
head(sort(abs(dfbeta(lm.out.sat)[,5]), decreasing = TRUE)) # none
```

```
##  New Hampshire South Carolina  West Virginia      Minnesota           Iowa
##     0.11007425     0.08234351     0.06791298     0.06327397     0.06127127
##  Massachusetts
##     0.05739453
```

Based on this, we would be most interested in examining the effects of removing the Utah data point from the model.

```
lm.out.sat3 <- lm(total ~ expend + salary + ratio + log(takers),
                  data = df_sat, subset = (states != "Utah"))
sumary(lm.out.sat3)
```

```
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 1141.1474    45.6325  25.0074   <2e-16
## expend         5.4620     8.6654   0.6303   0.5317
## salary         1.5511     1.9488   0.7959   0.4304
## ratio          -0.5362     2.8496  -0.1882   0.8516
## log(takers)   -79.9219     4.9288 -16.2151   <2e-16
##
## n = 49, p = 5, Residual SE = 26.36704, R-Squared = 0.88
```

Refitting the model without the Utah data point actually has very little affect on the estimators than expected. The point is not a very influential point after all.

f)

Check the structure of the relationship between the predictors and the response

To check the structure of the relationship between the predictors and response, it is most helpful to construct a partial regression plot for each predictor by regressing the response on all covariates except one, in a "leave-on-out"-like manner. We will compare this to regressing the left-out predictor on the other predictors in order to understand what happens when each predictor's effect is removed.

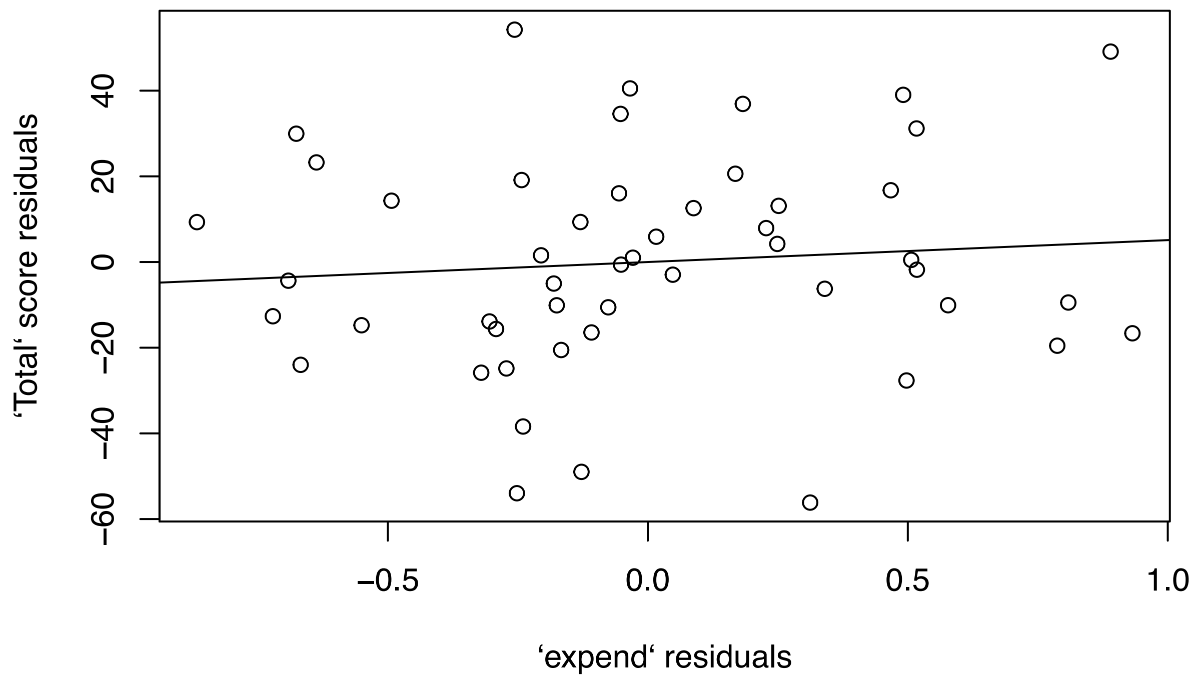
```r
d <- residuals(lm(total ~ log(takers) + salary + ratio, data = df_sat))
m <- residuals(lm(expend ~ log(takers) + salary + ratio, data = df_sat))
plot(m, d, xlab = "`expend` residuals", ylab = "`Total` score residuals")

coef(lm(d~m))
```

```
##   (Intercept)            m
## 1.004859e-15 5.118164e+00
```

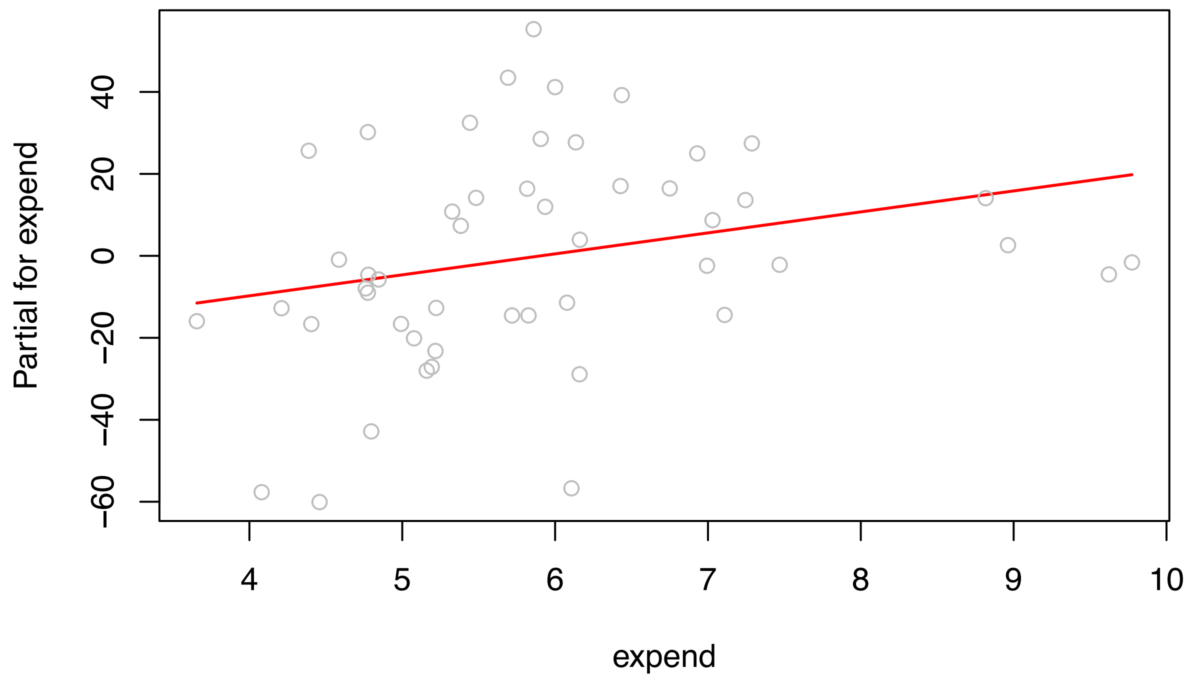
```r
coef(lm.out.sat1)
```

```
##   (Intercept)       expend       salary        ratio  log(takers)
## 1144.3210467    5.1181644    1.6256969   -0.7933901  -79.7745231
```

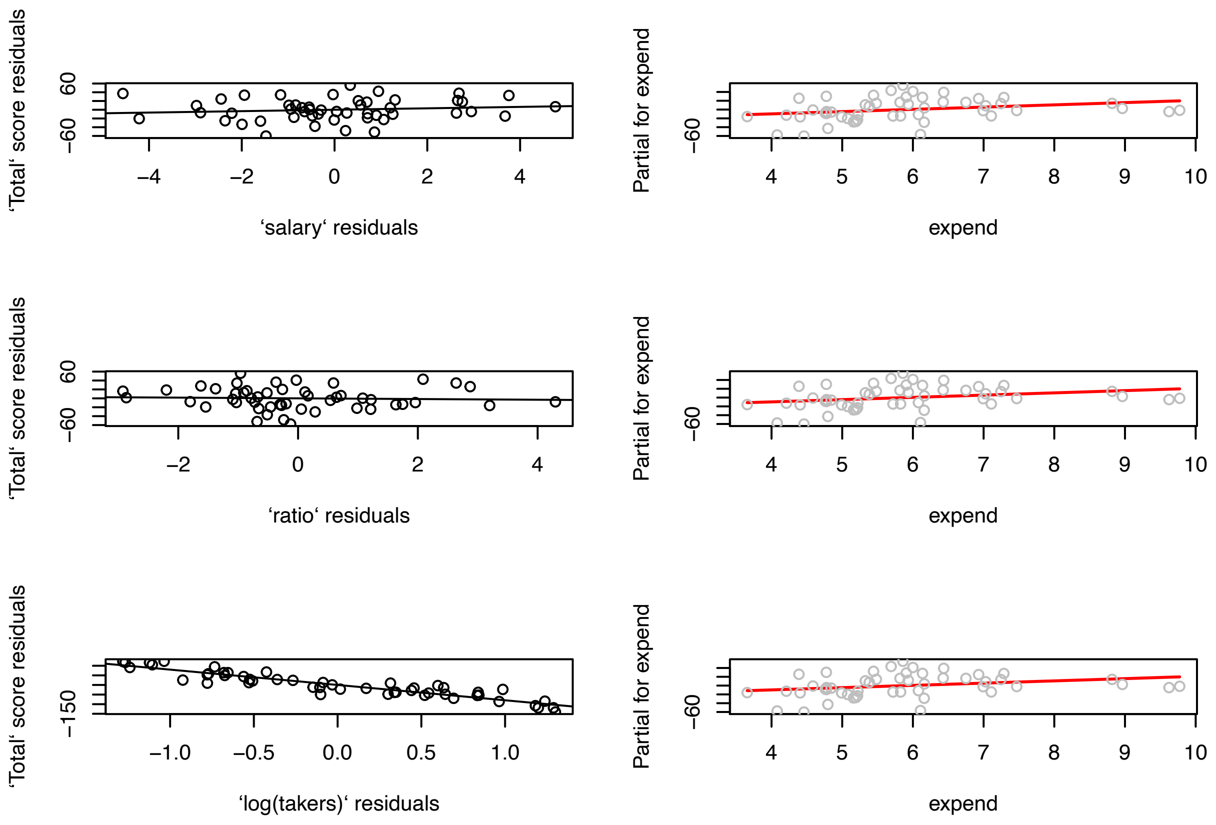
```r
abline(0, coef(lm.out.sat1)['expend'])
```



```r
termplot(lm.out.sat1, partial.resid = TRUE, terms = 1)
```



Note that the regression line for the plot gives the same coefficient for `expend` as the original model. In the partial residuals plot, we do not see any unusual or interesting structural trends like clusters, etc.



When we repeat this process for the other three covariates, we still do not come across anything unusual so we can proceed assuming a linearly structured relationship between the predictors and the response.

g)

Compare the coefficient on the `expend` covariate in the full model regressing on `expend`, `salary`, `ratio`, and `takers`, against the coefficient if you regress `total` on `expend` only

```r
lm.out.sat$coefficients['expend']
```

```
##   expend
## 4.462594
```

```r
lm.out.satnew <- lm(total ~ expend, data = df_sat)
lm.out.satnew$coefficients['expend']
```

```
##    expend
## -20.89217
```

If we regress `total` on `expend` only, and not on the other covariates as before, we see that the coefficient on `expend` is highly negative compared to previously being positive in the original model. For every 1 unit increase in school expenditure, total SAT score goes down by 20 units in this simplest model. This does not make sense! But the reason we observe this phenomenon is that we are not accounting for other important factors that contribute to the state's average SAT score but are confounding this model. When we do control for (i.e. keep constant) student/teacher `ratio`, average teacher `salary`, and percentage of students taking the SAT (`takers`), we see that a one unit increase in school expenditure is now associated with a 4.5 unit increase in SAT score. This is much more sensible.

3.

```r
library(MASS)
df_fgl <- data.frame(fgl)
```

a)

```r
lm.out.fgl <- lm(Al ~ RI + Na + Si + Ba, data = df_fgl)

sumary(lm.out.fgl)
```

```
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 15.746403   3.031944  5.1935 4.885e-07
## RI          -0.096435   0.010378 -9.2923 < 2.2e-16
## Na          -0.083879   0.034264 -2.4480   0.01519
## Si          -0.182085   0.040084 -4.5426 9.387e-06
## Ba           0.497171   0.054392  9.1405 < 2.2e-16
##
## n = 214, p = 5, Residual SE = 0.37163, R-Squared = 0.46
```

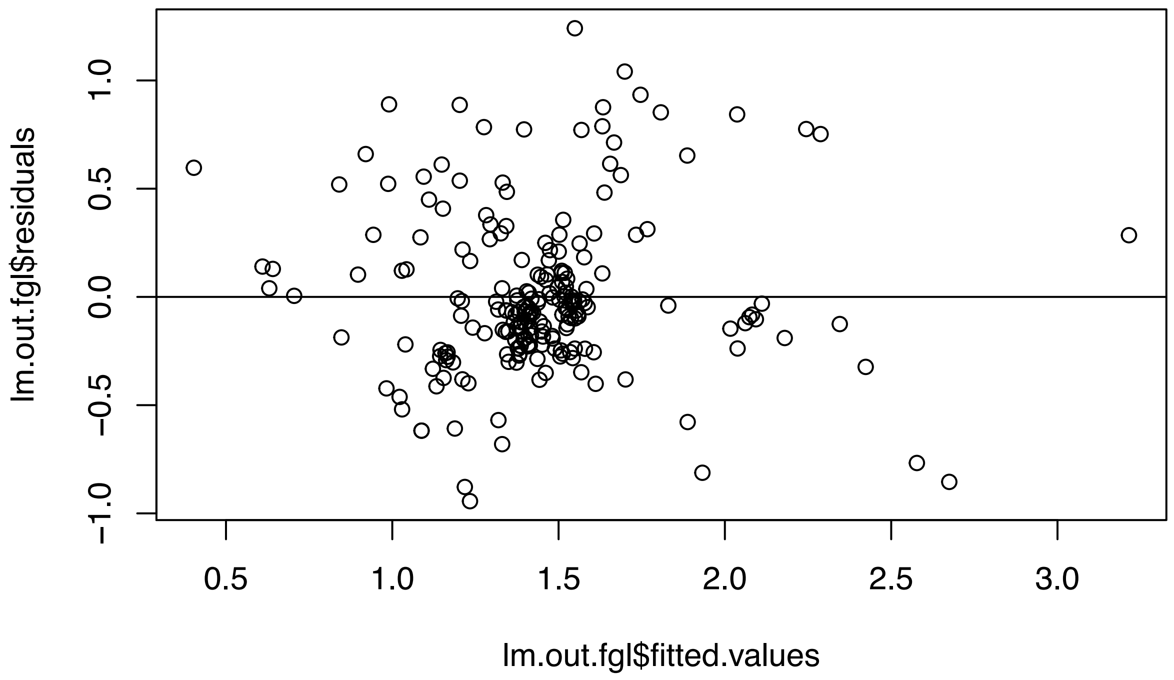
We ran a linear regression for the following model:

$$\hat{AL} = \hat{\beta}_0 + \hat{\beta}_1 RI + \hat{\beta}_2 Na + \hat{\beta_3} Si + \hat{\beta}_4 Ba$$

From the output, we see that at the 5% alpha level, all four of the regression coefficients as well as the intercept term estimates appear significant. Of these four, the coefficient on `Na` is has the largest p-value around `0.0152`, while the other coefficients all have p-values extremely close to `0`. This gives us reason to be suspicious about the coefficient for Na, depending on how well the diagnostics and assumptions hold up.

Plot of Residuals vs. Fitted Values:

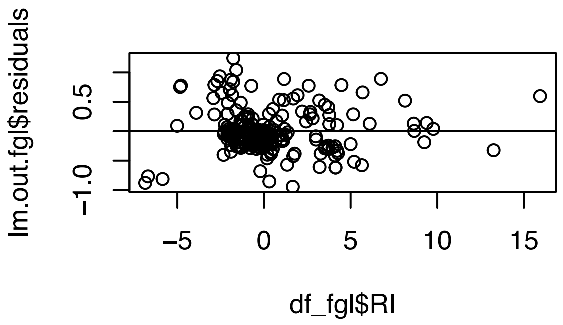
```
plot(y = lm.out.fgl$residuals, x = lm.out.fgl$fitted.values)
abline(h=0)
```

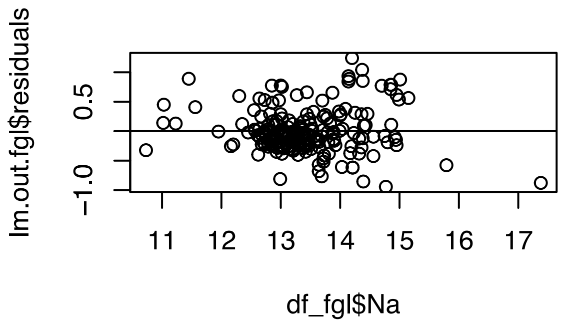


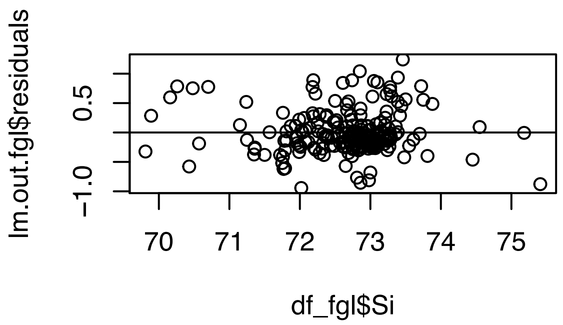
The plot of residuals vs. fitted values appears to be nonlinear with no clear trend in the residuals and centered around 0. The plot is is relatively homoskedastic, but we do see some residuals at the tails (far left and far right) that could be considered influential points/outliers, so we need to be cautious of these points as we proceed.

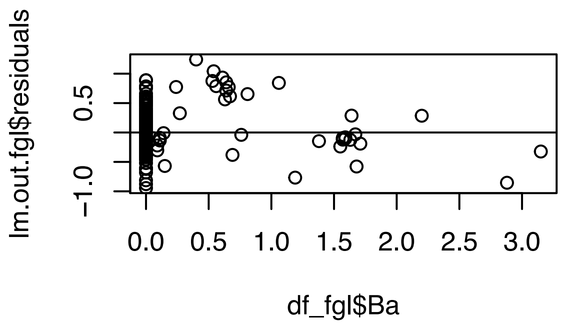
Plot of Residuals vs. each Covariate

```
par(mfrow = c(2,2))

plot(y = lm.out.fgl$residuals, x = df_fgl$RI)
abline(h=0)
plot(y = lm.out.fgl$residuals, x = df_fgl$Na)
abline(h=0)
plot(y = lm.out.fgl$residuals, x = df_fgl$Si)
abline(h=0)
plot(y = lm.out.fgl$residuals, x = df_fgl$Ba)
abline(h=0)
```







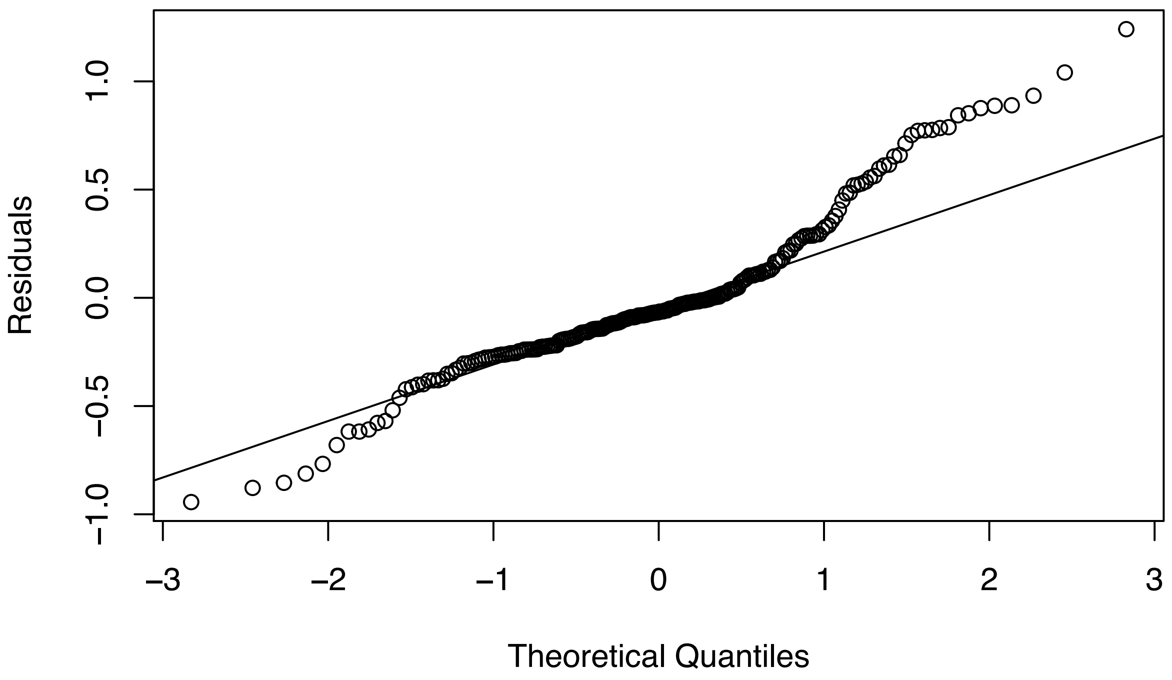


For the most part, the plots of the residuals vs. each covariate in our model look pretty homoskedastic, centered around 0, and with no obvious trends. We note that the residuals vs. X plot for Ba looks atypical because there are a lot of points for which the Ba value equals 0. Nevertheless, the residuals for these points are appear normally distributed around 0 which is a good sign. However, for those nonzero Ba points, there appears to be some heteroskedasticity and a slight negatively linear trend in these residuals.

b)

Now we will check using both a normal Q-Q plot and the Shapiro-Wilk test to see if the residuals are normally distributed which is a key assumption in using a linear regression model.

```r
qqnorm(lm.out.fgl$residuals, ylab = "Residuals", main = "")
qqline(lm.out.fgl$residuals)
```



In the above Q-Q plot, we see that the tails of the residual distribution are very far from the normal line. It is clear that there is some non-normality in the residuals and thus a violation of the normality assumption due to long-tailed errors.

```
shapiro.test(lm.out.fgl$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lm.out.fgl$residuals
## W = 0.9433, p-value = 2.033e-07
```

Formally, after conducting a Shapiro-Wilk test for the null hypothesis that the residuals are normal, we get a very small p-value of `2.044e-07` so the null hypothesis can definitely be rejected. Specifically, there is sufficient evidence to conclude that the residuals are not normally distributed. This validates the conclusions we drew from the Q-Q plot alone.

c)

Bootstrap the Residual:

```
set.seed(819)
X <- cbind(1, df_fgl$RI, df_fgl$Na, df_fgl$Si, df_fgl$Ba)
all_betas_res <- matrix(ncol = 5)
all_beta_ses_res <- matrix(ncol = 5)
colnames(all_betas_res) <- c("Intercept", "RI", "Na", "Si", "Ba")
colnames(all_beta_ses_res) <- c("Intercept", "RI", "Na", "Si", "Ba")
B <- 1000 # 1000 trials
n <- nrow(df_fgl)
for (b in 1:B){
  # first work with original model lm.out.fgl with original data df_fgl
  Y_boot <- X %*% lm.out.fgl$coefficients +
```

```r
    lm.out.fgl$residuals[sample(x = 1:n, size = n, replace = TRUE)]
  model_boot_res <- lm(Y_boot ~ df_fgl$RI + df_fgl$Na + df_fgl$Si + df_fgl$Ba)
  if (b == 1){
    all_betas_res <- rbind(model_boot_res$coefficients)
    all_beta_ses_res <- rbind(summary(model_boot_res)$coefficients[,2])
  }
  else {
    all_betas_res <- rbind(all_betas_res, model_boot_res$coefficients)
    all_beta_ses_res <- rbind(all_beta_ses_res, summary(model_boot_res)$coefficients[,2])
  }
}
ff <- function(x){return (round(sqrt(var(x)), 4))}
ff2 <- function(x){return (round(median(x), 4))}
Empirical_SDs_BRes <- apply(all_betas_res, 2, FUN = ff)
Median_model_estimate_of_SE_BRes <- apply(all_beta_ses_res, 2, FUN = ff2)
```

Bootstrap the Sample:

```r
set.seed(819)
all_betas_sam <- matrix(ncol = 5)
all_beta_ses_sam <- matrix(ncol = 5)
colnames(all_betas_sam) <- c("Intercept", "RI", "Na", "Si", "Ba")
colnames(all_beta_ses_sam) <- c("Intercept", "RI", "Na", "Si", "Ba")

B <- 1000 # 1000 trials
n <- nrow(df_fgl)
for (bb in 1:B){
  s <- sample(x = 1:n, size = n, replace = TRUE)
  model_boot <- lm(Al[s] ~ RI[s] + Na[s] + Si[s] + Ba[s], data = df_fgl)
  if (bb == 1){
    all_betas_sam <- rbind(model_boot$coefficients)
    all_beta_ses_sam <- rbind(summary(model_boot)$coefficients[,2])
  }
  else {
    all_betas_sam <- rbind(all_betas_sam, model_boot$coefficients)
    all_beta_ses_sam <- rbind(all_beta_ses_sam, summary(model_boot)$coefficients[,2])
  }
}
Empirical_SDs_BSam <- apply(all_betas_sam, 2, FUN = ff)
Median_model_estimate_of_SE_BSam <- apply(all_beta_ses_sam, 2, FUN = ff2)
Original_linear_model_SE_Betas <- summary(lm.out.fgl)$coefficients[,2]
```

Discussion

We are interested in comparing the following quantities resulting from bootstrapping the sample and bootstrapping the residuals:

```r
Empirical_SDs_BRes
```

```
## (Intercept)   df_fgl$RI   df_fgl$Na   df_fgl$Si   df_fgl$Ba
##      2.9965      0.0105      0.0333      0.0397      0.0513
```

```
Empirical_SDs_BSam
```

```
## (Intercept)       RI[s]       Na[s]       Si[s]       Ba[s]
##      5.1187      0.0141      0.0568      0.0640      0.0810
```

```
Original_linear_model_SE_Betas
```

```
## (Intercept)          RI          Na          Si          Ba
##  3.03194356  0.01037801  0.03426400  0.04008384  0.05439194
```

```
Median_model_estimate_of_SE_BRes
```

```
## (Intercept)   df_fgl$RI   df_fgl$Na   df_fgl$Si   df_fgl$Ba
##      2.9916      0.0102      0.0338      0.0395      0.0537
```

```
Median_model_estimate_of_SE_BSam
```

```
## (Intercept)       RI[s]       Na[s]       Si[s]       Ba[s]
##      3.0717      0.0104      0.0346      0.0405      0.0554
```

Overall, we compare the empirical SD of the beta estimates from bootstrapping the sample, bootstrapping the residual, From the original linear model without bootstrapping, we see that the bootstrapping residuals method produces empirical SDs for all covariates that are very similar to the values we expect based on the model assumptions (median model estimate of SE for residual bootstrap). These values are nearly identical to the standard errors of the beta estimates from the original linear model. However, when we bootstrap the sample, we get higher empirical standard deviations (i.e. we see more variability) compared to the median model estimate of the standard errors. The actual variability in our estimates from bootstrapping the sample is larger than what we expect.

In this case, and specifically for the `Na` covariate, the diagnostics we performed earlier suggested that we cannot assume a normality of the residuals. So we cannot trust the bootstrapping residuals method since it inherently assumes the linear model is true and its assumptions are valid. But here it is not. So based on the bootstrapping the sample method, since the standard errors are quite different for `Na`'s beta estimate, we CANNOT trust the p-value from the original fitted model because if we were to recalculate the t-value (-0.08388 / 0.0568 = -1.476761) and p-value (0.14145) using the empirical SD from bootstrapping the sample, the coefficient is no longer be significant.

4.

a)

```r
set.seed(819)
beta2_hats <- c()
for (ii in 1:1000){
  n <- 100
  beta0 <- 1; beta1 <- 1; beta2 <- 0
  X1 <- rnorm(n)
  X2 <- 0.9 * X1 + sqrt(1 - 0.9^2) * rnorm(n)
  Y <- beta0 + beta1*X1 + beta2*X2 + rnorm(n)
  lm.out.4a <- lm(Y ~ X1 + X2)
  beta2_hat <- lm.out.4a$coefficients[3] # coef on X2
  beta2_hats <- c(beta2_hats, beta2_hat)
}

get_sd <- function(x){
  return (round(sqrt(var(x)), 4))
}

(empirical_sd <- get_sd(beta2_hats))
```

```
## [1] 0.2327
```

b)

```
set.seed(819)
n <- 100
beta0 <- 1; beta1 <- 1; beta2 <- 0
X1 <- rnorm(n)
X2 <- 0.9 * X1 + sqrt(1 - 0.9^2) * rnorm(n)
Y <- beta0 + beta1*X1 + beta2*X2 + rnorm(n)
lm.out.4b <- lm(Y ~ X1 + X2)
beta2_hat_se <- summary(lm.out.4b)$coefficients[3,2] # se of coef on X2

beta2_hat_se
```

```
## [1] 0.2484447
```

```
empirical_sd
```

```
## [1] 0.2327
```

Yes, the standard error of $\hat{\beta}_2$ from the linear model of one simulation is approximately close to the empirical estimate of the true standard deviation of $\hat{\beta}_2$ from 1000 simulations. This is because the normality assumption about the error terms and therefore about the response holds true in both attempts. Also, the assumption that the true model is linear is clearly held, since that is where the Yi's are generated from.

c)

```
set.seed(819)
n <- 100
beta0 <- 1; beta1 <- 1; beta2 <- 0
X1 <- rnorm(n)
X2 <- 0.9 * X1 + sqrt(1 - 0.9^2) * rnorm(n)
Y <- beta0 + beta1*X1 + beta2*X2 + rnorm(n)


beta2_hats_permtest <- c()
for (xx in 1:1000){
  lm.out.permtest <- lm(Y ~ X1 + sample(X2))
  this_beta2hat <- lm.out.permtest$coefficients[3]
  beta2_hats_permtest <- c(beta2_hats_permtest, this_beta2hat)
}

(empirical_sd_permtest <- get_sd(beta2_hats_permtest))
```

```
## [1] 0.1099
```

```
empirical_sd
```

```
## [1] 0.2327
```

Now, when we simulate the data once at the beginning and then run a permutation test with 1000 permutations of X2 (Y, X1 constant), we get a much smaller empirical standard deviation than we did in part a (1000 full data simulations, 1000 models) and part b (1 data simulation, 1 model). The standard deviation we get now is about half of what it was before (0.11 vs 0.23). In part a, we had 1000 different data sets with 1000 different X1s, X2s, and Ys. Here, we have only 1 dataset of Y and X1 with 1000 different X2s. Since X1 and X2 are highly correlated, permuting X2 1000 times generates newer/more unpredictable X1,X2 pairs that are not necessarily correlated, and since X2 contains a `sqrt(1-0.9^2) = 0.436` ≈ 0.5 ‘ multiplier on its error, the standard deviation of estimates after the permutation tests is cut by a factor of half.