Homework 6

Alyssa Vanderbeek (amv2187)

3 December 2018

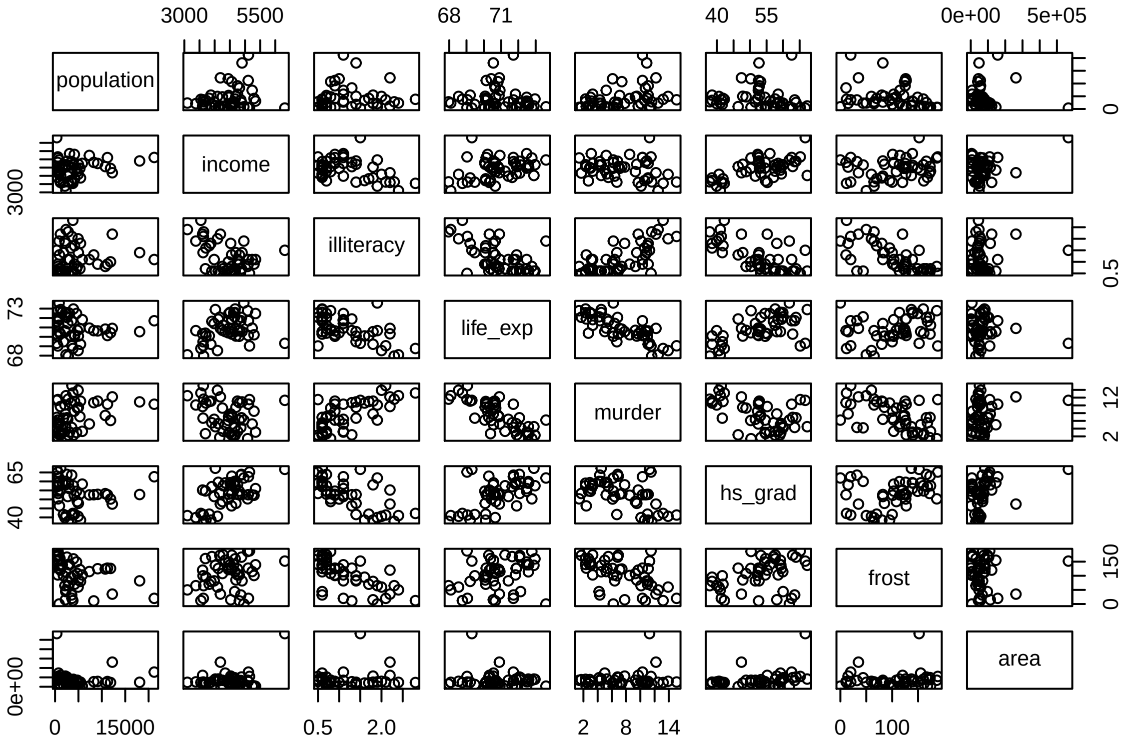
```
states = state.x77 %>% # load data from faraway package
  as.data.frame() %>%
  janitor::clean_names()
```

1. Explore the dataset and generate appropriate descriptive statistics and relevant graphs for all variables of interest (continuous and categorical) – no test required. Be selective! Even if you create 20 plots, you don't want to show them all.

```
# table of summary stats
states %>%
  skimr::skim_to_list() %>%
  as.data.frame %>%
  dplyr::select(1, 2, 5:11) %>%
  `colnames<-`(c(' ', 'NA', 'Mean', 'Std. Dev.', 'Min', '1st Q', 'Median', '3rd Q', 'Max')) %>%
  knitr::kable()
```

	NA	Mean	Std. Dev.	Min	1st Q	Median	3rd Q	Max
area	0	70735.88	85327.3	1049	36985.25	54277	81162.5	566432
frost	0	104.46	51.98	0	66.25	114.5	139.75	188
hs_grad	0	53.11	8.08	37.8	48.05	53.25	59.15	67.3
illiteracy	0	1.17	0.61	0.5	0.62	0.95	1.58	2.8
income	0	4435.8	614.47	3098	3992.75	4519	4813.5	6315
life_exp	0	70.88	1.34	67.96	70.12	70.67	71.89	73.6
murder	0	7.38	3.69	1.4	4.35	6.85	10.67	15.1
population	0	4246.42	4464.49	365	1079.5	2838.5	4968.5	21198

```
# scatterplot to assess correlation between vars
states %>%
  pairs
```




```r
# correlation matrix to evaluate what is seen in scatterplots
states %>%
  cor
```

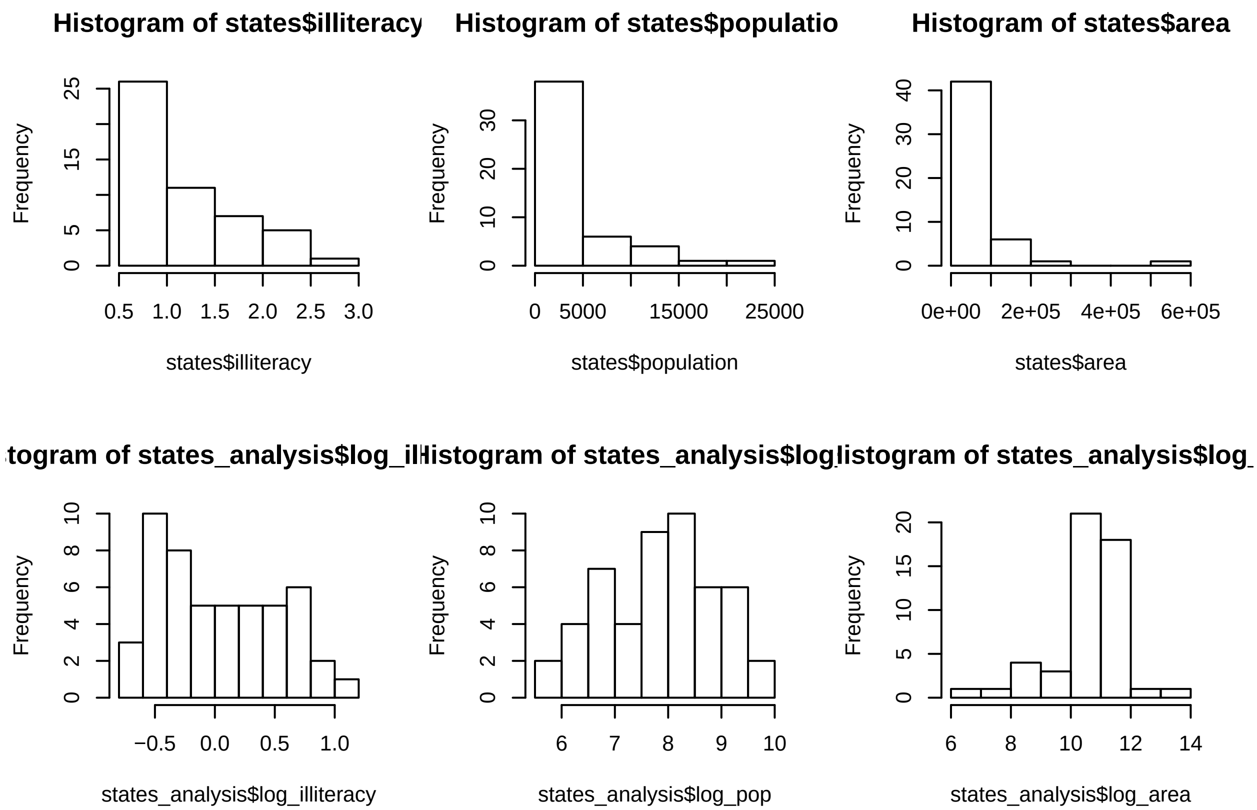
```
##              population     income  illiteracy    life_exp     murder
## population   1.00000000  0.2082276  0.10762237 -0.06805195  0.3436428
## income       0.20822756  1.0000000 -0.43707519  0.34025534 -0.2300776
## illiteracy   0.10762237 -0.4370752  1.00000000 -0.58847793  0.7029752
## life_exp    -0.06805195  0.3402553 -0.58847793  1.00000000 -0.7808458
## murder       0.34364275 -0.2300776  0.70297520 -0.78084575  1.0000000
## hs_grad     -0.09848975  0.6199323 -0.65718861  0.58221620 -0.4879710
## frost       -0.33215245  0.2262822 -0.67194697  0.26206801 -0.5388834
## area         0.02254384  0.3633154  0.07726113 -0.10733194  0.2283902
##                 hs_grad     frost        area
## population  -0.09848975 -0.3321525  0.02254384
## income       0.61993232  0.2262822  0.36331544
## illiteracy  -0.65718861 -0.6719470  0.07726113
## life_exp     0.58221620  0.2620680 -0.10733194
## murder      -0.48797102 -0.5388834  0.22839021
## hs_grad      1.00000000  0.3667797  0.33354187
## frost        0.36677970  1.0000000  0.05922910
## area         0.33354187  0.0592291  1.00000000
```

It looks like murder is correlated both with life expectancy and illiteracy, suggesting that it is a potential confounder. Specifically, murder is positively associated with illiteracy (higher murder rate = higher illiteracy rate) and negatively associated with life expectancy (higher murder rate = lower life expectancy).

After examining the distribution of each vriable in the dataset, I chose to perform a log transformation on the estimates for area size, illiteracy rate, and population size, which were all skewed.

```r
states_analysis = states %>%
  mutate(log_area = log(area),
```

```
         log_illiteracy = log(illiteracy),
         log_pop = log(population)) %>%
  dplyr::select(-area, -population, -illiteracy)

par(mfrow = c(2, 3))
hist(states$illiteracy)
hist(states$population)
hist(states$area)
hist(states_analysis$log_illiteracy)
hist(states_analysis$log_pop)
hist(states_analysis$log_area)
```



2. Use automatic procedures to find a 'best subset' of the full model. Present the results and comment on the following

```
## backwards elimination
# summary(lm(life_exp ~ ., data = states_analysis))
# summary(lm(life_exp ~ murder + hs_grad + frost + log_area + log_illiteracy + log_pop, data = states_a
# summary(lm(life_exp ~ murder + hs_grad + frost + log_illiteracy + log_pop, data = states_analysis))

b.fit = lm(life_exp ~ murder + hs_grad + frost + log_pop, data = states_analysis)
summary(b.fit)
```

```
##
## Call:
## lm(formula = life_exp ~ murder + hs_grad + frost + log_pop, data = states_analysis)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -1.41760 -0.43880  0.02539  0.52066  1.63048
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 68.720810   1.416828  48.503  < 2e-16 ***
## murder      -0.290016   0.035440  -8.183 1.87e-10 ***
## hs_grad      0.054550   0.014758   3.696 0.000591 ***
## frost       -0.005174   0.002482  -2.085 0.042779 *
## log_pop      0.246836   0.112539   2.193 0.033491 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7137 on 45 degrees of freedom
## Multiple R-squared:  0.7404, Adjusted R-squared:  0.7173
## F-statistic: 32.09 on 4 and 45 DF,  p-value: 1.17e-12
```

```
## forwards process
# summary(lm(life_exp ~ murder, data = states_analysis))
# summary(lm(life_exp ~ hs_grad, data = states_analysis))
# summary(lm(life_exp ~ frost, data = states_analysis))
# summary(lm(life_exp ~ log_area, data = states_analysis))
# summary(lm(life_exp ~ log_illiteracy, data = states_analysis))
# summary(lm(life_exp ~ log_pop, data = states_analysis))
#
# # murder has lowest p-val. Start adding secondary vars
# summary(lm(life_exp ~ murder + hs_grad, data = states_analysis))
# summary(lm(life_exp ~ murder + frost, data = states_analysis))
# summary(lm(life_exp ~ murder + log_area, data = states_analysis))
# summary(lm(life_exp ~ murder + log_illiteracy, data = states_analysis))
# summary(lm(life_exp ~ murder + log_pop, data = states_analysis))
#
# # murder + hs_grad
# summary(lm(life_exp ~ murder + hs_grad + frost, data = states_analysis))
# summary(lm(life_exp ~ murder + hs_grad + log_area, data = states_analysis))
# summary(lm(life_exp ~ murder + hs_grad + log_illiteracy, data = states_analysis))
# summary(lm(life_exp ~ murder + hs_grad + log_pop, data = states_analysis))
#
# # murder + hs_grad + log_pop
# summary(lm(life_exp ~ murder + hs_grad + log_pop + frost, data = states_analysis))
# summary(lm(life_exp ~ murder + hs_grad + log_pop + log_area, data = states_analysis))
# summary(lm(life_exp ~ murder + hs_grad + log_pop + log_illiteracy, data = states_analysis))
#
# # murder + hs_grad + log_pop + frost
# summary(lm(life_exp ~ murder + hs_grad + log_pop + frost + log_area, data = states_analysis))
# summary(lm(life_exp ~ murder + hs_grad + log_pop + frost + log_illiteracy, data = states_analysis))

f.fit = lm(life_exp ~ murder + hs_grad + log_pop + frost, data = states_analysis)
summary(f.fit)
```

```
##
## Call:
## lm(formula = life_exp ~ murder + hs_grad + log_pop + frost, data = states_analysis)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -1.41760 -0.43880  0.02539  0.52066  1.63048
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 68.720810   1.416828  48.503  < 2e-16 ***
## murder      -0.290016   0.035440  -8.183 1.87e-10 ***
## hs_grad      0.054550   0.014758   3.696 0.000591 ***
## log_pop      0.246836   0.112539   2.193 0.033491 *
## frost       -0.005174   0.002482  -2.085 0.042779 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7137 on 45 degrees of freedom
## Multiple R-squared:  0.7404, Adjusted R-squared:  0.7173
## F-statistic: 32.09 on 4 and 45 DF,  p-value: 1.17e-12
```

```
## Stepwise
step.fit = step(lm(life_exp ~ ., data = states_analysis))
```

```
## Start:  AIC=-23.71
## life_exp ~ income + murder + hs_grad + frost + log_area + log_illiteracy +
##     log_pop
##
##                  Df Sum of Sq    RSS     AIC
## - income          1    0.0002 22.596 -25.712
## - log_illiteracy  1    0.1079 22.704 -25.475
## - log_area        1    0.2368 22.833 -25.192
## <none>                        22.596 -23.713
## - frost           1    1.1645 23.760 -23.200
## - log_pop         1    2.0155 24.611 -21.441
## - hs_grad         1    2.4822 25.078 -20.502
## - murder          1   24.0347 46.631  10.512
##
## Step:  AIC=-25.71
## life_exp ~ murder + hs_grad + frost + log_area + log_illiteracy +
##     log_pop
##
##                  Df Sum of Sq    RSS      AIC
## - log_illiteracy  1    0.1095 22.705 -27.4708
## - log_area        1    0.2616 22.858 -27.1370
## <none>                        22.596 -25.7125
## - frost           1    1.2628 23.859 -24.9936
## - log_pop         1    2.3859 24.982 -22.6937
## - hs_grad         1    4.4112 27.007 -18.7959
## - murder          1   24.4834 47.079   8.9907
##
## Step:  AIC=-27.47
## life_exp ~ murder + hs_grad + frost + log_area + log_pop
##
##            Df Sum of Sq    RSS     AIC
## - log_area  1    0.2157 22.921 -28.998
## <none>                  22.705 -27.471
## - log_pop   1    2.2792 24.985 -24.688
## - frost     1    2.3760 25.082 -24.495
## - hs_grad   1    4.9491 27.655 -19.612
```

```
## - murder    1   29.2296 51.935  11.899
##
## Step:  AIC=-29
## life_exp ~ murder + hs_grad + frost + log_pop
##
##           Df Sum of Sq    RSS     AIC
## <none>                 22.921 -28.998
## - frost    1     2.214 25.135 -26.387
## - log_pop  1     2.450 25.372 -25.920
## - hs_grad  1     6.959 29.881 -17.741
## - murder   1    34.109 57.031  14.578
```

All automatic processes conclude the same model, using percent increase in population size (log(population)), rate of high school graduation (hs_grad), murder rate per 100,000 (murder), and average number of days annually with temperatures below freezing (frost) as predictors of life expectancy.

'Frost' is the least significant predictor, with a p-value of 0.043. No variables were seen to be a "close call" at the 5% significance level.

There is a correlation between illiteracy (with and without log transformation) and high school graduation rate (-0.6571886), but my model includes only high school graduation rate as a predictor.

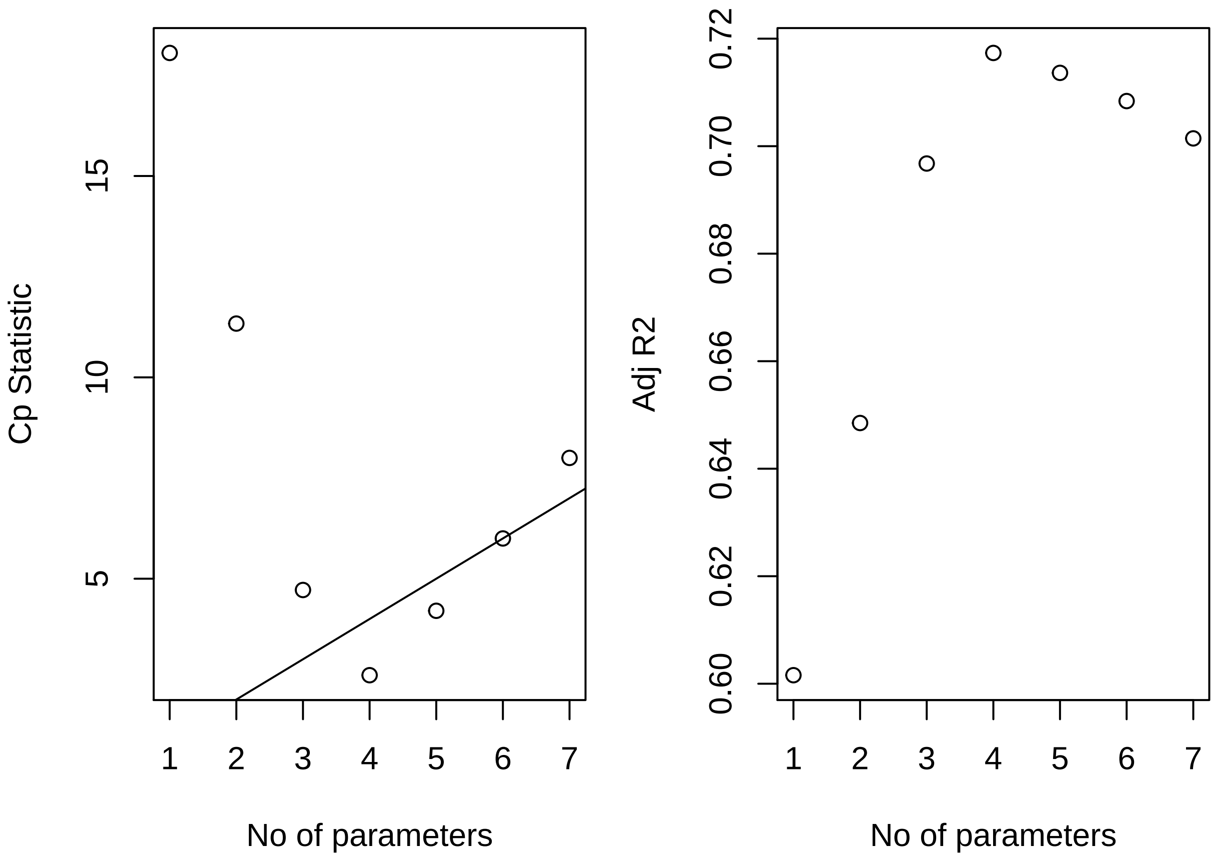
3. Use criterion-based procedures studied in class to guide your selection of the 'best subset'. Summarize your results (tabular or graphical)

```
best <- function(model, ...)
{
  subsets <- regsubsets(formula(model), model.frame(model), ...)
  subsets <- with(summary(subsets),
                  cbind(p = as.numeric(rownames(which)), which, rss, rsq, adjr2, cp, bic))

  return(subsets)
}

best(lm(life_exp ~ ., data = states_analysis)) %>%
  knitr::kable(., 'latex', caption = 'Criterion-based model building') %>%
  kableExtra::kable_styling(latex_options = c("hold_position")) %>%
  kableExtra::landscape()
```

Table 2: Criterion-based model building

p	(Intercept)	income	murder	hs_grad	frost	log_area	log_illiteracy	log_pop	rss	rsq	adjr2	cp	bic
1	1	0	1	0	0	0	0	0	34.46133	0.6097201	0.6015893	18.054999	-39.22051
2	1	0	1	1	0	0	0	0	29.77036	0.6628461	0.6484991	11.335656	-42.62472
3	1	0	1	1	0	0	0	1	25.13538	0.7153378	0.6967729	4.720403	-47.17452
4	1	0	1	1	1	0	0	1	22.92123	0.7404135	0.7173392	2.604837	-47.87315
5	1	0	1	1	1	1	0	1	22.70549	0.7428568	0.7136360	4.203829	-44.43397
6	1	0	1	1	1	1	1	1	22.59600	0.7440968	0.7083894	6.000318	-40.76364
7	1	1	1	1	1	1	1	1	22.59583	0.7440987	0.7014485	8.000000	-36.85199

```r
# leaps::leaps(x = states_analysis[, c(1, 3:8)], y = states_analysis$life_exp, nbest = 2, method = "Cp")

# leaps::leaps(x = states_analysis[, c(1, 3:8)], y = states_analysis$life_exp, nbest = 2, method = "adj

# Summary of models for each size (one model per size)
b = leaps::regsubsets(life_exp ~ ., data = states_analysis)
rs = summary(b)

# Plots of Cp and Adj-R2 as functions of parameters
par(mar = c(4, 4, 1, 1))
par(mfrow = c(1, 2))

plot(1:7, rs$cp, xlab = "No of parameters", ylab = "Cp Statistic")
abline(0, 1)
plot(1:7, rs$adjr2, xlab = "No of parameters", ylab = "Adj R2")
```



According to the Cp statistics and Adjusted R^2, the ideal number of parameters is 4; as seen in the table above, those parameters are murder, hs_grad, frost, and log_pop.

4. Compare the two 'subsets' from parts 2 and 3 and recommend a 'final' model. Using this 'final' model do the following. a) Identify any leverage and/or influential points and take appropriate measures. b) Check the model assumptions.

All analyses above recommend the same model using percent increase in population size (log(population)), rate of high school graduation (hs_grad), murder rate per 100,000 (murder), and average number of days annually with temperatures below freezing (frost) as predictors of life expectancy.

```r
life_exp_fit = b.fit

# rstandard function gives the INTERNALLY studentized residuals

stu_res = rstandard(life_exp_fit)
outliers_y = stu_res[abs(stu_res) > 2.5]

# Measures of influence:
# Gives DFFITS, Cook's Distance, Hat diagonal elements, and others.

influence.measures(life_exp_fit)
```

```
## Influence measures of
##   lm(formula = life_exp ~ murder + hs_grad + frost + log_pop, data = states_analysis) :
##
##       dfb.1_  dfb.mrdr  dfb.hs_g dfb.frst  dfb.lg_p    dffit cov.r
## 1   0.093164  1.54e-01 -0.065645 -0.09185 -0.095658  0.31415 1.199
## 2   0.082181 -3.25e-01 -0.245469 -0.09938  0.197451 -0.43712 1.444
## 3  -0.111760  9.09e-02 -0.197686  0.47189  0.165137 -0.54168 1.049
## 4   0.405014 -2.09e-02 -0.391545 -0.12787 -0.229081  0.54428 0.893
## 5  -0.113683  2.57e-02  0.117306 -0.05448  0.092384  0.17194 1.418
## 6  -0.253202  1.56e-01  0.226235  0.21261  0.115453  0.36539 1.081
## 7  -0.008355 -7.33e-02 -0.010773  0.02041  0.046640  0.11951 1.156
## 8  -0.255420  3.81e-02  0.030851  0.14241  0.306995 -0.36906 1.006
## 9  -0.000252  8.87e-05  0.000496 -0.00098  0.000304  0.00153 1.242
## 10 -0.011381 -3.84e-02  0.027384 -0.00619  0.003328 -0.07332 1.233
## 11  0.619189 -4.06e-01  0.566152 -1.53843 -0.867924  1.74328 0.645
## 12  0.042239 -7.07e-03  0.040647 -0.02273 -0.084505  0.14513 1.128
## 13  0.040766 -2.54e-02 -0.011148 -0.03507 -0.040331 -0.05634 1.258
## 14  0.020006  3.59e-04 -0.000341 -0.02039 -0.029438 -0.04467 1.160
## 15 -0.002542 -1.15e-02  0.001721  0.00138  0.006678  0.01842 1.200
## 16 -0.024034 -6.62e-02  0.070901 -0.03484  0.019021  0.17091 1.081
## 17  0.179977  4.82e-02 -0.309068  0.09013 -0.034027  0.40026 1.040
## 18 -0.087740 -5.03e-02  0.055557  0.10843  0.064083 -0.21257 1.208
## 19 -0.231935  2.90e-01  0.190679 -0.12807  0.118504 -0.57083 0.732
## 20  0.029818 -2.32e-02 -0.008596 -0.02062 -0.036916 -0.09682 1.104
## 21  0.088945  1.98e-01 -0.065584  0.05959 -0.178215 -0.32200 1.072
## 22 -0.238067  1.95e-01  0.081773  0.20996  0.205843  0.33627 1.134
## 23 -0.068348 -1.26e-01 -0.003045  0.09043  0.136503  0.25074 1.142
## 24 -0.241655 -1.32e-01  0.206069  0.10364  0.190293 -0.43729 1.015
## 25 -0.028918  4.12e-02 -0.029366  0.06005  0.053563  0.13361 1.106
## 26 -0.051619 -1.43e-02 -0.047074 -0.06411  0.128833 -0.27052 1.031
## 27  0.012590 -7.07e-02  0.015810 -0.00228 -0.003323  0.12965 1.138
## 28  0.187361 -4.27e-01 -0.263770 -0.27399  0.123993 -0.53116 1.437
## 29 -0.041228  3.78e-02  0.016411 -0.06156  0.048017 -0.16515 1.148
## 30  0.084452  1.27e-01  0.034420 -0.05363 -0.205562 -0.28469 1.048
## 31  0.003840  6.84e-02  0.025462  0.02257 -0.048751  0.10211 1.168
## 32 -0.046848  2.14e-02  0.019447  0.01661  0.049608  0.06314 1.257
## 33 -0.014411 -5.35e-03  0.032781 -0.00777 -0.004587 -0.04616 1.226
## 34  0.395237 -3.40e-01 -0.397658  0.18543 -0.205286  0.68500 0.955
## 35  0.086282 -3.83e-03 -0.013070 -0.06479 -0.113939 -0.13488 1.195
## 36  0.023677 -2.77e-02 -0.012390 -0.02817 -0.005569  0.05530 1.145
## 37 -0.038771  9.76e-02 -0.053195  0.15119  0.027664 -0.18490 1.247
## 38  0.191811  1.12e-01  0.079926 -0.19146 -0.378227 -0.46026 1.007
```

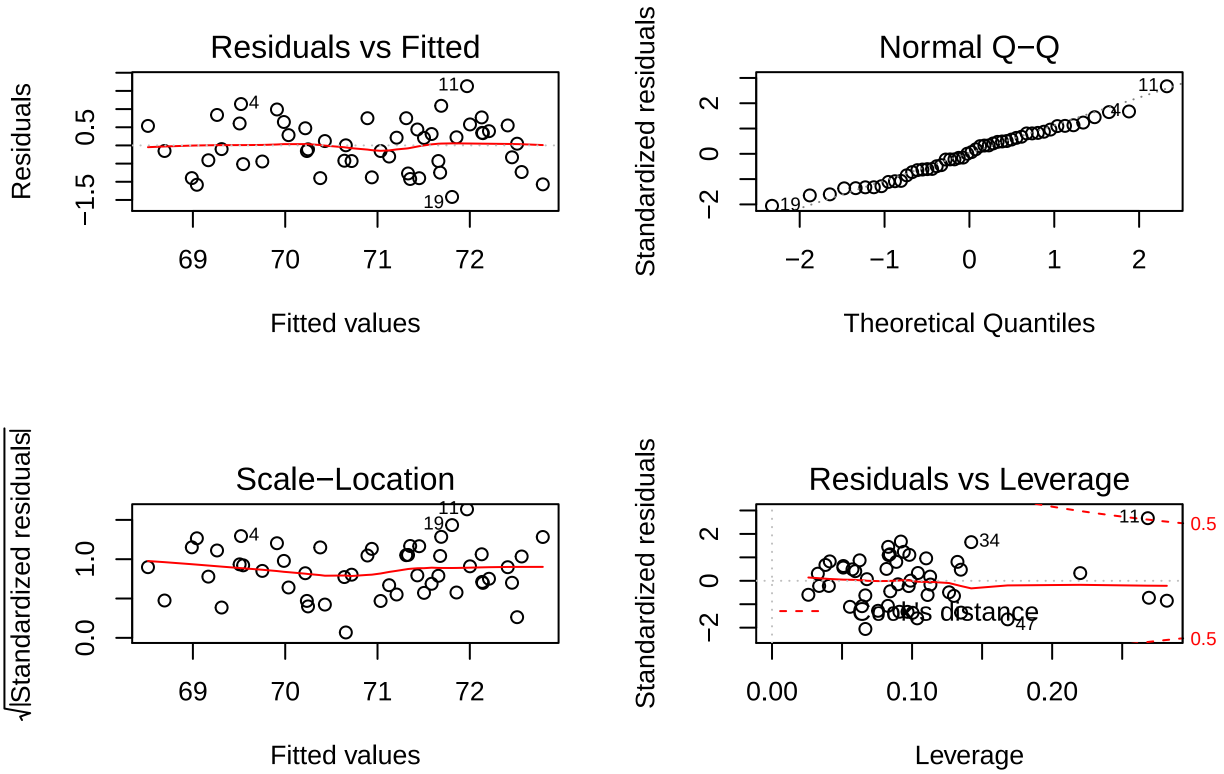
```
## 39  0.147326 -1.26e-01 -0.128814 -0.02902 -0.067792  0.18497 1.261
## 40 -0.331641 -7.49e-02  0.394591  0.04123  0.163742 -0.55501 0.930
## 41  0.062820 -5.91e-02 -0.050707  0.01987 -0.038415  0.11216 1.234
## 42  0.084329  3.96e-02 -0.134423 -0.00500 -0.011693  0.22505 1.096
## 43 -0.132948  1.25e-01  0.033883 -0.09183  0.197414  0.44100 0.961
## 44 -0.108486  1.84e-02  0.255592 -0.00818 -0.036620  0.34331 1.058
## 45  0.115480  5.28e-02 -0.008974  0.10503 -0.204807  0.33120 1.066
## 46  0.001017 -6.91e-03  0.011578 -0.00348 -0.012402 -0.04074 1.152
## 47  0.023176  3.31e-01 -0.349437  0.57841 -0.024418 -0.75432 0.987
## 48 -0.332040  1.66e-01  0.359058  0.03700  0.131277 -0.42175 1.010
## 49 -0.028050 -8.11e-02 -0.025452  0.05038  0.085101  0.14969 1.184
## 50 -0.006833 -1.15e-01 -0.089306 -0.07840  0.135985 -0.24759 1.228
##      cook.d    hat inf
## 1  1.99e-02 0.1324
## 2  3.86e-02 0.2691   *
## 3  5.76e-02 0.1350
## 4  5.68e-02 0.0920
## 5  6.03e-03 0.2200   *
## 6  2.66e-02 0.0981
## 7  2.91e-03 0.0574
## 8  2.68e-02 0.0757
## 9  4.78e-07 0.0989
## 10 1.10e-03 0.0976
## 11 5.23e-01 0.2683   *
## 12 4.27e-03 0.0509
## 13 6.49e-04 0.1131
## 14 4.08e-04 0.0406
## 15 6.94e-05 0.0677
## 16 5.88e-03 0.0412
## 17 3.17e-02 0.0941
## 18 9.17e-03 0.1111
## 19 6.04e-02 0.0667
## 20 1.90e-03 0.0260
## 21 2.07e-02 0.0827
## 22 2.27e-02 0.1101
## 23 1.27e-02 0.0887
## 24 3.76e-02 0.0967
## 25 3.61e-03 0.0381
## 26 1.46e-02 0.0556
## 27 3.41e-03 0.0509
## 28 5.68e-02 0.2819   *
## 29 5.53e-03 0.0666
## 30 1.61e-02 0.0643
## 31 2.12e-03 0.0594
## 32 8.15e-04 0.1129
## 33 4.36e-04 0.0899
## 34 9.02e-02 0.1422
## 35 3.70e-03 0.0844
## 36 6.24e-04 0.0327
## 37 6.96e-03 0.1264
## 38 4.15e-02 0.1003
## 39 6.96e-03 0.1348
## 40 5.94e-02 0.1037
## 41 2.57e-03 0.1042
```

```
## 42 1.02e-02 0.0626
## 43 3.79e-02 0.0830
## 44 2.34e-02 0.0841
## 45 2.18e-02 0.0833
## 46 3.39e-04 0.0337
## 47 1.09e-01 0.1682
## 48 3.50e-02 0.0909
## 49 4.56e-03 0.0817
## 50 1.24e-02 0.1299
```

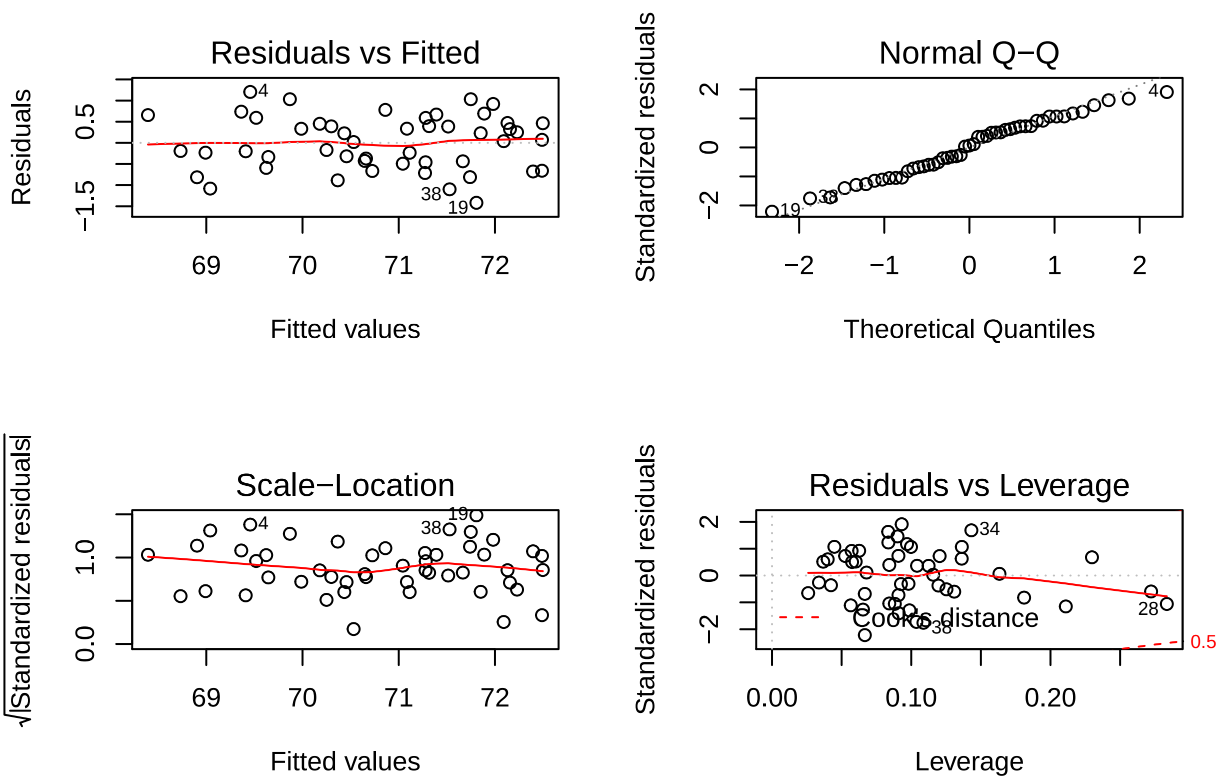
```r
# Look at the Cook's distance lines / influential point output and notice obs 11 as potential Y outlier

par(mfrow = c(2, 2))
plot(life_exp_fit)

# Examine results with and without observations 5 and 28 that have very high survivals (>2000)
fit_nooutlier = lm(life_exp ~ murder + hs_grad + log_pop + frost, data = states_analysis[-11, ])
summary(fit_nooutlier) # look at the results of the fitted model without the influential point
```

```
##
## Call:
## lm(formula = life_exp ~ murder + hs_grad + log_pop + frost, data = states_analysis[-11,
##     ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.41708 -0.45880  0.03924  0.46286  1.20332
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 67.906960   1.344438  50.510  < 2e-16 ***
## murder      -0.276679   0.033203  -8.333 1.35e-10 ***
## hs_grad      0.046799   0.013953   3.354  0.00165 **
## log_pop      0.337449   0.109043   3.095  0.00342 **
## frost       -0.001632   0.002610  -0.625  0.53499
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6621 on 44 degrees of freedom
## Multiple R-squared:  0.7611, Adjusted R-squared:  0.7394
## F-statistic: 35.05 on 4 and 44 DF,  p-value: 3.709e-13
```

```r
par(mfrow = c(2, 2))
plot(life_exp_fit)
```




```r
plot(fit_nooutlier)
```



The 11th entry showed evidence of being an influential outlier (according to Cook's distance and measure of influence). To check to see whether this entry had a significant impact on the model and its assumptions, I compared diagnostics of the model with and without the 11th point. The diagnostic plots above show that, in fact, the model assumptions (1. residuals have mean zero, 2. residuals have equal variance, 3. residuals are independent) are met for both models - with and without the potential influential point. However, we can see that without the point, the 'frost' variable is no longer a significant predictor of life expectancy, with a p-value of 0.53.

Using the 'final' model chosen in part 4, focus on MSE to test the model predictive ability a) Use a 10-fold cross-validation

```
kfold_cv = lapply(1:10, function(i){
  # create 10-fold training datasets
  data_train <- trainControl(method = "cv", number = 10)

  # Fit the model used above
  model_caret <- train((life_exp ~ murder + hs_grad + log_pop + frost),
                       data = states_analysis,
                       trControl = data_train,
                       method = 'lm',
                       na.action = na.pass)

  #return(list(model_caret$results, model_caret$resample))
  return(model_caret$results)
})

do.call("rbind", kfold_cv) %>%
  dplyr::select(-intercept)
```

```
##          RMSE  Rsquared       MAE    RMSESD RsquaredSD     MAESD
## 1  0.7550783 0.7792779 0.6623956 0.2569564  0.1930803 0.2172051
## 2  0.7083751 0.7574450 0.6213683 0.2752593  0.1610955 0.2362386
## 3  0.7696515 0.7385279 0.6668232 0.2340967  0.2801737 0.2136671
## 4  0.7662203 0.7575877 0.6627813 0.2307730  0.1815555 0.1977176
## 5  0.7491045 0.7862992 0.6385504 0.1877932  0.1671487 0.1534449
## 6  0.7566143 0.7247942 0.6437209 0.1884641  0.1652502 0.1443960
## 7  0.7240307 0.7984173 0.6349796 0.2801609  0.1519946 0.2486162
## 8  0.7409409 0.7355363 0.6388265 0.2408287  0.2321093 0.2098676
## 9  0.7488288 0.7643271 0.6340541 0.1763392  0.1456109 0.1430852
## 10 0.7586537 0.7420388 0.6576273 0.2292051  0.1656177 0.1898041
```

(b) Experiment a new, but simple bootstrap technique called "residual sampling".

```
# Perform a regression model with the original sample; calculate predicted values and residuals.
states_analysis = states_analysis %>%
  modelr::add_predictions(life_exp_fit) %>% # add predicted birthweight
  modelr::add_residuals(life_exp_fit) # residual of observed bwt - predicted bwt

# boot.res <- lapply(1:10, function(i, data = states_analysis){
#   data %>%
#     rowwise %>%
#     mutate(rand_res = sample(resid, replace = T, size = 1), # random sampling of residuals
#            boot_y = pred + rand_res) # new observations
#
#   new_pred = coef(lm(boot_y ~ murder + hs_grad + log_pop + frost, data = data)) # regress new obs
#
#   return(new_pred)
# })
```

```r
# function to bootstrap residuals and regress new predictions
boot.res <- function(data, index){
  data = data %>%
    rowwise %>%
    mutate(rand_res = sample(resid, replace = T, size = 1), # Randomly resample the residuals (with rep
           boot_y = pred + rand_res) # New observations by adding the original predicted values to the

  new_pred = coef(lm(boot_y ~ murder + hs_grad + log_pop + frost, data = data, subset = index)) # regre

  return(new_pred)
}

boot::boot(states_analysis, boot.res, 10)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot::boot(data = states_analysis, statistic = boot.res, R = 10)
##
##
## Bootstrap Statistics :
##          original        bias    std. error
## t1* 68.389499190 -0.400765069  1.57982661
## t2* -0.285326627  0.002826820  0.03163437
## t3*  0.053390681  0.010897592  0.03144217
## t4*  0.276870897 -0.010002273  0.13898372
## t5* -0.004169772 -0.001016192  0.00361087
```

```r
boot::boot(states_analysis, boot.res, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot::boot(data = states_analysis, statistic = boot.res, R = 1000)
##
##
## Bootstrap Statistics :
##          original          bias    std. error
## t1* 68.389499190  0.0725883880 1.531297990
## t2* -0.285326627  0.0018217664 0.036249535
## t3*  0.053390681  0.0013007637 0.016690977
## t4*  0.276870897 -0.0156432770 0.118714908
## t5* -0.004169772 -0.0001660881 0.003333786
```