p8130_hw5

Catherine

12/2/2018

Problem 1

Explore the dataset and generate appropriate descriptive statistics and relevant graphs for all variables of interest (continuous and categorical) – no test required.

```
state_data = state.x77 %>% as.tibble() %>% janitor::clean_names()

summary_df = skimr::skim_to_wide(state_data) %>%
  mutate(min = p0,
         max = p100,
         median = p50,
         IQR = (as.numeric(p75) - as.numeric(p25)),
         range = (as.numeric(p100) - as.numeric(p0))) %>%
  select(type, variable, mean, sd, min, median, max, range, IQR, hist)

knitr::kable(summary_df, digits = 3, caption = "Descriptive Statistics for All Variables")
```

Table 1: Descriptive Statistics for All Variables

type	variable	mean	sd	min	median	max	range	IQR	hist
numeric	area	70735.88	85327.3	1049	54277	566432	565383.00	44177.25	
numeric	frost	104.46	51.98	0	114.5	188	188.00	73.50	
numeric	hs_grad	53.11	8.08	37.8	53.25	67.3	29.50	11.10	
numeric	illiteracy	1.17	0.61	0.5	0.95	2.8	2.30	0.96	
numeric	income	4435.8	614.47	3098	4519	6315	3217.00	820.75	
numeric	life_exp	70.88	1.34	67.96	70.67	73.6	5.64	1.77	
numeric	murder	7.38	3.69	1.4	6.85	15.1	13.70	6.32	
numeric	population	4246.42	4464.49	365	2838.5	21198	20833.00	3889.00	

Comment

All the variables are continuous. Therefore, I chose mean and median as measures of location, range, IQR and sd as measures of dispersion.

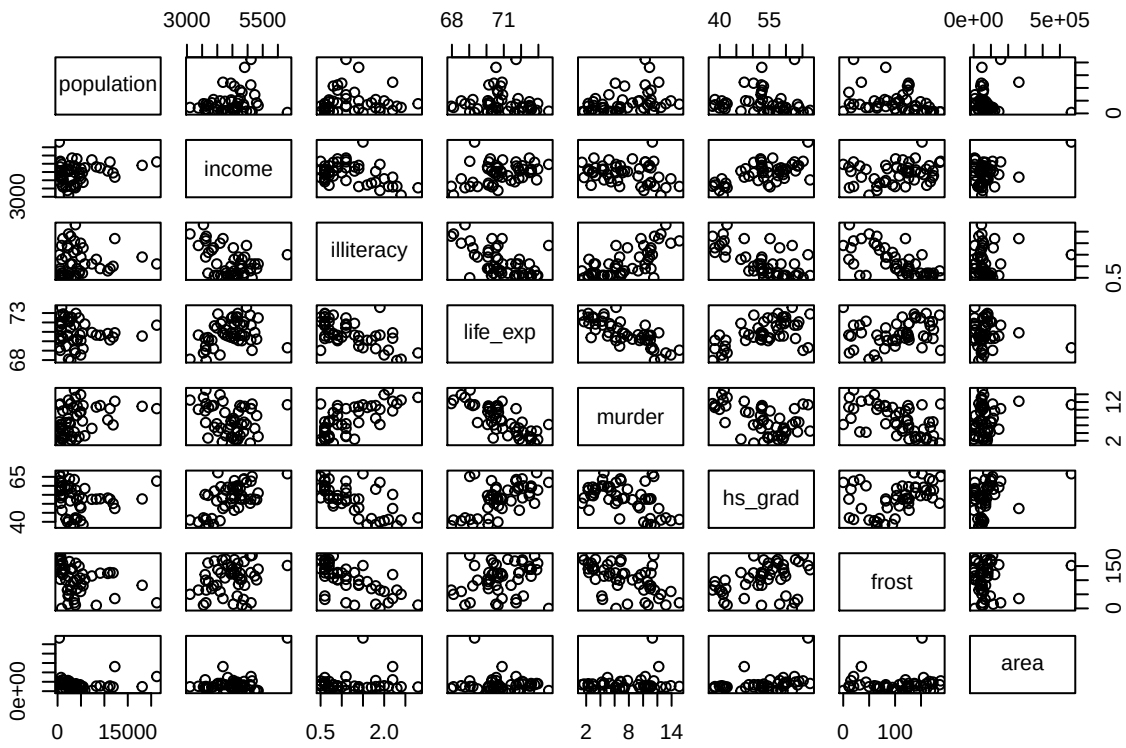
```
cor(state_data) %>% knitr::kable(digits = 3, caption = "Correlation Table")
```

Table 2: Correlation Table

	population	income	illiteracy	life_exp	murder	hs_grad	frost	area
population	1.000	0.208	0.108	-0.068	0.344	-0.098	-0.332	0.023
income	0.208	1.000	-0.437	0.340	-0.230	0.620	0.226	0.363
illiteracy	0.108	-0.437	1.000	-0.588	0.703	-0.657	-0.672	0.077
life_exp	-0.068	0.340	-0.588	1.000	-0.781	0.582	0.262	-0.107
murder	0.344	-0.230	0.703	-0.781	1.000	-0.488	-0.539	0.228
hs_grad	-0.098	0.620	-0.657	0.582	-0.488	1.000	0.367	0.334
frost	-0.332	0.226	-0.672	0.262	-0.539	0.367	1.000	0.059

	population	income	illiteracy	life_exp	murder	hs_grad	frost	area
area	0.023	0.363	0.077	-0.107	0.228	0.334	0.059	1.000

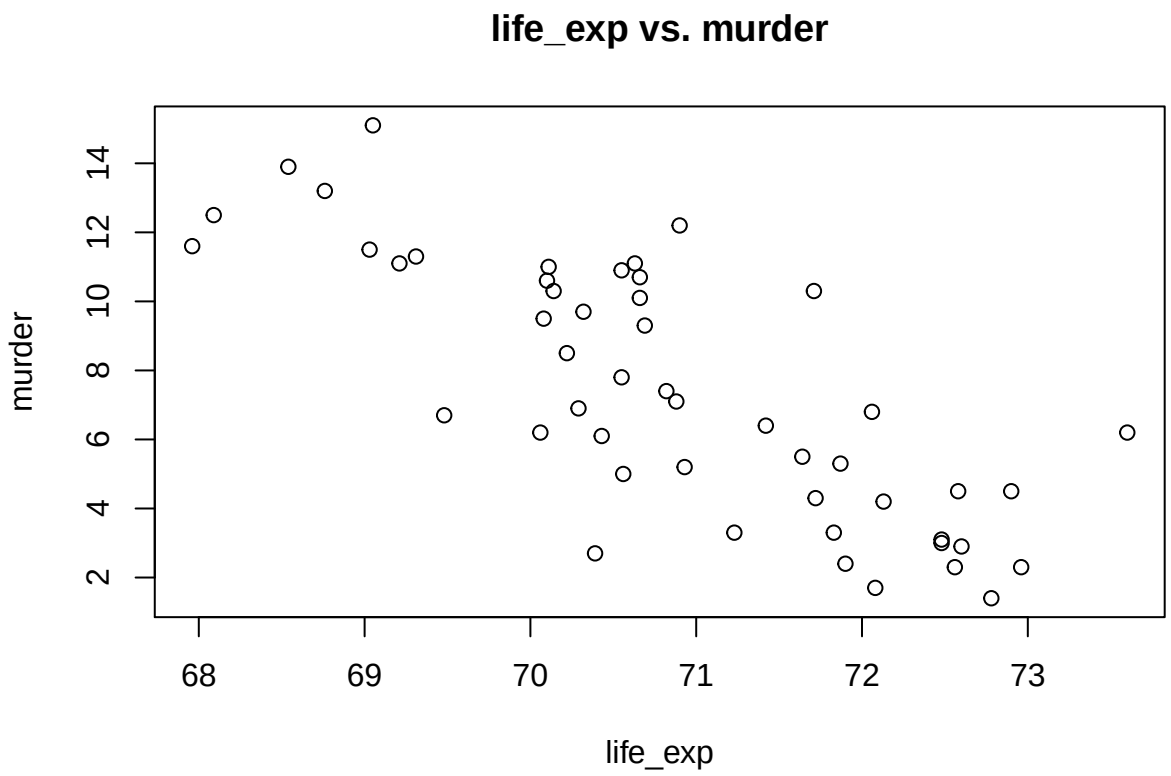
```
pairs(state_data)
```



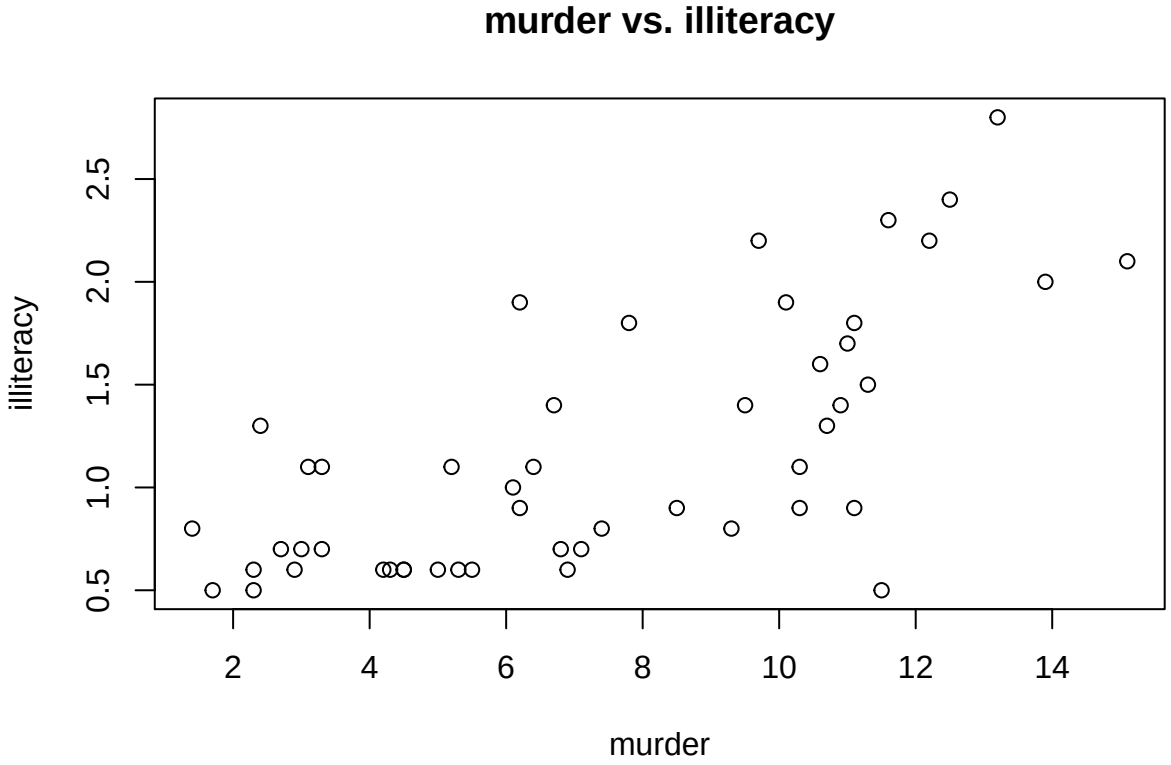
Comment

From the table, `life_exp` and `murder` have a strong negative correlation. Also, `murder` has a strong positive correaltion with `illiteracy`, which need more investigation.

```
state_data %>%
  select(life_exp, murder) %>%
  plot(main = "life_exp vs. murder")
```



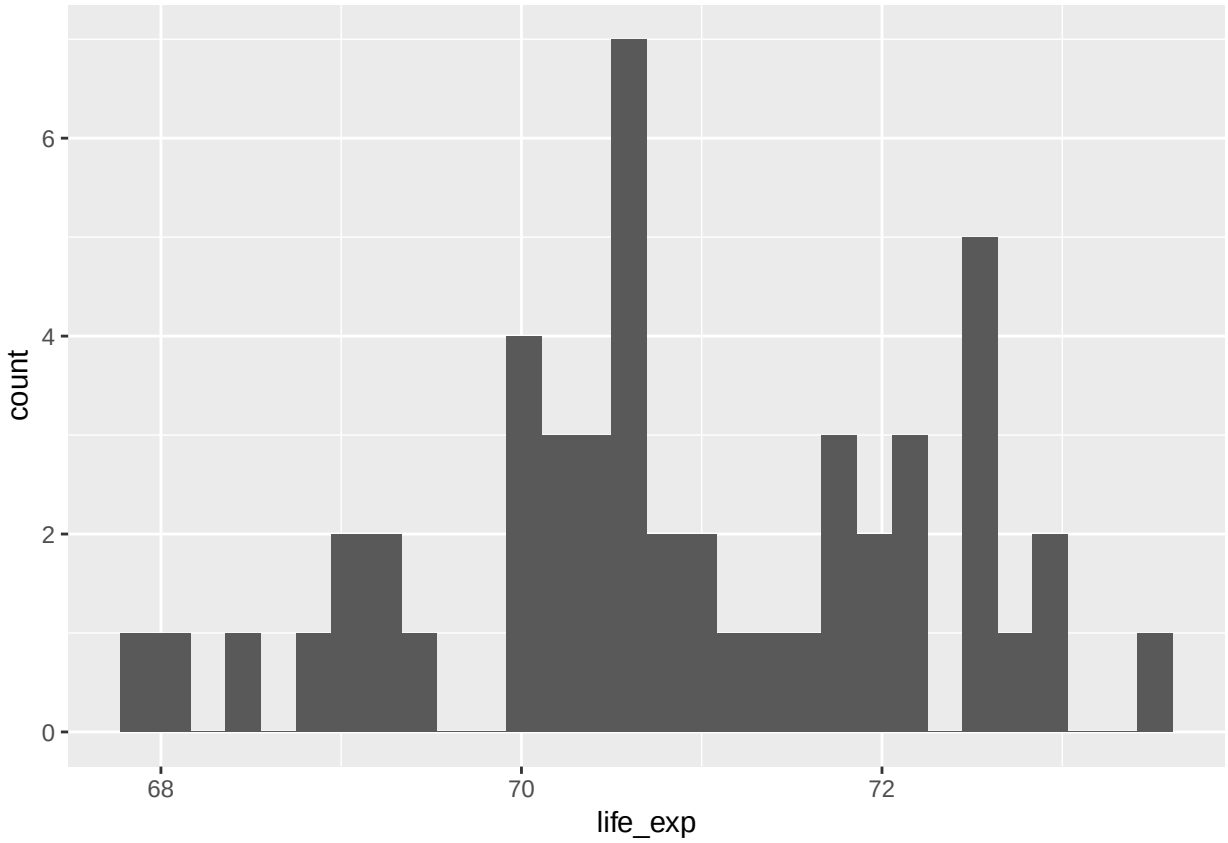

```
state_data %>%
  select(murder, illiteracy) %>%
  plot(main = "murder vs. illiteracy")
```




```
state_data %>%
```

```
  ggplot(aes(x = life_exp)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Comment

Make plot to show the correlation between `life_exp` and `murder`, `murder` and `illiteracy`. Also, check the distribution of `life_exp`, which is approximately normal distribution. I decided not to do any transformation.

Problem 2

Use automatic procedures to find a 'best subset' of the full model.

Backward elimination

Start with fitting a model with all variables.

```
fit_all = lm(life_exp ~ ., data = state_data)
summary(fit_all)$coef %>% knitr::kable(caption = "Regression model with all variables")
```

Table 3: Regression model with all variables

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	70.9432241	1.7479754	40.5859402	0.0000000

	Estimate	Std. Error	t value	Pr(>|t|)
population	0.0000518	0.0000292	1.7747731	0.0831835
income	-0.0000218	0.0002444	-0.0892060	0.9293422
illiteracy	0.0338203	0.3662799	0.0923346	0.9268712
murder	-0.3011232	0.0466207	-6.4589973	0.0000001
hs_grad	0.0489295	0.0233233	2.0978818	0.0419718
frost	-0.0057350	0.0031432	-1.8245568	0.0751868
area	-0.0000001	0.0000017	-0.0442593	0.9649075

Take out non-significant variables, starting with the highest p-value

```
# No area
step1 = update(fit_all, . ~ . -area)
summary(step1)
```

```
##
## Call:
## lm(formula = life_exp ~ population + income + illiteracy + murder +
##     hs_grad + frost, data = state_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.49047 -0.52533 -0.02546  0.57160  1.50374
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.099e+01  1.387e+00  51.165  < 2e-16 ***
## population   5.188e-05  2.879e-05   1.802   0.0785 .
## income      -2.444e-05  2.343e-04  -0.104   0.9174
## illiteracy   2.846e-02  3.416e-01   0.083   0.9340
## murder      -3.018e-01  4.334e-02  -6.963 1.45e-08 ***
## hs_grad      4.847e-02  2.067e-02   2.345   0.0237 *
## frost       -5.776e-03  2.970e-03  -1.945   0.0584 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7361 on 43 degrees of freedom
## Multiple R-squared:  0.7361, Adjusted R-squared:  0.6993
## F-statistic: 19.99 on 6 and 43 DF,  p-value: 5.362e-11
```

```
# No income
step2 = update(step1, . ~ . -income)
summary(step2)
```

```
##
## Call:
## lm(formula = life_exp ~ population + illiteracy + murder + hs_grad +
##     frost, data = state_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47248 -0.54771 -0.02936  0.58145  1.49381
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.095e+01  1.319e+00  53.786  < 2e-16 ***
## population   5.090e-05  2.691e-05   1.892   0.0651 .
## illiteracy   2.906e-02  3.377e-01   0.086   0.9318
## murder      -3.020e-01  4.282e-02  -7.053 9.57e-09 ***
## hs_grad      4.733e-02  1.732e-02   2.733   0.0090 **
## frost       -5.806e-03  2.923e-03  -1.986   0.0532 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7278 on 44 degrees of freedom
## Multiple R-squared:  0.7361, Adjusted R-squared:  0.7061
## F-statistic: 24.54 on 5 and 44 DF,  p-value: 1.021e-11
```

```
# No illiteracy
step3 = update(step2, . ~ . -illiteracy)
summary(step3)
```

```
##
## Call:
## lm(formula = life_exp ~ population + murder + hs_grad + frost,
##     data = state_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47095 -0.53464 -0.03701  0.57621  1.50683
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.103e+01  9.529e-01  74.542  < 2e-16 ***
## population   5.014e-05  2.512e-05   1.996  0.05201 .
## murder      -3.001e-01  3.661e-02  -8.199 1.77e-10 ***
## hs_grad      4.658e-02  1.483e-02   3.142  0.00297 **
## frost       -5.943e-03  2.421e-03  -2.455  0.01802 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7197 on 45 degrees of freedom
## Multiple R-squared:  0.736,  Adjusted R-squared:  0.7126
## F-statistic: 31.37 on 4 and 45 DF,  p-value: 1.696e-12
```

```
# No population
step4 = update(step3, . ~ . -population)
summary(step4)
```

```
##
## Call:
## lm(formula = life_exp ~ murder + hs_grad + frost, data = state_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5015 -0.5391  0.1014  0.5921  1.2268
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 71.036379   0.983262  72.246  < 2e-16 ***
## murder      -0.283065   0.036731  -7.706 8.04e-10 ***
## hs_grad      0.049949   0.015201   3.286  0.00195 **
## frost       -0.006912   0.002447  -2.824  0.00699 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7427 on 46 degrees of freedom
## Multiple R-squared:  0.7127, Adjusted R-squared:  0.6939
## F-statistic: 38.03 on 3 and 46 DF,  p-value: 1.634e-12
```

Comment

From the Backward elimination, we got `life_exp ~ population + murder + hs_grad + frost`. I keep the `population` as it contributes to the goodness of fit for the model.

life expectancy = 71 - 0.3murder + 0.0466hs_grad + -0.00594frost + 0.0000501population

Forward elimination

Take in significant variables, starting with the lowest p-value

Step 1: Fit simple linear regressions for all variables, look for the variable with lowest p-value

```
fit1 = lm(life_exp ~ population, data = state_data)
tidy(fit1)
```

```
## # A tibble: 2 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  71.0       0.265       267.    7.90e-78
## 2 population   -0.0000205 0.0000433    -0.473 6.39e- 1
```

```
fit2 = lm(life_exp ~ income, data = state_data)
tidy(fit2)
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) 67.6       1.33         50.9  1.98e-43
## 2 income       0.000743  0.000297      2.51 1.56e- 2
```

```
fit3 = lm(life_exp ~ illiteracy, data = state_data)
tidy(fit3)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    72.4      0.338    214.   3.47e-73
## 2 illiteracy     -1.30     0.257     -5.04 6.97e- 6
```

```
fit4 = lm(life_exp ~ murder, data = state_data)
tidy(fit4)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   73.0      0.270     270.   4.72e-78
```

```
## 2 murder        -0.284    0.0328     -8.66 2.26e-11
```

```
fit5 = lm(life_exp ~ hs_grad, data = state_data)
tidy(fit5)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  65.7       1.05       62.8  9.92e-48
## 2 hs_grad       0.0968    0.0195      4.96 9.20e- 6
```

```
fit6 = lm(life_exp ~ frost, data = state_data)
tidy(fit6)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) 70.2       0.419      168.   4.33e-68
## 2 frost        0.00677   0.00360      1.88 6.60e- 2
```

Step 2: Enter first the one with the lowest p-value: murder. Enter the one with the lowest p-value in the rest

```
forward1 = lm(life_exp ~ murder, data = state_data)
tidy(forward1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   73.0      0.270     270.   4.72e-78
## 2 murder        -0.284    0.0328     -8.66 2.26e-11
```

```
fit1 = update(forward1, . ~ . + illiteracy)
tidy(fit1)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   73.0      0.286    256.    1.56e-75
## 2 murder        -0.264    0.0464    -5.69  7.96e- 7
## 3 illiteracy    -0.172    0.281     -0.613 5.43e- 1
```

```
fit2 = update(forward1, . ~ . + hs_grad)
tidy(fit2)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  70.3       1.02       69.2  5.91e-49
## 2 murder       -0.237     0.0353     -6.72 2.18e- 8
## 3 hs_grad       0.0439    0.0161      2.72 9.09e- 3
```

```
fit3 = update(forward1, . ~ . + income)
tidy(fit3)
```

```
## # A tibble: 3 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  71.2       0.967        73.6  3.32e-50
```

```
## 2 murder       -0.270     0.0328       -8.21 1.22e-10
## 3 income        0.000370  0.000197      1.88 6.66e- 2
```

```
fit4 = update(forward1, . ~ . + frost)
tidy(fit4)
```

```
## # A tibble: 3 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  73.9       0.500      148.   2.36e-64
## 2 murder       -0.328     0.0375      -8.74 2.05e-11
## 3 frost        -0.00578   0.00266     -2.17 3.52e- 2
```

```
fit5 = update(forward1, . ~ . + population)
tidy(fit5)
```

```
## # A tibble: 3 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  72.9       0.258        282.   1.55e-77
## 2 murder       -0.312     0.0332       -9.42 2.15e-12
## 3 population    0.0000683 0.0000274     2.49 1.64e- 2
```

Step 3: Enter the one with the lowest p-value: hs_grad. Enter the one with the lowest p-value in the rest.

```
forward2 = update(forward1, . ~ . + hs_grad)
tidy(forward2)
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  70.3       1.02       69.2  5.91e-49
## 2 murder       -0.237     0.0353     -6.72 2.18e- 8
## 3 hs_grad       0.0439    0.0161      2.72 9.09e- 3
```

```
fit1 = update(forward2, . ~ . + illiteracy)
tidy(fit1)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  69.7       1.22      57.1   2.41e-44
## 2 murder       -0.258     0.0435    -5.93  3.63e- 7
## 3 hs_grad       0.0518    0.0188     2.76  8.25e- 3
## 4 illiteracy    0.254     0.305      0.833 4.09e- 1
```

```
fit2 = update(forward2, . ~ . + income)
tidy(fit2)
```

```
## # A tibble: 4 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  70.1        1.10        64.0   1.33e-46
## 2 murder       -0.239      0.0358      -6.66  2.92e- 8
## 3 hs_grad       0.0391     0.0203       1.92  6.05e- 2
## 4 income        0.0000953  0.000239     0.398 6.92e- 1
```

```
fit3 = update(forward2, . ~ . + frost)
tidy(fit3)
```

```
## # A tibble: 4 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  71.0       0.983       72.2  5.25e-49
## 2 murder       -0.283     0.0367      -7.71 8.04e-10
## 3 hs_grad       0.0499    0.0152       3.29 1.95e- 3
## 4 frost        -0.00691   0.00245     -2.82 6.99e- 3
```

```
fit4 = update(forward2, . ~ . + population)
tidy(fit4)
```

```
## # A tibble: 4 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  70.4       0.969         72.7  3.95e-49
## 2 murder       -0.266     0.0357        -7.45 1.91e- 9
## 3 hs_grad       0.0407    0.0154         2.64 1.12e- 2
## 4 population    0.0000625 0.0000259      2.41 1.99e- 2
```

Step 4: Enter the one with the lowest p-value: frost. Enter the one with the lowest p-value in the rest

```
forward3 = update(forward2, . ~ . + frost)
tidy(forward3)
```

```
## # A tibble: 4 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  71.0       0.983       72.2  5.25e-49
## 2 murder       -0.283     0.0367      -7.71 8.04e-10
## 3 hs_grad       0.0499    0.0152       3.29 1.95e- 3
## 4 frost        -0.00691   0.00245     -2.82 6.99e- 3
```

```
fit1 = update(forward3, . ~ . + illiteracy)
tidy(fit1)
```

```
## # A tibble: 5 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  71.5       1.32       54.2   1.28e-42
## 2 murder       -0.273     0.0411     -6.64  3.50e- 8
## 3 hs_grad       0.0450    0.0178      2.53  1.49e- 2
## 4 frost        -0.00768   0.00283    -2.72  9.36e- 3
## 5 illiteracy   -0.182     0.328      -0.554 5.82e- 1
```

```
fit2 = update(forward3, . ~ . + income)
tidy(fit2)
```

```
## # A tibble: 5 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  70.8       1.05        67.4   7.53e-47
## 2 murder       -0.286     0.0373      -7.66  1.07e- 9
## 3 hs_grad       0.0436    0.0190       2.30  2.64e- 2
## 4 frost        -0.00698   0.00247     -2.83  6.96e- 3
```

```
## 5 income        0.000127  0.000223     0.571 5.71e- 1
```

```
fit3 = update(forward3, . ~ . + population)
tidy(fit3)
```

```
## # A tibble: 5 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  71.0       0.953         74.5  8.61e-49
## 2 murder       -0.300     0.0366        -8.20 1.77e-10
## 3 hs_grad       0.0466    0.0148         3.14 2.97e- 3
## 4 frost        -0.00594   0.00242       -2.46 1.80e- 2
## 5 population    0.0000501 0.0000251      2.00 5.20e- 2
```

Step 5: Enter the one with the lowest p-value: population. Enter the one with the lowest p-value in the rest

```
forward4 = update(forward3, . ~ . + population)
tidy(forward4)
```

```
## # A tibble: 5 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  71.0       0.953         74.5  8.61e-49
## 2 murder       -0.300     0.0366        -8.20 1.77e-10
## 3 hs_grad       0.0466    0.0148         3.14 2.97e- 3
## 4 frost        -0.00594   0.00242       -2.46 1.80e- 2
## 5 population    0.0000501 0.0000251      2.00 5.20e- 2
```

```
fit1 = update(forward4, . ~ . + income)
tidy(fit1)
```

```
## # A tibble: 6 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  71.1       1.03         69.1   1.66e-46
## 2 murder       -0.300     0.0370       -8.10  2.91e-10
## 3 hs_grad       0.0478    0.0186        2.57  1.37e- 2
## 4 frost        -0.00591   0.00247      -2.39  2.10e- 2
## 5 population    0.0000511 0.0000271     1.89  6.57e- 2
## 6 income       -0.0000248 0.000232     -0.107 9.15e- 1
```

```
fit2 = update(forward4, . ~ . + illiteracy)
tidy(fit2)
```

```
## # A tibble: 6 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  70.9       1.32        53.8    8.77e-42
## 2 murder       -0.302     0.0428      -7.05   9.57e- 9
## 3 hs_grad       0.0473    0.0173       2.73   9.00e- 3
## 4 frost        -0.00581   0.00292     -1.99   5.32e- 2
## 5 population    0.0000509 0.0000269    1.89   6.51e- 2
## 6 illiteracy    0.0291    0.338        0.0861 9.32e- 1
```

P-value of all new added variables are larger than **0.1**, which means that they are not significant predictors, and we stop here.

The model we obtained is `life_exp ~ murder + hs_grad + frost + population`.

life expectancy = 71 - 0.3murder + 0.0466hs_grad + -0.00594frost + 0.0000501population

```
fit_final = lm(life_exp ~ murder + hs_grad + frost + population, data = state_data)
summary(fit_final)
```

```
##
## Call:
## lm(formula = life_exp ~ murder + hs_grad + frost + population,
##     data = state_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47095 -0.53464 -0.03701  0.57621  1.50683
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.103e+01  9.529e-01  74.542  < 2e-16 ***
## murder      -3.001e-01  3.661e-02  -8.199 1.77e-10 ***
## hs_grad      4.658e-02  1.483e-02   3.142  0.00297 **
## frost       -5.943e-03  2.421e-03  -2.455  0.01802 *
## population   5.014e-05  2.512e-05   1.996  0.05201 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7197 on 45 degrees of freedom
## Multiple R-squared:  0.736,  Adjusted R-squared:  0.7126
## F-statistic: 31.37 on 4 and 45 DF,  p-value: 1.696e-12
```

Stepwise elimination

```
fit_all = lm(life_exp ~ ., data = state_data)
step_wise = step(fit_all, direction = 'backward')
```

```
## Start:  AIC=-22.18
## life_exp ~ population + income + illiteracy + murder + hs_grad +
##     frost + area
##
##              Df Sum of Sq    RSS     AIC
## - area        1    0.0011 23.298 -24.182
## - income      1    0.0044 23.302 -24.175
## - illiteracy  1    0.0047 23.302 -24.174
## <none>                    23.297 -22.185
## - population  1    1.7472 25.044 -20.569
## - frost       1    1.8466 25.144 -20.371
## - hs_grad     1    2.4413 25.738 -19.202
## - murder      1   23.1411 46.438  10.305
##
## Step:  AIC=-24.18
## life_exp ~ population + income + illiteracy + murder + hs_grad +
##     frost
##
##              Df Sum of Sq    RSS     AIC
## - illiteracy  1    0.0038 23.302 -26.174
## - income      1    0.0059 23.304 -26.170
```

```
## <none>                    23.298 -24.182
## - population  1    1.7599 25.058 -22.541
## - frost       1    2.0488 25.347 -21.968
## - hs_grad     1    2.9804 26.279 -20.163
## - murder      1   26.2721 49.570  11.569
##
## Step:  AIC=-26.17
## life_exp ~ population + income + murder + hs_grad + frost
##
##              Df Sum of Sq    RSS     AIC
## - income      1     0.006 23.308 -28.161
## <none>                    23.302 -26.174
## - population  1     1.887 25.189 -24.280
## - frost       1     3.037 26.339 -22.048
## - hs_grad     1     3.495 26.797 -21.187
## - murder      1    34.739 58.041  17.456
##
## Step:  AIC=-28.16
## life_exp ~ population + murder + hs_grad + frost
##
##              Df Sum of Sq    RSS     AIC
## <none>                    23.308 -28.161
## - population  1     2.064 25.372 -25.920
## - frost       1     3.122 26.430 -23.877
## - hs_grad     1     5.112 28.420 -20.246
## - murder      1    34.816 58.124  15.528
```

Comment

The model we obtained is `life_exp ~ murder + hs_grad + frost + population` with the smallest `AIC`.

life expectancy = 71 - 0.3murder + 0.0466hs_grad + -0.00594frost + 0.0000501population

Present the results and comment on the following:

a) Do the procedures generate the same model?

Answer: Yes, all three precedures generate the same model, which is `life_exp ~ murder + hs_grad + frost + population`.

b) Is there any variable a close call? What was your decision: keep or discard? Provide arguments for your choice.

Answer: In the backward and forwad elimination, `population` is a close call. My decision is keep it in our model. First, by look at the `Adjusted R-square` changes of adding `population` in our model, I found out `Adjusted R-square` incresased. Also, the cut-off p-value is 0.1, we can be less strict in this procedures and drop the variable in next step if needed.

c) Is there any association between 'Illiteracy' and 'HS graduation rate'? Does your 'subset' contain both?

```
cor(state_data$illiteracy, state_data$hs_grad)
```

```
## [1] -0.6571886
```

Answer: Yes, from the correlation table in problem 1, we can see 'Illiteracy' and 'HS graduation rate' has a negative correlation which is `-0.657`, which means that higher HS graduation rate may decrease Illiteracy. The 'subset' doesn't contain both.

Problem 3

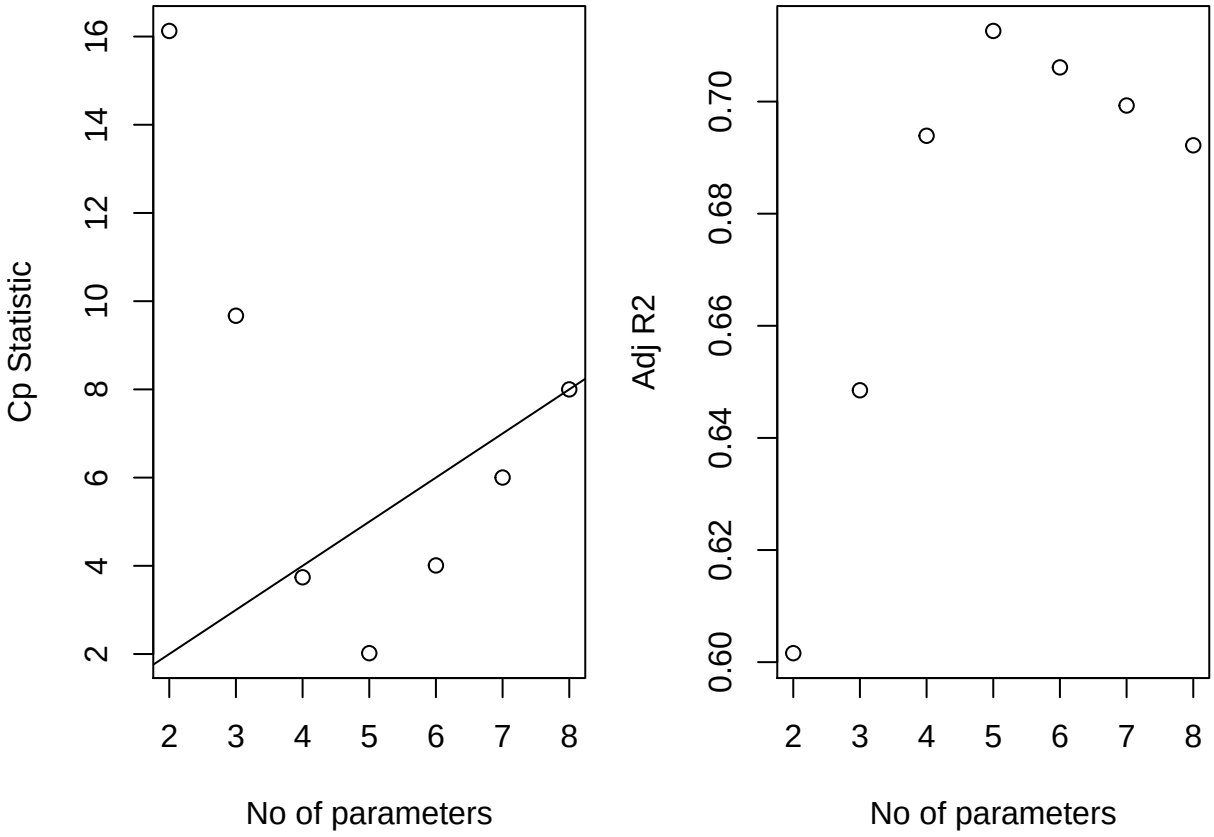
Use criterion-based procedures studied in class to guide your selection of the 'best subset'. Summarize your results

```
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)
}
```

```
summary_best = round(best(fit_all, nbest = 1), 4) %>% as.tibble()

summary_best
```

```
## # A tibble: 7 x 14
##       p `(Intercept)` population income illiteracy murder hs_grad frost
##   <dbl>         <dbl>      <dbl>  <dbl>      <dbl>  <dbl>   <dbl> <dbl>
## 1     1             1          0      0          0      1       0     0
## 2     2             1          0      0          0      1       1     0
## 3     3             1          0      0          0      1       1     1
## 4     4             1          1      0          0      1       1     1
## 5     5             1          1      1          0      1       1     1
## 6     6             1          1      1          1      1       1     1
## 7     7             1          1      1          1      1       1     1
## # ... with 6 more variables: area <dbl>, rss <dbl>, rsq <dbl>,
## #   adjr2 <dbl>, cp <dbl>, bic <dbl>
```

```
par(mar=c(4,4,1,1))
par(mfrow=c(1,2))

plot(2:8, summary_best$cp, xlab="No of parameters", ylab="Cp Statistic")
abline(0,1)

plot(2:8, summary_best$adjr2, xlab="No of parameters", ylab="Adj R2")
```



Comment

By doing models comparation in terms of AIC, R-adj and Cp, we obtained a model with 5 parameters (4 predictors) because 5 parameters' model has `Cp` value less than `n` and highest `Adj R2`. In the `Best` table, the best model with 4 predictors is `life_exp ~ murder + hs_grad + frost + population`.

Problem 4

Compare the two 'subsets' from parts 2 and 3 and recommend a 'final' model. Using this 'final' model do the following:

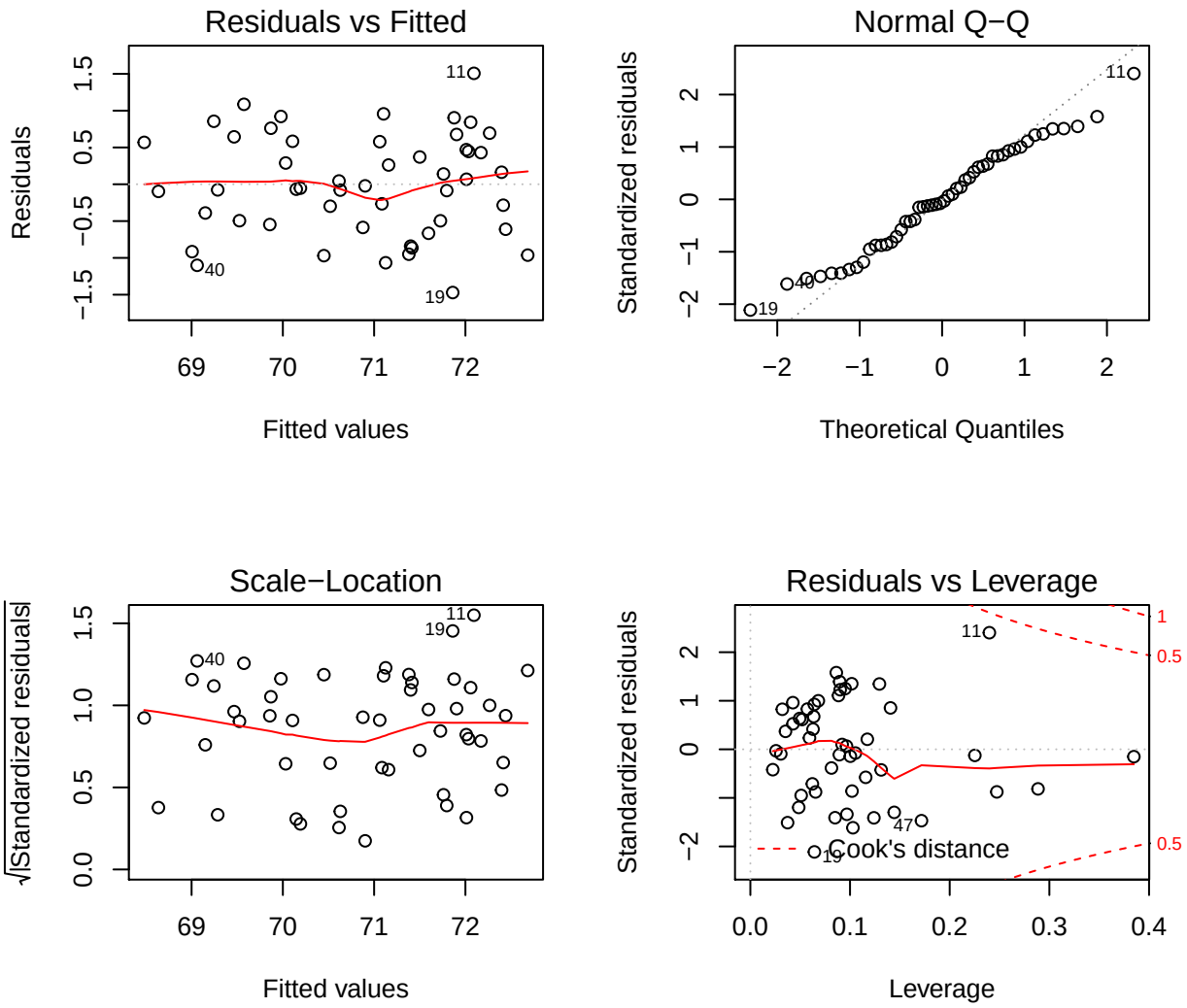
Answer: Actually, we found the exactly same model from part 2 and 3, which was amazing. Our final model is `life_exp ~ murder + hs_grad + frost + population`.

a) Identify any leverage and/or influential points and take appropriate measures.

Diagnostics Plot

First, let's take a look at the plots of our model.

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



Comment

In `Residuals vs Fitted` and `Quantile-Quantile Plot`, R detect three potential outliers, which are `40`, `19` and `11` observations.

In `Residuals vs Leverage`, R detect some observations at the upper right or lower right corner and cases outside of a dashed line, which are `11`, `47` and `19`.

Detect outliers in Y using 'studentized residuals'

```
sr = rstandard(fit_final) %>% as_tibble()
outlier_y = sr %>% filter(abs(value) >2.5)
max(abs(sr))
```

```
## [1] 2.401331
```

Comment

As the max R_i value is 2.4013305. There are no oulier in Y being detected.

Detect outliners in X using Leverage values

```
fit_hat = hatvalues(fit_final) %>% as_tibble() %>% mutate(n = 1:50)
outlier_x = cbind(fit_hat %>% filter(abs(value) > 0.2),
                  fit_hat %>% filter(abs(value) > (2*5/50)))
```

```r
names(outlier_x) = c("over_0.2", "observation_n","over_2p/n", "observation_n")
knitr::kable(outlier_x, caption = "Outliers of X")
```

Table 4: Outliers of X

over_0.2	observation_n	over_2p/n	observation_n
0.2472792	2	0.2472792	2
0.3847592	5	0.3847592	5
0.2397924	11	0.2397924	11
0.2886092	28	0.2886092	28
0.2252274	32	0.2252274	32

Comment

By taking look at the h_{ii} values, we detect five observations with $h_{ii} > 0.2$ and $hii > 2p/n$. They are the same observations: 2, 5, 11, 28, 32. They are the potential outliers in X.

Influencial Observation

Not all outliers are influential. Therefore, we need to test the influence of the outliers. Using DFFITS test the difference of fitted value with/without an observation and Cook's Distance to find concerned values.

```r
tb = influence.measures(step_wise)[["infmat"]] %>% as_tibble() %>%
  mutate(n = 1:50) %>%
  select(dffit, cook.d, n) %>%
  filter(abs(dffit)>1|abs(cook.d)>0.5)
knitr::kable(tb, caption = "Influential observation")
```

Table 5: Influential observation

dffit	cook.d	n
1.428239	0.3637786	11

Comment

Consider `DFFITS` and `Cook's Distance`, we found out an influencitial outlier `11`. The difference is large between fitted value with/without `11` observation. Next, we can take a look at the change of fitted value with/without `11`.

```r
without11 = state_data[-11,]
fit_with11 = lm(life_exp ~ murder + hs_grad + frost + population, data = state_data)
fit_without11 = lm(life_exp ~ murder + hs_grad + frost + population, data = without11)

sum1 = summary(fit_with11)$coef
sum2 = summary(fit_without11)$coef

knitr::kable(sum1, caption = "Model with observation 11")
```

Table 6: Model with observation 11

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	71.0271285	0.9528530	74.541541	0.0000000
murder	-0.3001488	0.0366095	-8.198669	0.0000000
hs_grad	0.0465822	0.0148271	3.141704	0.0029681

	Estimate	Std. Error	t value	Pr(>|t|)
frost	-0.0059433	0.0024209	-2.455017	0.0180178
population	0.0000501	0.0000251	1.996017	0.0520051

```
knitr::kable(sum2, caption = "Model without observation 11")
```

Table 7: Model without observation 11

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	71.0568289	0.8998400	78.966074	0.0000000
murder	-0.2906102	0.0347726	-8.357439	0.0000000
hs_grad	0.0372794	0.0144710	2.576148	0.0134247
frost	-0.0030995	0.0025449	-1.217924	0.2297440
population	0.0000636	0.0000243	2.617722	0.0120913

```
(sum1[2]-sum2[2])/sum1[2]
```

```
## [1] 0.0317797
```

```
(sum1[3]-sum2[3])/sum1[3]
```

```
## [1] 0.1997081
```

```
(sum1[4]-sum2[4])/sum1[4]
```

```
## [1] 0.4784953
```

```
(sum1[5]-sum2[5])/sum1[5]
```

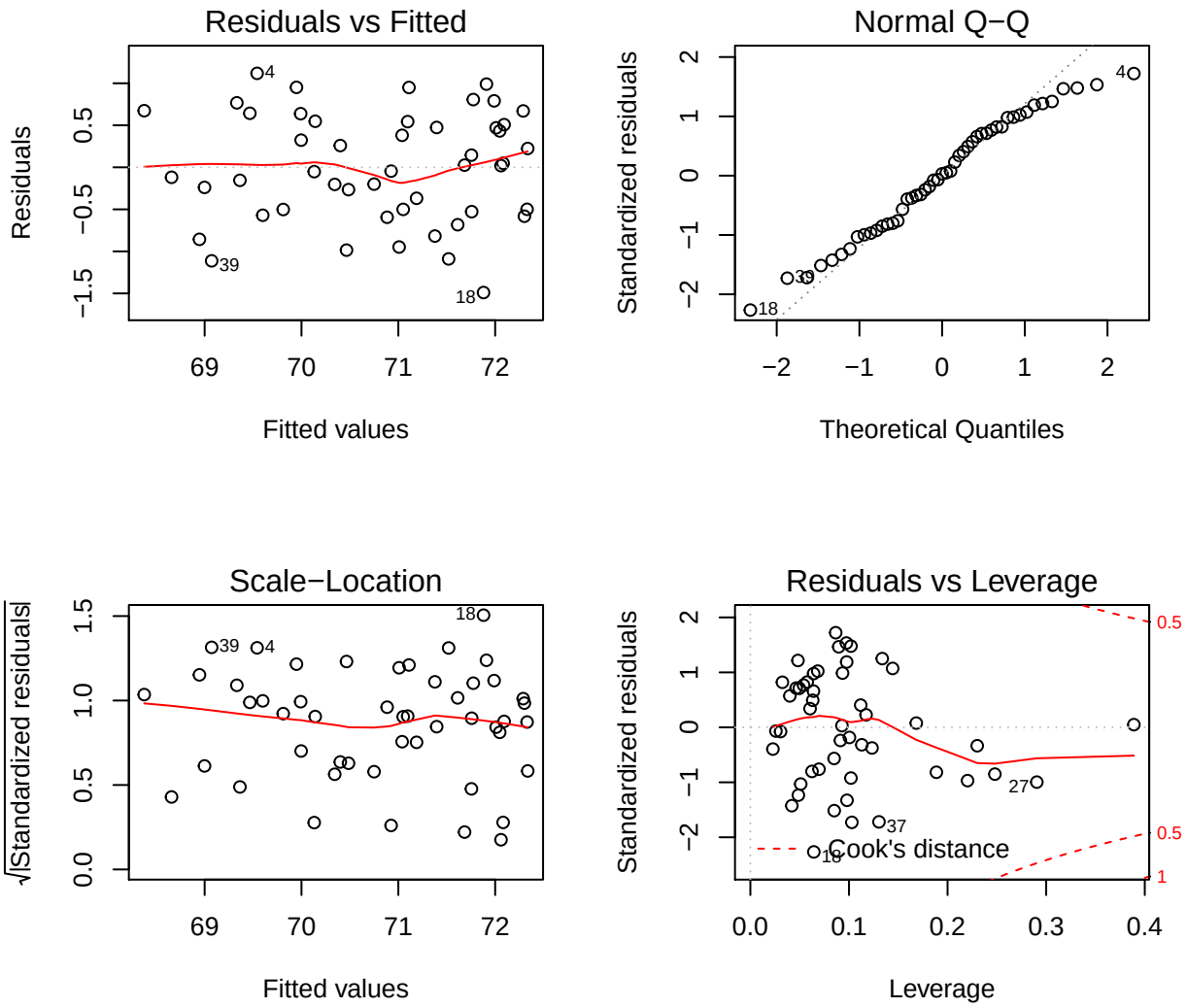
```
## [1] -0.2689781
```

Comment

After calculating the coefficient changes for each variables, we found that the changes are significant for `hs_grad`, `frost` and `population`. Especially for `frost`, the `frost` value of `11` observation is `0`. It is reasonable for us to consider the possibility of missing value. Therefore, `11` need to be exclude from our dataset.

b) Check the model assumptions.

```
fit_final_without11 = lm(life_exp ~ murder + hs_grad + frost + population, data = without11)

par(mfrow = c(2,2))
plot(fit_final_without11)
```



Comment

In the *Residuals vs Fitted Plot* and *Scale-Location Plot*, residual values bounce around 0 and form a horizontal 'band' around zero. Our model fulfill equal variance assumption. There are several values stand from the random pattern but they are not problematic.

In the *Quantile–Quantile Plot*, there is a approximately straight line with small departures from normality which are not concerning. There are presence of outliers but they are not concerning with our former test.

In the *Residuals vs Leverage Plot*, there are no outlying values at the upper right or lower right corner, which indicates no influential cases.

Problem 5

Using the 'final' model chosen in part 4, focus on MSE to test the model predictive ability:

a) Use a 10-fold cross-validation (10 repeats).

```
# define training control
train_control = trainControl(method ="cv", number = 10, savePredictions = TRUE)
```

```r
# train the model
model = train(life_exp ~ murder + hs_grad + frost + population,
              data = without11,
              trControl = train_control,
              method = 'lm')

model
```

```
## Linear Regression
##
## 49 samples
##  4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 44, 44, 44, 44, 43, 44, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.6930092  0.7966426  0.6060899
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# Examine model prediction for each fold
predictions = model$resample
```

Comment

For 10-fold cross-validation, the `RMSE` is about 0.7, `Rsquare` is about 0.8, `MAE` is about 0.6. There are about 80% of variation explained by our model.

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

Do it step by step

i) Perform a regression model with the original sample; calculate predicted values residuals

```r
fit_final_without11 = lm(life_exp ~ murder + hs_grad + frost + population, data = without11)

residuals = resid(fit_final_without11)
```

ii) Randomly resample the residuals (with replacement)

```r
set.seed(1)
boot_sample = function(df) {
  sample_frac(df, replace = TRUE)
}
wh = sample(1:length(residuals), replace = TRUE)
```

iii) Construct new values by adding the original predicted values to the bootstrap residuals

```r
pred_y = predict(fit_final_without11)
boot.Y = pred_y + residuals[wh]
```

iv) Regress on the original X variable(s).

```r
boot.lm = lm(boot.Y ~ murder + hs_grad + frost + population, data = without11)
anova(boot.lm)["Residuals","Mean Sq"]
```

```
## [1] 0.3753003
```

v) Repeat steps (ii) – (iv) 10 times and 1,000 times

```
residual_reg = function(model) {

  residuals = resid(model)

  wh = sample(1:length(residuals), replace = TRUE)

  pred_y = predict(fit_final_without11)

  boot.Y = pred_y + residuals[wh]

  boot.lm = lm(boot.Y ~ murder + hs_grad + frost + population, data = without11)

  MSE = anova(boot.lm)["Residuals","Mean Sq"]

  as.numeric(MSE)
}
```

Rerun 10 times and 1000 times

```
set.seed(2)
boot_straps_10 = data_frame(
  strap_number = 1:10,
  strap_sample = rerun(10, residual_reg(fit_final_without11))) %>%
  mutate(strap_sample, MSE = as.numeric(strap_sample)) %>%
  select(strap_number, MSE)

boot_straps_1000 = data_frame(
  strap_number = 1:1000,
  strap_sample = rerun(1000, residual_reg(fit_final_without11))) %>%
  mutate(strap_sample, MSE = as.numeric(strap_sample)) %>%
  select(strap_number, MSE)
```

vi) Summarize the MSE for all repetitions.

```
summary(boot_straps_10$MSE)
```

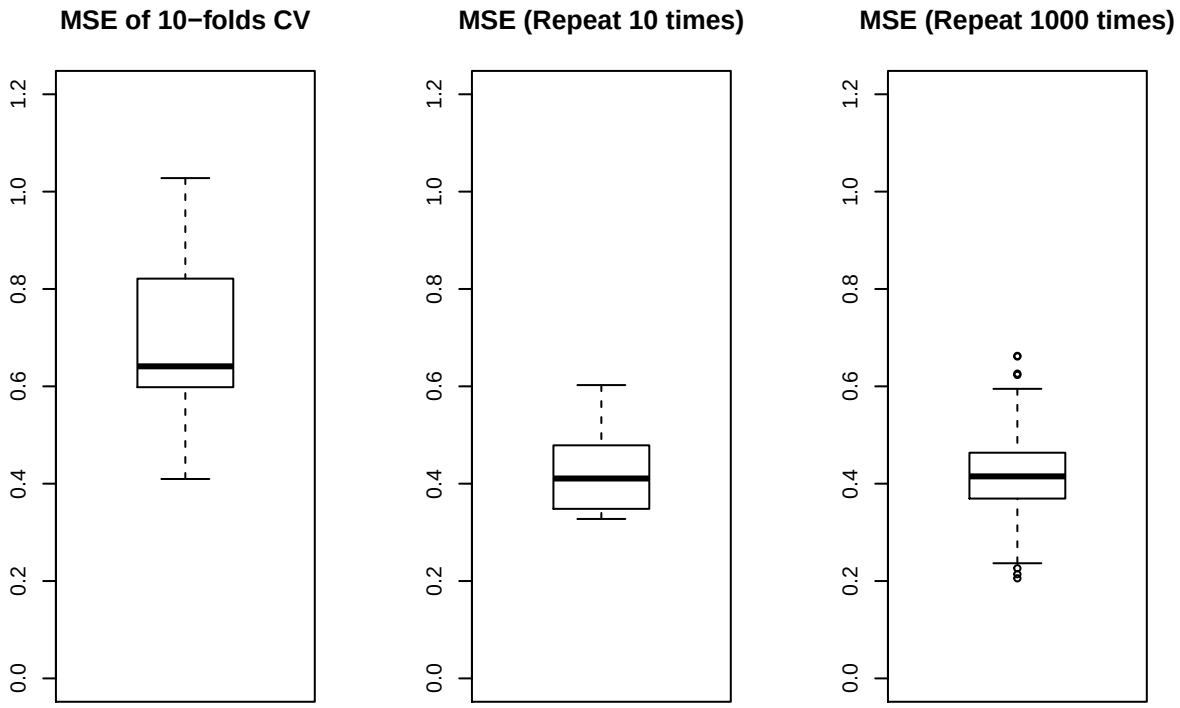
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3275  0.3537  0.4106  0.4316  0.4743  0.6026
```

```
summary(boot_straps_1000$MSE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2056  0.3696  0.4151  0.4160  0.4636  0.6625
```

c) In a paragraph, compare the MSE values generated by the two methods a) and b). Briefly comment on the differences and your recommendation for assessing model performance.

```
par(mfrow = c(1,3))
boxplot(model$resample$RMSE, main = "MSE of 10-folds CV", ylim = c(0, 1.2))
boxplot(boot_straps_10$MSE, main = "MSE (Repeat 10 times)", ylim = c(0, 1.2) )
boxplot(boot_straps_1000$MSE, main = "MSE (Repeat 1000 times)", ylim = c(0, 1.2))
```



Comment

The bootstrap methods has a lower MSE compared to 10-folds Cross Validation. Drawing samples with replacement from the observed data mimics drawing samples from the underlying distribution. Recalculating regression parameters for the 'new' samples gives an idea of the distribution of regression coefficients. Also, bootstrap doesn't require any assumption. Therefore, I would recommend bootstrap method.