HW5_hh2767

Haoran Hu

2018-11-30

```
life_exp = state.x77 %>%
  as.tibble()
```

Problem1.Descriptive statistics

In this part, I will use some functions that I created in previous homework to generate descriptive statistics. There seems to be no categorical variables, i.e., all variables can be treated as continuous variable.

```
#A function for descirptive statistics for continuous variables. The input is a dataframe.
descriptive_statistics_cont = function(x){
  data_mean = purrr::map_df(x, mean, na.rm = TRUE)
  data_quantile = purrr::map_df(x, quantile, na.rm = TRUE)
  data_sd = purrr::map_df(x, sd, na.rm = TRUE)
  n = colSums(!is.na(x))
  na = colSums(is.na(x))
  variable_names = colnames(x)

 output1 = tibble(
Variables = variable_names,
Minimun = as.numeric(data_quantile[1,]),
`1st_quantile` = as.numeric(data_quantile[2,]),
Median = as.numeric(data_quantile[3,]),
Mean = as.numeric(round(data_mean[1,], 3)),
`3rd_quantile` = as.numeric(data_quantile[4,]),
Max = as.numeric(data_quantile[5,])
)

 output2 = tibble(
Variables = variable_names,
Standard_deviation = as.numeric(round(data_sd, 3)),
Sample_sizes = n,
Number_of_NA = na
)

 list(output1, output2)
}

#create a dunction to get mode
getmode = function(v) {
   uniqv = unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

#A function for descriptive statistics for discrete variables, the input is a vector
descriptive_statistics_disc = function(x){
```

```
  levels = unique(x)
  n_levels = length(levels)
  levels = levels %>% tibble()
  n_total = length(x)
  if (n_levels <= 2) {
    level1 = as.numeric(levels[1,1])
    level2 = as.numeric(levels[2,1])
    n_level1 = sum(x == level1)
    n_level2 = sum(x == level2)
    tibble(`levels` = c(level1, level2),
           proportion = c(round(n_level1 / n_total, 4), round(n_level2 / n_total, 4))
           )

  } else {
    x_quantile = quantile(x) %>% tibble()
    x_mode = getmode(x)
    tibble(
      min = as.numeric(x_quantile[1,]),
      `1st_quantile` = as.numeric(x_quantile[2,]),
      median = as.numeric(x_quantile[3,]),
      mode = as.numeric(x_mode),
      `3rd_quantile` = as.numeric(x_quantile[4,]),
      max = as.numeric(x_quantile[5,])
    )
  }
}
```

```
life_exp_descrip = descriptive_statistics_cont(life_exp)

discrip_table1 = life_exp_descrip[[1]]

discrip_table2 = life_exp_descrip[[2]]

knitr::kable(discrip_table1)
```

Variables	Minimun	1st_quantile	Median	Mean	3rd_quantile	Max
Population	365.00	1079.5000	2838.500	4246.420	4968.5000	21198.0
Income	3098.00	3992.7500	4519.000	4435.800	4813.5000	6315.0
Illiteracy	0.50	0.6250	0.950	1.170	1.5750	2.8
Life Exp	67.96	70.1175	70.675	70.879	71.8925	73.6
Murder	1.40	4.3500	6.850	7.378	10.6750	15.1
HS Grad	37.80	48.0500	53.250	53.108	59.1500	67.3
Frost	0.00	66.2500	114.500	104.460	139.7500	188.0
Area	1049.00	36985.2500	54277.000	70735.880	81162.5000	566432.0

```
knitr::kable(discrip_table2)
```

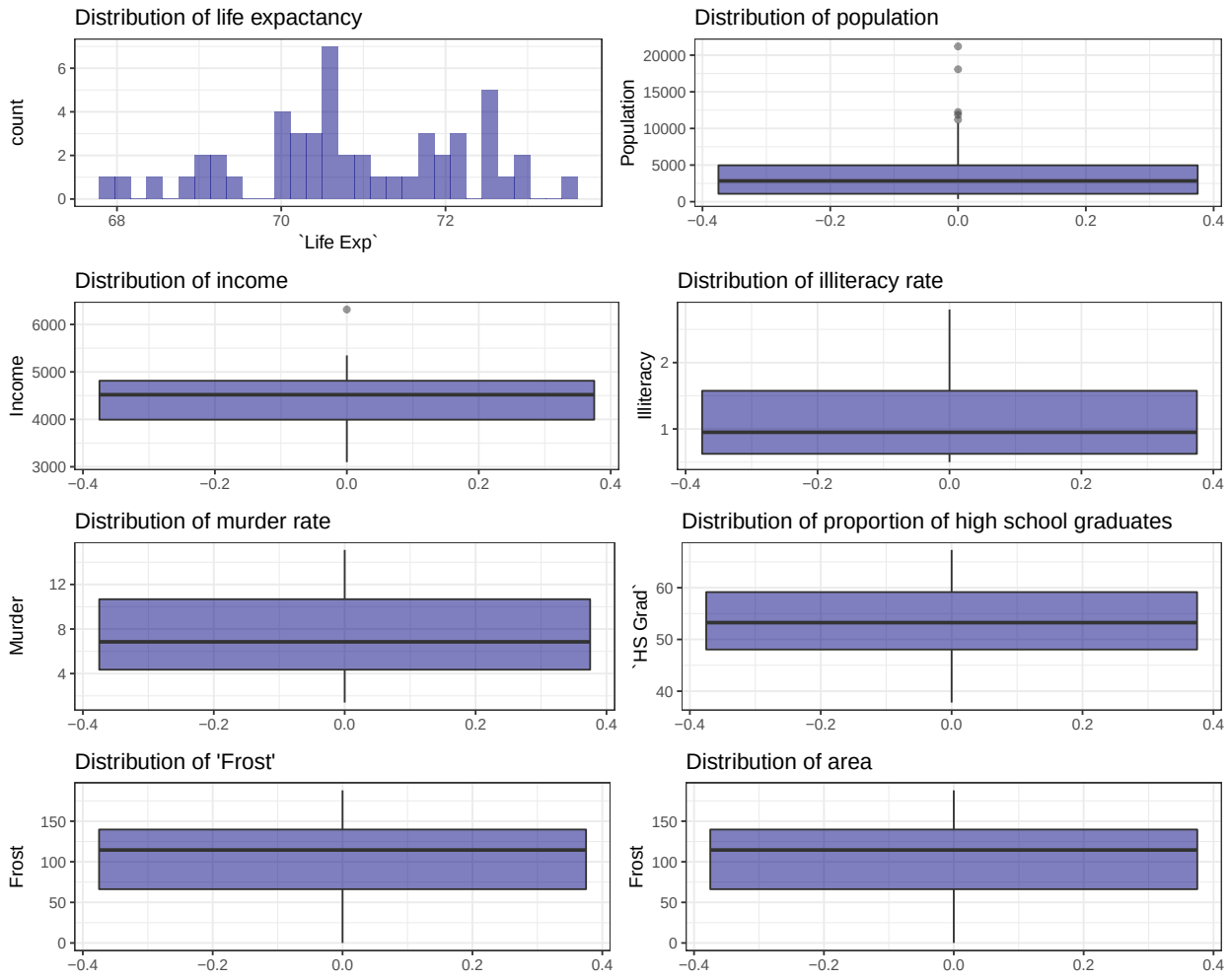
Variables	Standard_deviation	Sample_sizes	Number_of_NA
Population	4464.491	50	0
Income	614.470	50	0

Variables	Standard_deviation	Sample_sizes	Number_of_NA
Illiteracy	0.610	50	0
Life Exp	1.342	50	0
Murder	3.692	50	0
HS Grad	8.077	50	0
Frost	51.981	50	0
Area	85327.300	50	0

```
life_plot = ggplot(life_exp, aes(x = `Life Exp`)) + geom_histogram(fill = "navy", alpha = 0.5) + labs(ti
population_plot = ggplot(life_exp, aes(y = Population)) + geom_boxplot(fill = "navy", alpha = 0.5) + lab
income_plot =  ggplot(life_exp, aes(y = Income)) + geom_boxplot(fill = "navy", alpha = 0.5) + labs(title
illiteracy_plot =  ggplot(life_exp, aes(y = Illiteracy)) + geom_boxplot(fill = "navy", alpha = 0.5) + la
murder_plot = ggplot(life_exp, aes(y = Murder)) + geom_boxplot(fill = "navy", alpha = 0.5) + labs(title
hs_grad_plot = ggplot(life_exp, aes(y = `HS Grad`)) + geom_boxplot(fill = "navy", alpha = 0.5) + labs(ti
frost_plot = ggplot(life_exp, aes(y = Frost)) + geom_boxplot(fill = "navy", alpha = 0.5) + labs(title =
area_plot = ggplot(life_exp, aes(y = Frost)) + geom_boxplot(fill = "navy", alpha = 0.5) + labs(title =

(life_plot + population_plot) / (income_plot + illiteracy_plot) /  (murder_plot + hs_grad_plot) / (frost
```

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



Problem2.Building model

Next, I will use automatic procedures to select subsets and build several good models. The original model is:

```
origin_fit <- lm(`Life Exp` ~ ., data = life_exp)
```

Backward elimination

First, let's try backward elimination.In this part, we set$\alpha_{crit} = 0.10$.

```
summary(origin_fit)
```

```
##
## Call:
## lm(formula = `Life Exp` ~ ., data = life_exp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.48895 -0.51232 -0.02747  0.57002  1.49447
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.094e+01  1.748e+00  40.586  < 2e-16 ***
## Population   5.180e-05  2.919e-05   1.775   0.0832 .
## Income      -2.180e-05  2.444e-04  -0.089   0.9293
## Illiteracy   3.382e-02  3.663e-01   0.092   0.9269
## Murder      -3.011e-01  4.662e-02  -6.459 8.68e-08 ***
## `HS Grad`    4.893e-02  2.332e-02   2.098   0.0420 *
## Frost       -5.735e-03  3.143e-03  -1.825   0.0752 .
## Area        -7.383e-08  1.668e-06  -0.044   0.9649
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7448 on 42 degrees of freedom
## Multiple R-squared:  0.7362, Adjusted R-squared:  0.6922
## F-statistic: 16.74 on 7 and 42 DF,  p-value: 2.534e-10
```

```
#No Area
step1 = update(origin_fit, . ~ . -Area)
summary(step1)
```

```
##
## Call:
## lm(formula = `Life Exp` ~ Population + Income + Illiteracy +
##     Murder + `HS Grad` + Frost, data = life_exp)
##
## 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 Illiteracy
step2 = update(step1, . ~ . -Illiteracy)
summary(step2)
```

```
##
## Call:
## lm(formula = `Life Exp` ~ Population + Income + Murder + `HS Grad` +
##     Frost, data = life_exp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4892 -0.5122 -0.0329  0.5645  1.5166
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.107e+01  1.029e+00  69.067  < 2e-16 ***
## Population   5.115e-05  2.709e-05   1.888   0.0657 .
## Income      -2.477e-05  2.316e-04  -0.107   0.9153
## Murder      -3.000e-01  3.704e-02  -8.099 2.91e-10 ***
## `HS Grad`    4.776e-02  1.859e-02   2.569   0.0137 *
## Frost       -5.910e-03  2.468e-03  -2.395   0.0210 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7277 on 44 degrees of freedom
## Multiple R-squared:  0.7361, Adjusted R-squared:  0.7061
## F-statistic: 24.55 on 5 and 44 DF,  p-value: 1.019e-11
```

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

```
##
## Call:
## lm(formula = `Life Exp` ~ Population + Murder + `HS Grad` + Frost,
##     data = life_exp)
##
```

```
## 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
```

Therefore, using backward elimination, we selected 'Population', 'Murder', 'HS Grad' and 'Frost' as predictors. The model is given by:

`Life Exp` ~ `Population` + `Murder` + `HS Grad` + `Frost`.

The fitted regression line is:

`Life Exp` = 71 + 0.00005`Population` - 0.3`Murder` + 0.047`Hs Grad` - 0.006`Frost`

Forward elimination

Second, let's use forward elimination. In this part, we also set$\alpha_{crit} = 0.10$.

```
# Step 1:  Fit simple linear regressions for all variables,look for the variable with lowest p-value
fit1 <- lm(`Life Exp` ~ Population, data = life_exp)
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 = life_exp)
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 = life_exp)
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 = life_exp)
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 = life_exp)
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 = life_exp)
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
```

```
fit7 <- lm(`Life Exp` ~ Area, data = life_exp)
tidy(fit7)
```

```
## # A tibble: 2 x 5
##   term            estimate  std.error statistic  p.value
##   <chr>              <dbl>      <dbl>     <dbl>    <dbl>
## 1 (Intercept)  71.0        0.249        285.    3.46e-79
## 2 Area         -0.00000169 0.00000226    -0.748 4.58e- 1
```

```
# Enter first the one with the lowest p-value: Murder
forward1 <- lm(`Life Exp`~ Murder, data = life_exp)
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
```

```
# Step 2: Enter the one with the lowest p-value in the rest
fit1 <- update(forward1, . ~ . +Population, data = life_exp)
tidy(fit1)
```

```
## # 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
```

```
fit2 <- update(forward1, . ~ . +Income, data = life_exp)
tidy(fit2)
```

```
## # 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
```

```
fit3 <- update(forward1, . ~ . +Illiteracy, data = life_exp)
tidy(fit3)
```

```
## # 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
```

```
fit4 <- update(forward1, . ~ . +`HS Grad`, data = life_exp)
tidy(fit4)
```

```
## # 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
```

```
fit5 <- update(forward1, . ~ . + Frost, data = life_exp)
tidy(fit5)
```

```
## # 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
```

```r
fit6 <- update(forward1, . ~ . +Area, data = life_exp)
tidy(fit6)
```

```
## # A tibble: 3 x 5
##   term            estimate  std.error statistic  p.value
##   <chr>              <dbl>      <dbl>     <dbl>    <dbl>
## 1 (Intercept)  72.9        0.275        265.    2.73e-76
## 2 Murder       -0.290      0.0338        -8.58  3.47e-11
## 3 Area          0.00000118 0.00000146     0.806 4.24e- 1
```

```r
# Enter the one with the lowest p-value: 'HS Grad'
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
```

```r
# Step 3: Enter the one with the lowest p-value in the rest
fit1 <- update(forward2, . ~ . +Population, data = life_exp)
tidy(fit1)
```

```
## # 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
```

```r
fit2 <- update(forward2, . ~ . +Income, data = life_exp)
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
```

```r
fit3 <- update(forward2, . ~ . +Illiteracy, data = life_exp)
tidy(fit3)
```

```
## # 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
```

```
fit4 <- update(forward2, . ~ . +Frost, data = life_exp)
tidy(fit4)
```

```
## # 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
```

```
fit5 <- update(forward2, . ~ . +Area, data = life_exp)
tidy(fit5)
```

```
## # A tibble: 4 x 5
##   term            estimate  std.error statistic  p.value
##   <chr>              <dbl>      <dbl>     <dbl>    <dbl>
## 1 (Intercept)  69.9        1.16          60.1   2.30e-45
## 2 Murder       -0.224      0.0404        -5.56  1.30e- 6
## 3 `HS Grad`     0.0504     0.0190         2.65  1.10e- 2
## 4 Area         -0.00000106 0.00000162    -0.658 5.14e- 1
```

```
# Enter the one with the lowest p-value: Frost
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
```

```
# Step 4: Enter the one with the lowest p-value in the rest
fit1 <- update(forward3, . ~ . +Population, data = life_exp)
tidy(fit1)
```

```
## # 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
```

```
fit2 <- update(forward3, . ~ . +Income, data = life_exp)
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, . ~ . +Illiteracy, data = life_exp)
tidy(fit3)
```

```
## # 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
```

```
fit4 <- update(forward3, . ~ . +Area, data = life_exp)
tidy(fit4)
```

```
## # A tibble: 5 x 5
##   term             estimate  std.error statistic  p.value
##   <chr>               <dbl>      <dbl>     <dbl>    <dbl>
## 1 (Intercept)  70.9         1.15          61.7   3.92e-45
## 2 Murder       -0.279       0.0427        -6.52  5.34e- 8
## 3 `HS Grad`     0.0519      0.0179         2.91  5.66e- 3
## 4 Frost        -0.00682     0.00251       -2.71  9.40e- 3
## 5 Area         -0.000000329 0.00000154    -0.214 8.32e- 1
```

```
# Enter the one with the lowest p-value: Population
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
```

```r
# Step 5: Enter the one with the lowest p-value in the rest
fit1 <- update(forward4, . ~ . +Income, data = life_exp)
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
```

```r
fit2 <- update(forward4, . ~ . +Illiteracy, data = life_exp)
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
```

```r
fit3 <- update(forward4, . ~ . +Area, data = life_exp)
tidy(fit3)
```

```
## # A tibble: 6 x 5
##   term        estimate  std.error statistic  p.value
##   <chr>          <dbl>      <dbl>     <dbl>    <dbl>
## 1 (Intercept)  7.10e+1 1.12         63.7    5.81e-45
## 2 Murder      -2.99e-1 0.0428       -7.00   1.16e- 8
## 3 `HS Grad`    4.69e-2 0.0175        2.68   1.03e- 2
## 4 Frost       -5.93e-3 0.00248      -2.39   2.11e- 2
## 5 Population   5.00e-5 0.0000255     1.96   5.61e- 2
## 6 Area        -5.79e-8 0.00000150   -0.0386 9.69e- 1
```

```r
# P-value of all new added variables are larger than 0.10, which means that they
# are not significant predictor, and we stop here.
```

Therefore, using backward elimination, we selected 'Murder', 'HS Grad', 'Frost' and 'Population' as predictors. The model is given by: `Life Exp` ~ `Population` + `Murder` + `HS Grad` + `Frost`

The fitted regression line is:

`Life Exp` = 71 - 0.3`Murder` + 0.047`Hs Grad` - 0.006`Frost`+ 0.00005`Population`

Stepwise elimination

Next, I will use stepwise elimination.

```
step(origin_fit, 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
```

```
##
## Call:
## lm(formula = `Life Exp` ~ Population + Murder + `HS Grad` + Frost,
##     data = life_exp)
##
## Coefficients:
## (Intercept)   Population       Murder    `HS Grad`        Frost
##   7.103e+01    5.014e-05   -3.001e-01    4.658e-02   -5.943e-03
```

Using stepwise elimination, we also selected 'Murder', 'HS Grad', 'Frost' and 'Population' as predictors. The model is given by:

`Life Exp` ~ `Population` + `Murder` + `HS Grad` + `Frost`

The fitted regression line is:

`Life Exp` = 71 + 0.00005`Population` - 0.3`Murder` + 0.047`Hs Grad` - 0.006`Frost`

a)

The three automatic procedures generates the same models:

`Life Exp` ~ `Population` + `Murder` + `HS Grad` + `Frost`

The fitted regression line is also the same:

`Life Exp` = 71 + 0.00005`Population` - 0.3`Murder` + 0.047`Hs Grad` - 0.006`Frost`

b)

The `Population` variable is a close call. In both forward and backward elimination, if we set$\alpha_{crit} = 0.05$, the variable will not be included in the model. The automatic procedures, however, are of exploratory purpose, and we should be less stringent. Therefore, we set$\alpha_{crit} = 0.10$ and in this case, we will keep `Population` variable in the model.

c)

In this part, I will check the correlation between "Illiteracy" and "HS Grad".

```
cor(life_exp$Illiteracy, life_exp$`HS Grad`)
```

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

The correlation between "Illiteracy" and "HS Grad" is -0.66, which means there is a relatively strong linear relationship between the two variables. They are negetively correlated because high high school graduates proportion leads to low illiteracy rate. If we include both of them into the model, they may cause collinearty problem. Fortunately, none of my 'subsets' contains both.

Problem3.Using criterion-base procedures to select model

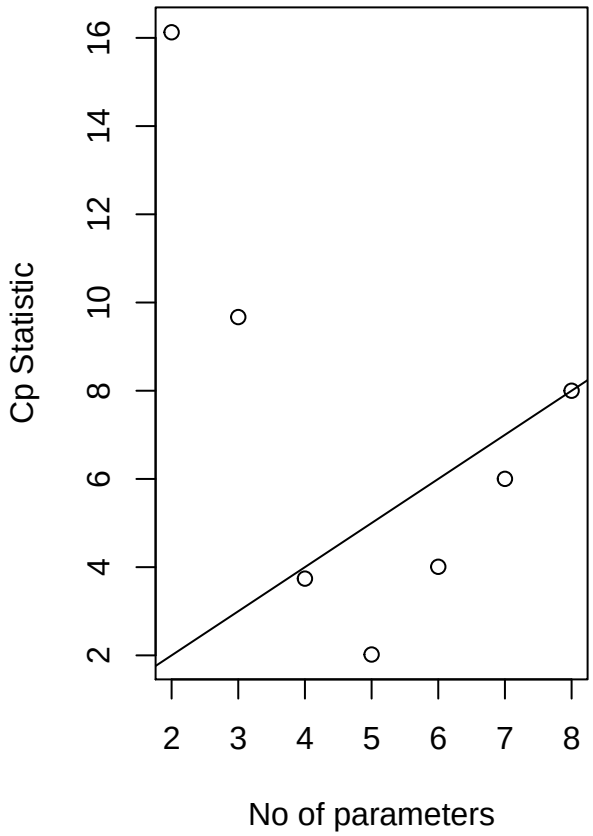
```r
# Summary of models for each size (one model per size)
best_model = regsubsets(`Life Exp` ~ ., data = life_exp)
   (rs = summary(best_model))
```

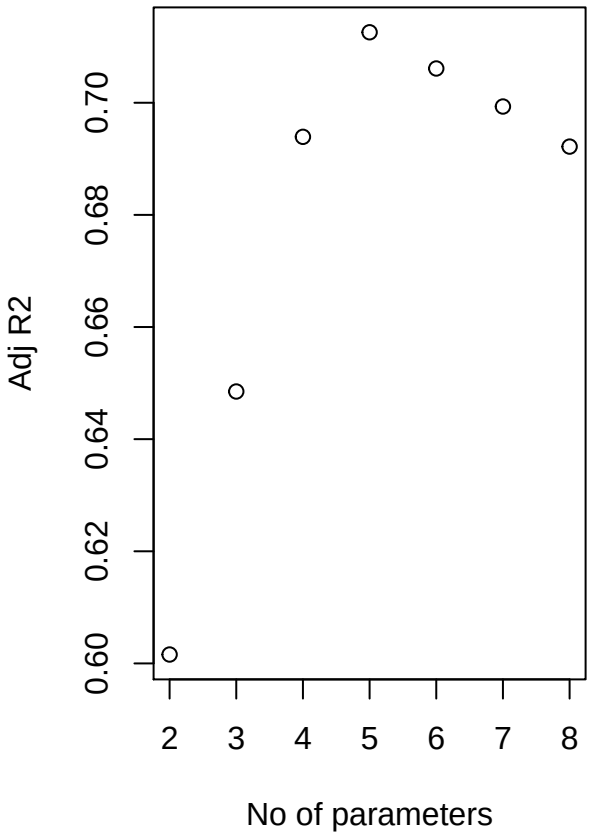
```
## Subset selection object
## Call: regsubsets.formula(`Life Exp` ~ ., data = life_exp)
## 7 Variables  (and intercept)
##            Forced in Forced out
## Population     FALSE      FALSE
## Income         FALSE      FALSE
## Illiteracy     FALSE      FALSE
## Murder         FALSE      FALSE
## `HS Grad`      FALSE      FALSE
## Frost          FALSE      FALSE
## Area           FALSE      FALSE
## 1 subsets of each size up to 7
## Selection Algorithm: exhaustive
##          Population Income Illiteracy Murder `HS Grad` Frost Area
## 1  ( 1 ) " "        " "    " "        "*"    " "       " "   " "
## 2  ( 1 ) " "        " "    " "        "*"    "*"       " "   " "
## 3  ( 1 ) " "        " "    " "        "*"    "*"       "*"   " "
## 4  ( 1 ) "*"        " "    " "        "*"    "*"       "*"   " "
## 5  ( 1 ) "*"        "*"    " "        "*"    "*"       "*"   " "
## 6  ( 1 ) "*"        "*"    "*"        "*"    "*"       "*"   " "
## 7  ( 1 ) "*"        "*"    "*"        "*"    "*"       "*"   "*"
```

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

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

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





We can also look at it in tabular form.

```
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)
}
# Select the 'best' model of all subsets for the full model
round(best(origin_fit, nbest = 1), 4)
```

```
##   p (Intercept) Population Income Illiteracy Murder `HS Grad` Frost Area
## 1 1           1          0      0          0      1         0     0    0
## 2 2           1          0      0          0      1         1     0    0
## 3 3           1          0      0          0      1         1     1    0
## 4 4           1          1      0          0      1         1     1    0
## 5 5           1          1      1          0      1         1     1    0
## 6 6           1          1      1          1      1         1     1    0
## 7 7           1          1      1          1      1         1     1    1
##       rss    rsq  adjr2      cp      bic
## 1 34.4613 0.6097 0.6016 16.1268 -39.2205
## 2 29.7704 0.6628 0.6485  9.6699 -42.6247
## 3 25.3716 0.7127 0.6939  3.7399 -46.7068
## 4 23.3080 0.7360 0.7126  2.0197 -47.0364
```

```
## 5 23.3020 0.7361 0.7061  4.0087 -43.1374
## 6 23.2982 0.7361 0.6993  6.0020 -39.2334
## 7 23.2971 0.7362 0.6922  8.0000 -35.3237
```

From the plots, we can see that for the model with four parameters, adjusted R-squared is close to the maximum, meaning the model has a good fit of the data. In addition, from the left panel, we konw that for the model with four parameters, number of parameters is close to and less than Cp Statistic, meaning the model has low bias. The table above also shows the BIC of model with 4 parameters is only slightly greater than that of model with 5 parameters. Considering about parsimony, the model with 4 parameters is better. Therefore, using criterion-based procedures, we choose 'Murder', 'HS Grad', and 'Frost' as the best subset of predictors. The model is given by: `Life Exp` ~ `Murder` + `HS Grad` + `Frost`

Problem4 recommendation for a final model and doing model diagnostics

The model built in part 2 is given by:

Model_L = `Life Exp` ~ `Population` + `Murder` + `HS Grad` + `Frost`

The model built in part 3 is given by:

Model_S = `Life Exp` ~ `Murder` + `HS Grad` + `Frost`

Model_S is nested in Model_L, so we use ANOVA to choose one from them.

```
Model_S = lm(`Life Exp` ~ Murder + `HS Grad` + Frost, data = life_exp)
Model_L = lm(`Life Exp` ~ Population + Murder + `HS Grad` + Frost, data = life_exp)
anova(Model_S, Model_L)
```

```
## Analysis of Variance Table
##
## Model 1: `Life Exp` ~ Murder + `HS Grad` + Frost
## Model 2: `Life Exp` ~ Population + Murder + `HS Grad` + Frost
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     46 25.372
## 2     45 23.308  1    2.0636 3.9841 0.05201 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the partial F-test, the p-value = $0.052 > 0.05$. Then, we fail to reject the null and conclude that the larger model is not superior. Therefore, we recommend the model built in part 3. That makes sence, because population seems not to have an association with life expectance. The model is given by:

Model_S = `Life Exp` ~ `Murder` + `HS Grad` + `Frost`

To get the fitted regression line:

```
summary(Model_S)
```

```
##
## Call:
## lm(formula = `Life Exp` ~ Murder + `HS Grad` + Frost, data = life_exp)
##
```

```
## 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
```

Then, the fitted regression line is:

`Life Exp` = 71.04 -0.28`Murder` + 0.05`HS Grad` - 0.007`Frost`

a)

I will use `influence.measures` and diagnostic plots to identify outliers and influential points.

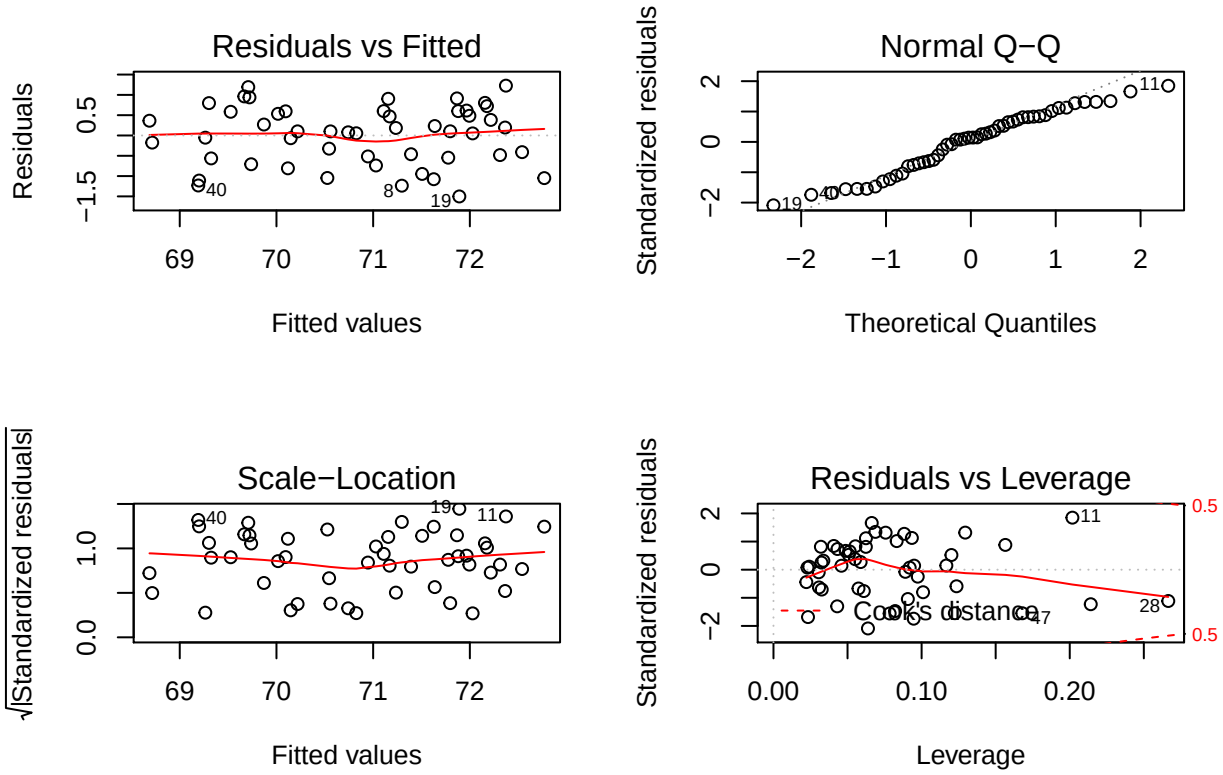
```
influence.measures(Model_S)
```

```
## Influence measures of
##   lm(formula = `Life Exp` ~ Murder + `HS Grad` + Frost, data = life_exp) :
##
##       dfb.1_  dfb.Mrdr  dfb..HSG dfb.Frst   dffit cov.r   cook.d    hat
## 1   0.020962  0.093207 -0.033585 -0.04133  0.1911 1.212 9.28e-03 0.1201
## 2   0.568667 -0.510752 -0.457047 -0.28342 -0.6450 1.216 1.03e-01 0.2142
## 3   0.019269  0.120753 -0.254344  0.50352 -0.5873 1.005 8.35e-02 0.1224
## 4   0.270008 -0.031940 -0.278704 -0.04440  0.3794 1.013 3.54e-02 0.0757
## 5  -0.175463  0.089072  0.274620 -0.23119  0.3784 1.209 3.60e-02 0.1565
## 6  -0.288857  0.192455  0.244268  0.21372  0.3995 1.037 3.93e-02 0.0883
## 7   0.054134 -0.094926 -0.024045  0.00797  0.1505 1.104 5.73e-03 0.0487
## 8  -0.051896  0.085444 -0.016772  0.06084 -0.2659 0.866 1.69e-02 0.0233
## 9  -0.001171  0.003564  0.013987 -0.03479  0.0459 1.204 5.37e-04 0.0949
## 10 -0.014692 -0.042057  0.029926 -0.00842 -0.0806 1.203 1.66e-03 0.0974
## 11 -0.026095 -0.306720  0.440340 -0.84137  0.9557 1.004 2.16e-01 0.2018
## 12 -0.015525 -0.007337  0.026574  0.00224  0.0589 1.120 8.86e-04 0.0337
## 13 -0.036507  0.066083  0.012430  0.05322  0.0894 1.142 2.04e-03 0.0551
## 14  0.000966  0.000762 -0.001297  0.00387  0.0112 1.117 3.22e-05 0.0230
## 15  0.014268 -0.042757  0.003049 -0.00310  0.0671 1.153 1.15e-03 0.0588
## 16 -0.015076 -0.066007  0.070249 -0.04401  0.1732 1.069 7.55e-03 0.0407
## 17  0.211172  0.041321 -0.279943  0.09706  0.3632 1.077 3.28e-02 0.0934
## 18 -0.079217 -0.059140  0.061992  0.12265 -0.2678 1.148 1.81e-02 0.1010
## 19 -0.218718  0.306937  0.178345 -0.17777 -0.5671 0.783 7.44e-02 0.0638
## 20  0.002554 -0.019630 -0.002494 -0.00688 -0.0660 1.098 1.11e-03 0.0222
## 21 -0.040226  0.111588 -0.024877  0.07519 -0.1641 1.114 6.81e-03 0.0574
## 22 -0.175893  0.297319  0.073482  0.21085  0.3683 0.999 3.33e-02 0.0688
```

```
## 23  0.068768 -0.157332 -0.031176  0.06797  0.2891 1.043 2.08e-02 0.0624
## 24 -0.175594 -0.135141  0.212086  0.05309 -0.4617 0.954 5.16e-02 0.0784
## 25  0.019871  0.055721 -0.045037  0.05464  0.1476 1.064 5.48e-03 0.0320
## 26  0.077900 -0.003224 -0.077333 -0.12998 -0.2785 0.982 1.91e-02 0.0430
## 27  0.014233 -0.066931  0.015439 -0.00120  0.1216 1.123 3.76e-03 0.0509
## 28  0.546692 -0.544474 -0.370503 -0.43122 -0.6731 1.334 1.13e-01 0.2665
## 29 -0.009964  0.051541  0.011808 -0.09893 -0.1927 1.106 9.37e-03 0.0610
## 30 -0.058490  0.061791  0.036497  0.00717 -0.1118 1.088 3.17e-03 0.0308
## 31 -0.016496  0.021758  0.011082  0.01363  0.0304 1.142 2.36e-04 0.0459
## 32 -0.058901  0.103302  0.049772  0.00326  0.1553 1.089 6.09e-03 0.0432
## 33 -0.013823 -0.002992  0.017465 -0.00344 -0.0238 1.198 1.44e-04 0.0890
## 34  0.284799 -0.281741 -0.291841  0.20760  0.5123 1.075 6.45e-02 0.1295
## 35 -0.000476  0.003232 -0.000725  0.00689  0.0166 1.118 7.02e-05 0.0242
## 36  0.024362 -0.023553 -0.009749 -0.02317  0.0458 1.122 5.35e-04 0.0324
## 37 -0.032765  0.120935 -0.069453  0.18077 -0.2200 1.209 1.23e-02 0.1236
## 38 -0.066381  0.038440  0.066425 -0.03671 -0.1290 1.080 4.20e-03 0.0326
## 39  0.044825 -0.041090 -0.037178 -0.00240  0.0531 1.234 7.21e-04 0.1167
## 40 -0.341829 -0.065804  0.407941 -0.01265 -0.5767 0.918 7.94e-02 0.0946
## 41  0.011120 -0.013620 -0.009913  0.00735  0.0229 1.202 1.33e-04 0.0919
## 42  0.105152  0.035934 -0.124388 -0.00124  0.2084 1.099 1.09e-02 0.0624
## 43  0.024292  0.164316  0.006729 -0.18702  0.4514 0.913 4.90e-02 0.0663
## 44 -0.182657  0.013593  0.236391  0.00333  0.3063 1.088 2.34e-02 0.0832
## 45 -0.032136  0.020000  0.011745  0.10378  0.1502 1.110 5.72e-03 0.0515
## 46 -0.005136 -0.003355  0.005614  0.00021 -0.0162 1.125 6.68e-05 0.0305
## 47  0.007009  0.309432 -0.327995  0.58047 -0.7075 1.059 1.21e-01 0.1680
## 48 -0.390621  0.198027  0.382489 -0.00575 -0.4458 0.981 4.84e-02 0.0821
## 49  0.086372 -0.120195 -0.061795  0.03991  0.2006 1.088 1.01e-02 0.0553
## 50  0.226031 -0.165164 -0.175086 -0.20503 -0.3301 1.091 2.72e-02 0.0907
##    inf
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11   *
## 12
## 13
## 14
## 15
## 16
## 17
## 18
## 19
## 20
## 21
## 22
## 23
## 24
## 25
```

```
## 26
## 27
## 28   *
## 29
## 30
## 31
## 32
## 33
## 34
## 35
## 36
## 37
## 38
## 39
## 40
## 41
## 42
## 43
## 44
## 45
## 46
## 47
## 48
## 49
## 50
```

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



The result shows that the leverage values(hat values) for all observations are less than 0.3, i.e., there's no observation with high leverage value. In addition, none of them has a Cook's distance greater than 0. We can be more stringent about influential points, though — sample point 11 is an outlier in almost each plot, and sample point 28 has the highest leverage value. These two points are possiblly problematic. By looking at the data, we can find out that `Frost` value for observation 11 is 0. Since observation 11 represents Hawaii and the temperature in Hawaii is high all year round, that should not be a data error. Observation 28 represents Nevada, whose population is very small and `Frost` value is very high. That also makes sence and should not be a data error. Next, we remove these data points and see how will they affect the model.

```
life_exp_noinf = life_exp[-c(11, 28),]

Model_S_noinf = lm(`Life Exp` ~ Murder + `HS Grad` + Frost, data = life_exp_noinf)

summary(Model_S_noinf)
```

```
##
## Call:
## lm(formula = `Life Exp` ~ Murder + `HS Grad` + Frost, data = life_exp_noinf)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.51727 -0.47272  0.06809  0.54146  1.26417
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 70.444020   1.056398  66.683  < 2e-16 ***
```

```
## Murder      -0.248414   0.040082  -6.198 1.73e-07 ***
## `HS Grad`    0.049525   0.015734   3.148  0.00295 **
## Frost       -0.003572   0.002771  -1.289  0.20422
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7164 on 44 degrees of freedom
## Multiple R-squared:  0.7085, Adjusted R-squared:  0.6886
## F-statistic: 35.64 on 3 and 44 DF,  p-value: 7.636e-12
```

In the new model, `Frost` becomes non-significant, and that's because observation 11 and observation 28 are both outliers in `Frost`.

Discussion:

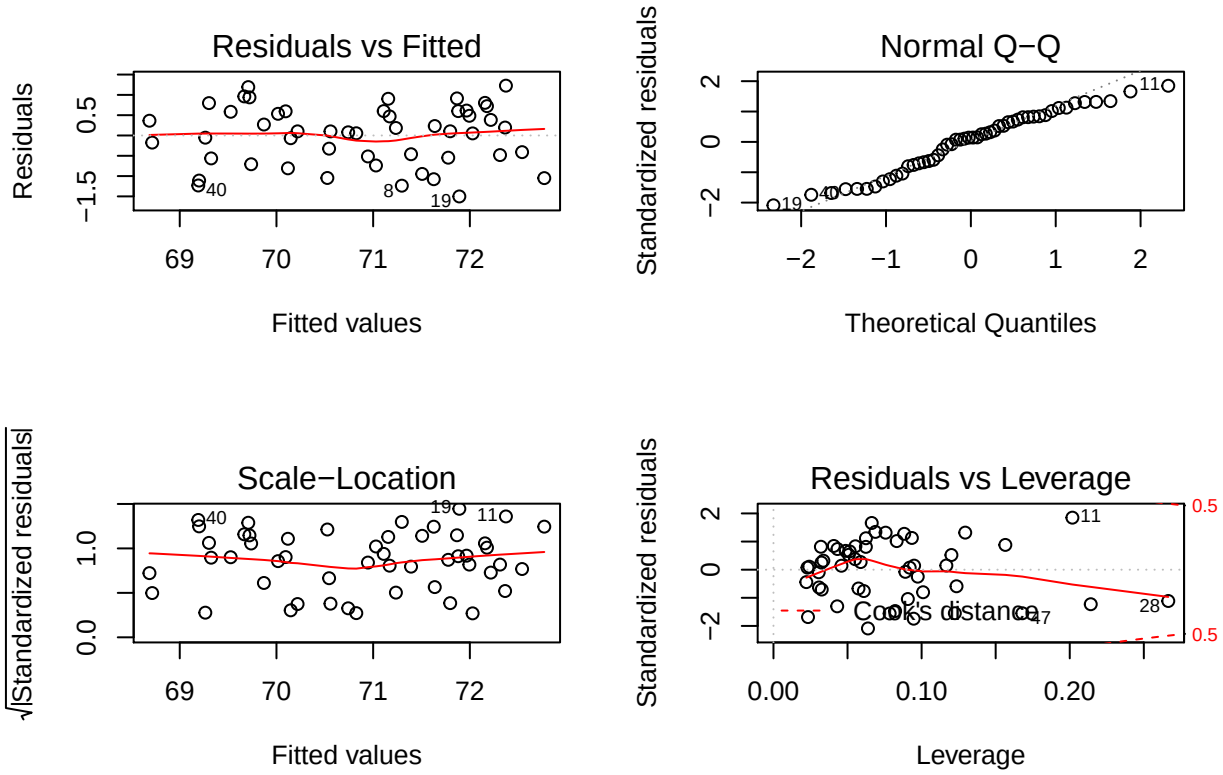
In order for the sample to be representative of all the states, we had better keep the two data points in the model.

In later analysis, will keep the two observations.

b)

Next, let's look at diagnostic plots again.

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



As the Residual vs Fitted plot shows, residuals forms a horizontal linear band around 0, they are equally spread throughout the range of fitted values, and there is a random pattern. Also, from the scale-location plot, we can see a horizontal line with equally spread points. These facts show that residuals have constant variance. In the QQplot, we can see an overall nice line shape. Although there are some deviations on tails, which means the model is not doing a good job capturing the ends of the distribution, the model is still acceptible. Therefore, the normality assumption of the residuals holds true.

Probelm5 Testing predictive ability

a) 10-fold cross-validation

10-fold cross-validation of the model is given by:

```
set.seed(1)

# Use 10-fold validation and create the training sets
data_train <- trainControl(method = "cv", number = 10)

# Fit the 3-variables model that we chose in previous part
model_cv <- train(`Life Exp` ~ Murder + `HS Grad` + Frost,
                  data = life_exp,
                  trControl = data_train,
                  method = 'lm',
                  na.action = na.pass)
```

```
model_cv$resample
```

```
##         RMSE  Rsquared       MAE Resample
## 1  0.8596780 0.8043734 0.7592786   Fold01
## 2  0.7409967 0.7937309 0.5512412   Fold02
## 3  0.8062552 0.9276387 0.6601515   Fold03
## 4  1.0405615 0.5671494 0.8588002   Fold04
## 5  0.4984804 0.9870190 0.3993902   Fold05
## 6  0.8479497 0.6312466 0.7667338   Fold06
## 7  0.6343097 0.8978960 0.3734799   Fold07
## 8  0.6152349 0.8751867 0.5527385   Fold08
## 9  0.7643245 0.6038997 0.7352997   Fold09
## 10 0.7901490 0.7809610 0.7685668   Fold10
```

Next, we calculate

$$CV_{10} = \frac{1}{10}\sum_{i=0}^{10} MSE_i$$

```
mean((model_cv$resample$RMSE))
```

```
## [1] 0.759794
```

Therefore, $CV_{10} = 0.759794$

b) "Residual sampling" bootstrap

I use the following code to experiment the new bootstrap technique:

```
pred = predict(Model_S) %>% as.tibble()
resid = residuals(Model_S) %>% as.tibble()

#use the following function to perform the "Residual sampling" bootstrap
#regression process described in the homework document for one time
boot_mse = function(df, residual, predict){
boot_resid = sample_frac(residual, replace = TRUE) #randomly resample the residuals (with replacement)
y_star = predict + boot_resid %>% as.tibble()
life_exp_new = cbind(y_star, select(df, -`Life Exp`))
model_new = lm(value ~ Murder + `HS Grad` + Frost, data = life_exp_new)
sm = summary(model_new)
mse_new = mean(sm$residuals^2) #get MSE for the new model
mse_new
}
```

Next, I will repeat the steps above for 10 times and 1000 times, respectively.

Results of repeating for 10 times:

```
boot_mse_10 = data_frame(
  strap_number = 1:10,
  strap_mse = rerun(10, boot_mse(life_exp, resid, pred))
) %>% unnest()

summary(boot_mse_10$strap_mse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3663  0.4156  0.4516  0.4659  0.5207  0.5749
```

```
sd(boot_mse_10$strap_mse)
```

```
## [1] 0.07186886
```

Results of repeating for 1000 times:

```
boot_mse_1000 = data_frame(
  strap_number = 1:1000,
  strap_mse = rerun(1000, boot_mse(life_exp, resid, pred))
) %>% unnest()

summary(boot_mse_1000$strap_mse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2078  0.4193  0.4622  0.4667  0.5188  0.6977
```

```
sd(boot_mse_1000$strap_mse)
```

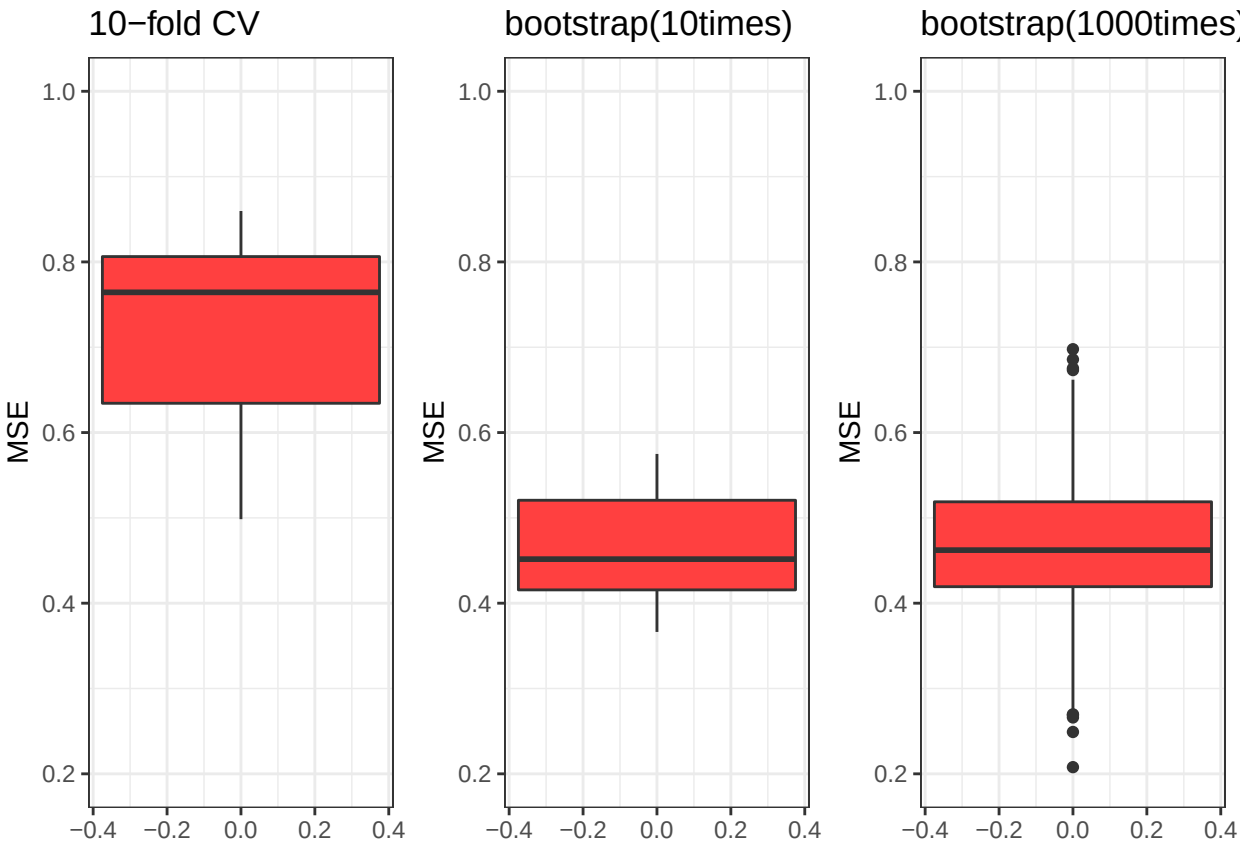
```
## [1] 0.07275573
```

c) Comparing methods a) and b)

First, let's make some plots to compare the MSE generated by the two methods.

```
cv_plot = model_cv$resample$RMSE %>% as.tibble() %>%
ggplot(aes(y = value)) + geom_boxplot(fill = "brown1") + labs(title = "10-fold CV", y = "MSE") + ylim(0
boot10_plot = boot_mse_10 %>% ggplot(aes(y = strap_mse)) + geom_boxplot(fill = "brown1") + labs(title =
boot1000_plot = boot_mse_1000 %>% ggplot(aes(y = strap_mse)) + geom_boxplot(fill = "brown1") + labs(tit
cv_plot + boot10_plot + boot1000_plot
```

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```



The plots shows that the new bootstrap method generates lower MSEs than 10-fold cross validation, while the results are similar between 10-time bootstrap and 1000-time bootstrap.The cross-validation method splits data into training data and testing data, and the MSEs are calculated based on model prediction on the testing data. Therefore, the cross-validation method reflects predictive ability of the model. The new bootstrap method generates lower MSEs, but that's because it calculates MSEs within the training data. The MSEs generated by the new bootstrap method only shows "goodness of fit" of the model instead of predictive ability. Therefore, I recommand cross-validation for assessing model performance.