Ordinary Least Squares Regression

CJ 702: Advanced Criminal Justice Statistics

Thomas Bryan Smith*

February 03, 2025

Contents

1 Setting up your environment

```
# Load Packages
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## Warning: package 'dplyr' was built under R version 4.3.3

## Warning: package 'stringr' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3
```

*University of Mississippi, tbsmit10@olemiss.edu

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.3.3
```

```
##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(bda)
```

```
## Warning: package 'bda' was built under R version 4.3.3
```

```
## Loading required package: boot
```

```
## Warning: package 'boot' was built under R version 4.3.3
```

```
##
## Attaching package: 'boot'
##
## The following object is masked from 'package:psych':
##
##     logit
##
## bda - 18.3.2
```

```r
library(mediation)
```

```
## Warning: package 'mediation' was built under R version 4.3.3
```

```
## Loading required package: MASS
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
##
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loading required package: mvtnorm

## Warning: package 'mvtnorm' was built under R version 4.3.3

## Loading required package: sandwich

## Warning: package 'sandwich' was built under R version 4.3.3

## Warning in check_dep_version(): ABI version mismatch:
## lme4 was built with Matrix ABI version 1
## Current Matrix ABI version is 0
## Please re-install lme4 from source or restore original 'Matrix' package

## mediation: Causal Mediation Analysis
## Version: 4.5.0
##
##
## Attaching package: 'mediation'
##
## The following object is masked from 'package:psych':
##
##     mediate
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.3.3

## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:boot':
##
##     logit
##
## The following object is masked from 'package:psych':
##
##     logit
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```r
# =========================================================================== #

# Load the USArrests dataset.
# This is a built-in dataset for practicing with and testing R functions.
# It is loaded with the data() function, and does not need to be assigned.
data("USArrests")

# Check your data:
head(USArrests)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```r
# Take note of the variables:
## Murder: Murder arrests (per 100,000)
## Assault: Assault arrests (per 100,000)
## Rape: Rape arrests (per 100,000)
## UrbanPop: Percent urban population

# =========================================================================== #

# Load the NCVS dataset we have been working with:
person <- readRDS("./Data/person.rds")

# Check your data:
head(person)
```

```
## # A tibble: 6 x 12
##   ID       IDHH   YEAR PER_WGT VIOLENT VLNT_WGT NONVIOLENT NVLNT_WGT   YIH EDUC
##   <fct>    <fct> <dbl>   <dbl>   <dbl>    <dbl>      <dbl>     <dbl> <dbl> <fct>
## 1 2000966~ 2000~  2000   1063.       0       NA          0        NA    10 FE
## 2 2000951~ 2000~  2000    894.       0       NA          0        NA     9 FE
## 3 2000470~ 2000~  2000   1317.       0       NA          0        NA     9 NHSE
## 4 2000205~ 2000~  2000   1093.       0       NA          0        NA     4 FE
## 5 2000361~ 2000~  2000   1101.       1    2202.          0        NA     4 FE
## 6 2000879~ 2000~  2000   1063.       0       NA          0        NA     6 HSE
## # i 2 more variables: AGE <fct>, SEX <fct>
```

```r
# Take note of the variables:
## ID: Person ID                           (numeric)
## IDHH: Household ID                      (numeric)
## PER_WGT: Person Weight                  (numeric)
## VIOLENT: Violent victimization count    (numeric, count, ratio)
```

```r
## VLNT_WGT: Violent victimization weight       (numeric)
## NONVIOLENT: Nonviolent victimization count   (numeric, count, ratio)
## NVLNT_WGT: Nonviolent victimization weight   (numeric)
# YIH: Years in household                       (numeric, years, interval)
# EDUC: Education level                         (factor, ordinal)
# AGE: Age                                      (factor, years, ordinal)
# SEX: Sex                                      (factor, nominal)

# Check for missingness:
missing <- person %>%
  filter(!complete.cases(VIOLENT, NONVIOLENT,
                         YIH, EDUC, AGE, SEX)) %>%
  nrow()
n <- person %>% nrow()
missing / n
```

```
## [1] 0.08812478
```

```r
# Satisfied with sufficiently low missingness,
# you can perform listwise deletion:
person <- person %>%
  filter(complete.cases(VIOLENT, NONVIOLENT,
                        YIH, EDUC, AGE, SEX))

# If you want to create any scales or indices,
# you can check the internal consistency of a set
# of measures with psych's alpha() function:
person %>% dplyr::select(VIOLENT, NONVIOLENT) %>% alpha()
```

```
## Number of categories should be increased  in order to count frequencies.
```

```
##
## Reliability analysis
## Call: alpha(x = .)
##
##   raw_alpha std.alpha G6(smc) average_r  S/N    ase  mean   sd median_r
##       0.12      0.14   0.075     0.075 0.16 0.0076 0.023 0.15    0.075
##
##     95% confidence boundaries
##          lower alpha upper
## Feldt     0.11  0.12  0.14
## Duhachek  0.11  0.12  0.14
##
##  Reliability if an item is dropped:
##            raw_alpha std.alpha G6(smc) average_r   S/N alpha se var.r med.r
## VIOLENT        0.045     0.075  0.0057     0.075 0.082       NA     0 0.075
## NONVIOLENT     0.125     0.075  0.0057     0.075 0.082       NA     0 0.075
##
##  Item statistics
##                n raw.r std.r r.cor r.drop  mean   sd
## VIOLENT    41742  0.56  0.73   0.2  0.075 0.008 0.15
## NONVIOLENT 41742  0.87  0.73   0.2  0.075 0.037 0.24
```

```r
# These variables were never really intended to be
# aggregated, so Cronbach's Alpha is very low (around 0.14).
# In your own research, you want to be aiming for 0.7 or
# above. For a modern peer-review publication, you should also be
# supplementing it with exploratory factory analysis (EFA),
# confirmatory factor analysis (CFA), or item response theory (IRT).
# All of these can be performed in R!

# If you did want to create an additive scale, you could use
# the composite() function:
person %>%
  dplyr::select(VIOLENT, NONVIOLENT) %>%
  mutate(VIC = rowSums(.))
```

```
## # A tibble: 41,742 x 3
##    VIOLENT NONVIOLENT   VIC
##      <dbl>      <dbl> <dbl>
##  1       0          0     0
##  2       0          0     0
##  3       0          0     0
##  4       0          0     0
##  5       1          0     1
##  6       0          0     0
##  7       0          0     0
##  8       0          0     0
##  9       0          0     0
## 10       0          1     1
## # i 41,732 more rows
```

```r
# If you wanted to retain this variable, you would need to
# assign this string of functions over the original person object.
```

2 Estimating Linear Models

- In this section we use the function `lm()` to estimate **ordinary least squares** (OLS) regression models using R's USArrest data.
  - Variables in these data include:
    - `Murder`: murder arrests (per 100,000).
    - `Assault`: assault arrests (per 100,000).
    - `Rape`: rape arrests (per 100,000).
    - `UrbanPop`: percent of population living in an urban area.
- However, for the exercises at the end of each section you will be using a subset of variables drawn from the second wave of the National Crime Victimization Survey (NCVS).
  - Variables in these data include:
    - `VIOLENT`: violent victimization count (numeric, count, ratio).
    - `NONVIOLENT`: non-violent victimization count (numeric, count, ratio).
    - `YIH`: years in household (numeric, years, interval).
    - `EDUC`: education level (factor, ordinal).

    * `SEX`: male v. female (factor, nominal).

- When using the `lm()` function you should typically assign the results to an object. **Linear model objects** are extremely versatile in that they can be used in conjunction with an array of other functions in R:
  - `summary()`: The most common function you will use in conjunction with `lm()`. When used on a linear model object it will provide you full details on the results of the linear model you have fit to your data.
  - `AIC()` and `BIC()`: Used to estimate **Akaike's Information Criterion** and **Bayesian Information Criterion** respectively.
  - `coefficients()`: Prints a named numeric vector containing the coefficients from your model in the order in which they appear in the model.
  - `confint()`: Prints the confidence intervals for each variable in the model at a level specified by you with the `level=` option.
  - `fitted()` and `predict()`: Prints marginal estimates for each observation's score on the dependent variable based on the fitted linear model.
  - `residuals()`: Prints the unstandardized, unstudentized residuals for each observation in your model.
  - `anova()`: Depending on its use, this function can print either the **Analysis of Variance** table for your linear model (`anova(model1)`), or a comparison of the fit of a null model with an alternative model (`anova(model1, model2)`.
  - `vcov()`: Prints the variance-covariance matrix of your linear model.
  - `influence()`: Prints numeric estimates of the influence each observation has on your model (how much the predicted scores would differ if the observation in question were not included in the data).
  - `calc.relimp()`: Estimates the relative importance of each of the independent variables in your model on the dependent variable. Performs a similar role as `influence()` but rather than identifying influential observations it identifies influential variables.
- The models estimated in this section are as follows:

$$assault = \beta_0 + \beta_1 urbanpop + \epsilon$$

$$murder = \beta_0 + \beta_1 assault + \beta_2 urbanpop + \epsilon$$

- You will be required to estimate the following models using NCVS:

$$education = \beta_0 + \beta_1 violent + \epsilon$$

$$education = \beta_0 + \beta_1 violent + \beta_2 non - violent + \beta_3 sex + \beta_4 age + \beta_5 hhyears + \epsilon$$

```
# Start out with the correlation coefficient:
USArrests %>% dplyr::select(Murder, Assault, Rape, UrbanPop) %>% cor()
```

```
##              Murder   Assault      Rape   UrbanPop
## Murder   1.00000000 0.8018733 0.5635788 0.06957262
## Assault  0.80187331 1.0000000 0.6652412 0.25887170
## Rape     0.56357883 0.6652412 1.0000000 0.41134124
## UrbanPop 0.06957262 0.2588717 0.4113412 1.00000000
```

```
# To estimate a bivariate linear model you use the lm() function.

# The lm() function is a little different in that you need to specify
# the data you are working with and tell R how you want to build
```

```r
# your linear model. The 'data = ' option is where you indicate
# the data frame you are working with, and is an option in many
# other R functions that analyze data.

# You specify the regression equation at the beginning of the function
# input. If you want the equation 'Murder = b0 + (b1 * UrbanPop) + e',
# then you would enter 'Murder ~ UrbanPop', where '~' is used to denote
# approximate equality. The intercept and error terms are assumed by the
# function. Putting this all together, we get the following:
lm(Assault ~ UrbanPop, data = USArrests)
```

```
##
## Call:
## lm(formula = Assault ~ UrbanPop, data = USArrests)
##
## Coefficients:
## (Intercept)     UrbanPop
##       73.08         1.49
```

```r
# However, you'll notice that it only returns the slope and intercept
# coefficients. As ever, you need to assign the results to an object
# if you want to explore them further:
m1 <- lm(Assault ~ UrbanPop, data = USArrests)

# Once assigned, you can use the summary() function to return a
# complete read out of the results:
summary(m1)
```

```
##
## Call:
## lm(formula = Assault ~ UrbanPop, data = USArrests)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -150.78  -61.85  -18.68   58.05  196.85
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  73.0766    53.8508   1.357   0.1811
## UrbanPop      1.4904     0.8027   1.857   0.0695 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 81.33 on 48 degrees of freedom
## Multiple R-squared:  0.06701,    Adjusted R-squared:  0.04758
## F-statistic: 3.448 on 1 and 48 DF,  p-value: 0.06948
```

```r
# For efficiency, you can combine this into a single line of code:
summary(m1 <- lm(Assault ~ UrbanPop, data = USArrests))
```

```
##
## Call:
```

```
## lm(formula = Assault ~ UrbanPop, data = USArrests)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -150.78  -61.85  -18.68   58.05  196.85
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  73.0766    53.8508   1.357   0.1811
## UrbanPop      1.4904     0.8027   1.857   0.0695 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 81.33 on 48 degrees of freedom
## Multiple R-squared:  0.06701,    Adjusted R-squared:  0.04758
## F-statistic: 3.448 on 1 and 48 DF,  p-value: 0.06948
```

```r
## Interpretation:
### b0: the intercept, the average value of the DV when all IVs are 0.
### b1: the average change in the DV for each interval increase in the IV.
### Pr(>|t|): P-value, probability of observing the current (or a more extreme)
###           effect size under the assumption that the null hypothesis is true.
### Degrees of freedom: n - (k + 1); n = # obs; k = # IVs.
### R-squared: Proportion of variance in the DV explained by the IVs.
### F-test: Overall model significance.

# ============================================================================ #

# Multiple Regression:
summary(m1 <- lm(Murder ~ Assault + UrbanPop, data = USArrests))
```

```
##
## Call:
## lm(formula = Murder ~ Assault + UrbanPop, data = USArrests)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5530 -1.7093 -0.3677  1.2284  7.5985
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.207153   1.740790   1.842   0.0717 .
## Assault      0.043910   0.004579   9.590 1.22e-12 ***
## UrbanPop    -0.044510   0.026363  -1.688   0.0980 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.58 on 47 degrees of freedom
## Multiple R-squared:  0.6634, Adjusted R-squared:  0.6491
## F-statistic: 46.32 on 2 and 47 DF,  p-value: 7.704e-12
```

```r
### b1-k: the average change in the DV for each interval increase in the IV,
###       net of all other IVs in the model.
```

```r
# ============================================================================ #

# In the above example, we are estimating the change in the Assault
# rate per 100,000 people for each interval increase (1 percentahe point)
# in the Urban Population Percentage.

# As you'll recall, the UrbanPop variable is a percentage.
# It is not reasonable to assume that 0 percent of the population
# in a given state resides in a city. The lowest we observe is 32%:
min(USArrests$UrbanPop)
```

```
## [1] 32
```

```r
# So, we might want to mean center and / or standardize the variable.
## Mean centered:
summary(m1 <- lm(Assault ~ scale(UrbanPop, scale = FALSE), data = USArrests))
```

```
##
## Call:
## lm(formula = Assault ~ scale(UrbanPop, scale = FALSE), data = USArrests)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -150.78  -61.85  -18.68   58.05  196.85
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    170.7600    11.5019  14.846   <2e-16 ***
## scale(UrbanPop, scale = FALSE)   1.4904     0.8027   1.857   0.0695 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 81.33 on 48 degrees of freedom
## Multiple R-squared:  0.06701,    Adjusted R-squared:  0.04758
## F-statistic: 3.448 on 1 and 48 DF,  p-value: 0.06948
```

```r
## Z-score standardized, but uncentered:
summary(m1 <- lm(Assault ~ scale(UrbanPop, center = FALSE), data = USArrests))
```

```
##
## Call:
## lm(formula = Assault ~ scale(UrbanPop, center = FALSE), data = USArrests)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -150.78  -61.85  -18.68   58.05  196.85
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        73.08      53.85   1.357   0.1811
## scale(UrbanPop, center = FALSE)   101.01      54.40   1.857   0.0695 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 81.33 on 48 degrees of freedom
## Multiple R-squared:  0.06701,    Adjusted R-squared:  0.04758
## F-statistic: 3.448 on 1 and 48 DF,  p-value: 0.06948
```

```r
## Z-score standardized and mean centered:
summary(m1 <- lm(Assault ~ scale(UrbanPop), data = USArrests))
```

```
##
## Call:
## lm(formula = Assault ~ scale(UrbanPop), data = USArrests)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -150.78  -61.85  -18.68   58.05  196.85
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       170.76      11.50  14.846   <2e-16 ***
## scale(UrbanPop)    21.57      11.62   1.857   0.0695 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 81.33 on 48 degrees of freedom
## Multiple R-squared:  0.06701,    Adjusted R-squared:  0.04758
## F-statistic: 3.448 on 1 and 48 DF,  p-value: 0.06948
```

```r
# In the above example, we are estimating the change in the Assault
# rate per 100,000 people for each standard deviation increase in the
# Urban Population Percentage.

# ============================================================================ #

# If you want to fit an intercept-only model, that is a model that is
# specified such that 'y = b0 + e' (includes no independent variables)
# you simple need to replace the independent variable with a 1.
summary(m1 <- lm(Assault ~ 1, data = USArrests))
```

```
##
## Call:
## lm(formula = Assault ~ 1, data = USArrests)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -125.76  -61.76  -11.76   78.24  166.24
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   170.76      11.79   14.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 83.34 on 49 degrees of freedom
```

```r
# ============================================================================ #

# If you want to fit the model to a subset of your data (e.g., males),
# you can use the 'subset = ' option. This performs the same function
# as R's subset() function, but within the lm() function:
summary(m1 <- lm(Assault ~ scale(UrbanPop),
                 data = USArrests,
                 subset = Murder < 8))
```

```
##
## Call:
## lm(formula = Assault ~ scale(UrbanPop), data = USArrests, subset = Murder <
##     8)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -100.025  -21.752   -0.576   18.058  111.843
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      114.489      8.026  14.265 8.31e-14 ***
## scale(UrbanPop)   26.144      8.036   3.253  0.00316 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.45 on 26 degrees of freedom
## Multiple R-squared:  0.2893, Adjusted R-squared:  0.262
## F-statistic: 10.58 on 1 and 26 DF,  p-value: 0.003155
```

```r
# In the above example, we are estimating the change in the Assault
# rate per 100,000 people for each standard deviation increase in the
# Urban Population Percentage, but only for states with a below average
# murder rate.

# ============================================================================ #

# You can also specify the way that lm() deals with missingness using
# the 'na.action = ' option. If you want the model to return an error
# when there is any missing present, you can set this to 'na.fail':
summary(m1 <- lm(Murder ~ scale(Assault) + scale(UrbanPop),
                 data = USArrests,
                 na.action = na.fail))
```

```
##
## Call:
## lm(formula = Murder ~ scale(Assault) + scale(UrbanPop), data = USArrests,
##     na.action = na.fail)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -4.5530 -1.7093 -0.3677  1.2284  7.5985
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.7880     0.3649  21.344  < 2e-16 ***
## scale(Assault)     3.6594     0.3816   9.590 1.22e-12 ***
## scale(UrbanPop)   -0.6443     0.3816  -1.688    0.098 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.58 on 47 degrees of freedom
## Multiple R-squared:  0.6634, Adjusted R-squared:  0.6491
## F-statistic: 46.32 on 2 and 47 DF,  p-value: 7.704e-12
```

```
# However, the default setting is 'na.omit', like so:
summary(m1 <- lm(Murder ~ scale(Assault) + scale(UrbanPop),
                 data = USArrests,
                 na.action = na.omit))
```

```
##
## Call:
## lm(formula = Murder ~ scale(Assault) + scale(UrbanPop), data = USArrests,
##     na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5530 -1.7093 -0.3677  1.2284  7.5985
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.7880     0.3649  21.344  < 2e-16 ***
## scale(Assault)     3.6594     0.3816   9.590 1.22e-12 ***
## scale(UrbanPop)   -0.6443     0.3816  -1.688    0.098 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.58 on 47 degrees of freedom
## Multiple R-squared:  0.6634, Adjusted R-squared:  0.6491
## F-statistic: 46.32 on 2 and 47 DF,  p-value: 7.704e-12
```

```
# ============================================================================ #

# You can introduce polynomials by raising a given variable to
# the appropriate power.
## Quadratic:
summary(lm(Assault ~ UrbanPop + I(UrbanPop^2),
           data = USArrests))
```

```
##
## Call:
## lm(formula = Assault ~ UrbanPop + I(UrbanPop^2), data = USArrests)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -152.04  -60.21  -17.09   58.47  195.61
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.048e+02  2.106e+02   0.498    0.621
## UrbanPop      4.455e-01  6.749e+00   0.066    0.948
## I(UrbanPop^2) 8.167e-03  5.237e-02   0.156    0.877
##
## Residual standard error: 82.17 on 47 degrees of freedom
## Multiple R-squared:  0.0675, Adjusted R-squared:  0.02782
## F-statistic: 1.701 on 2 and 47 DF,  p-value: 0.1935
```

```
## Cubic:
summary(lm(income ~ UrbanPop + I(UrbanPop^2) + I(UrbanPop^3),
           data = USArrests))
```

```
## Error in h(simpleError(msg, call)): error in evaluating the argument 'object' in selecting a method f
```

```
## Quartic:
summary(lm(income ~ UrbanPop + I(UrbanPop^2) + I(UrbanPop^3) + I(UrbanPop^4),
           data = USArrests))
```

```
## Error in h(simpleError(msg, call)): error in evaluating the argument 'object' in selecting a method f
```

```
### But we will get more into this when and if we cover non-linear models.

# ============================================================================ #

# Mediation
dir <- lm(Murder ~ UrbanPop, data = USArrests)
ind1 <- lm(Assault ~ UrbanPop, data = USArrests)
ind2 <- lm(Murder ~ Assault + UrbanPop, data = USArrests)

## Sobel Test
### This test is used to check the statistical significance
### of the mediation effect. It cannot help you find effect
### size. For that, you'll want to either interpret the
### above models, or run a nonparametric bootstrap.
mediation.test(mv = USArrests$Assault,
               iv = USArrests$UrbanPop,
               dv = USArrests$Murder)
```

```
##              Sobel    Aroian    Goodman
## z.value 1.82295357 1.81347475 1.83258260
## p.value 0.06831042 0.06975863 0.06686467
```

```
## Nonparametric Bootstrap
bsm <- mediate(ind1, ind2,
               treat = "UrbanPop",
               mediator = "Assault",
               boot = TRUE,
               sims = 1000)
```

```
## Running nonparametric bootstrap
```

```r
summary(bsm)
```

```
##
## Causal Mediation Analysis
##
## Nonparametric Bootstrap Confidence Intervals with the Percentile Method
##
##                  Estimate 95% CI Lower 95% CI Upper p-value
## ACME              0.06545     -0.00854         0.15   0.078 .
## ADE              -0.04451     -0.10252         0.00   0.062 .
## Total Effect      0.02093     -0.07056         0.10   0.650
## Prop. Mediated    3.12616    -16.50860        23.72   0.576
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Sample Size Used: 50
##
##
## Simulations: 1000
```

```r
## ACME: Average Causal Mediation Effect
### This is how much of the total effect is explained
### by the mediator. The average change in the DV in
### response to an interval increase in the IV via the MV.

## ADE: Average Direct Effect
### This is how much of the total effect is explained
### by the independent variable. The average change in
### the DV in response to an interval increase in the IV,
### controlling for the effect of the MV.

## Total Effect
### This is the combined effect of both the independent
### variable and the mediator. It's non-significant here
### because the indirect (ACME) and direct (ADE) effects
### appear to cancel eachother out.

## Prop. Mediated
### The indirect effect divided by the total effect.

# ============================================================================ #

# Moderation
## To integrate a moderation effect, you simply multiply the
## two variables you wish to examine. The interaction term
## is interpreted as the additional increase in the DV
## for each interval increase in the interacting variables.
mod <- lm(Murder ~ Assault * UrbanPop, data = USArrests)

# ============================================================================ #
```

```r
# Post-estimation functions
## Akaike's Information Criterion (AIC)
AIC(m1)
```

```
## [1] 241.5826
```

```r
## Bayesian Information CRiterion (BIC)
BIC(m1)
```

```
## [1] 249.2307
```

```r
## You can extract the coefficients as a named numeric vector:
coefficients(m1)
```

```
##     (Intercept)  scale(Assault) scale(UrbanPop)
##       7.7880000       3.6593525      -0.6442786
```

```r
## You can generate a named matrix of confidence intervals:
confint(m1, level = 0.95)
```

```
##                     2.5 %    97.5 %
## (Intercept)      7.053956 8.5220440
## scale(Assault)   2.891688 4.4270173
## scale(UrbanPop) -1.411943 0.1233862
```

```r
## You can generated a named numeric vector of predicted marginal scores:
fitted(m1)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##      10.988294      12.618968      12.555841       9.324520      11.275846
##       Colorado    Connecticut       Delaware        Florida        Georgia
##       8.692966       4.609941      10.452967      14.356149       9.801524
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##       1.532642       6.072782      10.446362       5.275797       3.129014
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##       5.319106       5.678793      11.203040       4.581645      13.397937
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##       5.966346      11.110416       3.430979      12.621370       7.907391
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##       5.634283       4.926319      10.667112       3.217434       6.227403
##     New Mexico       New York North Carolina   North Dakota           Ohio
##      12.605756      10.532380      16.001835       3.224640       5.138062
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##       6.810844       7.206634       4.656854       6.975073      13.321527
##   South Dakota      Tennessee          Texas           Utah        Vermont
##       4.980438       8.836106       8.472215       4.915509       3.890496
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##       7.252946       6.324832       5.027951       2.596689       7.606027
```

```r
predict(m1)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##      10.988294      12.618968      12.555841       9.324520      11.275846
##       Colorado    Connecticut       Delaware        Florida        Georgia
##       8.692966       4.609941      10.452967      14.356149       9.801524
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##       1.532642       6.072782      10.446362       5.275797       3.129014
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##       5.319106       5.678793      11.203040       4.581645      13.397937
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##       5.966346      11.110416       3.430979      12.621370       7.907391
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##       5.634283       4.926319      10.667112       3.217434       6.227403
##     New Mexico       New York North Carolina   North Dakota           Ohio
##      12.605756      10.532380      16.001835       3.224640       5.138062
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##       6.810844       7.206634       4.656854       6.975073      13.321527
##   South Dakota      Tennessee          Texas           Utah        Vermont
##       4.980438       8.836106       8.472215       4.915509       3.890496
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##       7.252946       6.324832       5.027951       2.596689       7.606027
```

```r
## You can convert your OLS model into an ANOVA model:
anova(m1)
```

```
## Analysis of Variance Table
##
## Response: Murder
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## scale(Assault)   1 597.70  597.70 89.7874 1.769e-12 ***
## scale(UrbanPop)  1  18.98   18.98  2.8507   0.09796 .
## Residuals       47 312.87    6.66
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
## You can print your variance-covariance matrix:
vcov(m1)
```

```
##                   (Intercept) scale(Assault) scale(UrbanPop)
## (Intercept)      1.331374e-01  -3.604543e-17    7.332240e-17
## scale(Assault)  -3.604543e-17   1.456127e-01   -3.769501e-02
## scale(UrbanPop)  7.332240e-17  -3.769501e-02    1.456127e-01
```

```r
## You can estimate the level of influence each observation has
## on your results. This is typically used to detect outliers:
influence(m1)
```

```
## $hat
##        Alabama         Alaska        Arizona       Arkansas     California
##     0.04395885     0.09410554     0.07293432     0.04918497     0.09740137
```

```
##       Colorado    Connecticut       Delaware        Florida        Georgia
##     0.03580000     0.05187580     0.03451853     0.10449088     0.03039710
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##     0.12130042     0.03651888     0.05828160     0.03024515     0.05989200
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##     0.03005536     0.04330248     0.03896407     0.05434894     0.07105766
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##     0.06500272     0.04313164     0.05116771     0.11080688     0.02193865
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##     0.04115952     0.03391389     0.05394834     0.06007199     0.08048527
##     New Mexico       New York North Carolina   North Dakota           Ohio
##     0.05839711     0.06953715     0.18314307     0.09281906     0.04196627
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##     0.02231792     0.02081931     0.04149316     0.06746040     0.10684300
##   South Dakota      Tennessee          Texas           Utah        Vermont
##     0.07032822     0.02646005     0.04060430     0.05683544     0.14625247
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##     0.02100773     0.02970422     0.09654741     0.06420657     0.02299663
##
## $coefficients
##                  (Intercept) scale(Assault) scale(UrbanPop)
## Alabama         0.0462680060   0.0464381142   -3.661469e-02
## Alaska         -0.0578205881  -0.0898311738    9.474956e-02
## Arizona        -0.0961278338  -0.1282855218   -6.478009e-02
## Arkansas       -0.0110330652  -0.0061395207    1.367610e-02
## California     -0.0504287600  -0.0445357268   -7.898150e-02
## Colorado       -0.0164481691  -0.0031664994   -1.362796e-02
## Connecticut    -0.0276322742   0.0282279707   -2.963100e-02
## Delaware       -0.0943149617  -0.0713098434   -2.449113e-02
## Florida         0.0233130233   0.0436560597    1.246322e-02
## Georgia         0.1567337637   0.0997551922   -8.703554e-02
## Hawaii          0.0857484934  -0.1696824208    1.494699e-01
## Idaho          -0.0720882169   0.0317503270    5.042595e-02
## Illinois       -0.0009846174  -0.0006747401   -1.037250e-03
## Indiana         0.0396843195  -0.0296626388    6.168129e-03
## Iowa           -0.0197639754   0.0264646980    5.047605e-03
## Kansas          0.0140398454  -0.0104004035    3.147654e-03
## Kentucky        0.0840643277  -0.0458720551   -6.836538e-02
## Louisiana       0.0873424227   0.0888974727   -2.018070e-02
## Maine          -0.0524854287   0.0455224601    4.201346e-02
## Maryland       -0.0451682858  -0.0753205945    1.484948e-02
## Massachusetts  -0.0335048173   0.0223213506   -5.174181e-02
## Michigan        0.0206838151   0.0194441375    7.302153e-03
## Minnesota      -0.0154079601   0.0201089076   -5.705277e-03
## Mississippi     0.0782424045   0.1235732983   -1.507989e-01
## Missouri        0.0223423341   0.0001737707    6.979684e-03
## Montana         0.0076283215  -0.0043118184   -5.627348e-03
## Nebraska       -0.0129661094   0.0108026584    4.392507e-04
## Nevada          0.0324060015   0.0247508309    2.891081e-02
## New Hampshire  -0.0237770125   0.0310611786    7.949863e-03
## New Jersey      0.0255046863  -0.0156398901    4.622910e-02
## New Mexico     -0.0256107113  -0.0361627919    1.309242e-03
## New York        0.0122008108   0.0084456428    1.541140e-02
## North Carolina -0.0734972093  -0.1898768556    1.555763e-01
```

```
## North Dakota   -0.0534543922  0.0657016719   6.416095e-02
## Ohio            0.0451328192 -0.0384170673   4.004367e-02
## Oklahoma       -0.0043131325  0.0013260430  -1.091256e-03
## Oregon         -0.0471135454  0.0086167399  -7.079730e-03
## Pennsylvania    0.0342855361 -0.0334713309   2.427848e-02
## Rhode Island   -0.0766739230  0.0289244608  -1.234829e-01
## South Carolina  0.0241496909  0.0425903723  -4.088639e-02
## South Dakota   -0.0253947210  0.0180455669   3.209959e-02
## Tennessee       0.0896500229  0.0317519513  -4.955208e-02
## Texas           0.0881343237  0.0100492750   8.723978e-02
## Utah           -0.0363777340  0.0345225061  -4.601918e-02
## Vermont        -0.0396017738  0.0378206211   8.384473e-02
## Virginia        0.0254762841 -0.0036691883  -3.611909e-03
## Washington     -0.0479200591  0.0231926549  -3.120497e-02
## West Virginia   0.0148773525 -0.0098021059  -2.529737e-02
## Wisconsin       0.0000707538 -0.0001099832   3.076595e-05
## Wyoming        -0.0164999822  0.0003254542   6.359752e-03
##
## $sigma
##        Alabama         Alaska        Arizona       Arkansas     California
##       2.586573       2.576235       2.517143       2.606779       2.583958
##       Colorado    Connecticut       Delaware        Florida        Georgia
##       2.605266       2.600431       2.516910       2.602909       2.346719
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##       2.539774       2.555283       2.607976       2.592024       2.604156
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##       2.605992       2.536563       2.530442       2.580700       2.588163
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##       2.597026       2.603716       2.605637       2.550636       2.602893
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##       2.607404       2.606292       2.597613       2.602443       2.601746
##     New Mexico       New York North Carolina   North Dakota           Ohio
##       2.601542       2.606542       2.561596       2.580835       2.587572
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##       2.607796       2.585240       2.596219       2.550223       2.602552
##   South Dakota      Tennessee          Texas           Utah        Vermont
##       2.601731       2.525142       2.529145       2.594948       2.593997
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##       2.601356       2.584665       2.605901       2.607985       2.605212
##
## $wt.res
##        Alabama         Alaska        Arizona       Arkansas     California
##    2.211705876   -2.618967521   -4.455840797   -0.524520212   -2.275846477
##       Colorado    Connecticut       Delaware        Florida        Georgia
##   -0.792966233   -1.309941393   -4.552967384    1.043851247    7.598475578
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##    3.767358262   -3.472781804   -0.046361617    1.924203064   -0.929013572
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##    0.680893638    4.021206698    4.196960319   -2.481645072   -2.097936664
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##   -1.566345654    0.989584413   -0.730978505    3.478630382    1.092608676
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##    0.365717173   -0.626318907    1.532887583   -1.117433997    1.172596742
##     New Mexico       New York North Carolina   North Dakota           Ohio
```

```
##   -1.205755989    0.567620055   -3.001835255   -2.424640287    2.161938157
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##   -0.210843618   -2.306633694    1.643146036   -3.575073459    1.078473276
##   South Dakota      Tennessee          Texas           Utah        Vermont
##   -1.180437769    4.363893957    4.227784567   -1.715509472   -1.690495829
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##    1.247054259   -2.324831546    0.672049136    0.003310547   -0.806026914
```

```r
## You can compare the fit of two different models with ANOVA:
anova(dir, ind2)
```

```
## Analysis of Variance Table
##
## Model 1: Murder ~ UrbanPop
## Model 2: Murder ~ Assault + UrbanPop
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     48 925.05
## 2     47 312.87  1    612.18 91.962 1.216e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
## The 'car' package will let you find the variance inflation factor (VIF):
vif(m1)
```

```
##  scale(Assault) scale(UrbanPop)
##        1.071828        1.071828
```

3 Visualizing Linear Models

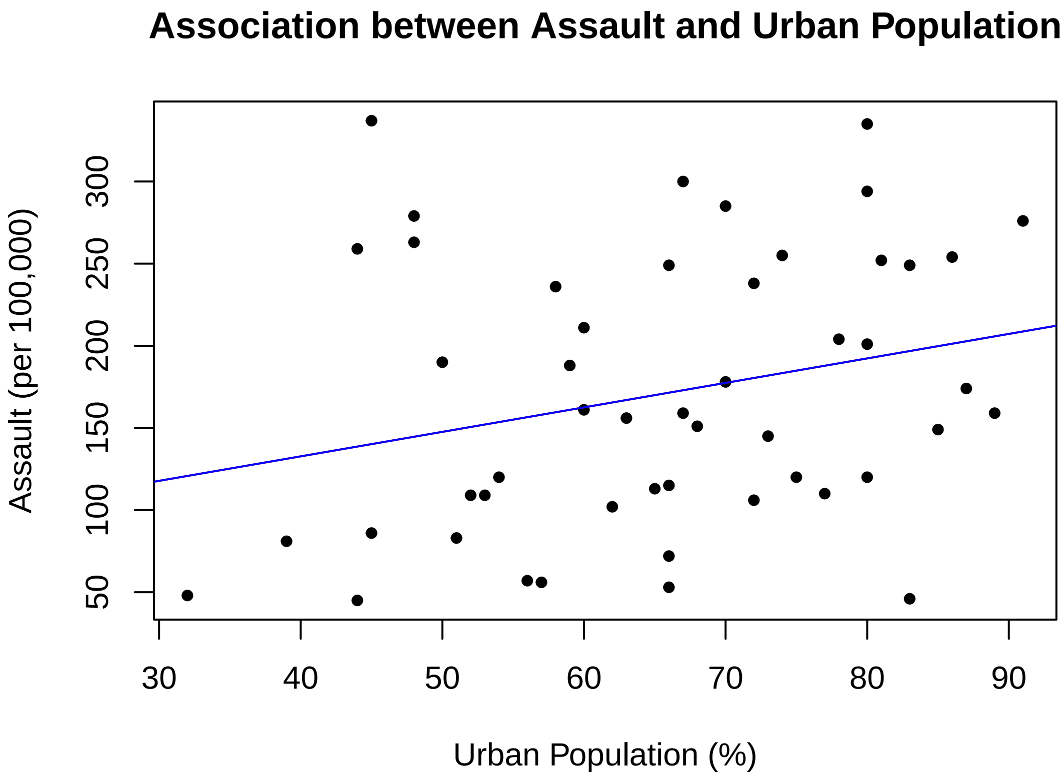
- You can plot bivariate linear models with the `plot()` and `abline()` **base R** functions.
  - The `abline()` function is used on linear model objects in much the same way as the functions I have discussed above.
  - Alternatively, you could plot a regression line by with the predicted marginal scores you have generated using `predict()` or `fitted()` functions. For multivariate models you can use these predicted scores to plot a predicted / actual scatterplot which will give you an idea of how well your model fits the data.
- Alternatively, the more versatile ggplot2 can be used for more aesthetically pleasing graphs. You would be using a combination of the `ggplot()`, `geom_point()`, and `geom_smooth()` functions to plot your bivariate regression with the `labs()` function as a way of assigning labels to the plot.
- For additional details on the functionality of **base R** and **ggplot2** graphical functions see the materials from **3 Descriptive Statistics and Graphics**.

```r
# Bivariate regression results

## Use the par() function to define the graph margins.
## If you find that your plot is not displaying an axis title,
## then you may want to try running this code before using the
## plot() function. It can also be used to visualize multiple plots.
par(mar = c(5, 5, 4, 4),
    mfrow = c(1, 1))
```

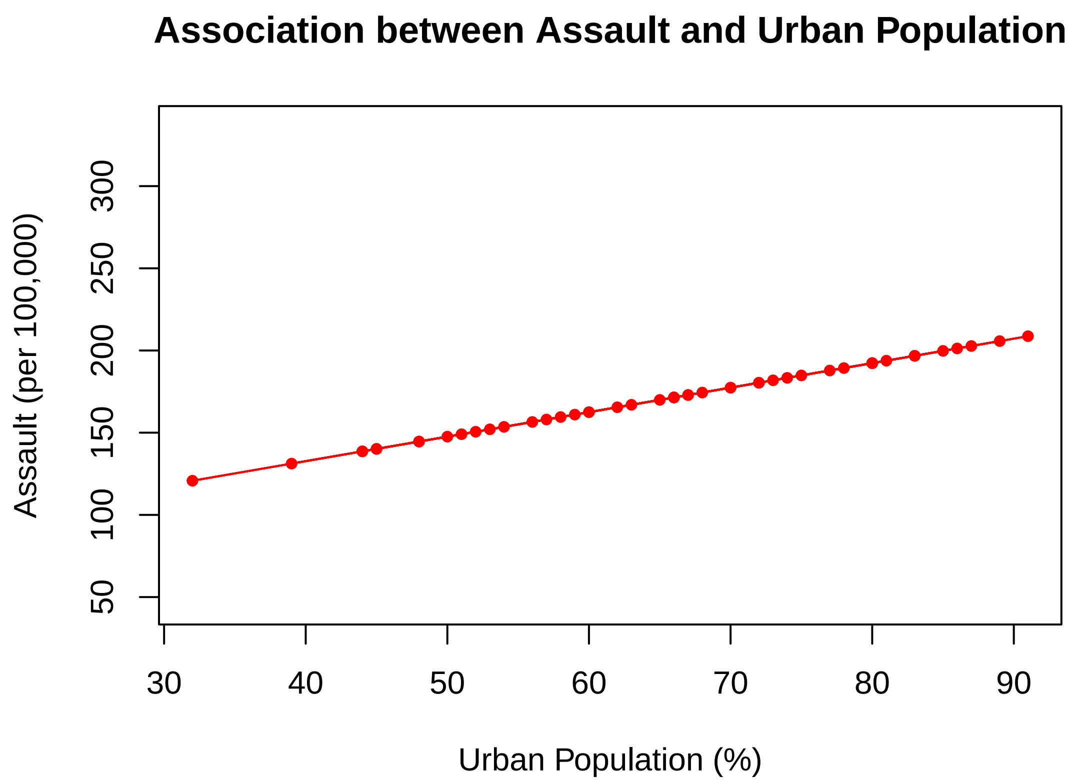
```r
## Scatter plot for your DV and IV:
plot(USArrests$UrbanPop, USArrests$Assault,
     main = "Association between Assault and Urban Population",
     xlab = "Urban Population (%)",
     ylab = "Assault (per 100,000)",
     pch = 20)

## Plot the results of the OLS regression, as generated by the lm() function:
abline(lm(Assault ~ UrbanPop, USArrests), col = "red")

## If you have already fit the linear model and assigned it to an object,
## you can instead just replace the lm() function within abline() with that
## object:
ols <- lm(Assault ~ UrbanPop, USArrests)
summary(ols)
```

```
##
## Call:
## lm(formula = Assault ~ UrbanPop, data = USArrests)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -150.78  -61.85  -18.68   58.05  196.85
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  73.0766    53.8508   1.357   0.1811
## UrbanPop      1.4904     0.8027   1.857   0.0695 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 81.33 on 48 degrees of freedom
## Multiple R-squared:  0.06701,    Adjusted R-squared:  0.04758
## F-statistic: 3.448 on 1 and 48 DF,  p-value: 0.06948
```

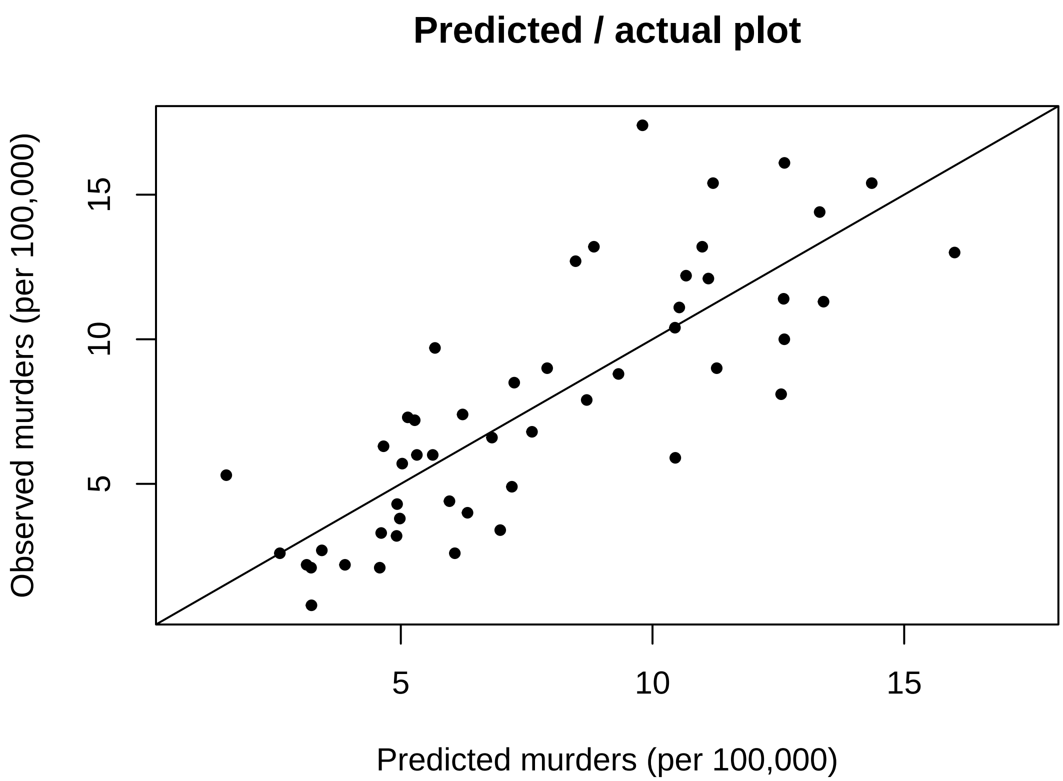
```r
abline(ols, col = "blue")
```




```r
## You can also plot a line with the predicted scores from the regression:
predictions <- predict(ols)

plot(USArrests$UrbanPop, predictions,
     type = "o", col = "red",
     xlim = c(min(USArrests$UrbanPop), max(USArrests$UrbanPop)),
     ylim = c(min(USArrests$Assault), max(USArrests$Assault)),
     main = "Association between Assault and Urban Population",
     xlab = "Urban Population (%)",
     ylab = "Assault (per 100,000)",
     pch = 20)
```




```r
# ============================================================================ #

# Multiple regression results

## Predicted / actual plot
### Predicted values of a multivariate linear model can be a good
### way of visualizing the goodness-of-fit. You can examine the
### model residuals by generating a scatterplot with both
### predicted and observed scores on the dependent variable.
predictions <- predict(m1)

plot(predictions, USArrests$Murder,
     xlim = c(min(USArrests$Murder), max(USArrests$Murder)),
     ylim = c(min(USArrests$Murder), max(USArrests$Murder)),
     main = "Predicted / actual plot",
     xlab = "Predicted murders (per 100,000)",
     ylab = "Observed murders (per 100,000)",
     pch = 20)
abline(a = 0, b = 1)
```

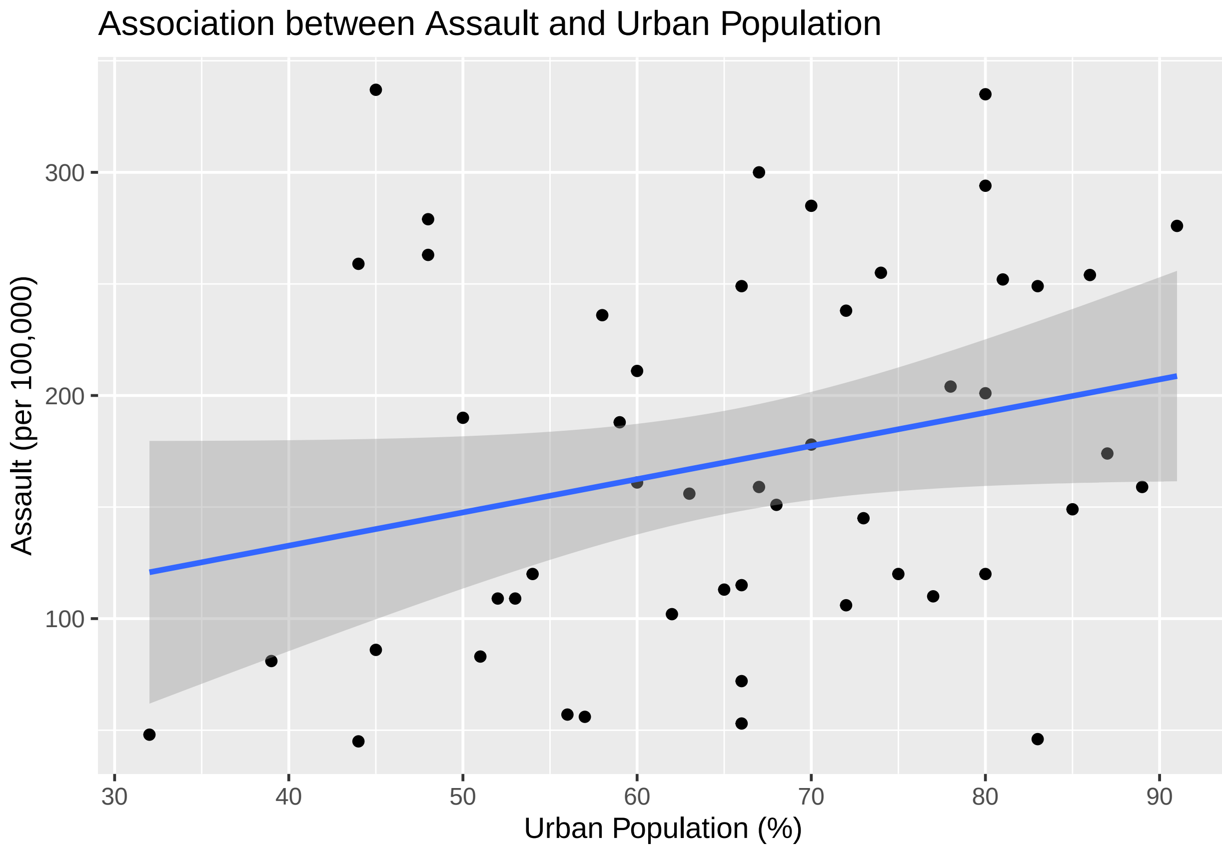



```r
## 3D scatter plot
### Alternatively, if you are only working with three variables
### you could create a 3D plot:
scatter3d(Murder ~ Assault + UrbanPop,
          data = USArrests)
```

```
## Loading required namespace: rgl

## Loading required namespace: mgcv
```

```r
# ============================================================================ #

# ggplot2

## You can (and should) replicate all of the above scatterplots using ggplot2.
## Here is an example of the basic bivariate scatterplot.
ggplot(USArrests, aes(y = Assault, x = UrbanPop)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Association between Assault and Urban Population",
       y = "Assault (per 100,000)",
       x = "Urban Population (%)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



4 Regression Diagnostics and Testing Assumptions

- This section presents a wide array of linear model diagnostic tests you may want to run, especially if evaluating whether or not the assumptions of OLS linear regression hold in the case of the model you have fitted to the data. Information on how to interpret each of these diagnostics are embedded in the chunks of R code. Here I simply present to you the types of tests and accompanying functions one might consider when assessing each of the following topics/assumptions.

Identifying Outliers

- Certain observations in your data may be considered outliers, cases which do not conform to the distribution of a variable. Depending on your field this may or may not be a priority. In some fields it may be required that you identify and appropriately respond to outliers. In other fields it may be the convention to simply not worry about outliers. Regardless, R offers some useful tools for identifying outliers.
  - *Outlier Test*
    - `outlierTest()`
  - *Q-Q Plot*
    - `qqPlot()`
  - *Leverage Plot*
    - `leveragePlots()`
  - *Cook's D Plot*

    * `plot()`

Assumption 1: Y is a linear function of X

- The observed relationship between the independent and dependent variables must be linear for a linear model to be appropriate for the data.
  - *Scatter Plot*
    * `scatter.smooth()`
    * `ggplot() + geom_point() + geom_smooth()`
  - *Component and Residual Plot*
    * `crPlots()`
  - *Ceres Plot*
    * `ceresPlots()`

Assumption 2: Multivariate Normality

- All variables in the linear model need to be multivariate normal (when scatterplotting dependent and independent variables the distribution must be multivariate normal). Alternatively this assumption can simply require that all continuous variables in your model are normally distributed (appropriate skew, kurtosis, etc.)
  - *Histogram*
    * `hist()`
  - *Q-Q Plot*
    * `qqplot()`

Assumption 3: Little or No Multicollinearity

- Multicollinearity is the phenomenon wherein the independent variables of a multivariate regression model are too highly correlated with one another. Ideally,
  - *Variance Inflation Factors*
    * `vif()`

Assumption 4: Little or No Autocorrelation

- Autocorrelation is when the residuals of a multivariate regression model are not independent (i.e. if running a time-series model you would expect the residuals at time 2 to be dependent on th residuals at time 1). Typically not required for cross-sectional models as those in this workshop. That being said, I have still demonstrated the diagnostics one might wish to run when testing for autocorrelation.
  - *Durbin Watson Test*
    * `durbinWatsonTest()`
  - *ACF Plot*
    * `acf()`

Assumption 5: Homoscedasticity (Little or No Heteroscedasticity)

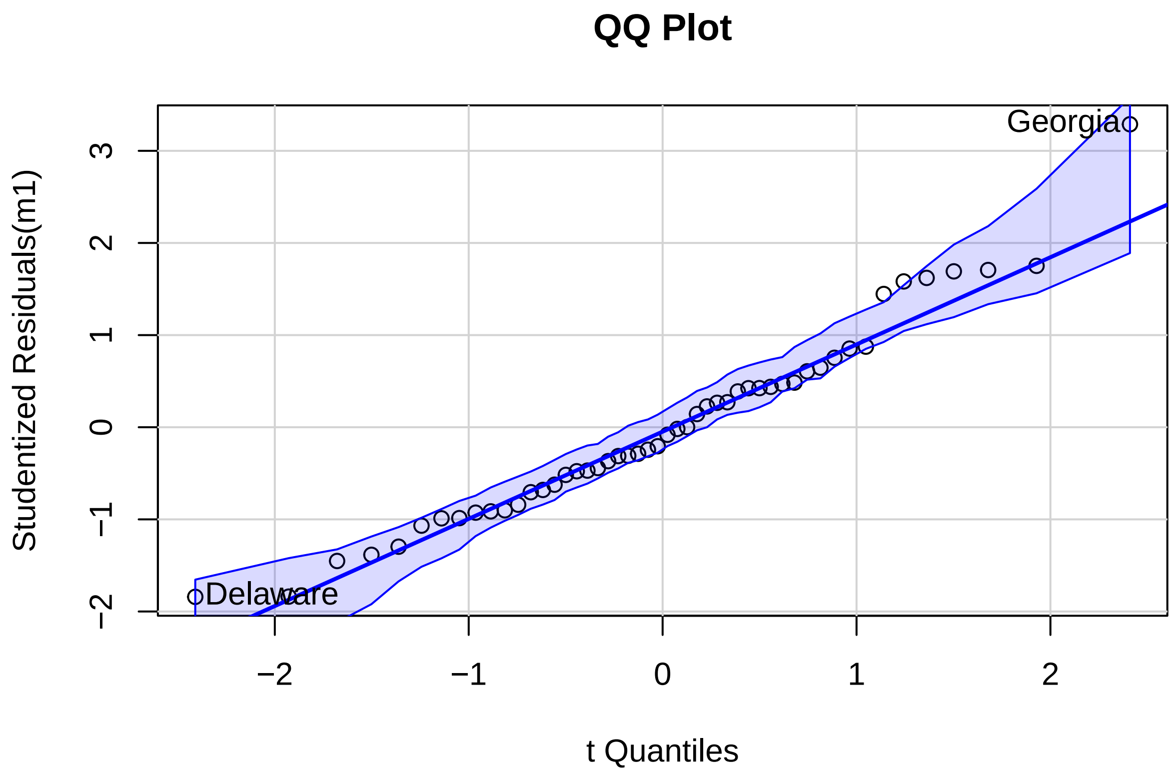
- Homoscedasticity is the condition of a linear model wherein the residuals are evenly distributed at all points of the regression. Conversely, heteroscedasticity is the condition of the linear model wherein the residuals are unevenly distributed at different points of the regression.

    - *Non-Constant Error Variance Test*
        * `ncvTest()`
    - *Residual Plot*
        * `predict()` or `fitted()`, `plot()` and `abline()`
    - *Spread Level Plot*
        * `spreadLevelPlot()`
- The `gvlma()` function from the **gvlma** package can also be useful for assessing skewness, kurtosis, and homoscedasticity in your linear models.
- Using the `plot()` function on a linear model object after running `par(mfrow=c(2,2))` will generate a battery of visualizations (residuals v. fitted scatterplot, QQ plot, scale-location plot, and residuals v. leverage plot) which can be used to assess the fit and assumptions of your linear model.
    - This use of the `par()` function will change how RStudio presents graphics to you. To revert to defaults simply run the following: `par(mfrow=c(1,1))`.

```r
# Note on terminology:
## Studentized residuals are estimated by dividing each residual by the
## standard error. Studentization follows a similar logic to standardization
## (z-scores).

# ========================================================================= #

# Checking for outliers and influential observations

## Bonferonni p-value for most extreme observations
outlierTest(m1)
```

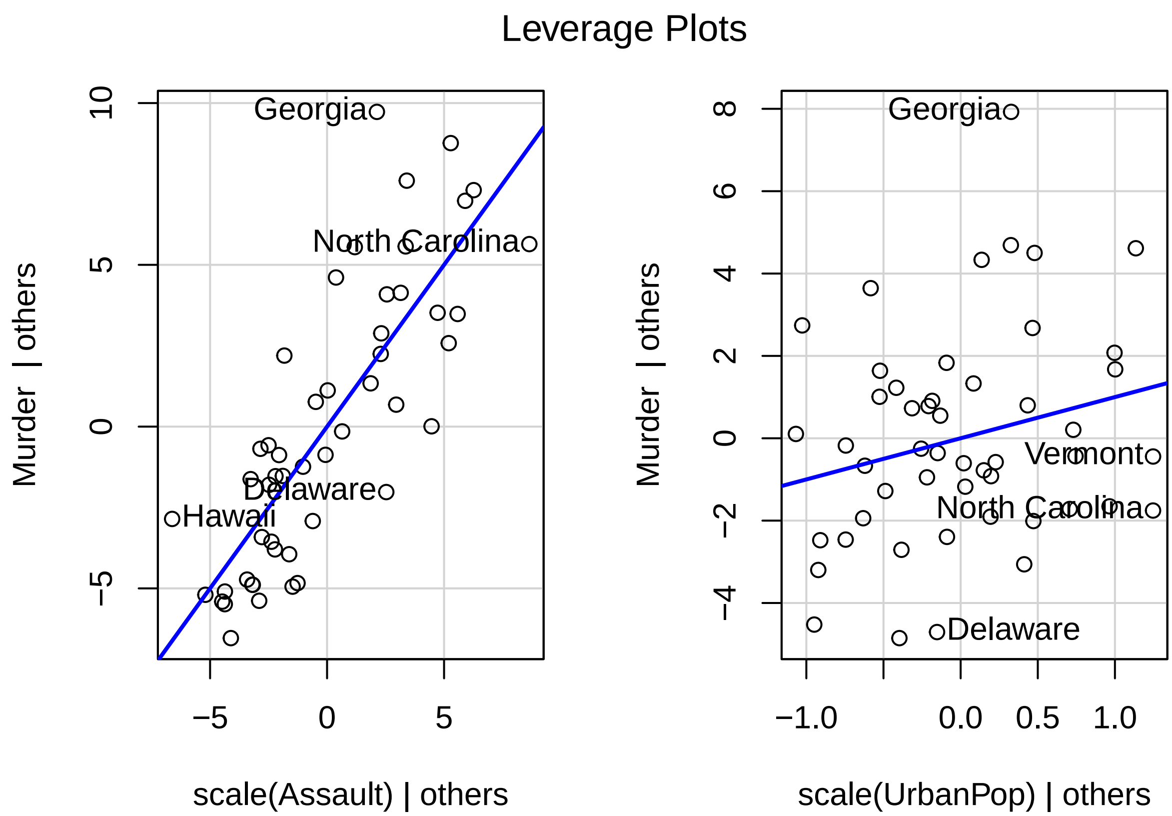
```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##         rstudent unadjusted p-value Bonferroni p
## Georgia 3.288277          0.0019369     0.096846
```

```r
## QQ plot
qqPlot(m1, main = "QQ Plot")
```

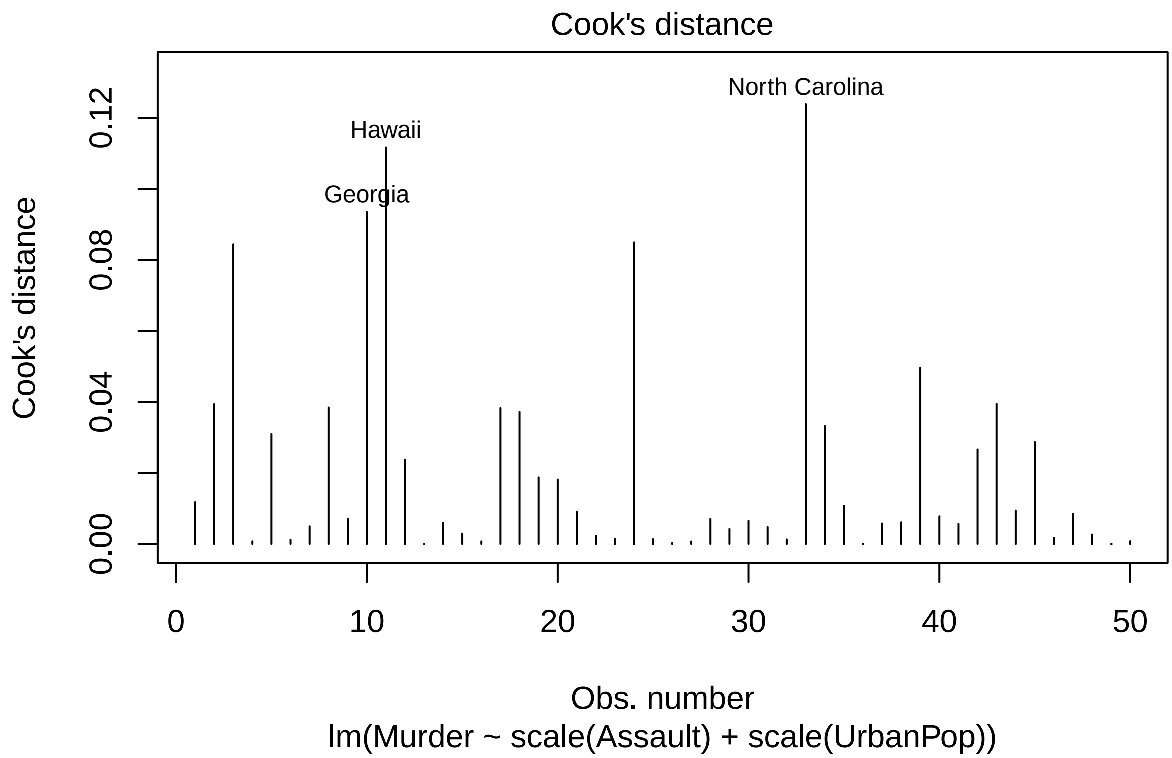



```
## Delaware  Georgia
##        8       10
```

```
## Leverage Plot
leveragePlots(m1)
```

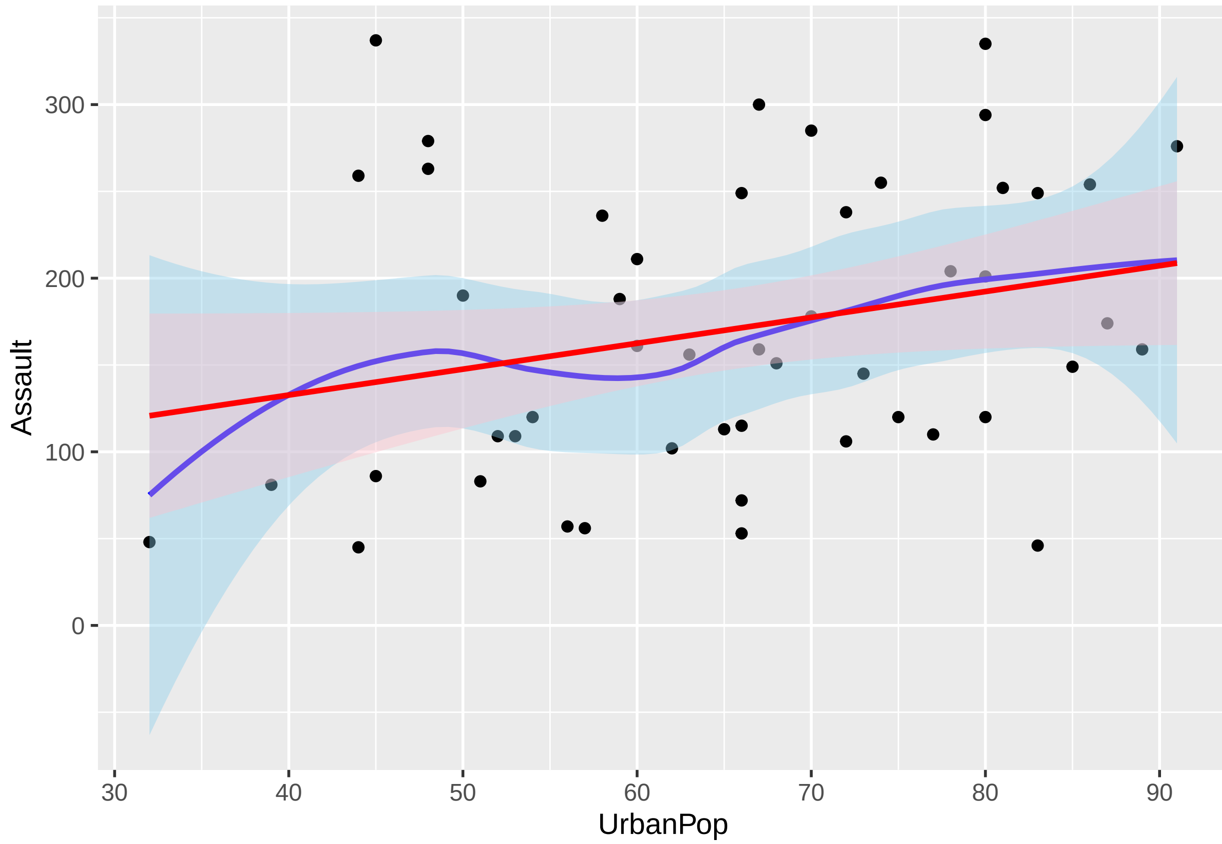



```
## Cook's D plot [with 4/(n-k-1) as the cutoff]
cutoff <- 4 / ((nrow(USArrests) - length(m1$coefficients) - 2))
plot(m1, which = 4, cook.levels = cutoff)
```

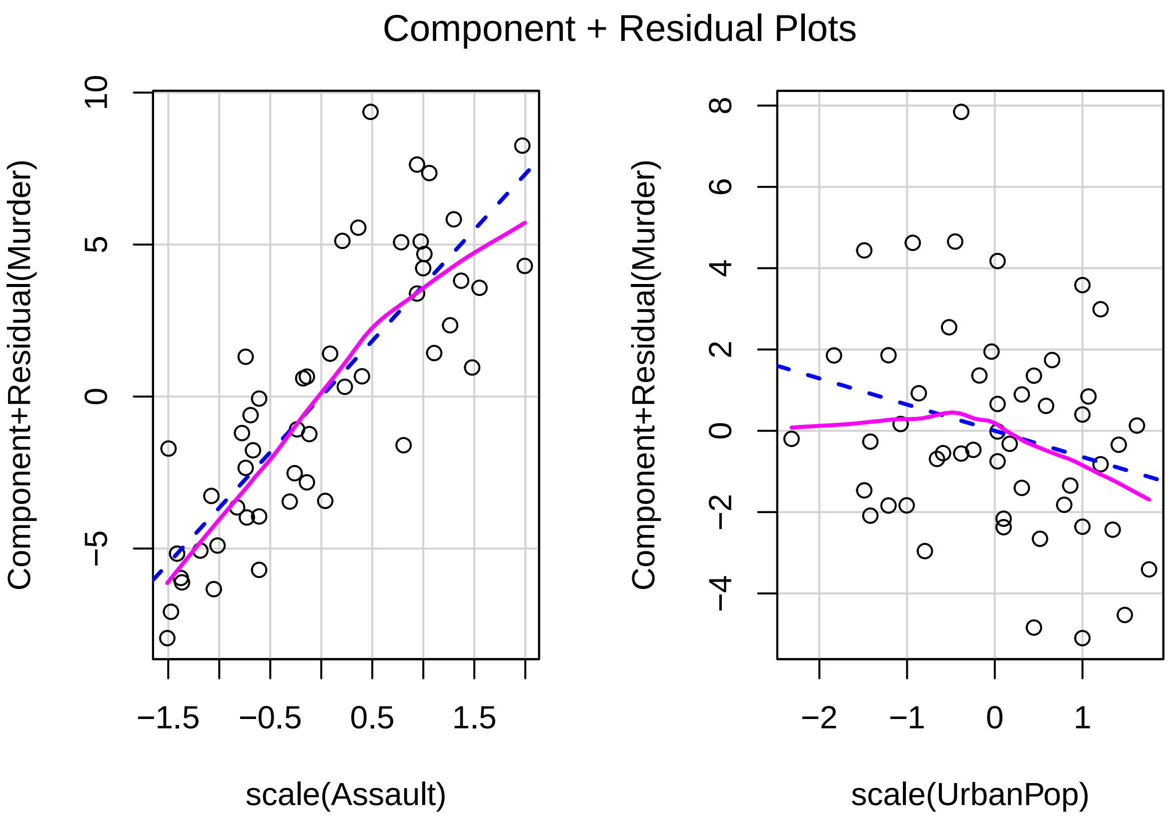



```r
# ========================================================================== #

# Assumption 1: Y is a linear function of X
## Technically this assumption is simply that the model is linear in parameters.
## By this definition, simply using the lm() function satisfies the assumption.

## However, you can use the scatter.smooth() function to see if the relationship
## between two variablesis actually linear. From the following plot you can see
## that, despite being slightly squiggly, you can cleanly fit a linear
## model within the bounds of the 95% confidence interval.

ggplot(USArrests, aes(y = Assault, x = UrbanPop)) +
  geom_point() +
  geom_smooth(color = "blue", fill = "skyblue") +
  geom_smooth(method = "lm", color = "red", fill = "pink")
```

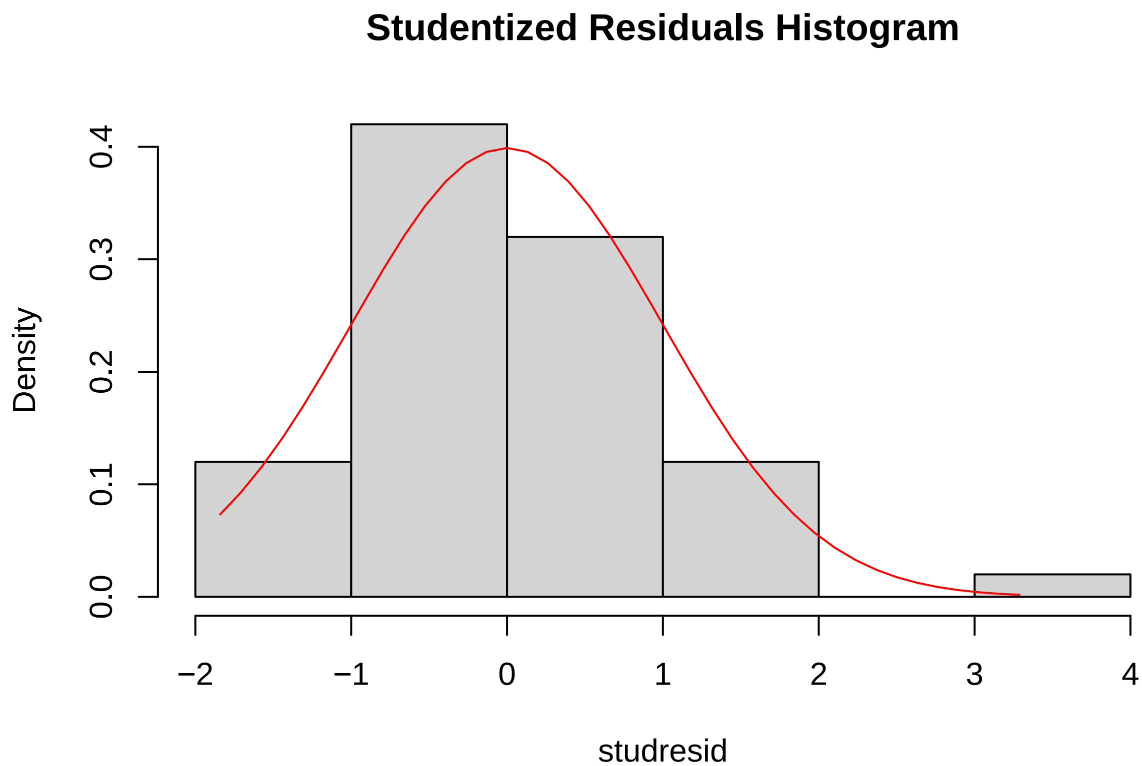
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



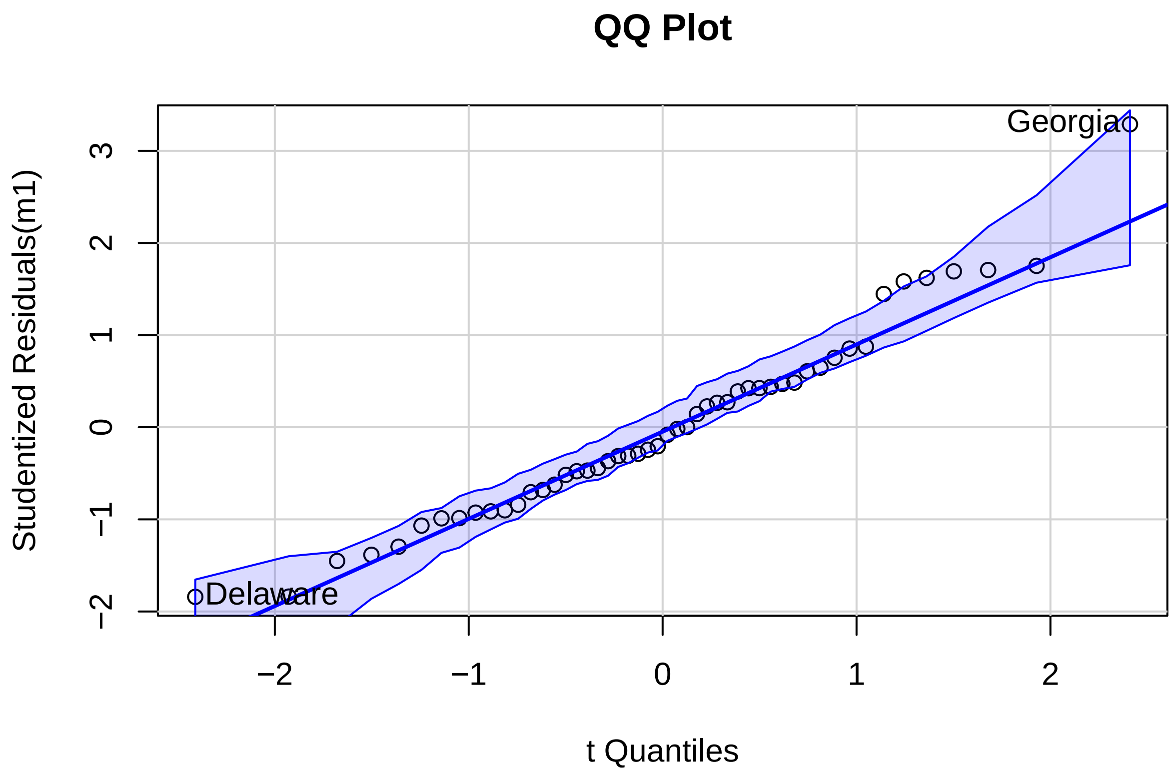

```r
## Component and Residual Plots
### The following function will generate similar visualizations for
### multivariate models. Keep in mind that a plot will be generated
### for each variable. This can be a problem with saturated models.
crPlots(m1)
```




```r
# ============================================================================ #

# Assumption 2: Multivariate Normality

## Histograms
studresid <- studres(m1)
hist(studresid, freq = FALSE,
     main = "Studentized Residuals Histogram")

xfit <- seq(min(studresid), max(studresid), length = 40)
yfit <- dnorm(xfit)

lines(xfit, yfit, col = "red")
```




```
## QQ Plots
qqPlot(m1, main = "QQ Plot")
```




```
## Delaware  Georgia
##        8       10
```

```r
# ========================================================================== #

# Assumption 3: Little or no multicollinearity

## Variance inflation factors
vif(m1)
```

```
##  scale(Assault) scale(UrbanPop)
##        1.071828        1.071828
```

```r
vif(m1) > 2
```

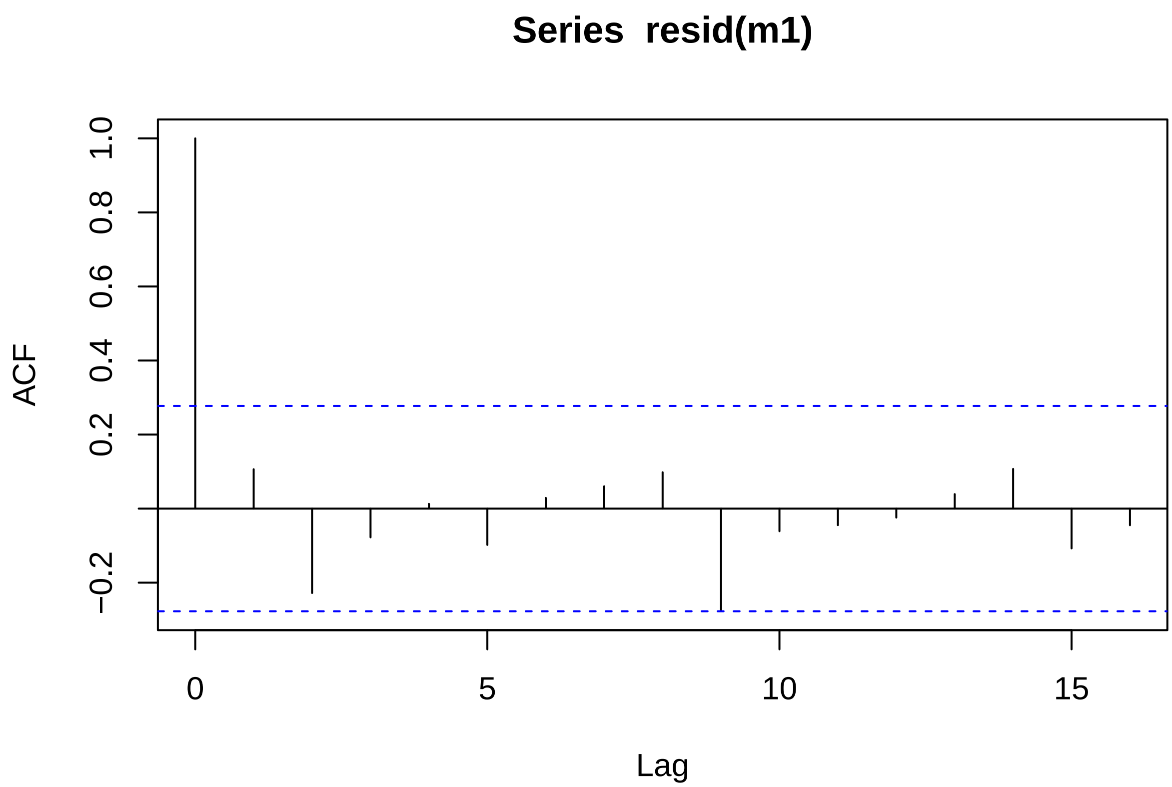
```
##  scale(Assault) scale(UrbanPop)
##           FALSE           FALSE
```

```r
### VIF = 1 / (1 - R-Squared)
### Estimated for all variables.
### A VIF over 10 suggests that there may be multicollinearity.
### A VIF over 100 suggests that there is definitely multicollinearity.

### The square root of the VIF gives you an estimate for
```

```r
### how much larger the SE is when compared with the SE if the
### variable in question was uncorrelated with any other variable.

### You want this value to be lower than 2 when possible.

### The second line of code sets up a boolean output where TRUE
### indicates possible multicollinearity in the associated variable.

# ============================================================================ #

# Assumption 4: Little or no autocorrelation

## Autocorrelation is the correlation between the same variable
## at different points in time (past behavior predicting future behavior).
## Not as relevant in our cross-sectional models.
## Regardless, this is how you'd test for autocorrelation.

## Durbin Watson Test
### We want the D-W statistic to approximately equal 2.
### A D-W statistic which is significantly different from 2 indicates
### that the linear model's residuals are correlated. This suggests
### the presence of autocorrelation.
durbinWatsonTest(m1)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.1064333      1.769422   0.424
##  Alternative hypothesis: rho != 0
```

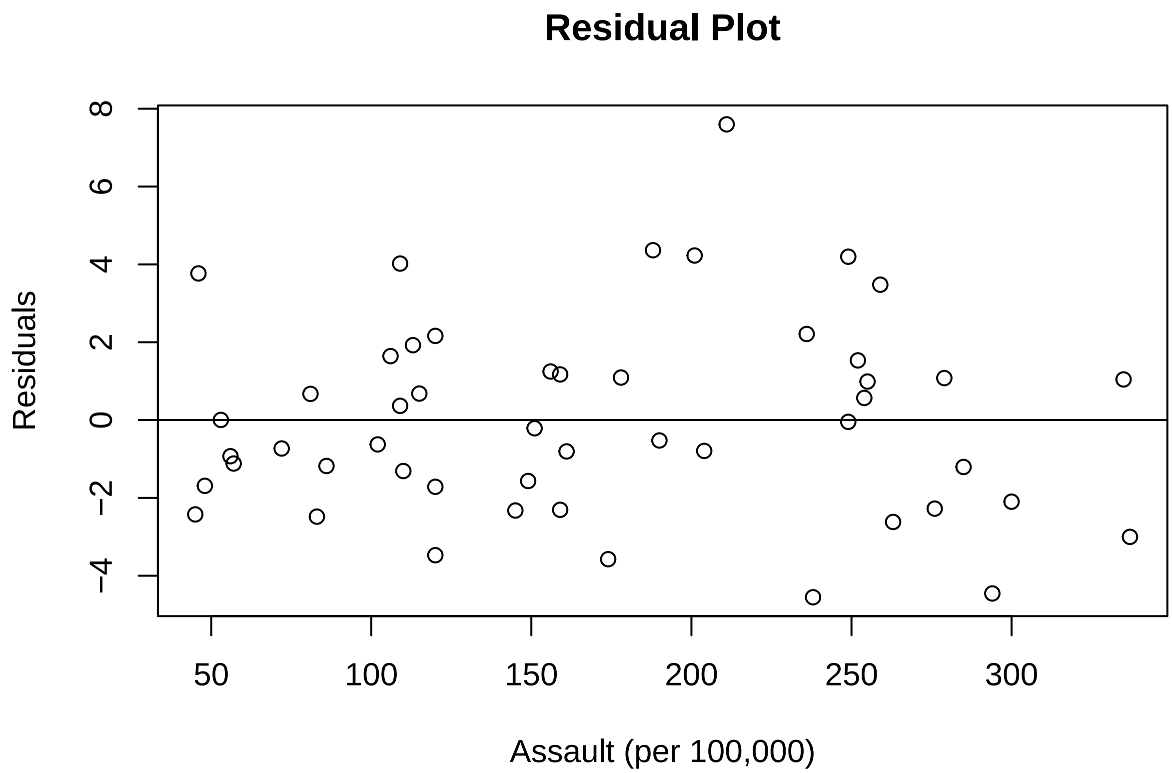
```r
## ACF Plot
### In this plot we want all the lines following the first to fall
### within the two dashed blue lines. No autocorrelation present!
resid(m1) |> acf()
```



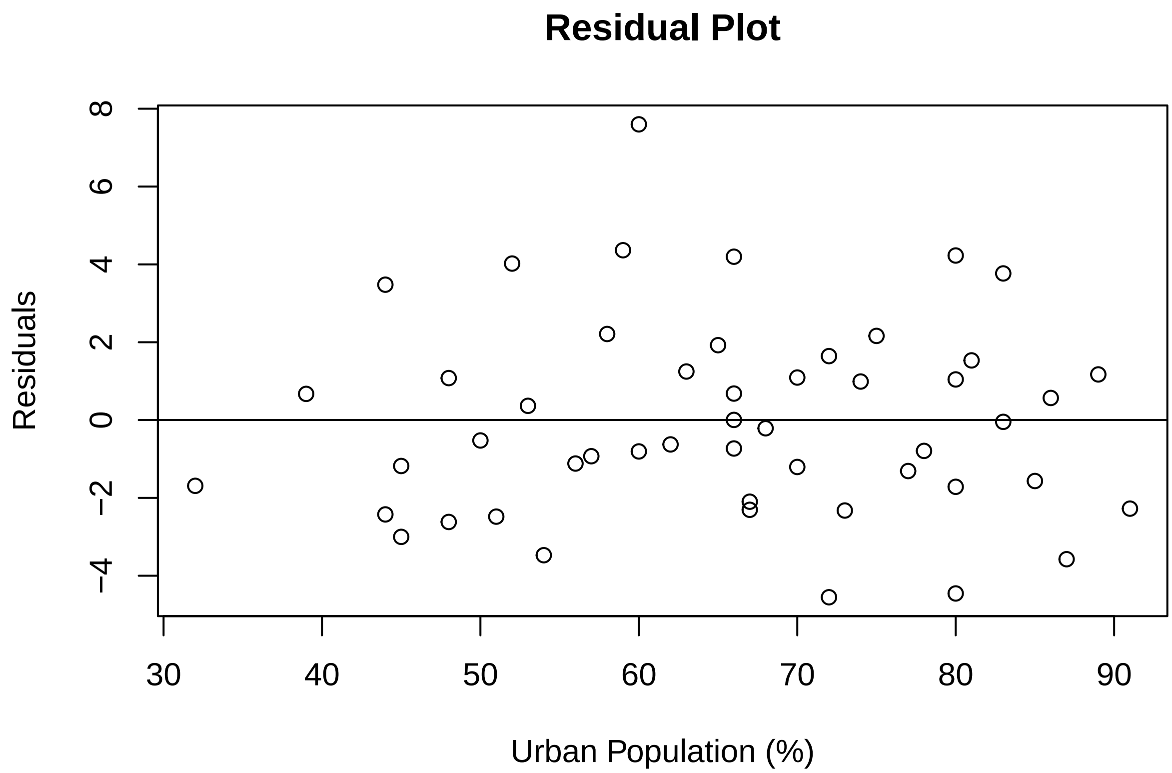

```r
# ============================================================================ #

# Assumption 5: Homoscedasticity (Little or no heteroscedasticity)

## Non-constant error variance test
ncvTest(m1)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 2.134843, Df = 1, p = 0.14399
```

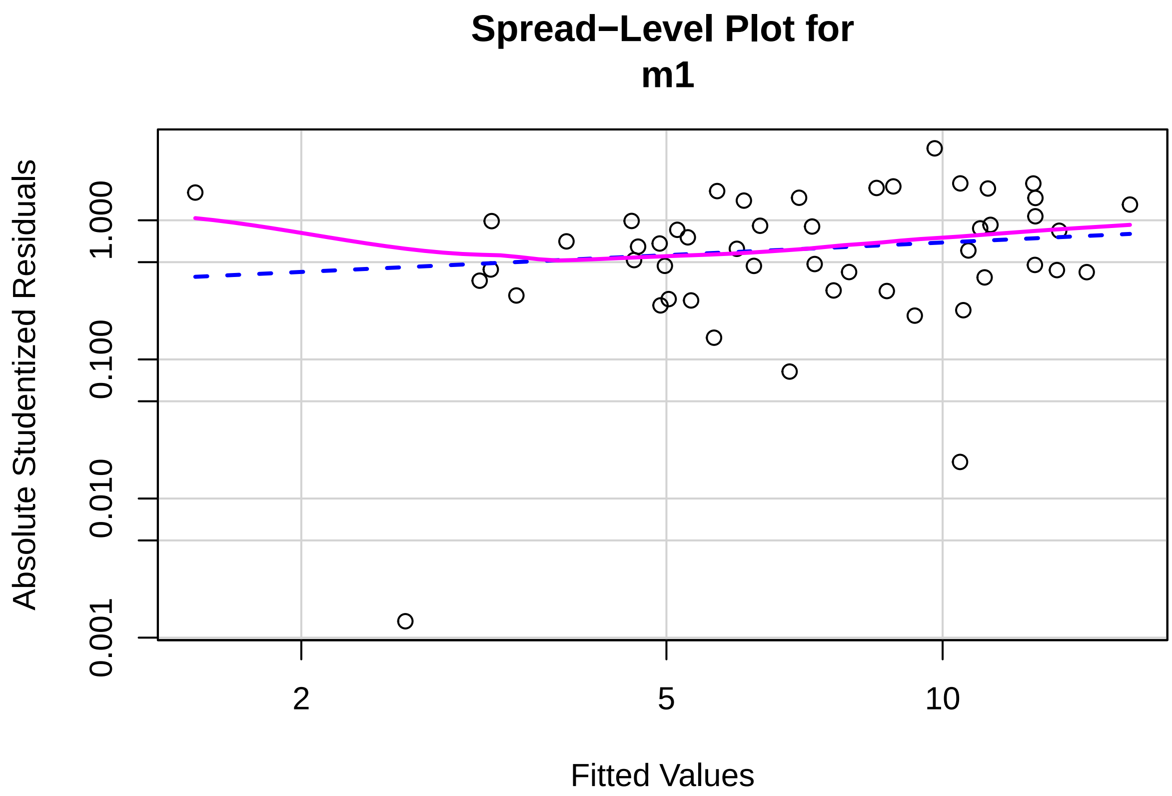
```r
### This function performs a chi-square test where the null hypothesis
### is homoscedasticity. We want to see a chi-square with p > 0.05.
### A high chi-square with p < 0.05 implies that the residuals are not
### consistent at all points of the regression line. You can visualize
### this concept with residual plots and spread level plots.

## Residual plots
residuals <- resid(m1)

plot(USArrests$Assault, residuals,
     ylab = "Residuals",
     xlab = "Assault (per 100,000)",
     main = "Residual Plot")
abline(lm(residuals ~ USArrests$Assault))
```




```r
plot(USArrests$UrbanPop, residuals,
     ylab = "Residuals",
     xlab = "Urban Population (%)",
     main = "Residual Plot")
abline(lm(residuals ~ USArrests$UrbanPop))
```




```r
## Spread level plots
### You want the blue dashed line to conform to the solid pink line.
spreadLevelPlot(m1)
```

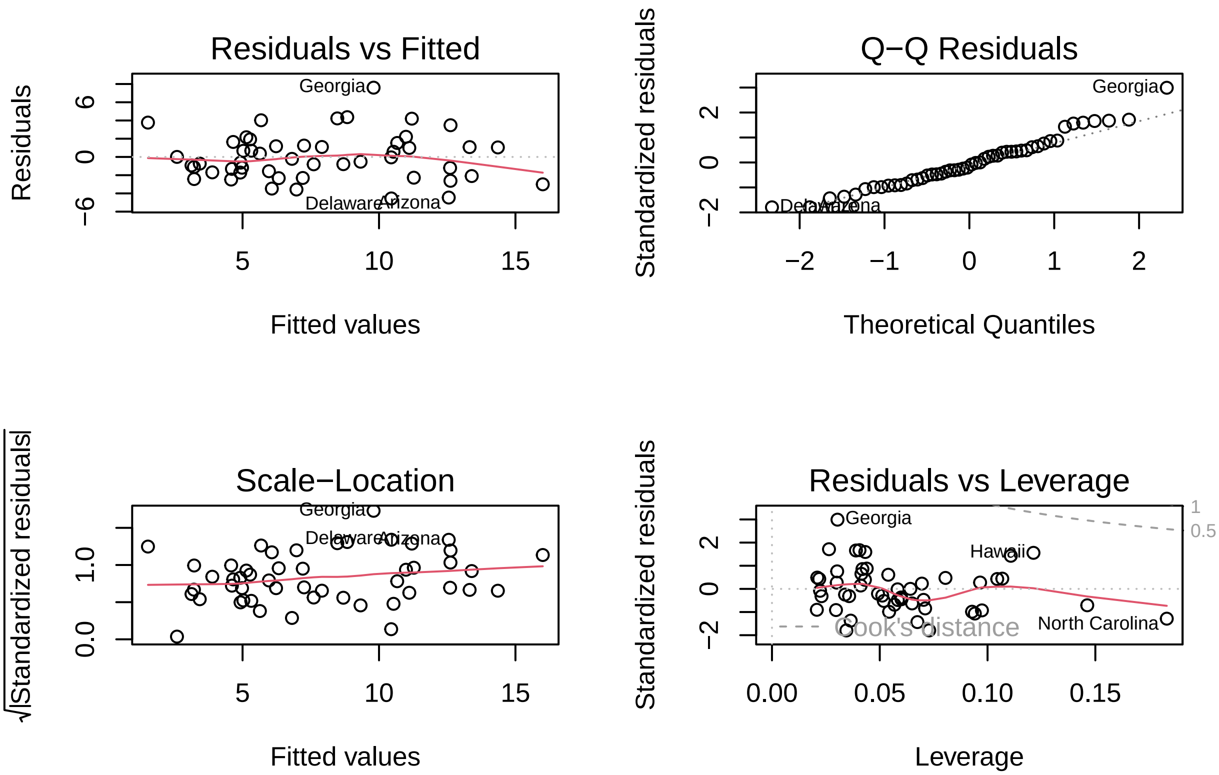



```
##
## Suggested power transformation:  0.6974395
```

```
# ============================================================================ #

# Additional diagnostics
## The gvlma package is very helpful for evaluating model skewness, kurtosis,
## and heteroscedasticity. It will inform you of any violated assumptions.
summary(gv <- gvlma(m1))
```

```
## Error in h(simpleError(msg, call)): error in evaluating the argument 'object' in selecting a method f
```

```
## The plot() function can also be used to automatically generate a battery
## of fit visualizations, including a residuals v. fitted plot, QQ plot,
## scale-location plot, and residuals v. leverage plot.
par(mfrow = c(2, 2))
plot(m1)
```




```r
## Before proceeding, execute this code to reset your 'plots' tab
## to its original settings:
par(mfrow = c(1, 1))
```