Week 3 - Data Screening and Cleaning Tutorial

Professor Christopher J. Schmank

2025-02-04

Exploratory data analysis (EDA)

Remember: Like in univariate we will **ALWAYS** begin with EDA before checking assumptions, running inferential tests, or statistical models

1. Physically observe raw data (i.e., spreadsheet) [CRUCIAL STEP!]
2. Compute summary statistics (e.g., means, medians, and standard deviations)
3. Create and assess data visualizations
4. Assess specific inferential test assumptions (e.g., Univariate: Outliers, Normality, Linearity; Homoscedasticity; Multivariate Outliers, Normality, Collinearity)

Steps to Data Screening and Cleaning

1. Start with traditional Exploratory Data Analysis
    a. Eyeball Check
    b. Data Visualizations
    c. Univariate Statistics Descriptions
2. Evaluate presence/pattern of missing data and impute data if necessary
3. Assess linearity and homoscedasticity (i.e., variance sameness assessments)
4. Assess normality and univariate outliers
5. Assess/identify multivariate outliers
6. Assess multicollinearity and singularity

Load/Install Required `R Packages`

```
# install.packages("tidyverse", dependencies = TRUE)
# install.packages("psych", dependencies = TRUE)
# install.packages("mice", dependencies = TRUE)
# install.packages("psych", dependencies = TRUE)
# install.packages("lmtest", dependencies = TRUE)
# install.packages("jmv", dependencies = TRUE)
# install.packages("MVN", dependencies = TRUE)
# install.packages("naniar", dependencies = TRUE)

library(mice)
library(naniar)
library(lmtest)
library(jmv)
library(MVN) # Can also use library(mvnormalTest) if installed
library(psych)
library(tidyverse)
```

Step 1A: Accuracy of the Data File

- Open data file in Excel or R; preview variables of interest
    - Must complete this step carefully—**do not rush!**
    - Use `View()` function to open data frame in R

- Sort ascending/descending for each variable to assess `minimums` and `maximums` ; take note of missing data/cases with missingness

General Data Importing Template:

```
dat <- <ImportDataFrame>
View(dat)
```

In Your R Environment Run the Following Lines of Code:

```
View(cattell)
View(attitude)
View(nhanes)
```

- What do you make of these data files?
- Are they tidy? If not, what would make them tidy?
- Do any of them have missing data?
- Would any of them be problematic for EDA? Why or Why not?

Step 1B: Examine Descriptive Statistics

- We will focus on using the `psych` package
  - Produces frequently requested descriptive statistics for psychological or psychometric studies
  - `describe()` runs descriptive statistics for all variables in data frame at once
  - `describe_by()` runs descriptive statistics for all variables in data frame at once broken up by grouping variable(s)

Descriptive Statistics Template:

```
describe(dat)
describeBy(dat, group=<GROUP>)
```

By default `describe()` and `describeBy()` will:

1. Remove missing data listwise — `na.rm = TRUE`
2. Calculate skew and kurtosis — `skew = TRUE`
3. Calculate the range — `ranges = TRUE`

`describe()` Function and Output

Initial EDA should focus on the following for each variable of interest:

1. `Means`
2. `Standard Deviations`
3. `Minimums & Maximums`
4. `Sample Sizes`
5. `Skewness & Kurtosis`

```
describe(attitude)
```

```
##            vars  n  mean    sd median trimmed   mad min max range  skew
## rating        1 30 64.63 12.17   65.5   65.21 10.38  40  85    45 -0.36
## complaints    2 30 66.60 13.31   65.0   67.08 14.83  37  90    53 -0.22
## privileges    3 30 53.13 12.24   51.5   52.75 10.38  30  83    53  0.38
## learning      4 30 56.37 11.74   56.5   56.58 14.83  34  75    41 -0.05
## raises        5 30 64.63 10.40   63.5   64.50 11.12  43  88    45  0.20
## critical      6 30 74.77  9.89   77.5   75.83  7.41  49  92    43 -0.87
## advance       7 30 42.93 10.29   41.0   41.83  8.90  25  72    47  0.85
##            kurtosis   se
## rating        -0.77 2.22
## complaints    -0.68 2.43
## privileges    -0.41 2.23
## learning      -1.22 2.14
## raises        -0.60 1.90
## critical       0.17 1.81
## advance        0.47 1.88
```

```
describe(nhanes)
```

```
##     vars  n   mean    sd median trimmed   mad   min   max range  skew kurtosis
## age    1 25   1.76  0.83   2.00    1.71  1.48   1.0   3.0   2.0  0.44    -1.47
## bmi    2 16  26.56  4.22  26.75   26.38  4.60  20.4  35.3  14.9  0.39    -0.83
## hyp    3 17   1.24  0.44   1.00    1.20  0.00   1.0   2.0   1.0  1.14    -0.73
## chl    4 15 191.40 45.22 187.00  190.31 28.17 113.0 284.0 171.0 -0.09    -0.43
##        se
## age  0.17
## bmi  1.05
## hyp  0.11
## chl 11.67
```

`describeBy()` Function and Output

The following data comes from the `datarium` package (`library(datarium)`) and assesses descriptive statistics for the anxiety scores (`t1`) across three groups (`anxiety`)

```
describeBy(x = t1,
           group = group,
           data = anxiety,
           mat = TRUE)
```

```
##     item group1 vars  n     mean       sd median  trimmed     mad  min  max
## X11    1   grp1    1 15 17.08667 1.628701   17.0 17.10769 1.33434 14.1 19.8
## X12    2   grp2    1 15 16.64667 1.565643   16.9 16.66923 1.63086 13.7 19.3
## X13    3   grp3    1 15 17.01333 1.321183   17.3 17.04615 1.33434 14.6 19.0
##     range       skew   kurtosis        se
## X11   5.7 -0.1029298 -0.8254734 0.4205288
## X12   5.6 -0.2002025 -1.0718963 0.4042473
## X13   4.4 -0.3718028 -1.1608155 0.3411279
```

Step 1C: Visualize Variables of Interest

Generate and assess univariate plots for all continuous variables to be used in the statistical model:

1. Histograms — useful for looking at the distribution of data

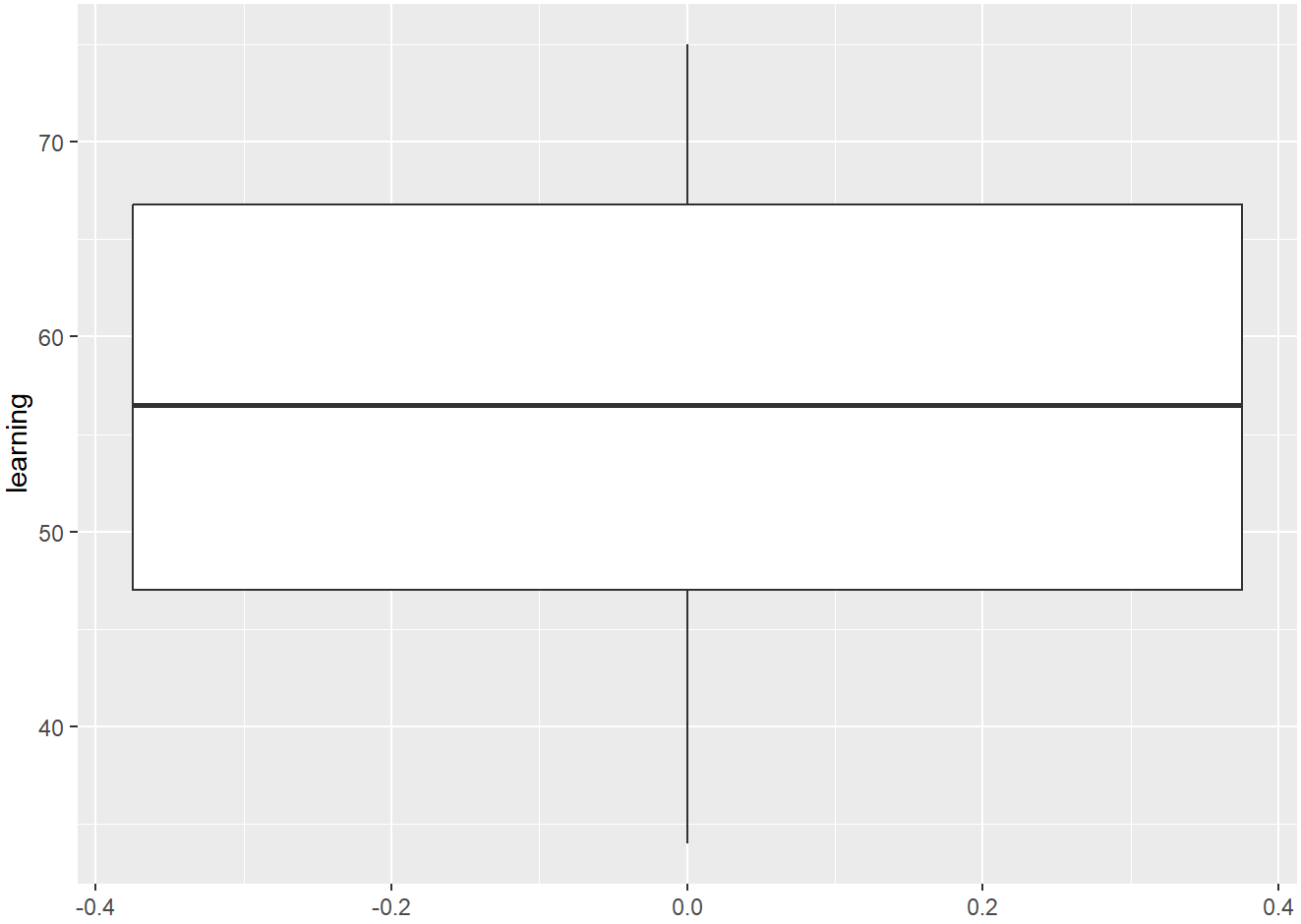
2. Boxplots — useful for quick assessment of outliers

Visualization Template:

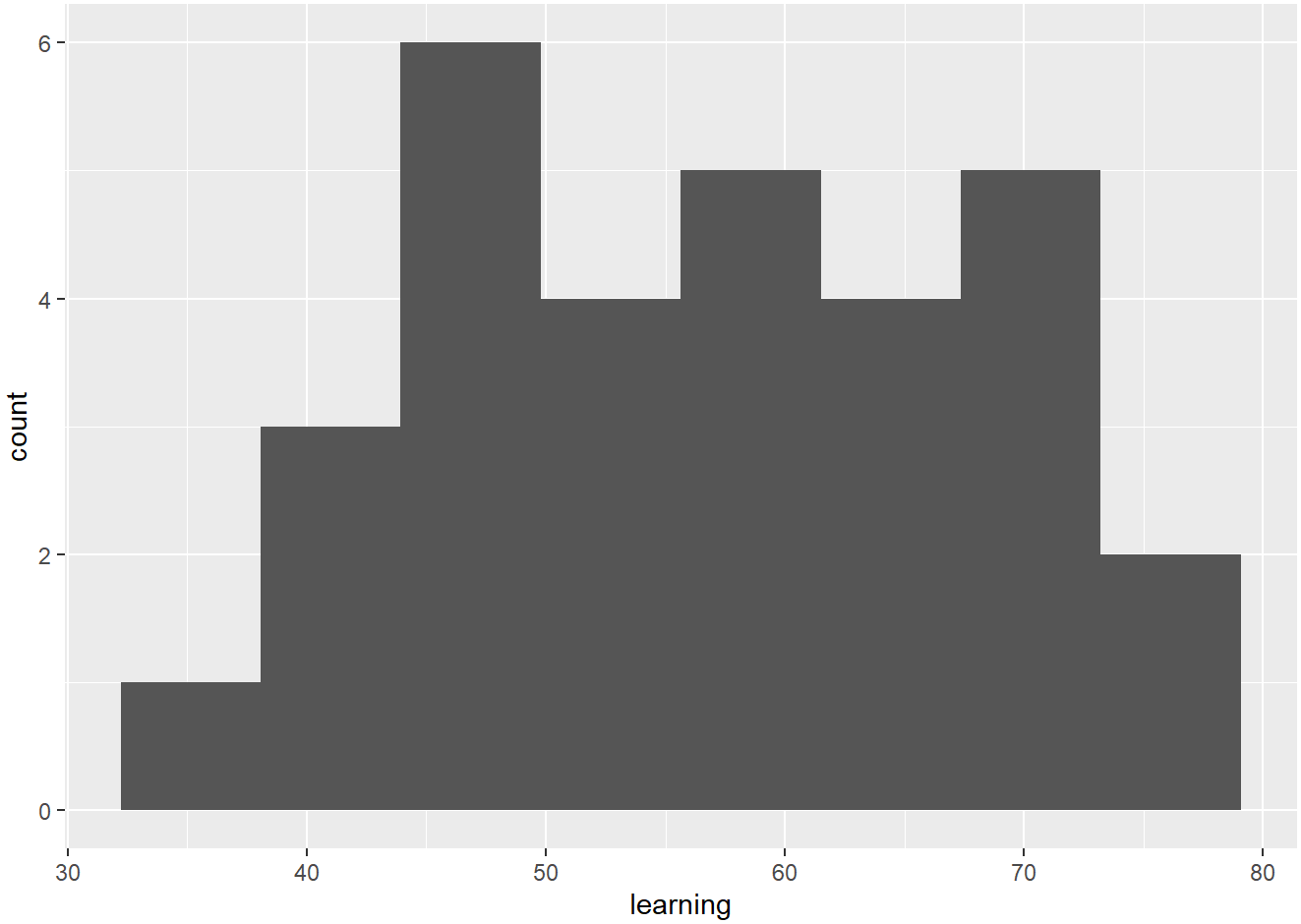
```
ggplot(data = <DATA>,
       mapping = aes(<MAPPINGS>)) +
  <GEOM_FUNCTION>()
```

`ggplot()` Function and Output

```
# Boxplot Visualization
ggplot(data = attitude,
       mapping = aes(y = learning)) +
  geom_boxplot()
```




```
# Histogram Visualization
ggplot(data = attitude,
       mapping = aes(x = learning)) +
  geom_histogram(bins = 8)
```



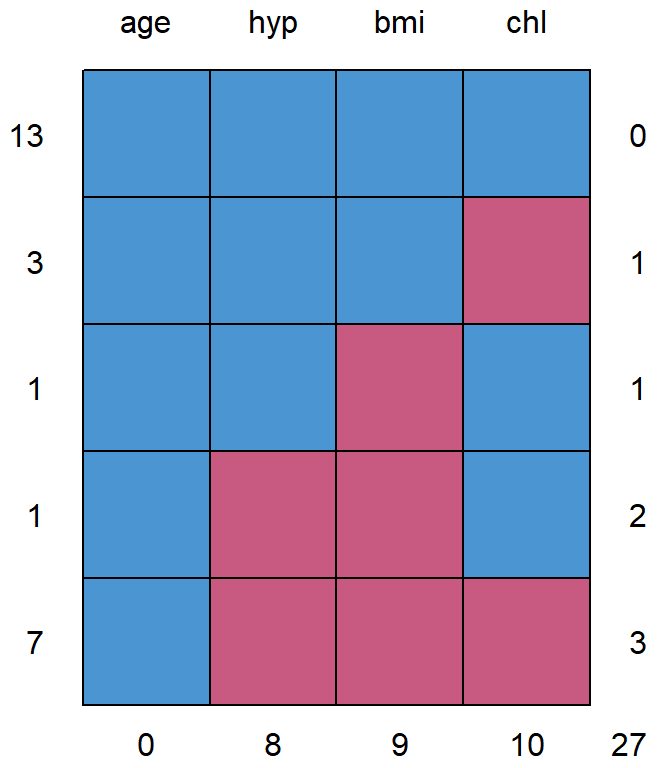
Step 2: Missing Data Analysis

Evaluation of Missing Data: Amount and Pattern

```
# Calculate Amount of Missing Data
sum(is.na(nhanes))/prod(dim(nhanes))
```

```
## [1] 0.27
```

```
# Generate Missing Data Pattern
md.pattern(nhanes)
```



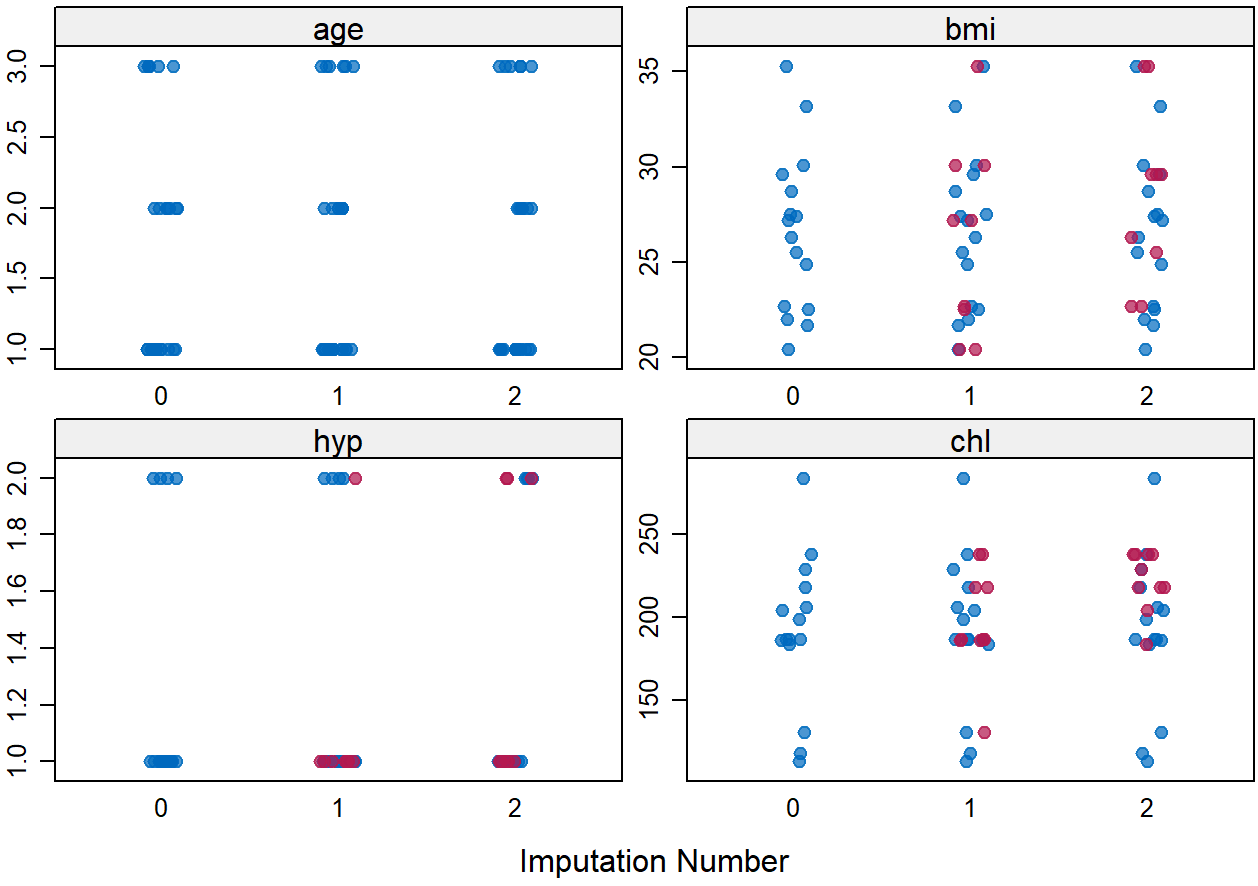

```
##    age hyp bmi chl
## 13   1   1   1   1  0
## 3    1   1   1   0  1
## 1    1   1   0   1  1
## 1    1   0   0   1  2
## 7    1   0   0   0  3
##      0   8   9  10 27
```

```
# This pattern indicates that there are 7 distinct patterns of missingness
# 13 participants have no missing data at all
# 3 participants are only missing cholesterol data
# 1 participant is missing BMI data
# 1 participant is missing a value for `hyp` and BMI
# 7 participants are missing all variables except for age
```

```
# Statistical test of Missingness Completely at Random
mcar_test(nhanes)
```

```
## # A tibble: 1 × 4
##   statistic    df p.value missing.patterns
##       <dbl> <dbl>   <dbl>            <int>
## 1      8.00     9   0.534                5
```

```
# Use the `mice()` function to impute missing data
imp_dat <- mice(nhanes,
                maxit = 100,
                m = 2,
                seed = 13,
                method = "pmm",
                print=FALSE)
# `maxit = ` is the number of iterations conducted (default = 5)
# `m =` the number of imputed data sets created (default = 5)
# `seed =` IMPORTANT: Allows for replicable imputations (default = NA)
# `method =` what type of imputation method is used --- see documentation
# `print = FALSE` code runs but no output listed

# Can visualize the imputed data using `stripplot()`
# Important to ensure that no imputed data falls outside sampled data!!
stripplot(imp_dat,
          pch = 19,
          xlab = "Imputation Number")
```



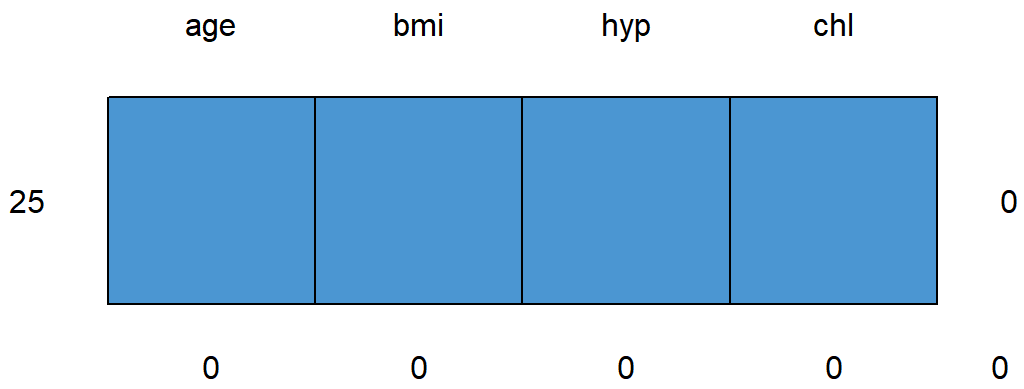

```
# `complete()` function can be used to pull a specific imputation dataset (default = 1)
dat_imp1 <- complete(imp_dat, action = 1)
dat_imp2 <- complete(imp_dat, action = 2)

# Can check the pattern of missingness again to show that missingness has been imputed
md.pattern(dat_imp1)
```

```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```

```
##    age bmi hyp chl
## 25   1   1   1   1 0
##      0   0   0   0 0
```

```
md.pattern(dat_imp2)
```

```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```




```
##    age bmi hyp chl
## 25   1   1   1   1 0
##      0   0   0   0 0
```

```
# Can assess redundancy of data sets using correlation coefficient (> .70 is good news!)
cor(dat_imp1$bmi,dat_imp2$bmi)
```

```
## [1] 0.8440181
```

```
cor(dat_imp1$hyp,dat_imp2$hyp)
```

```
## [1] 0.8017837
```

```
cor(dat_imp1$chl,dat_imp2$chl)
```

```
## [1] 0.8450673
```

Once data has been imputed you would choose an imputed data set to use throughout the remainder of the statistical analyses —always important to run data with missing values as well (if possible) to see how interpretation changes after imputation!

Step 3A: Check Linearity

Best way to assess **linearity** is:

- Correlation Matrix & Scatterplots

Correlation Matrix

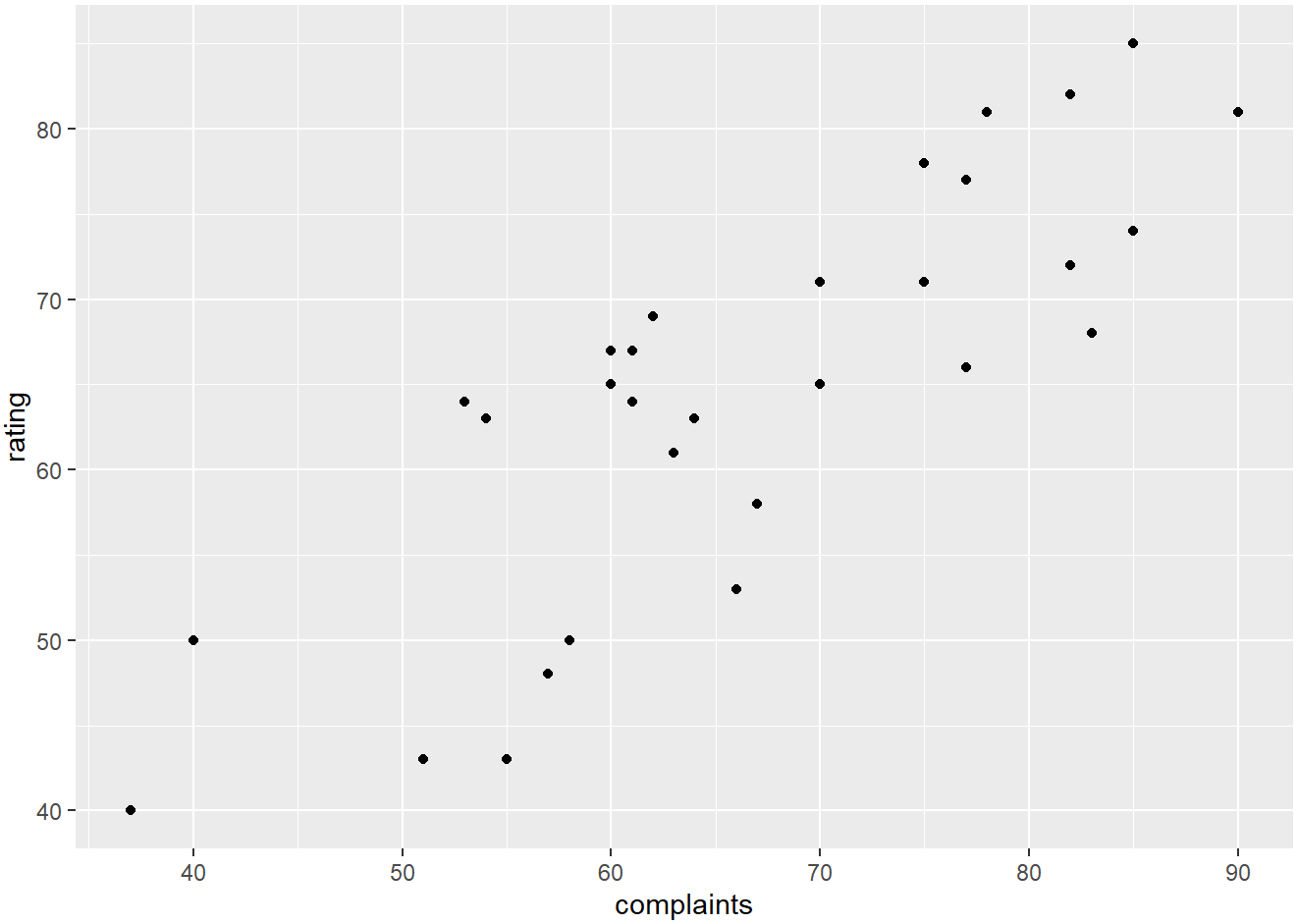
```
corrMatrix(attitude[1:3])
```

```
##
##  CORRELATION MATRIX
##
##  Correlation Matrix
##  ──────────────────────────────────────────────────────────────────────
##                                rating        complaints    privileges
##  ──────────────────────────────────────────────────────────────────────
##    rating        Pearson's r             —
##                  df                      —
##                  p-value                 —
##
##    complaints    Pearson's r     0.8254176             —
##                  df                     28             —
##                  p-value        < .0000001             —
##
##    privileges    Pearson's r     0.4261169     0.5582882             —
##                  df                     28            28             —
##                  p-value         0.0188770     0.0013455             —
##  ──────────────────────────────────────────────────────────────────────
```

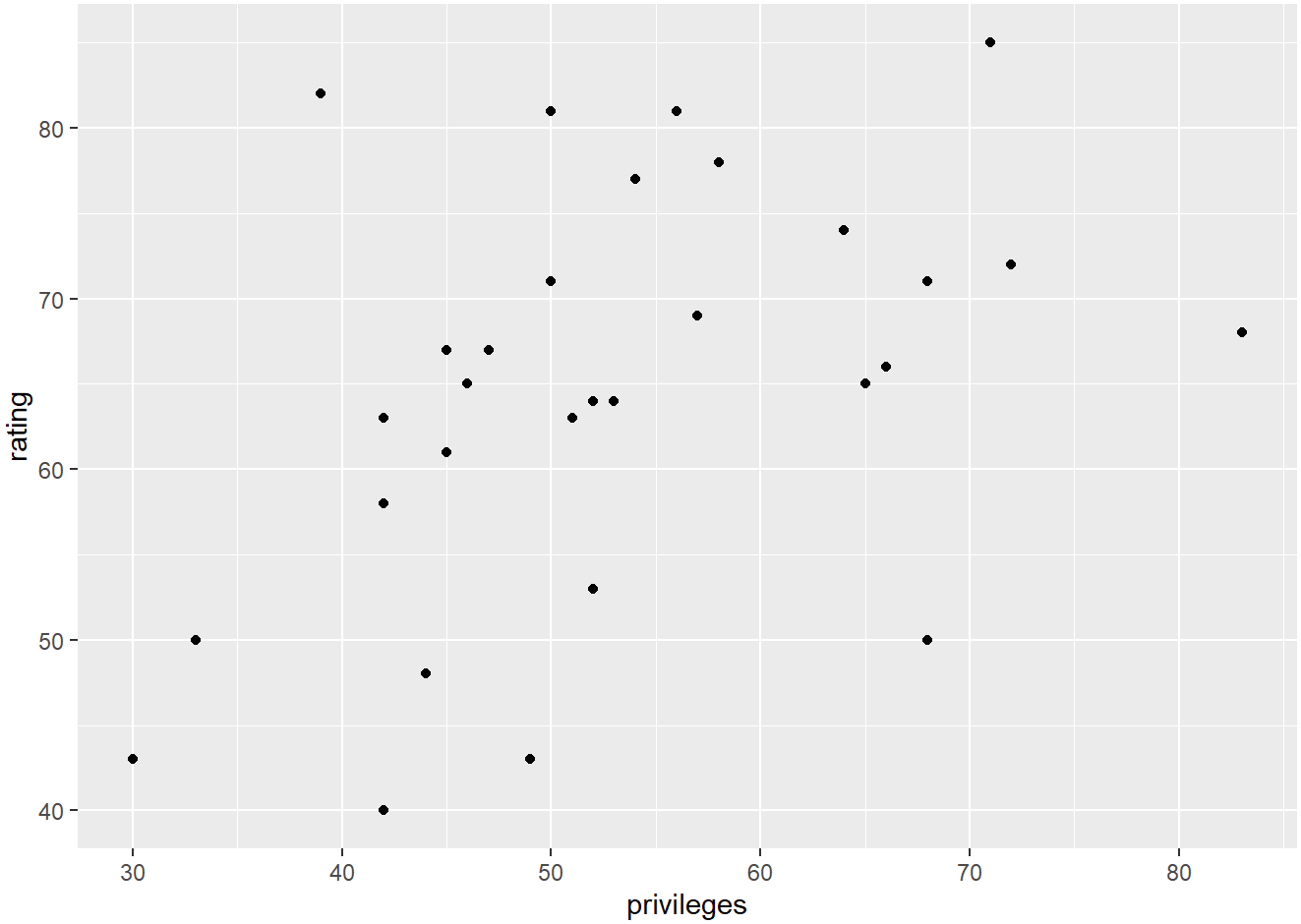
Scatterplot Template and Output

```
ggplot(data = <DATA>,
       mapping = aes(x = <X Variable>,
                     y = <Y Variable>)) +
  geom_point()
```

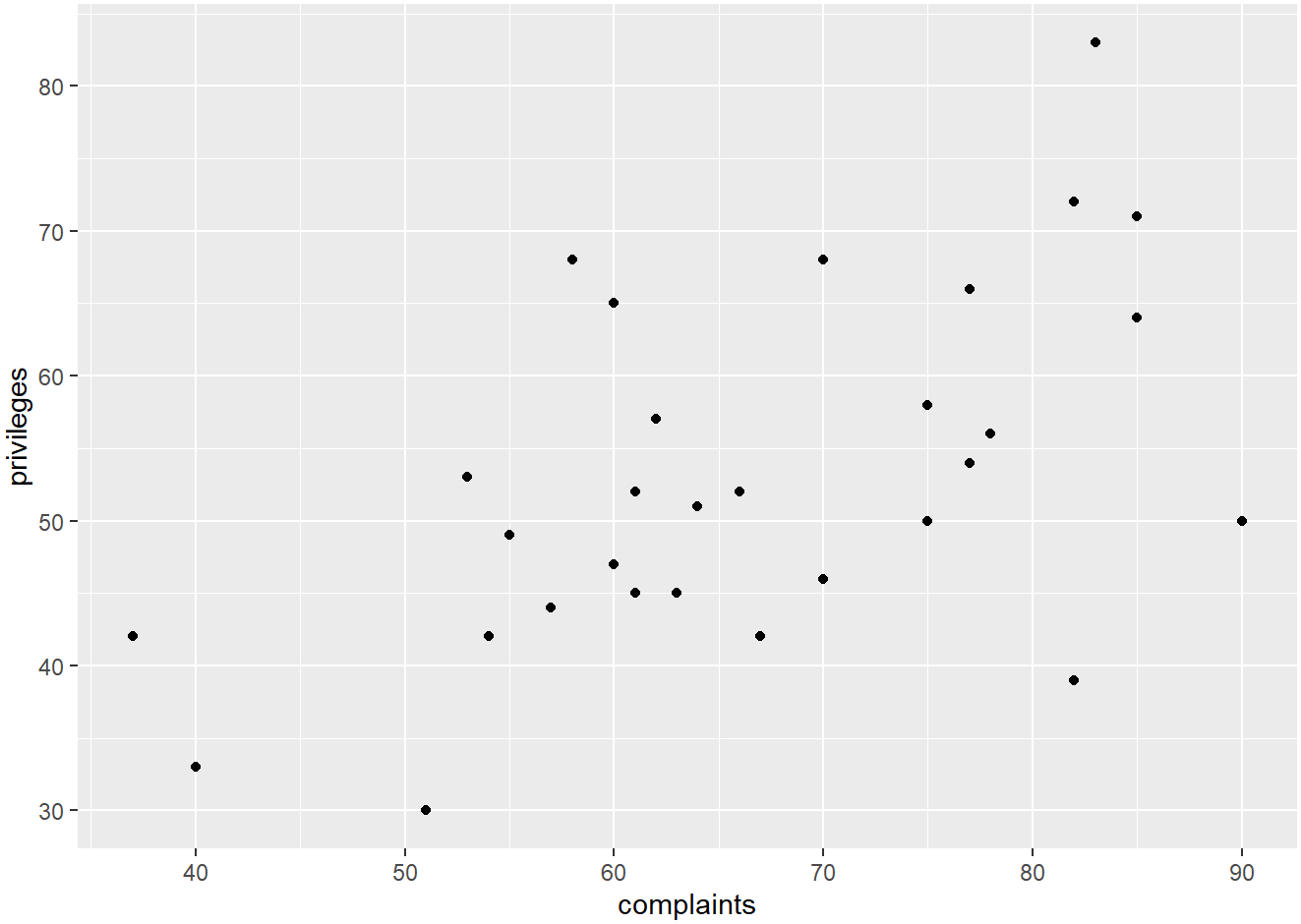
```
# Scatterplot of Ratings x Complaints
ggplot(data = attitude,
       mapping = aes(x = complaints,
                     y = rating)) +
  geom_point()
```




```
# Scatterplot of Ratings x Privileges
ggplot(data = attitude,
       mapping = aes(x = privileges,
                     y = rating)) +
  geom_point()
```




```
# Scatterplot of Privileges x Complaints
ggplot(data = attitude,
       mapping = aes(x = complaints,
                     y = privileges)) +
  geom_point()
```



Step 3B: Check Homoscedasticity

Best way to assess **homoscedasticity** is:

- Breusch-Pagan and Studentized Breusch Pagan Tests

Breusch-Pagan and Non-Constant Variance Test Template

```
# Create Linear Model
lm_model <- lm(dat$DV ~ dat$Predictor1 + dat$Predictor2 + ...
           + dat$PredictorK)

#Breusch-Pagan Test
bptest(lm_model, varformula = ~fitted.values(lm_model), stdentize=FALSE)

# Studentized Breusch-Pagan Test is a more conservative option
bptest(lm_model, varformula = ~fitted.values(lm_model), stdentize=TRUE)
```

Want the significance of these tests to be NON-STATISTICALLY significant $(p > .05)$

```
# Generates a linear model predicting attitude ratings from complaints and privilege data
bplm <- lm(rating~complaints+privileges, data = attitude)

#Breusch-Pagan Test
bptest(bplm, varformula = ~ fitted.values(bplm), studentize=FALSE)
```

```
##
##  Breusch-Pagan test
##
## data:  bplm
## BP = 0.97582, df = 1, p-value = 0.3232
```

```
#Breusch-Pagan Test (Studentized)
bptest(bplm, varformula = ~ fitted.values(bplm), studentize=TRUE)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  bplm
## BP = 2.5437, df = 1, p-value = 0.1107
```

Step 4A: Identify Univariate Outliers

Univariate Outlier Template

```
#Counts number of values greater than z = +/-3.29, p = .001
data[abs(scale(data$IvK)) > 3.29, ]

#Following code removes any univariate outliers from data with
data_no_uni_outliers <- data[!abs(scale(data$IvK)) > 3.29, ]
```

```
# Will want to run this for ALL variables to be entered into statistical model
attitude[abs(scale(attitude$rating)) > 3.29, ]
```

```
## [1] rating     complaints privileges learning   raises     critical   advance
## <0 rows> (or 0-length row.names)
```

```
attitude[abs(scale(attitude$complaints)) > 3.29, ]
```

```
## [1] rating     complaints privileges learning   raises     critical   advance
## <0 rows> (or 0-length row.names)
```

```
attitude[abs(scale(attitude$privileges)) > 3.29, ]
```

```
## [1] rating     complaints privileges learning   raises     critical   advance
## <0 rows> (or 0-length row.names)
```

```
# No univariate outliers present for these three variables
```

Step 4B: Assess Univariate Normality

Best ways to assess univariate normality is:

- Check histograms and box-plot visualizations
- Assess levels of skewness/kurtosis
    - Skewness values more extreme than ± 3.00 (Liberal) OR ± 1.00 (Conservative)

- Kurtosis values more extreme than ± 10.00 (Liberal) OR ± 1.00 (Conservative)

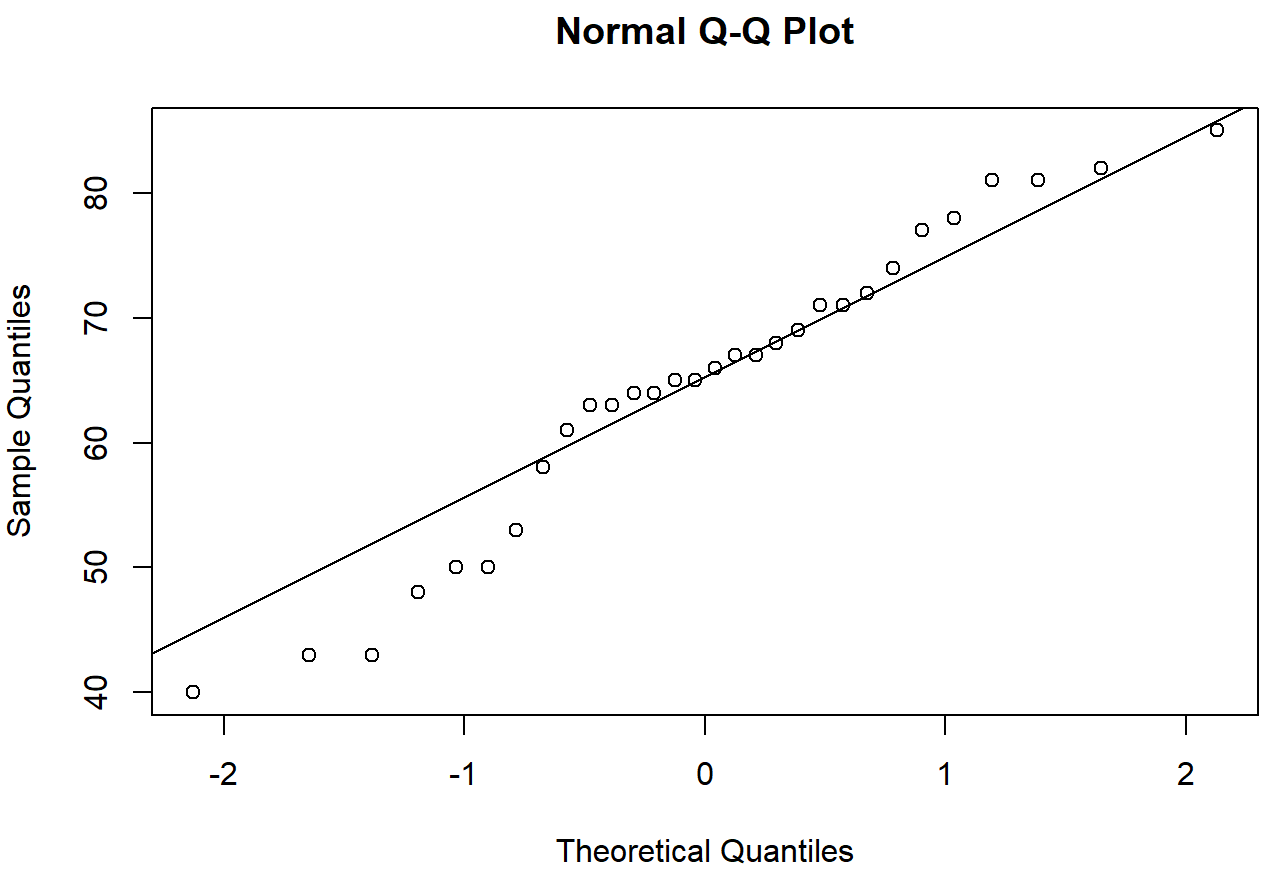
Can also use a statistical test and additional visualizations

- Shapiro-Wilks' test
- Q-Q Plots

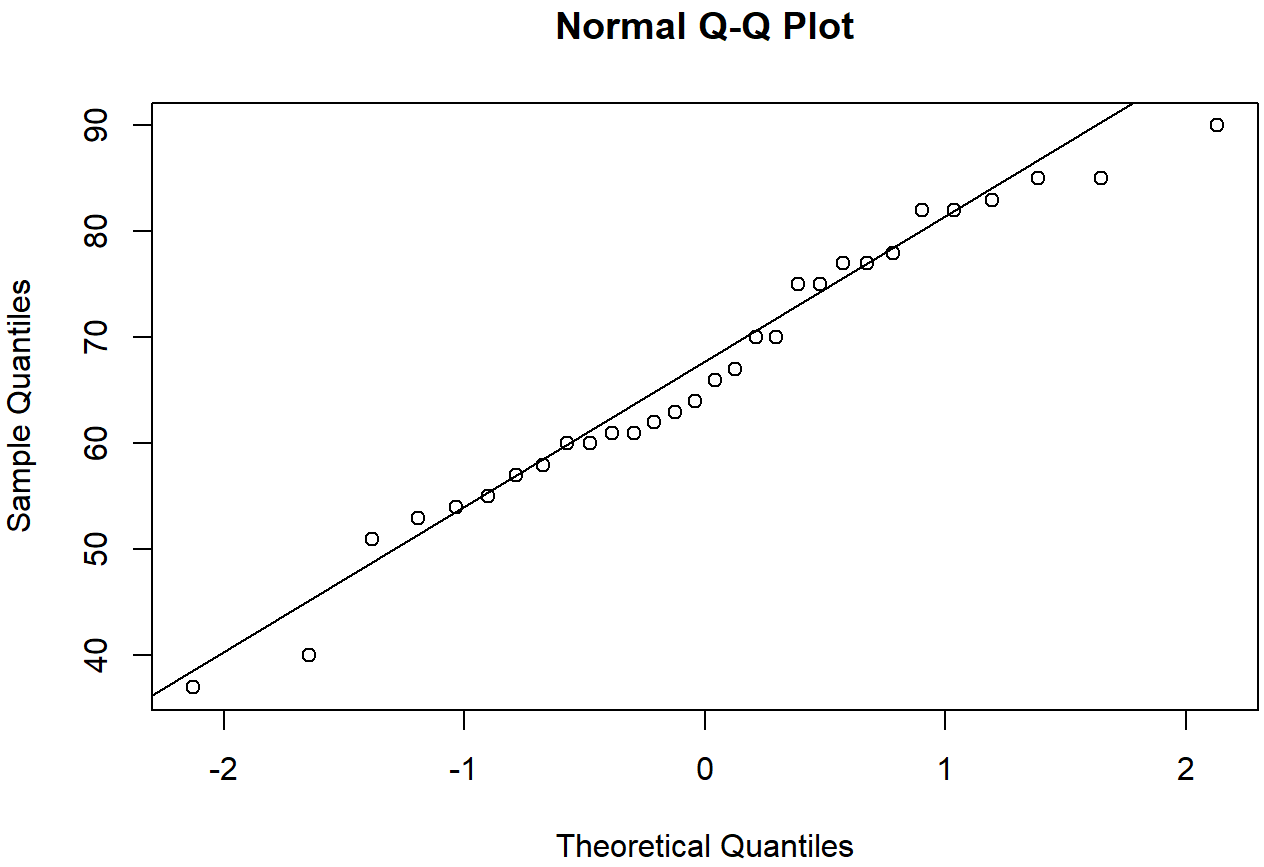
Shapiro-Wilks and Q-Q Plot Template

```
shapiro.test(x)
qqnorm(x)
qqline(x)
```

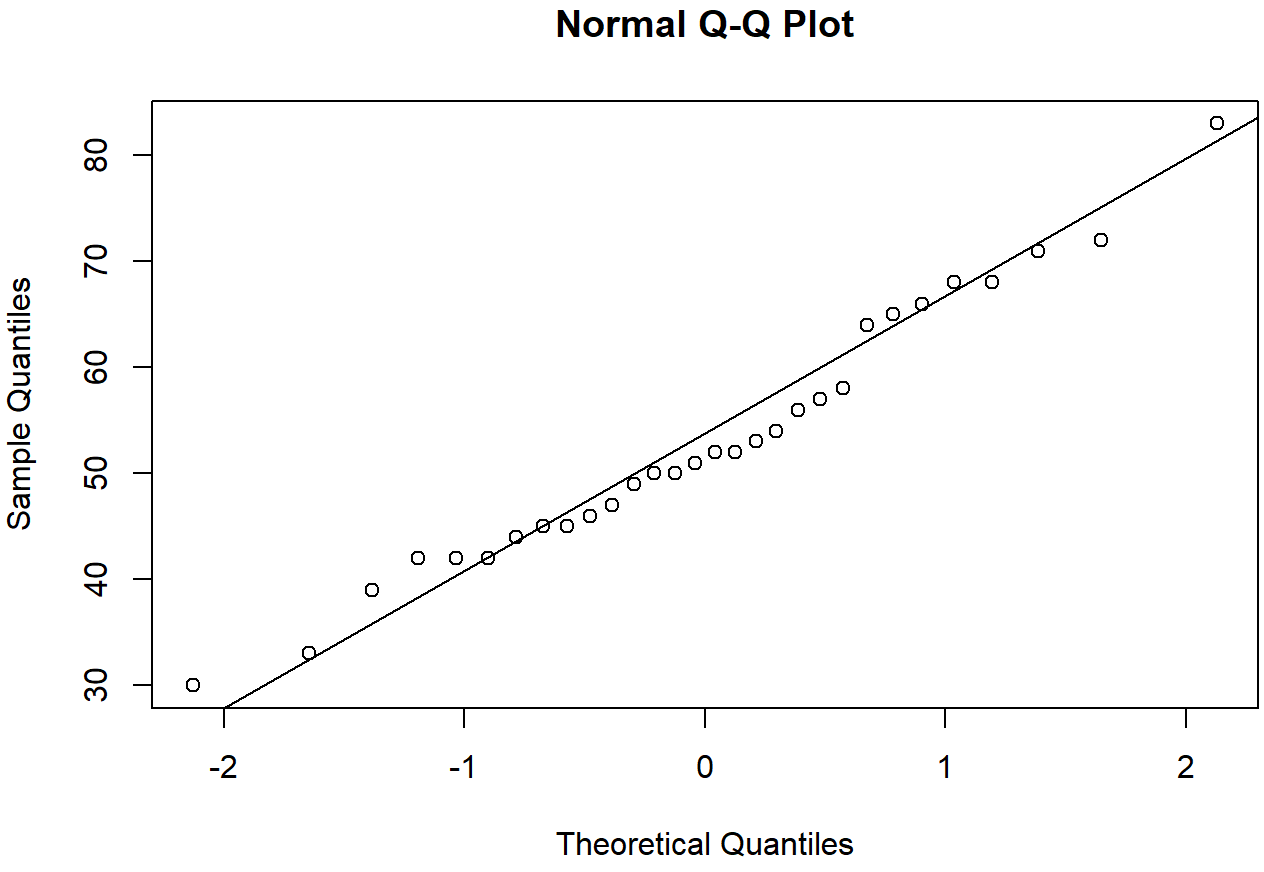
```
qqnorm(attitude$rating)
qqline(attitude$rating)
```




```
qqnorm(attitude$complaints)
qqline(attitude$complaints)
```




```
qqnorm(attitude$privileges)
qqline(attitude$privileges)
```



```
shapiro.test(attitude$rating)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  attitude$rating
## W = 0.9567, p-value = 0.2545
```

```
shapiro.test(attitude$complaints)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  attitude$complaints
## W = 0.97045, p-value = 0.5516
```

```
shapiro.test(attitude$privileges)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  attitude$privileges
## W = 0.97364, p-value = 0.6426
```

Want the significance of this test to be NON-STATISTICALLY significant $(p > .05)$

Step 4C: Assess Multivariate Normality

Best way to assess multivariate normality is:

- Statistical test triangulation
- Assessment of Residual Plots and Multivariate Q-Q plots

Multivariate Normality Statistical Test template

```
# Mardia's Test of Multivariate Normality
mardia <- mvn(data,
              mvnTest = "mardia",
              desc = FALSE)

# Henze-Zirkler's Test of Multivariate Normality
hz <- mvn(data,
          mvnTest = "hz",
          desc = FALSE)

# Energy Test of Multivariate Normality
energy <- mvn(data,
              mvnTest = "energy",
              desc = FALSE)

# Doornik-Hansen's Test of Multivariate Normality
dh <- mvn(data,
          mvnTest = "dh",
          desc = FALSE)

mardia$multivariateNormality
hz$multivariateNormality
energy$multivariateNormality
dh$multivariateNormality
```

```
# Mardia's Test of Multivariate Normality
mardia <- mvn(attitude[1:3],
              mvnTest = "mardia",
              desc = FALSE)

# Henze-Zirkler's Test of Multivariate Normality
hz <- mvn(attitude[1:3],
          mvnTest = "hz",
          desc = FALSE)

# Energy Test of Multivariate Normality
energy <- mvn(attitude[1:3],
              mvnTest = "energy",
              desc = FALSE)

# Doornik-Hansen's Test of Multivariate Normality
dh <- mvn(attitude[1:3],
          mvnTest = "dh",
          desc = FALSE)

mardia$multivariateNormality
```

```
##              Test         Statistic           p value Result
## 1 Mardia Skewness  6.15716330112075 0.801890686642304    YES
## 2 Mardia Kurtosis -1.09187365807145 0.274888637519459    YES
## 3             MVN              <NA>              <NA>    YES
```

```
hz$multivariateNormality
```

```
##            Test        HZ   p value MVN
## 1 Henze-Zirkler 0.6459459 0.3686721 YES
```

```
energy$multivariateNormality
```

```
##          Test Statistic p value MVN
## 1 E-statistic 0.8172546    0.51 YES
```

```
dh$multivariateNormality
```

```
##             Test        E df   p value MVN
## 1 Doornik-Hansen 4.170374  6 0.6536325 YES
```

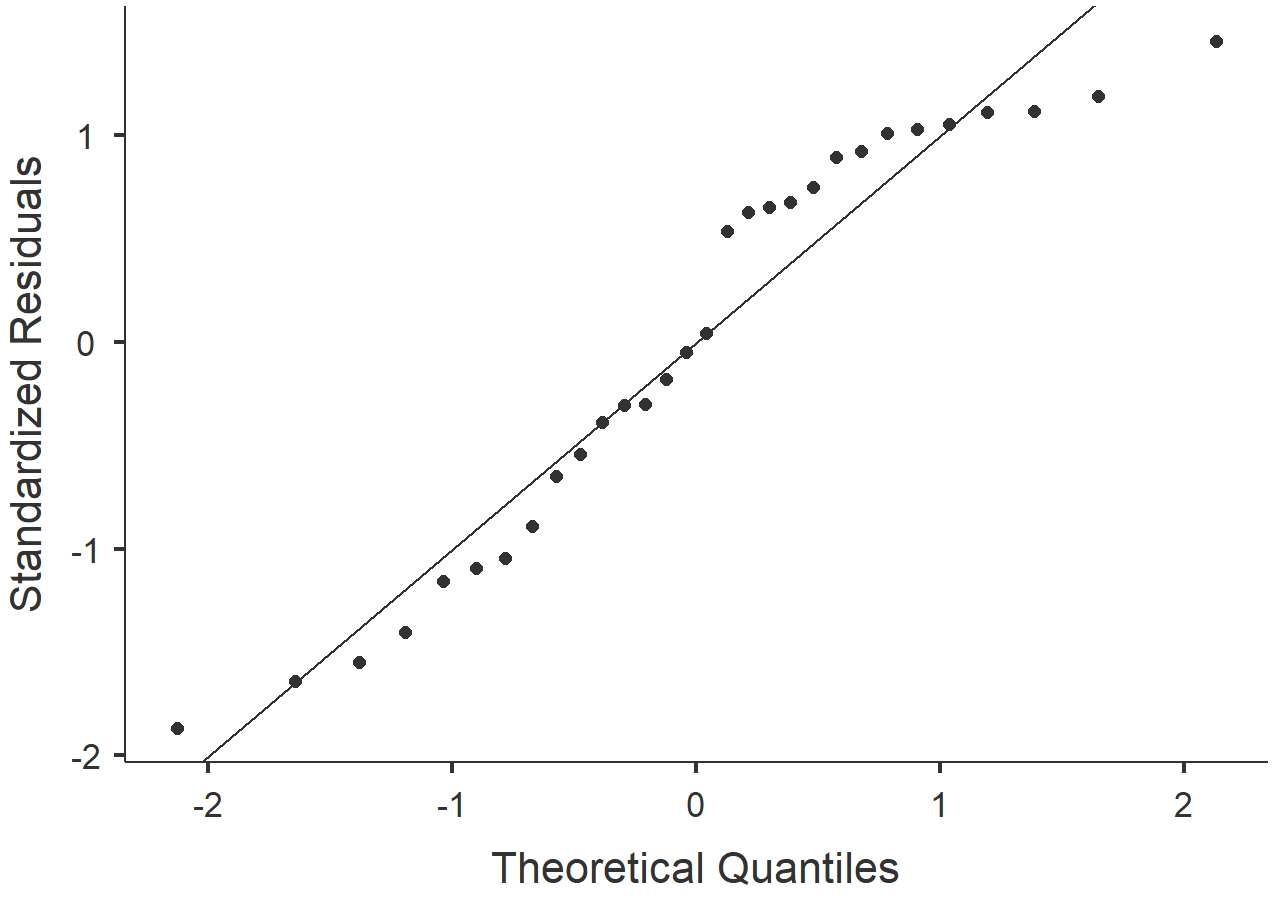
Want the significance of this test to be NON-STATISTICALLY significant $(p > .05)$

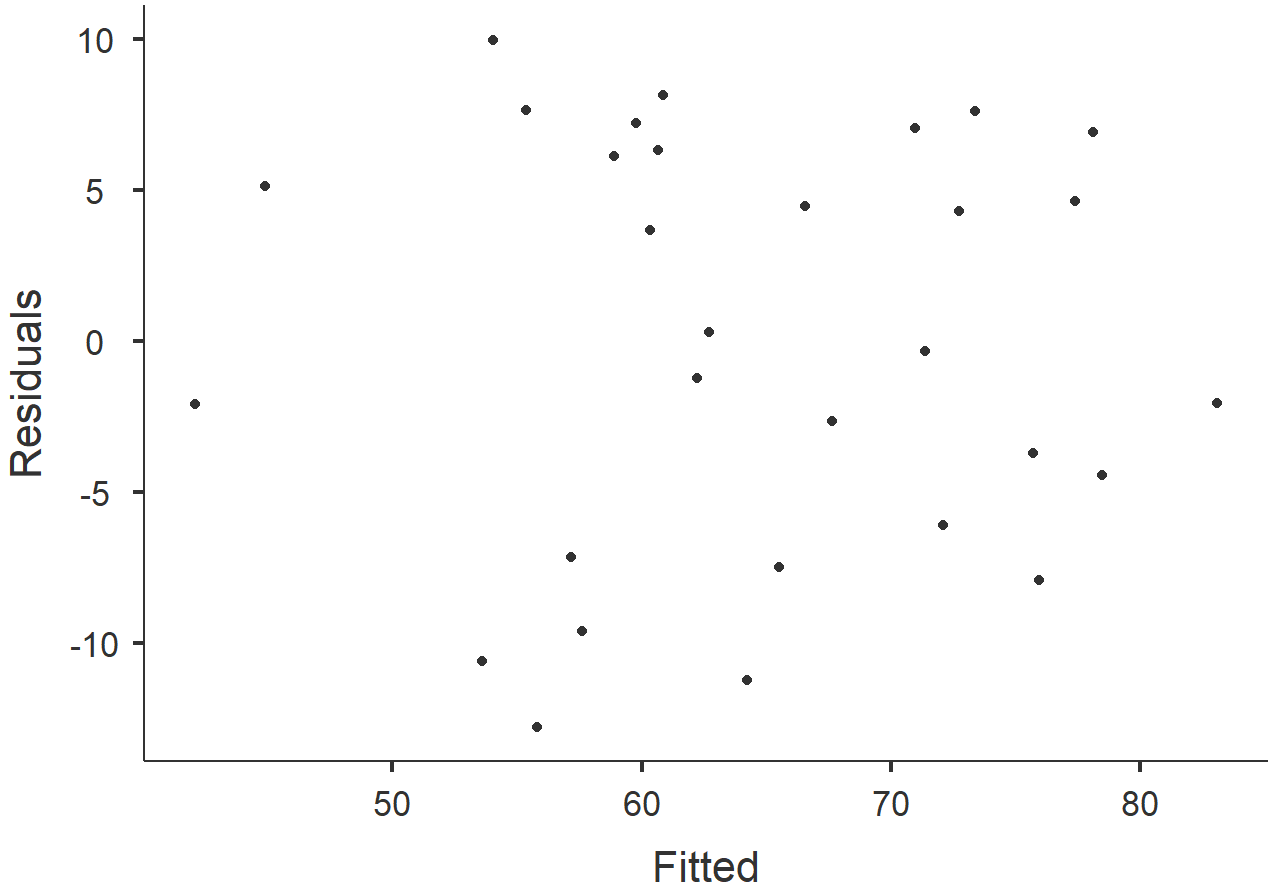
Residual and Q-Q plot template

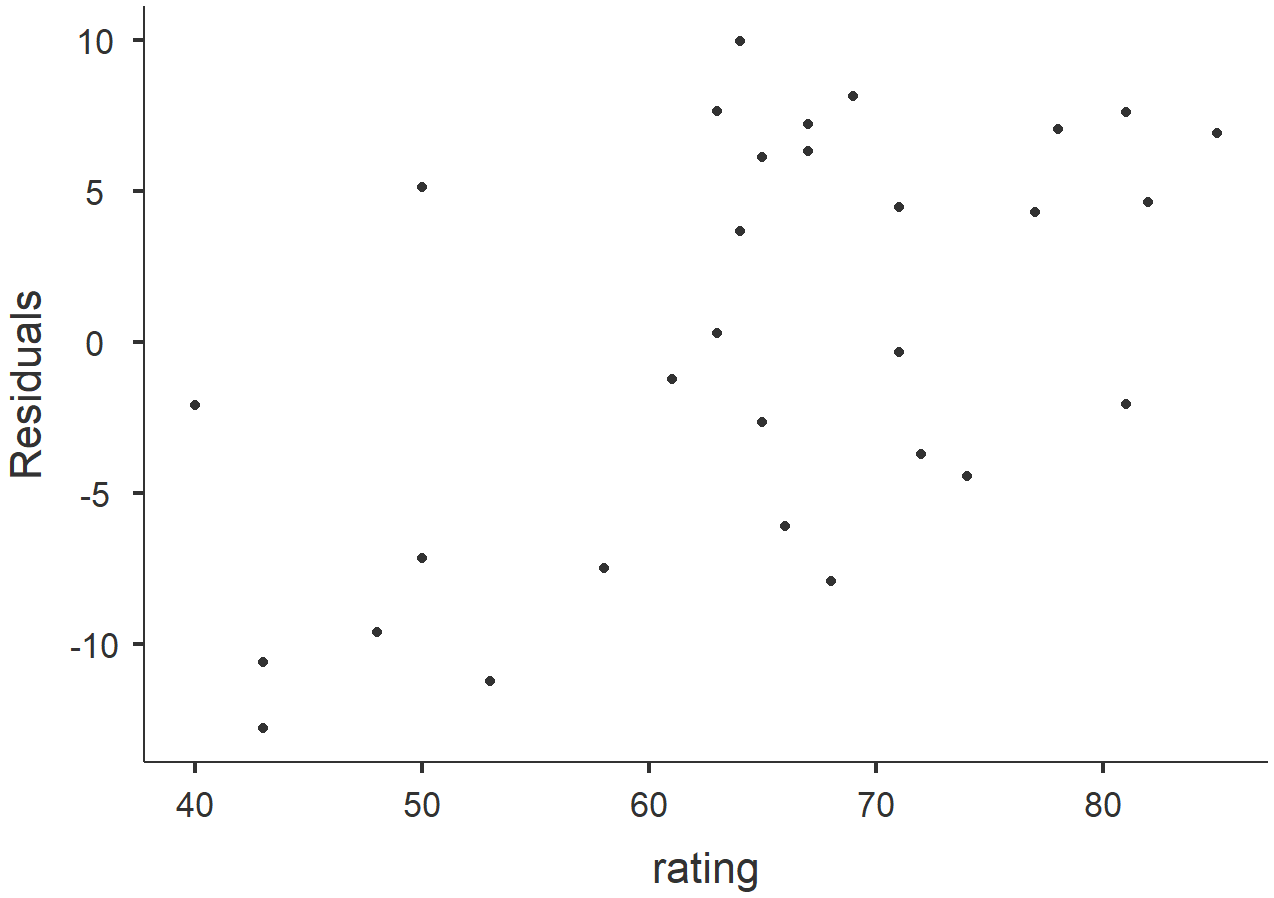
```
MNlm <- linReg(data = data,
               dep = <DV>,
               covs = c('IV1', 'IV2',..., 'IVK'),
               blocks = list(
                 list('IV1', 'IV2',..., 'IVK')),
               r = FALSE,
               r2 = FALSE,
               resPlots = TRUE,
               qqPlot = TRUE)

MNlm
```

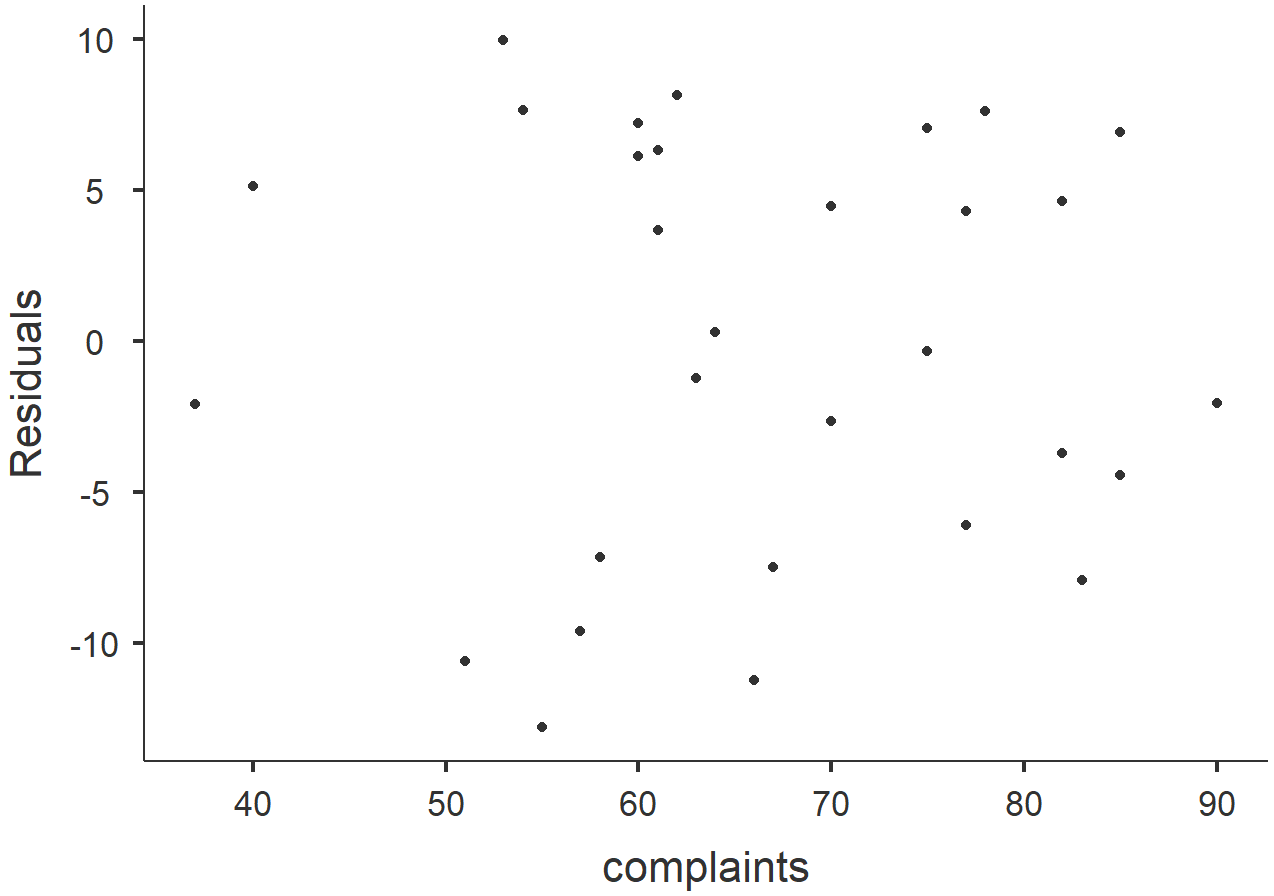
```
MNlm <- linReg(data = attitude,
               dep = 'rating',
               covs = c('complaints','privileges'),
               blocks = list(
                 list('complaints','privileges')),
               r = FALSE,
               r2 = FALSE,
               resPlots = TRUE,
               qqPlot = TRUE)

MNlm
```

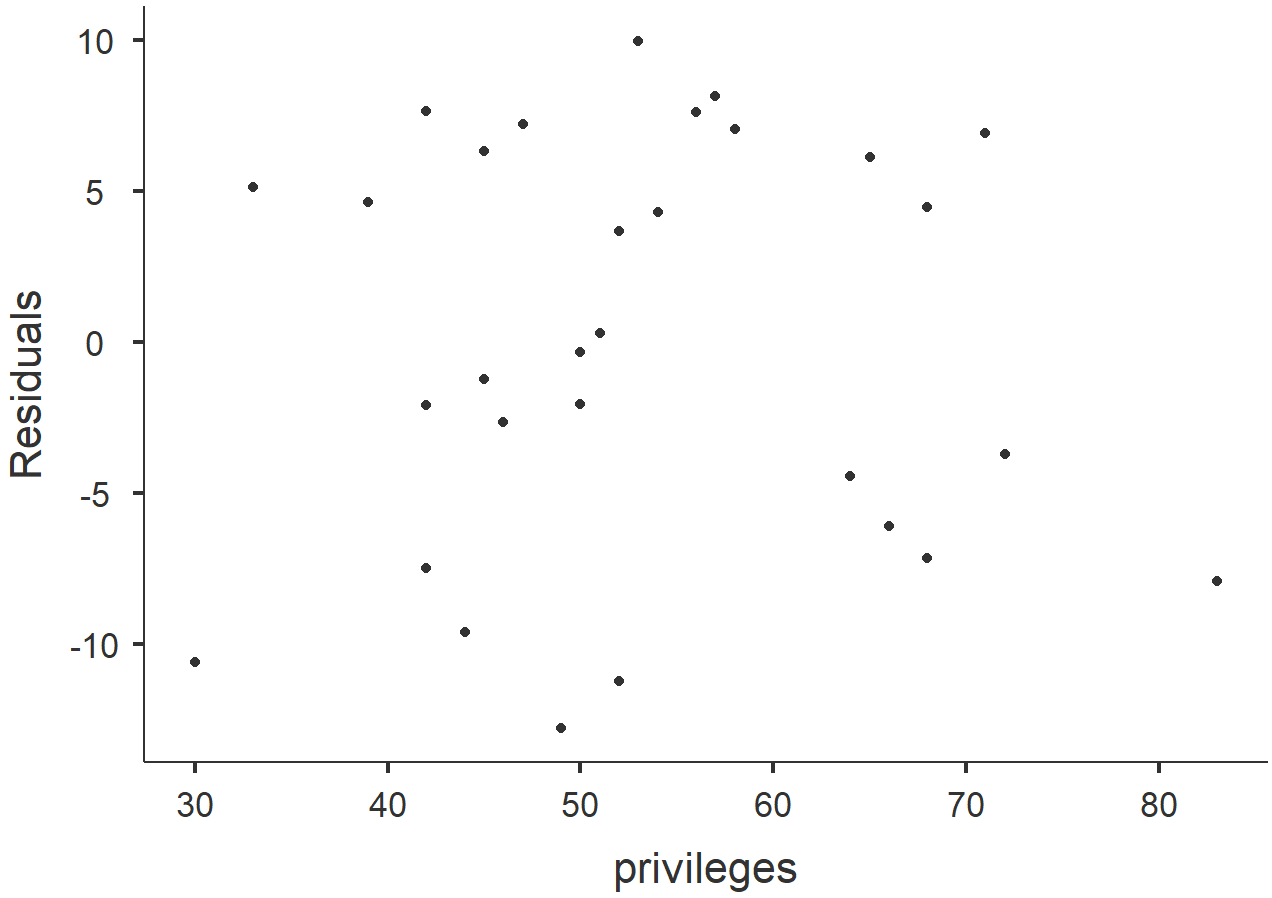
```
##
##  LINEAR REGRESSION
##
##  MODEL SPECIFIC RESULTS
##
##  MODEL 1
##
##  Model Coefficients - rating
##  ──────────────────────────────────────────────────────────────────────
##    Predictor     Estimate       SE          t            p
##  ──────────────────────────────────────────────────────────────────────
##    Intercept     15.32762017    7.1602343     2.1406590    0.0414872
##    complaints     0.78034340    0.1193876     6.5362166    0.0000005
##    privileges    -0.05015980    0.1299192    -0.3860845    0.7024597
##  ──────────────────────────────────────────────────────────────────────
##
##
##  ASSUMPTION CHECKS
```


Standardized Residuals
1
0
-1
-2
-2
-1
0
1
2
Theoretical Quantiles


Residuals
10
5
0
-5
-10
50
60
70
80
Fitted


Residuals
10
5
0
-5
-10
40
50
60
70
80
rating


Residuals
10
5
0
-5
-10
40
50
60
70
80
90
complaints



Step 5: Statistical Identification of Multivariate Outliers

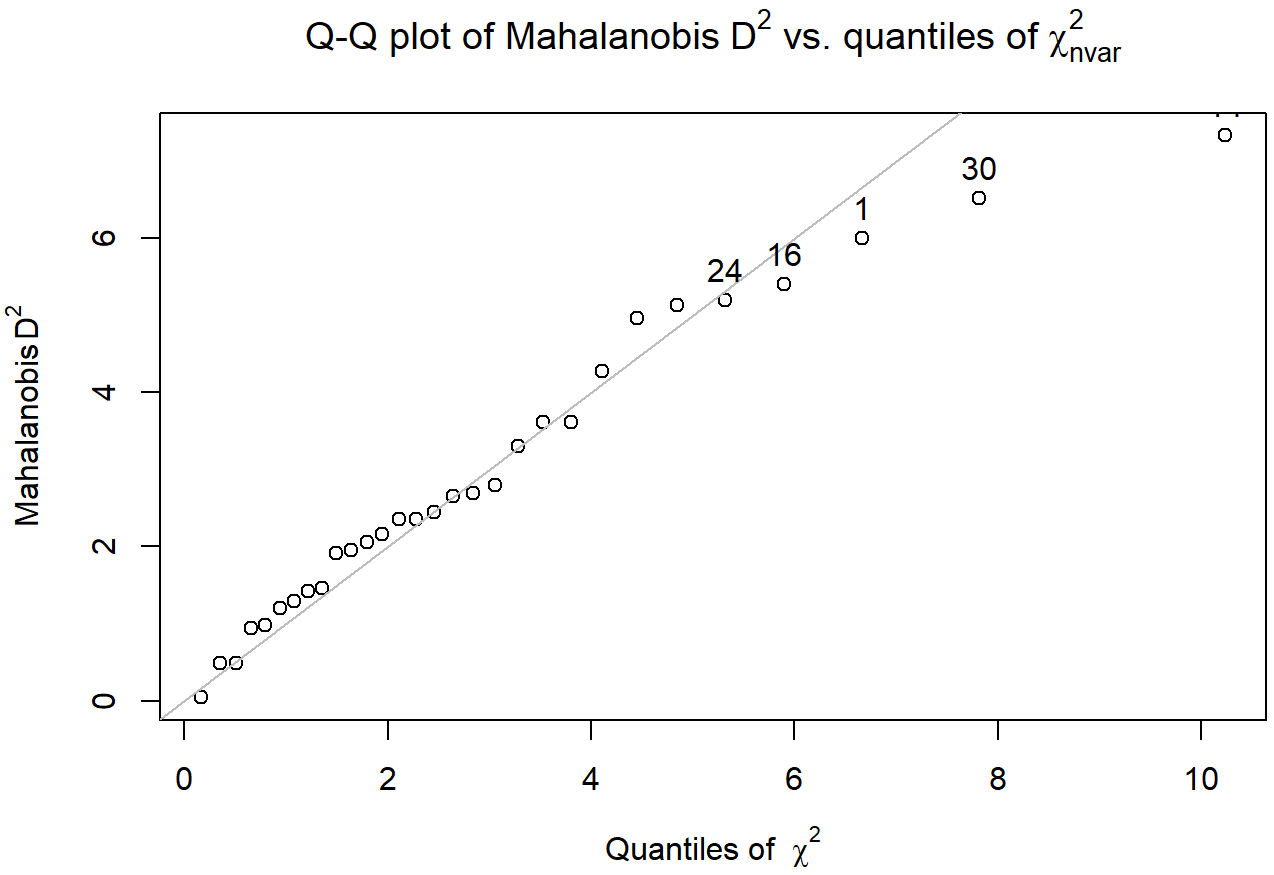
Two statistical assessments can be used to asses multivariate outliers

- Malhalanobis Distance
- Cook's Distance

Multivariate Outlier Assessment template

```
# Generate Mahalanobis Distance Values
data$mahal <- scale(outlier(data))

# Compare Standardized Mahalanobis Distance Values to 3.29
data[abs(data$mahal) > 3.29,]

# Cook's Distance
model_cook <- lm(dat$DV ~ dat$Predictor1 + dat$Predictor2 + ...
                 + dat$PredictorK)

data$cook <- scale(cooks.distance(model_cook))

# Compare Standardized Cooks Distance Values to 3.29
data[abs(data$cook) > 3.29,]
```

```
# Generate Mahalanobis Distance Values
attitude$mahalZ <- scale(outlier(attitude[1:3]))
```




```
# Generate Cook's Distance Values
CDlm <- lm(rating~complaints+privileges, data = attitude)
attitude$cookZ <- scale(cooks.distance(CDlm))

# Compare Standardized Mahalanobis Distance Values to 3.29
no_mv_outliers <- attitude[!abs(attitude$mahalZ) > 3.29,]
# Compare Standardized Cooks Distance Values to 3.29
no_mv_outliers <- attitude[!abs(attitude$cookZ) > 3.29,]
```

Step 6: Assess Multicollinearity and Singularity

Best way to assess Multicollinearity and Singularity:

- Bivariate Correlations
- VIF and Tolerance
- Determinant

VIF and Tolerance Template

```
MCvif <- linReg(data = dat,
                dep = <DV>,
                covs = c('IV1', 'IV2',..., 'IVK'),
                blocks = list(
                  list('IV1', 'IV2',..., 'IVK')),
                r = FALSE,
                r2 = FALSE,
                collin = TRUE)
MCvif
```

```
MCvif <- linReg(data = attitude,
                dep = 'rating',
                covs = c('complaints', 'privileges'),
                blocks = list(
                  list('complaints', 'privileges')),
                r = FALSE,
                r2 = FALSE,
                collin=TRUE)
MCvif
```

```
##
##  LINEAR REGRESSION
##
##  MODEL SPECIFIC RESULTS
##
##  MODEL 1
##
##  Model Coefficients - rating
##  ──────────────────────────────────────────────────────────────────────
##    Predictor     Estimate       SE           t             p
##  ──────────────────────────────────────────────────────────────────────
##    Intercept     15.32762017    7.1602343     2.1406590    0.0414872
##    complaints     0.78034340    0.1193876     6.5362166    0.0000005
##    privileges    -0.05015980    0.1299192    -0.3860845    0.7024597
##  ──────────────────────────────────────────────────────────────────────
##
##
##  ASSUMPTION CHECKS
##
##  Collinearity Statistics
##  ──────────────────────────────────────
##                  VIF         Tolerance
##  ──────────────────────────────────────
##    complaints    1.452825    0.6883143
##    privileges    1.452825    0.6883143
##  ──────────────────────────────────────
```

`Tolerance` values should be greater than .25 and definitely **NOT** less than .10 `VIF` values should be less than 4 and definitely **NOT** greater than 10

Determinant Template

```
det(cor(dat))
```

```
det(cor(attitude[1:3]))
```

```
## [1] 0.2181516
```

Transforming Data

- Square root transformation—Moderate Positive Skew

```
dat$V1T <- sqrt(dat$V1)
```

- Reflected square root transformation—Moderate Negative Skew

```
dat$V1T <- sqrt(max(dat$V1) + 1 - dat$V1)
```

- Log transformation—Substantial Positive Skew

```
dat$V1T <- log10(dat$V1)
```

- Reflected log transformation—Substantial Negative Skew

```
dat$V1T <- log10(max(dat$V1) + 1 - dat$V1)
```

- Inverse transformation—Severe Positive Skew

```
dat$V1T <- 1/(dat$V1)
```

- Reflected Inverse transformation—Severe Negative Skew

```
dat$V1T <- 1/(max(dat$V1) + 1 - dat$V1)
```