USING R FOR PROBABILITY AND STATISTICAL ANALYSIS – PART 2
WEEK 13 NOTES
STAT 330

Outline of Notes:

One Sample t Tests and Confidence Intervals

- The t-test is one method of testing statements about a population mean, μ. Similarly, t confidence intervals are one method for estimating a population mean.

- General syntax for the one-sample `t.test()` function in R (see `?t.test`):

  ```
  t.test(x, alternative="type", mu=#, conf.level=#)
  ```

  - `x` is a numeric vector of data values (i.e., the sample data)

  - `alternative` is the type of alternative hypothesis being tested – there are three possible options that can be used (must be enclosed in quotation marks!)
    - `"two.sided"` (default value) – testing $\mu \neq \mu_0$ (where μ_0 is the hypothesized value)
    - `"less"` – testing $\mu < \mu_0$
    - `"greater"` – testing $\mu > \mu_0$

  - `mu` is the hypothesized value of the mean (μ_0) that is being tested (default value is 0)

  - `conf.level` indicates the confidence level to be used for a confidence interval (default value is 0.95)

- Reminder from previous statistics class(es):

One Sample t Test Statistic	*One Sample t Confidence Interval for μ*
$t^* = \dfrac{\bar{x} - \mu_0}{s/\sqrt{n}}$	$\bar{x} \pm t_{n-1,\alpha/2}\left(\dfrac{s}{\sqrt{n}}\right)$

Warning! Warning! Warning!

In this class, we are only focused on using the software like a calculator. Proper analysis requires a human's touch! One should ideally check assumptions: random sample, normality, etc.

- Example: Suppose we are interested in doing a one-sample t-test using the petal widths in the R dataset called iris. Specifically we are interested in seeing if the true mean petal width is less than 1.2 cm. (For now we will ignore species type.)

  H_0: $\mu = 1.2$ H_a: $\mu < 1.2$

  - Observe the variable Petal.Width by using the following commands:

    ```
    > data(iris)
    ```

```
> iris$Petal.Width
> mean(iris$Petal.Width)          #sample mean
> sd(iris$Petal.Width)            #sample standard deviation
> length(iris$Petal.Width)        #sample size
> hist(iris$Petal.Width)          #Does assuming normality seem valid?
```

- Regardless of the normality assumption (since the sample size is large), conduct the hypothesis test mentioned above for the petal widths. Additionally, calculate a 90% confidence interval for the true mean petal width.

```
> t.test(iris$Petal.Width, alternative="less", mu=1.2, conf.level=.9)

        One Sample t-test

data:  iris$Petal.Width
t = -0.0107, df = 149, p-value = 0.4957
alternative hypothesis: true mean is less than 1.2
90 percent confidence interval:
     -Inf 1.279448
sample estimates:
mean of x
 1.199333
```

- Let's take a look at specific parts of the output:
    - `t = -0.0107` is the value of the test statistic
    - `p-value = 0.4957` is the p-value for the hypothesis test
    - `alternative hypothesis: true mean is less than 1.2` shows what is being tested

    -
      ```
      90 percent confidence interval:
           -Inf 1.279448
      ```
      Notice that what is calculated is known as a one-sided confidence interval. Anytime a one-sided test is done, a one-sided confidence interval is calculated. Thus, *to get the traditional two-sided confidence interval, you will need to rerun the `t-test()` function with the two-sided alternative*.

    -
      ```
      sample estimates:
      mean of x
       1.199333
      ```
      gives the value of the sample mean

- You can use R to verify the calculations of the test statistic (called `tstat`) and p-value (`pval`):

```
> tstat<-(mean(iris$Petal.Width)-1.2)/(sd(iris$Petal.Width)/
+ sqrt(length(iris$Petal.Width)))
> tstat
[1] -0.01071184
> pval<-pt(tstat,149)
> pval
[1] 0.4957338
```

- Use the `"two.sided"` alternative in R to obtain the traditional two-sided confidence interval (two equivalent commands are shown below). Since we are now only interested in the two-sided interval (and not in hypothesis testing), the `mu=` argument was not needed here.

```
> t.test(iris$Petal.Width, alternative="two.sided", conf.level=.9)
> t.test(iris$Petal.Width, conf.level=.9)      #two.sided is default
```

```
90 percent confidence interval:
 1.096323 1.302344
```

- It is possible to save the results of a t-test as an object in R. Then certain values can be extracted by using subsetting notation (`[]` or `$`).

```
> info<-t.test(iris$Petal.Width, alternative="two.sided")
```

```
> summary(info)
            Length Class  Mode
statistic   1      -none- numeric
parameter   1      -none- numeric
p.value     1      -none- numeric
conf.int    2      -none- numeric
estimate    1      -none- numeric
null.value  1      -none- numeric
alternative 1      -none- character
method      1      -none- character
data.name   1      -none- character
```

```
> info$statistic
      t
19.2706

> info$p.value
[1] 2.659021e-42

> info[1]
$statistic
      t
19.2706
```

- Store the values of `info$conf.int` in an R object. Then create a vector which contains only the lower value and upper value of the confidence interval.

```
> iris.int<-info$conf.int          #extracting the confidence interval
> iris.int                         #looking at what was just created
[1] 1.076353 1.322313
attr(,"conf.level")
[1] 0.95

> iris.int<-as.vector(iris.int)          #making iris.int a vector
> iris.int                               #looking at what was just created
[1] 1.076353 1.322313
```

Two Independent Samples t Tests and Confidence Intervals

- The t-test can be used to compare the means of two populations. Several versions of two-sample t-tests exist. In this section, we focus on two-sample t-tests where the samples are ***independent***.

- General syntax for the two-sample `t.test()` function in R (see `?t.test`):

  `t.test(x, y, alternative="type", mu=#, var.equal=FALSE, conf.level=#)`

  - `x` is a numeric vector of data values (i.e., the data from the first sample)

  - `y` is a numeric vector of data values (i.e., the data from the second sample)

  - `alternative` is the type of alternative being tested – there are three possible options that can be used (must be enclosed in quotation marks!)
    - `"two.sided"` (default value) – testing $\mu_1 - \mu_2 \neq D_0$ (where D_0 is the hypothesized value of the difference in means); note that when $D_0 = 0$, this is testing that $\mu_1 \neq \mu_2$.
    - `"less"` – testing $\mu_1 - \mu_2 < D_0$; note that when $D_0 = 0$, this is testing that $\mu_1 < \mu_2$.

        - `"greater"` – testing $\mu_1 - \mu_2 > D_0$; note that when $D_0 = 0$, this is testing that $\mu_1 > \mu_2$.

    - `mu` is the hypothesized value of difference in means (D_0) that is being tested (default value is 0, which is most often what's being tested in two-sample tests)

    - `conf.level` indicates the confidence level to be used for a confidence interval (default value is 0.95)

    - `var.equal` is a logical value indicating whether or not the population variances can be assumed equal (default value is `FALSE`)

- The type of t-test that is performed will depend on the value of the `var.equal` argument.
  - If **`var.equal=TRUE`**, then a *pooled* variance is used to estimate the common population variance.

Test Statistic

$$t^* = \frac{(\bar{x}_1 - \bar{x}_2) - D_0}{\sqrt{s_p^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

Confidence Interval for μ₁ – μ₂

$$(\bar{x}_1 - \bar{x}_2) \pm t_{n_1+n_2-2,\alpha/2}\sqrt{s_p^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}$$

Pooled Estimate of Common Variance

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

  - If **`var.equal=FALSE`**, then the Welch-Satterthwaite t-test is used (in which the degrees of freedom are estimated).

Test Statistic

$$t^* = \frac{(\bar{x}_1 - \bar{x}_2) - D_0}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Confidence Interval for μ₁ – μ₂

$$(\bar{x}_1 - \bar{x}_2) \pm t_{\nu,\alpha/2}\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

Estimated Degrees of Freedom

$$\nu = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{s_2^2}{n_2}\right)^2}{n_2 - 1}}$$

- Example: Suppose we want to compare the mean petal width of the versicolor irises (population 1) to the mean petal width of the virginica irises (population 2). We can consider the following hypotheses, testing whether or not there is a difference in mean petal width for the two species.

  H_0: $\mu_1 - \mu_2 = 0$ ($\mu_1 = \mu_2$) H_a: $\mu_1 - \mu_2 \neq 0$ ($\mu_1 \neq \mu_2$)

  - We are only comparing two of the three species of irises, so we will create objects with the appropriate subset of data in them. (Note that we do not *have* to create new objects. We could always pass the appropriate subset of the data to the R functions directly.)

```
> group1<-iris$Petal.Width[iris$Species=="versicolor"]        #versicolor
> group2<-iris$Petal.Width[iris$Species=="virginica"]         #virginica
```

  - Compare the variances of both of these datasets, using the `var()` function, to see whether or not the assumption of equal variances is reasonable. Since the variances are not very close, we will assume unequal variances. (Note: There is a statistical test that can be done to compare variances, but we will not cover that in this class. You are asked about this on the weekly assignment. Google can help you find the appropriate *function* for that test.)

```
> var(group1)
[1] 0.03910612
> var(group2)
[1] 0.07543265
```

  - Use the following code to compute the appropriate test statistic and p-value for the hypotheses listed above and to also compute a 95% confidence interval for $\mu_1 - \mu_2$.

```
> t.test(group1, group2, alternative="two.sided", mu=0, var.equal=F)

        Welch Two Sample t-test

data:  group1 and group2
t = -14.6254, df = 89.043, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.7951002 -0.6048998
sample estimates:
mean of x mean of y
    1.326     2.026
```

  - You can use R to verify the calculation of the test statistic (tstat) and p-value (pval):

```
> tstat<-(mean(group1)-mean(group2))/sqrt((var(group1)/length(group1))+
+ (var(group2)/length(group2)))
> tstat
[1] -14.62537
> pval<-2*pt(tstat,89.043) #stealing estimated df from t.test output
> pval
[1] 2.111743e-25
```

    Why doesn't the p-value match what was given in the result of `t.test()`??? Notice that in the output for `t.test()`, we are told "`p-value < 2.2e-16`". We are not given the actual p-value; we are just told a value that it is less than. The *minimum* p-value that will be given in built-in test functions is typically "`p-value < 2.2e-16`".

- Suppose that you switched group1 and group2 in the R code.

```
> t.test(group2, group1, alternative="two.sided", mu=0, var.equal=F)

        Welch Two Sample t-test

data:  group2 and group1
t = 14.6254, df = 89.043, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.6048998 0.7951002
sample estimates:
mean of x mean of y
    2.026     1.326
```

a. How was the test statistic affected?

b. How was the p-value affected?

c. How was the confidence interval affected?

- Example: Suppose we instead chose to assume that *the variances are equal*.

```
> t.test(group1, group2, alternative="two.sided", mu=0, var.equal=T)

        Two Sample t-test

data:  group1 and group2
t = -14.6254, df = 98, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.7949807 -0.6050193
sample estimates:
mean of x mean of y
    1.326     2.026
```

- Alternative syntax also exists for two-sample t-tests in R:

```
t.test(analysis.variable ~ grouping.variable)
#with any desired additional arguments
```

- Example: Consider the sleep dataset (`?sleep`) in R, which contains information on the increase in sleep time based on two different drugs.

```
> data(sleep)
> attach(sleep)    #now we can "call" variables in this object by name alone
```

  - Draw side-by-side boxplots using the following command.

```
> boxplot(extra ~ group, data=sleep)
```

  - Conduct an appropriate two-sided hypothesis test (using the equal-variance assumption) using both versions of commands (output is only shown for one of them below).

```
> t.test(extra[group==1], extra[group==2], var.equal=T)
> t.test(extra ~ group, var.equal=T)
        Two Sample t-test

data:  extra by group
t = -1.8608, df = 18, p-value = 0.07919
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.363874  0.203874
sample estimates:
mean in group 1 mean in group 2 
           0.75            2.33 
```

Paired t Tests and Confidence Intervals

- Careful reading of the `?sleep` help file reveals that the data is actually **paired**! The same ten patients appear twice, once in each group.

- For paired t-tests and confidence intervals, the observations are paired from each sample, and differences are calculated within those pairs. Then, the mean and standard deviation *of those differences* are used in the (one-sample) inferential procedures.

- General syntax for a paired t-test (two different ways to give the data to the function):

```
t.test(analysis.var ~ grouping.var, mu=#, alternative="type", paired=TRUE,
   conf.level=#)
t.test(x, y, mu=#, alternative="type", paired=TRUE, conf.level=#)
```

Test Statistic

$$t^* = \frac{\bar{d} - D_0}{s_d/\sqrt{n}}$$

Confidence Interval for $\mu_1 - \mu_2$

$$\bar{d} \pm t_{n-1,\alpha/2}\left(s_d/\sqrt{n}\right)$$

where

$\bar{d}$ is the mean of the paired differences
s_d is the standard deviation of the paired differences
n is the number of paired observations

- Example: Conduct a paired t-test on the sleep dataset (two equivalent ways to do this appear below, but the output is only shown once).

```
> t.test(extra[group==1], extra[group==2], paired=T)
> t.test(extra ~ group, paired=T)

        Paired t-test

data:  extra by group
t = -4.0621, df = 9, p-value = 0.002833
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.4598858 -0.7001142
sample estimates:
mean of the differences
                  -1.58
```

  - Notice that the p-value is actually smaller with the paired t-test. This is because when the data is paired, the paired t-test is more powerful at detecting differences than the independent samples t-test.

- Recall (from a previous statistics class) that a paired t-test is equivalent to a one-sample t-test on a new variable that is the *difference between each paired observation*. We can also use this technique in R.

```
> drugA<-extra[group==1]
> drugB<-extra[group==2]
> diffs<-drugA-drugB    #creating a vector of differences
> t.test(diffs)         #performing a one-sample t-test on the differences

        One Sample t-test

data:  diffs
t = -4.0621, df = 9, p-value = 0.002833
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -2.4598858 -0.7001142
sample estimates:
mean of x
    -1.58
```

Correlation and Regression

- Correlation is calculated with the `cor()` function.
  - The Pearson correlation coefficient (a measure of the strength of the *linear* relationship between two quantitative variables) is the default.
  - You can use options to obtain the Spearman's rank correlation, a nonparametric correlation (if you don't know what this is, that's OK!), among others.

- General syntax for `cor()`:

  `cor(x, y=NULL)`

  - `x` can be a vector, matrix, or data frame. *If `x` is a vector, then a second vector (`y`) must also be used.*
  - `y` is a vector (if needed)

- Example: Calculate the correlation between the petal lengths and widths of irises.

```
> cor(iris$Petal.Width, iris$Petal.Length)
[1] 0.9628654
```

- Example: Observe the dataset mtcars (see `?mtcars`), which is a data frame.

```
> data(mtcars)
> mtcars
```

- Notice that the data frame contains data for 11 numerical variables (the car names are the names of the rows, *not* an actual variable).

- To calculate the correlation between every pair of quantitative variables at the same time, the `cor()` function can be applied to the data frame. The result will be a matrix of correlations.

```
> cor(mtcars)
            mpg        cyl       disp         hp        drat
mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191
cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811
disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393
hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912
drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000
wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065
qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476
vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846
am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113
gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013
carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980
             wt        qsec         vs          am       gear
mpg  -0.8676594  0.41868403  0.6640389  0.59983243  0.4802848
cyl   0.7824958 -0.59124207 -0.8108118 -0.52260705 -0.4926866
disp  0.8879799 -0.43369788 -0.7104159 -0.59122704 -0.5555692
hp    0.6587479 -0.70822339 -0.7230967 -0.24320426 -0.1257043
drat -0.7124406  0.09120476  0.4402785  0.71271113  0.6996101
wt    1.0000000 -0.17471588 -0.5549157 -0.69249526 -0.5832870
qsec -0.1747159  1.00000000  0.7445354 -0.22986086 -0.2126822
vs   -0.5549157  0.74453544  1.0000000  0.16834512  0.2060233
am   -0.6924953 -0.22986086  0.1683451  1.00000000  0.7940588
gear -0.5832870 -0.21268223  0.2060233  0.79405876  1.0000000
carb  0.4276059 -0.65624923 -0.5696071  0.05753435  0.2740728
            carb
mpg  -0.55092507
cyl   0.52698829
disp  0.39497686
hp    0.74981247
drat -0.09078980
wt    0.42760594
qsec -0.65624923
vs   -0.56960714
am    0.05753435
gear  0.27407284
carb  1.00000000
```

- Notice that all of the diagonal values are 1. That is because the diagonal contains the correlation between a variable and itself (which is always 1). The off-diagonals contain the correlation between the corresponding variables. For example, the correlation between weight and mpg is -0.8676594. To verify this, we can calculate the correlation between just those two variables.

```
> cor(mtcars$mpg, mtcars$wt)
[1] -0.8676594
```

- Create a scatterplot of the mpg and weight data using the following command. Is there an approximate linear relationship?

```
> plot(mtcars$mpg, mtcars$wt)
```

- To fit a linear least-squares regression model, we use the `lm()` function (for other linear models, see more details with `?lm`) .
  - General syntax for simple linear regression with `lm()`:

```
lm(response.variable ~ explanatory.variable)
```

- Example: Find the least-squares regression line using the mtcars dataset, where mpg is the response (y) variable and weight is the explanatory/predictor (x) variable.

```
> lm(mtcars$mpg ~ mtcars$wt)

Call:
lm(formula = mtcars$mpg ~ mtcars$wt)

Coefficients:
(Intercept)    mtcars$wt
     37.285       -5.344
```

  - The linear least-squares regression model is:

$$\hat{y} = 37.285 - 5.344x$$

where $\hat{y}$ is the predicted/estimated value for mpg, and x is the weight.

  - The `lm()` function actually calculates much more than just the parameter estimates. To see additional, detailed information, the output needs to be saved as an object in R. Then, the `summary()` function can be applied to that object. Alternatively, the call to the `lm()` function can be nested in the `summary()` function.

```
> out<-lm(mtcars$mpg ~ mtcars$wt)       #saving the results to an object
> summary(out)                          #taking a closer look

Call:
lm(formula = mtcars$mpg ~ mtcars$wt)

Residuals:
    Min      1Q  Median      3Q     Max
-4.5432 -2.3647 -0.1252  1.4096  6.8727
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
mtcars$wt    -5.3445     0.5591  -9.559 1.29e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared: 0.7528,     Adjusted R-squared: 0.7446 
F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

- In the "coefficients" table, information for a t-test for each parameter (testing its significance) is included. Since both p-values are quite small, both the intercept and the weight should be included in the model (for this example).

- The `confint()` function can be used to calculate confidence intervals for the regression parameters. The general syntax is: `confint(output.object, level=0.95)` (default confidence level is 0.95).

```
> confint(out)
                2.5 %    97.5 %
(Intercept) 33.450500 41.119753
mtcars$wt   -6.486308 -4.202635
```

95% confidence interval for the y-intercept

95% confidence interval for the slope

- To fit a **multiple** linear regression model, we "add" additional predictor variables to our model statement. Use the following line of code to find a model that estimates miles per gallon (mpg) based on both weight (wt) and horsepower (hp).

```
> lm(mtcars$mpg ~ mtcars$wt + mtcars$hp)

Call:
lm(formula = mtcars$mpg ~ mtcars$wt + mtcars$hp)

Coefficients:
(Intercept)    mtcars$wt    mtcars$hp  
   37.22727     -3.87783     -0.03177
```

The multiple linear least-squares regression model is:

$$\hat{y} = 37.22727 - 3.87783x_1 - 0.03177x_2$$

where $\hat{y}$ is the predicted/estimated value for mpg, x_1 is the weight, and x_2 is the horsepower.

```
> out2<-lm(mtcars$mpg ~ mtcars$wt + mtcars$hp)
> summary(out2)

Call:
lm(formula = mtcars$mpg ~ mtcars$wt + mtcars$hp)

Residuals:
   Min     1Q Median     3Q    Max 
-3.941 -1.600 -0.182  1.050  5.854
```

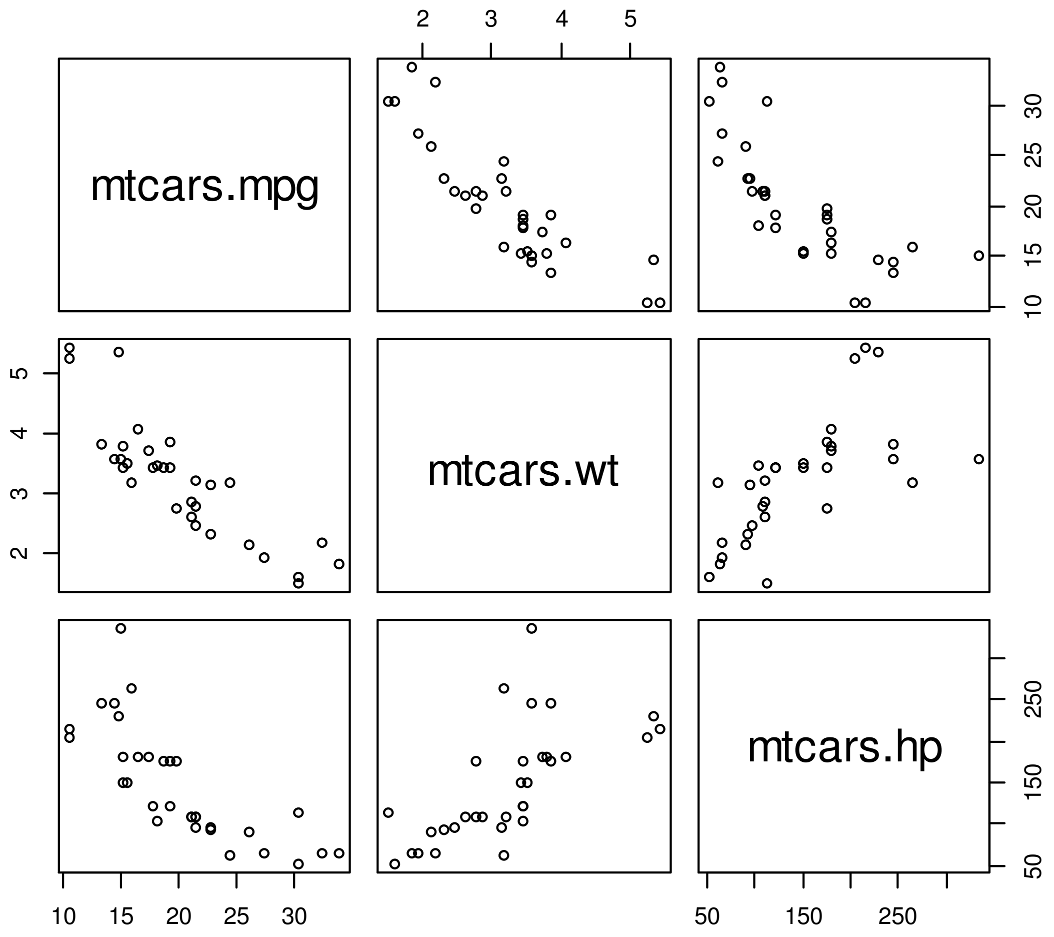
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
mtcars$wt   -3.87783    0.63273  -6.129 1.12e-06 ***
mtcars$hp   -0.03177    0.00903  -3.519  0.00145 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared: 0.8268,     Adjusted R-squared: 0.8148 
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

- Notice that all parameters included in this model are statistically significant.

- To visually see the relationships among pairs of the three variables, a scatterplot matrix can be created using the `pairs()` function.

```
> pairs(data.frame(mtcars$mpg,mtcars$wt,mtcars$hp))
```



Categorical Tests

- Suppose that you have a contingency table containing the number of observations falling into each combination of the categories for two different variables. To test whether or not there is a relationship between the two variables, the `chisq.test()` function can be used to carry out a chi-square test for independence.

- **Chi-Square Test for Independence**.
  - Hypotheses

    H_0: the two variables are independent
    H_a: the two variables are dependent

- General syntax for a chi-square test for independence:

  ```
  chisq.test(table)
  ```

- Example: Consider the VADeaths dataset. Test whether or not there is a relationship between age group (for which the levels are 50-54, 55-59, 60-64, 65-69, and 70-74) and population group (for which the levels are Rural Male, Rural Female, Urban Male, and Urban Female).

  H_0: age group and population group are independent for death rates in Virginia in 1940 (i.e., there is no relationship)
  H_a: age group and population group are dependent for death rates in Virginia in 1940 (i.e., there is a relationship)

  ```
  > data(VADeaths)
  > chisq.test(VADeaths)

          Pearson's Chi-squared test

  data:  VADeaths
  X-squared = 2.9208, df = 12, p-value = 0.9961
  ```

  - Since the p-value is so large, we *cannot* conclude that there is a relationship between age group and population group for death rates in Virginia in 1940.

- Suppose that you have a single categorical variable with *k* categories, and you would like to perform a test for the proportion in each category (simultaneously). To do this, chi-square goodness-of-fit test is used.

- **Chi-Square Goodness-of-Fit Test**
  - Hypotheses:

    H_0: $p_1 = p_{1,0}$, $p_2 = p_{2,0}$, $p_3 = p_{3,0}$, …, $p_k = p_{k,0}$ (where $p_{i,0}$ is the hypothesized proportion for category i)
    H_a: at least one inequality

- General syntax for a chi-square goodness-of-fit test:

  ```
  chisq.test(x, p=hypothesized.values)
  ```

- `x` is a vector or table containing sample data (the counts within each category)
- `p` is a vector of the same length as `x`, containing the hypothesized proportions (*which must sum to 1*)

- Example: Suppose a die is rolled 100 times, and 14 ones, 12 twos, 17 threes, 14 fours, 22 fives, and 21 sixes are observed. Test whether or not the die is a fair die (i.e., whether or not all six sides are equally likely).
  - H_0: $p_1 = p_2 = p_3 = p_4 = p_5 = p_6 = ⅙$ (the die is fair)
    H_a: at least one inequality (the die is not fair)

```
> rolls<-c(14,12,17,14,22,21)
> chisq.test(rolls, p=rep(1/6,6))       #p is a vector of 1/6, 6 times

        Chi-squared test for given probabilities

data:  rolls
X-squared = 5, df = 5, p-value = 0.4159
```

    - Since the p-value is large, we cannot conclude that the die is not a fair die.

  - Suppose instead that 8 ones, 27 twos, 9 threes, 11 fours, 17 fives, and 28 sixes were observed.

```
> rolls2<-c(8,27,9,11,17,28)
> chisq.test(rolls2, p=rep(1/6,6))

        Chi-squared test for given probabilities

data:  rolls2
X-squared = 24.08, df = 5, p-value = 0.0002096
```

    - In this case, since the p-value is small, we *would* conclude that the die is *not* a fair die.

- Note that all of the hypothesized proportions do **not** need to be the same in the Chi-Square Goodness-of-Fit test. The only restriction is that they sum to 1 (and that they are all non-negative, of course).

- Example: If you had suspicions that the die above was weighted in such a way that the two and six are twice as likely to occur as any of the other results, you could test the following hypotheses:
  H_0: $p_2 = p_6 = ¼$, $p_1 = p_3 = p_4 = p_5 = ⅛$ (the die is weighted in such a way)
  H_a: at least one inequality (the die is not weighted in such a way)

```
> chisq.test(rolls2, p=c(1/8,1/4,1/8,1/8,1/8,1/4))

        Chi-squared test for given probabilities

data:  rolls2
X-squared = 4.92, df = 5, p-value = 0.4257
```

    - Since the p-value is so large, we cannot can conclude that the die is not weighted in such a way that the two and six are twice as likely to occur as any of the other results.