MTH 3240 R Notes 10

1 One-Factor ANOVA

1.1 Data Considerations

- Suppose we have a data frame called `my.data` containing the following data:

```
group <- c("control", "control", "control", "trt1", "trt1", "trt1", "trt2",
           "trt2", "trt2", "trt3", "trt3", "trt3")

my.response.var <- c(22, 21, 19, 23, 23, 25, 25, 24, 25, 30, 27, 28)

my.data <- data.frame(Response = my.response.var, Treatment = group)
```

```
my.data

##    Response Treatment
## 1        22   control
## 2        21   control
## 3        19   control
## 4        23      trt1
## 5        23      trt1
## 6        25      trt1
## 7        25      trt2
## 8        24      trt2
## 9        25      trt2
## 10       30      trt3
## 11       27      trt3
## 12       28      trt3
```

  and we want to carry out a one-factor ANOVA. It's important that the response for different groups are ”stacked” in a single column (as above) and that the groups are recorded as text (as opposed to numerical values) so that R knows they're levels of a factor.

1.2 Fitting the ANOVA Model and Obtaining the ANOVA Table

- The following functions are used to carry out and summarize a one-factor ANOVA:

```
aov()       # Fit an ANOVA model, compute sums of squares, degrees of
            # freedom, mean squares, F statistic, p-value, fitted values
            # and residuals
summary()   # Print the ANOVA table
```

- The `aov()` function takes as arguments:

`formula`	a formula specifying the model, such as Y ˜ X, where Y is a numeric response variable and X is the factor.
`data`	a data frame from which the variables in the formula will be found.

  It returns the results bundled together in a so-called ***aov*** object.

- The `summary()` function takes an *aov* object and returns the ANOVA table.
- Below we fit an ANOVA model to the data in `my.data`, with `Treatment` as the factor and `Response` as the response:

```
my.anova <- aov(Response ~ Treatment, data = my.data)
```

  The ***formula*** is `Response ~ Treatment`, and indicates that `Response` is the response variable and `Treatment` is the factor (explanatory variable). (For more information on how to specify models in R, type `?formula`.)

  To look at the ANOVA table, we type:

```
summary(my.anova)

##             Df Sum Sq Mean Sq F value   Pr(>F)
## Treatment    3  90.00  30.000   18.95 0.000541
## Residuals    8  12.67   1.583
```

  From the output, we conclude that:

  – The treatment sum of squares is SSTr = 90.00 with 3 degrees of freedom.
  – The error sum of squares is SSE = 12.67 with 8 degrees of freedom.
  – The mean square for treatments is MSTr = 30.000.
  – The mean squared error is MSE=1.583.
  – The ANOVA F statistic is 18.95 and the p-value is 0.000541 (from the F distribution with numerator degrees of freedom 3 and denominator degrees of freedom 8).

1.3 Obtaining the Residuals and Fitted Values

- The object `my.anova` created above is a bundle of other objects. We can view their names using `names()`:

```
names(my.anova)

##  [1] "coefficients"  "residuals"     "effects"
##  [4] "rank"          "fitted.values" "assign"
##  [7] "qr"            "df.residual"   "contrasts"
## [10] "xlevels"       "call"          "terms"
## [13] "model"
```

- To access these, we use the dollar sign `$` with `my.anova`. For example, to access the residuals, type:

```
my.anova$residuals

##          1          2          3          4          5
##  1.3333333  0.3333333 -1.6666667 -0.6666667 -0.6666667
##          6          7          8          9         10
##  1.3333333  0.3333333 -0.6666667  0.3333333  1.6666667
##         11         12
## -1.3333333 -0.3333333
```

and to access the fitted values (group means), type:

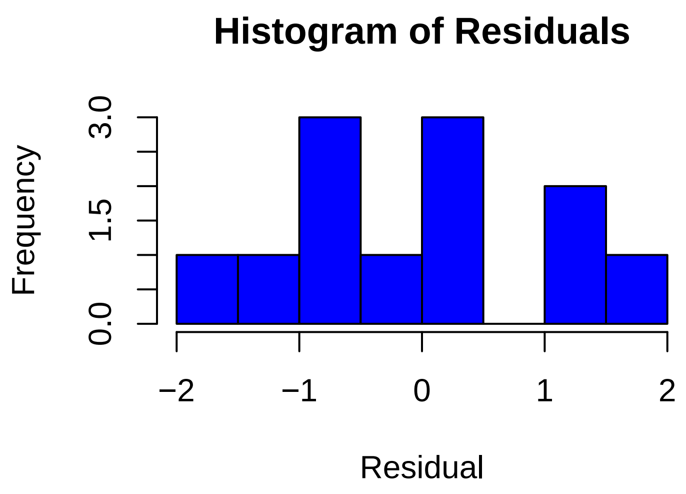
```
my.anova$fitted.values

##        1        2        3        4        5        6
## 20.66667 20.66667 20.66667 23.66667 23.66667 23.66667
##        7        8        9       10       11       12
## 24.66667 24.66667 24.66667 28.33333 28.33333 28.33333
```

1.4 Checking Assumptions by Plotting the Residuals

1.4.1 Checking the Normality Assumption

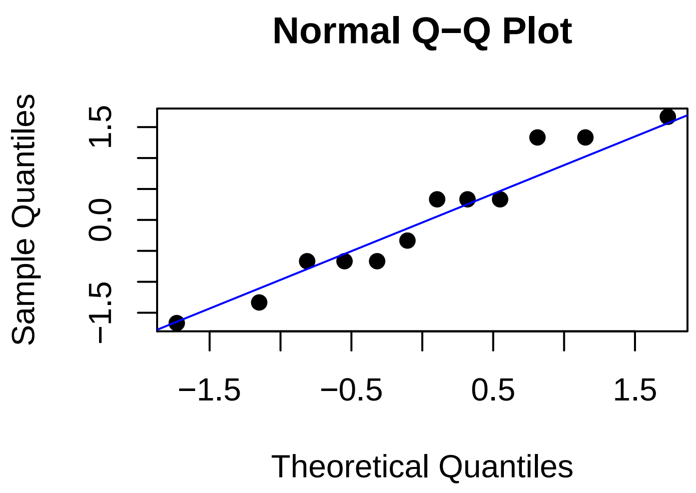
- To check the assumption that the samples were drawn from N(μ_i, σ) populations, or equivalently that the errors ϵ_{ij} in the ANOVA model follow a N(0, σ) distribution, we make a histogram or normal probability plot of the residuals.
- For example, here's the histogram:

```
hist(my.anova$residuals,
     xlab = "Residual",
     main = "Histogram of Residuals",
     col = "blue")
```



and here's the normal probability plot:

```r
qqnorm(my.anova$residuals, pch = 19)
qqline(my.anova$residuals, col = "blue")
```

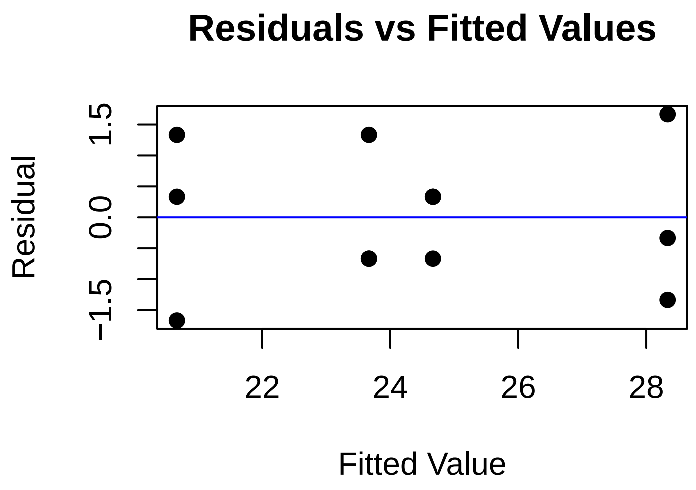


Both plots appear to support the normality assumption, so the results of the ANOVA F test are valid.

1.4.2 Checking the Equal Standard Deviation Assumption

- To check the assumption that the k populations (or treatment groups) all have the same standard deviation σ, we plot the residuals versus the fitted values (and add a horizontal line at $y = 0$ using `abline()`):

```r
plot(x = my.anova$fitted.values, y = my.anova$res, pch = 19,
  xlab = "Fitted Value", ylab = "Residual",
  main = "Residuals vs Fitted Values")

abline(h = 0, col = "blue")
```



The vertical spread of the points above and below the horizontal line reflects the size of the standard deviation σ within each group, and if the equal σ assumption is met, should be fairly constant from left to right in the plot.

Here, the small sample sizes make it difficult to assess, but there are no strong indications that the equal σ assumption isn't met, so the results of the ANOVA F test appear to be valid.

2 Summarizing and Graphing the Results of a One-Factor Study

2.1 Summarizing the Responses Separately for Each Treatment Group

- The easiest way to compute summary statistics separately for each treatment group is using `aggregate()`.

  For example, below we compute the mean of `Response` separately for each `Treatment` (using `my.data` created above):

```r
aggregate(Response ~ Treatment, data = my.data, FUN = mean)

##   Treatment Response
## 1   control 20.66667
## 2      trt1 23.66667
## 3      trt2 24.66667
## 4      trt3 28.33333
```

and here we compute the standard deviations:

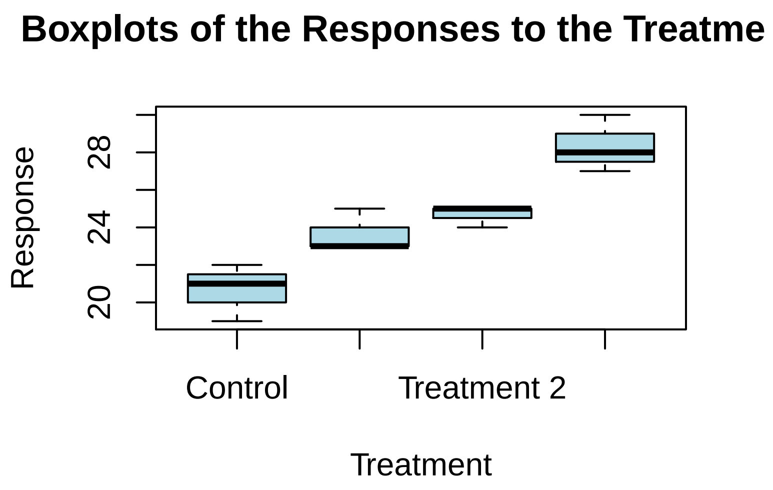
```
aggregate(Response ~ Treatment, data = my.data, FUN = sd)

##   Treatment  Response
## 1   control 1.5275252
## 2      trt1 1.1547005
## 3      trt2 0.5773503
## 4      trt3 1.5275252
```

2.2 Graphing the Responses to the Treatments

2.2.1 Boxplots

- The `boxplot()` function will accept a *formula* (as its first argument) and a data frame (via `data`), which is useful for making side-by-side boxplots of the response variable for several treatment groups.
- Here's an example (using `my.data` created above):

```
boxplot(Response ~ Treatment,
        data = my.data,
        col = "lightblue",
        names = c("Control", "Treatment 1", "Treatment 2", "Treatment 3"),
        main = "Boxplots of the Responses to the Treatments")
```



- Some comments:
  - Specifying `names`, as above, is used to specify labels for the horizontal axis.
  - Make sure that the names passed via `names` are in the same order that the groups are plotted in the graph. To check the order in which they're plotted, use `levels()`. For example, typing:

    ```
    levels(my.data$Treatment)

    ## [1] "control" "trt1"    "trt2"    "trt3"
    ```

    tells us that the groups will be plotted from left to right in the order `"control"`, `"trt1"`, `"trt2"`, `"trt3"`.

Section 2.2 Exercises

Exercise 1 Here's a data frame containing data from a **one-factor study** involving responses to three treatments in an experiment, where the true (unknown) mean responses are μ_1 μ_2, and μ_3.

```
Treatment <- c("trt1", "trt3", "trt2", "trt1", "trt3", "trt1", "trt1",
               "trt2", "trt3", "trt3", "trt1", "trt2", "trt2", "trt3",
               "trt2")
Y <- c(21, 31, 25, 24, 26, 24, 25, 25, 29, 27, 26, 23, 27, 25, 24)
my.data <- data.frame(Y, Treatment)
```

a) Make side-by-side boxplots of the three treatment groups:

```
boxplot(Y ~ Treatment, data = my.data,
        col = "lightblue",
        names = c("Trt1", "Trt2", "Trt3"))
```

Don't print the boxplots, just answer the following question. Based on the boxplots, which treatment group (**Trt1**, **Trt2**, or **Trt3**) has the highest values of Y?

b) **Report** the values of the three **group means**:

```
aggregate(Y ~ Treatment, data = my.data, FUN = mean)
```

c) Use `aov()` to carry out a **one-factor ANOVA**, then use `summary()` to look at the ANOVA table:

```
my.anova <- aov(Y ~ Treatment, data = my.data)
summary(my.anova)
```

For the ***F* test** of the hypotheses

$$\begin{aligned} H_0 &: \quad \mu_1 = \mu_2 = \mu_3 \\ H_a &: \quad \text{The } \mu_i\text{'s aren't all equal} \end{aligned}$$

report the following values from the ANOVA table and **answer** the **question**.

Test statistic: F = ____________

P-value: p = ____________

Based on the ANOVA F test, is there statistically significant evidence for any differences among the true means μ_1, μ_2, and μ_3 (Yes/No)?

3 (Optional for Spring 2020) The Kruskal-Wallis Test

- We can carry out a Kruskal-Wallis test using the function:

```
kruskal.test()              # Carry out a Kruskal-Wallis test
```

- The `kruskal.test()` function takes arguments `formula` (a *formula*) and `data` (a data frame containing the variables in the *formula*).
- For example, suppose we have the following data frame:

```
my.data

##    Response Factor
## 1        48   Trt1
## 2        51   Trt1
## 3        54   Trt1
## 4        48   Trt1
## 5        52   Trt1
## 6        44   Trt1
## 7        57   Trt2
## 8        53   Trt2
## 9        59   Trt2
## 10       53   Trt2
## 11       55   Trt2
## 12       56   Trt2
## 13       58   Trt3
## 14       59   Trt3
## 15       56   Trt3
## 16       54   Trt3
## 17       51   Trt3
## 18       56   Trt3
```

We carry out the Kruskal-Wallis test using the command:

```
kruskal.test(Response ~ Factor, data = my.data)

##
##  Kruskal-Wallis rank sum test
##
## data:  Response by Factor
## Kruskal-Wallis chi-squared = 8.5207, df = 2, p-value
## = 0.01412
```

From the output, we find that:

- The observed value of the test statistic is $K_w = 8.5207$.
- The p-value is 0.0142 (from the chi-squared distribution with 2 degrees of freedom).

Section 3.0 Exercises

Exercise 2 (**Optional for Spring 2020**) Here's a data frame containing data from a **one-factor study** involving samples from three populations whose true (unknown) means are μ_1 μ_2, and μ_3.

```
Group <- c("grp1", "grp1", "grp1", "grp1", "grp1", "grp2", "grp2", "grp2",
           "grp3", "grp2", "grp2", "grp3", "grp3", "grp3", "grp3")
Y <- c(4, 7, 3, 4, 4, 6, 5, 4, 9, 7, 9, 6, 8, 5, 4)
my.data <- data.frame(Y, Group)
```

Carry out a **Kruskal-Wallis test** of the hypotheses

$$\begin{aligned} H_0 &: \quad \mu_1 = \mu_2 = \mu_3 \\ H_a &: \quad \text{Not all } \mu_i\text{'s are equal} \end{aligned}$$

by typing:

```
kruskal.test(Y ~ Group, data = my.data)
```

Report the following values and **answer the question**.

Test statistic: K_w = ____________

P-value: p = ____________

Based on the Kruskal-Wallis test, is there statistically significant evidence for any differences among the true means μ_1, μ_2, and μ_3 (Yes/No)?

4 (Optional for Spring 2020) The Bonferroni Multiple Comparisons Test Procedure

- After the null hypothesis is rejected in a one-factor ANOVA F test, we're sometimes interested in deciding *which* group means differ from each other. We do this by performing a **multiple comparisons** test procedure. A function that will carry out such tests is given below:

```
pairwise.t.test()       # Carry out multiple pairwise t tests between
                        # group means
```

- The main arguments to be passed to `pairwise.t.test()` are `x`, a *vector* containing values of the response variable, `g`, a *character vector* or `factor` indicating the treatment groups (levels of the factor), and `p.adjust.method`, the method (if any) for adjusting the p-value of each pairwise t test to control the familywise Type I error rate.

- Two choices for `p.adjust.method` are:

  - `"none"`
  - `"bonferroni"`

  In the first case (`"none"`), no adjustment is made to the p-values of the pairwise t tests, and so to control the **familywise Type I** error rate at 0.05, each p-value should be compared to the ***Bonferroni-corrected level of significance***

$$\alpha_p = \frac{0.05}{k(k-1)/2},$$

  where k is the number of groups.

  In the second case (`"bonferroni"`), the p-values reported for each pairwise t test are ***Bonferroni-corrected p-values*** given by

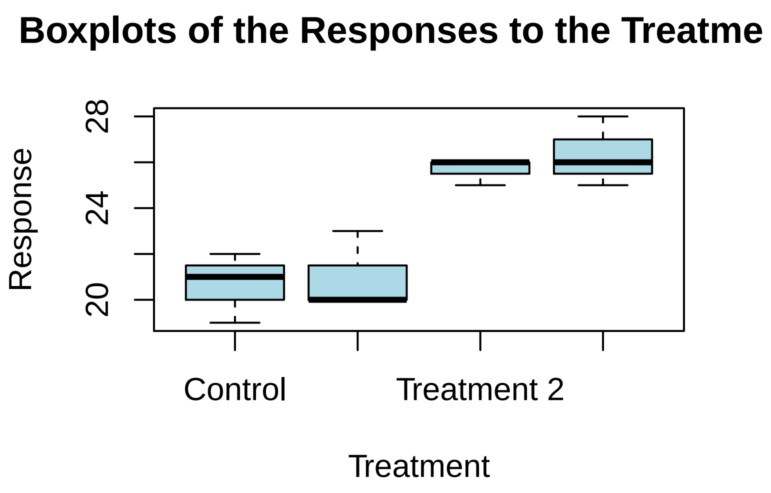
$$\frac{k(k-1)}{2}p,$$

  where p is the unadjusted p-value (as would be reported if `"none"` was specified). In this case, each of the Bonferroni-corrected p-values should be compared to 0.05.

- As an example, consider the following data from a one-factor study.

```
group <- c("control", "control", "control", "trt1", "trt1", "trt1", "trt2",
"trt2", "trt2", "trt3", "trt3", "trt3")

my.response.var <- c(22, 21, 19, 20, 23, 20, 26, 25, 26, 28, 25, 26)

my.data <- data.frame(Response = my.response.var, Treatment = group)
my.data

##    Response Treatment
## 1        22   control
## 2        21   control
## 3        19   control
## 4        20      trt1
## 5        23      trt1
## 6        20      trt1
## 7        26      trt2
## 8        25      trt2
## 9        26      trt2
## 10       28      trt3
## 11       25      trt3
## 12       26      trt3
```

Side-by-side boxplots are shown below.



A one-factor ANOVA F test determined that there are statistically significant differences among the 4 group means. To decide *which* means differ from each other, we type:

```
pairwise.t.test(my.data$Response, my.data$Treatment, p.adjust.method = "none")

##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  my.data$Response and my.data$Treatment
##
##      control trt1   trt2
## trt1 0.7802  -      -
## trt2 0.0025  0.0037 -
## trt3 0.0012  0.0017 0.5796
##
## P value adjustment method: none
```

Comparing each p-value from the output above to the Bonferroni-corrected level of significance

$$\alpha_p = \frac{0.05}{4(4-1)/2} = 0.0083,$$

we conclude that:

- The `trt1` and `control` means **don't differ** (p-value = 0.7802).
- The `trt2` and `control` means **differ** (p-value = 0.0025).
- The `trt2` and `trt1` means **differ** (p-value = 0.0037).
- The `trt3` and `control` means **differ** (p-value = 0.0012).
- The `trt3` and `trt1` means **differ** (p-value = 0.0017).
- The `trt3` and `trt2` means **don't differ** (p-value = 0.5796).

(Alternatively, we could've specified `p.adjust.method = "bonferroni"` in the call to `pairwise.t.test()`, in which case we would've compared each p-value to 0.05).

5 Two-Factor ANOVA

5.1 Data Considerations

- To carry out a ***two-factor ANOVA***, we first create a data frame called `my.data`, shown below.

```
my.response.var <- c(48, 51, 54, 48, 52, 44, 57, 53, 59, 53, 55, 56, 58, 59,
                     56, 54, 51, 56)

my.facA <- c("A1", "A1", "A1", "A1", "A1", "A1", "A2", "A2", "A2", "A2",
             "A2", "A2", "A3", "A3", "A3", "A3", "A3", "A3")

my.facB <- c("B1", "B1", "B1", "B2", "B2", "B2", "B1", "B1", "B1", "B2",
             "B2", "B2", "B1", "B1", "B1", "B2", "B2", "B2")

my.data <- data.frame(Response = my.response.var,
                      FactorA = my.facA,
                      FactorB = my.facB)
```

```
my.data

##    Response FactorA FactorB
## 1        48      A1      B1
## 2         51       A1       B1
## 3         54       A1       B1
## 4         48       A1       B2
## 5         52       A1       B2
## 6         44       A1       B2
## 7         57       A2       B1
## 8         53       A2       B1
## 9         59       A2       B1
## 10        53       A2       B2
## 11        55       A2       B2
## 12        56       A2       B2
## 13        58       A3       B1
## 14        59       A3       B1
## 15        56       A3       B1
## 16        54       A3       B2
## 17        51       A3       B2
## 18        56       A3       B2
```

- The two factors need to be stored as two separate columns containing `"character"` values (i.e. *text* as opposed to numbers) so that R knows they're *factors* rather than numerical explanatory variables.
- A two-factor ANOVA is carried out using `aov()`. It's carried out differently depending on on whether or not an ***interaction effect*** is included in the model.

5.2 Fitting the *Additive* Two-Factor ANOVA Model and Carrying Out the F Tests

- **If** we believe the effects of the two factors are **additive** (i.e. if we **don't** think there's an *interaction effect*), we fit the ANOVA model to the data and carry out the F tests using the following command:

```
my.anova <- aov(Response ~ FactorA + FactorB, data = my.data)
```

The ANOVA table is obtained as before, using `summary()`:

```
summary(my.anova)

##             Df Sum Sq Mean Sq F value  Pr(>F)
## FactorA      2 148.11   74.06  10.939 0.00138
## FactorB      1  37.56   37.56   5.547 0.03362
## Residuals   14  94.78    6.77
```

We see from the output that:

- The Factor A sum of squares is SSA = 148.11 with 2 degrees of freedom.
- The mean square for Factor A is MSA = 74.06.

- The F test statistic for a Factor A effect is $F_A = 10.939$, and the associated p-value is 0.00138.
- The Factor B sum of squares is SSB = 37.56 with 1 degree of freedom.
- The mean square for Factor B is MSB = 37.56.
- The F test statistic for a Factor B effect is $F_B = 5.547$ with a p-value of 0.03362.
- The error sum of squares is SSE = 94.78 with 14 degrees of freedom.
- The mean squared error is MSE = 6.77.

5.3 Fitting the *Interaction* Two-Factor ANOVA Model and Carrying Out the *F* Tests

- It's usually **more appropriate** to fit the two-factor ANOVA model **with** the ***interaction effect***, specified via the colon symbol `:`, for example `FactorA:FactorB`, as below:

```
my.anova <- aov(Response ~ FactorA + FactorB + FactorA:FactorB, data = my.data)
```

Now the ANOVA table includes a line representing the interaction between `FactorA` and `FactorB`:

```
summary(my.anova)

##                 Df Sum Sq Mean Sq F value Pr(>F)
## FactorA          2 148.11   74.06   9.801 0.0030
## FactorB          1  37.56   37.56   4.971 0.0457
## FactorA:FactorB  2   4.11    2.06   0.272 0.7664
## Residuals       12  90.67    7.56
```

From the output, we conclude that:

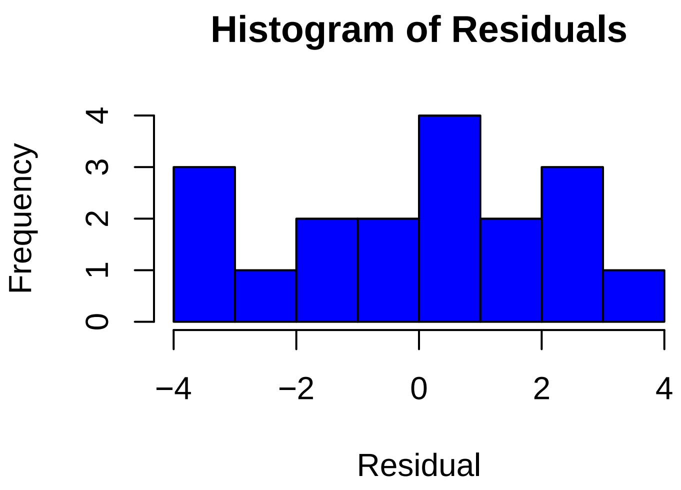
- The Factor A sum of squares is SSA = 148.11 with 2 degrees of freedom.
- The mean square for Factor A is MSA = 74.06.
- The F test statistic for a Factor A effect is $F_A = 9.801$, and the associated p-value is 0.0030.
- The Factor B sum of squares is SSB = 37.56 with 1 degree of freedom.
- The mean square for Factor B is MSB = 37.56.
- The F test statistic for a Factor B effect is $F_B = 4.971$ with a p-value of 0.0457.
- The AB interaction sum of squares is SSAB = 4.11 with 2 degrees of freedom.
- The mean square for the AB interaction is MSAB = 2.06.
- The F test statistic for an AB interaction effect is $F_{AB} = 0.272$ and the p-value is 0.7664.
- The error sum of squares is SSE = 90.67 with 12 degrees of freedom.
- The mean squared error is MSE = 7.56.

5.4 Residuals, Fitted Values, and Checking Assumptions

- The residuals and fitted values (group means) are obtained as in one-factor ANOVA (using the `$` operator, e.g. `my.anova$residuals` and `my.anova$fitted.values`).

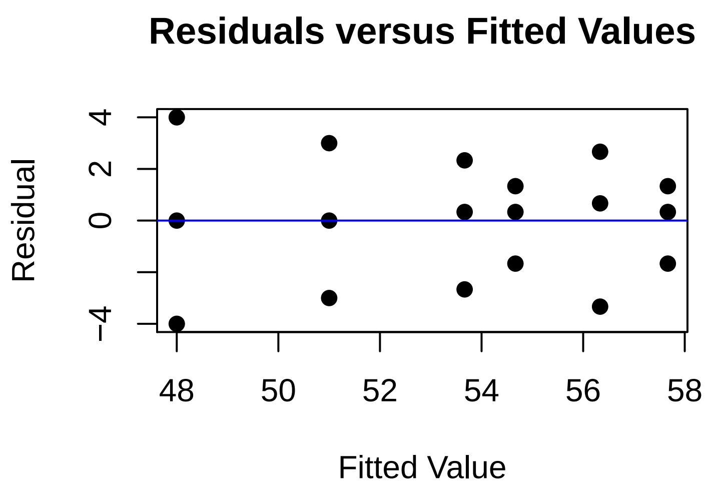
- The ***normality assumption*** is checked as in one-factor ANOVA too, by making a histogram or normal probability plot of the residuals. For example, a histogram of the residuals is made by typing:

```r
hist(my.anova$residuals, xlab = "Residual", col = "blue",
     main = "Histogram of Residuals")
```



- The ***equal standard deviation assumption*** is also checked as in one-factor ANOVA, by plotting the residuals versus the fitted values (group means). For example:

```r
plot(x = my.anova$fitted.values, y = my.anova$residuals,
     xlab = "Fitted Value",  ylab = "Residual", pch = 19,
     main = "Residuals versus Fitted Values")
abline(h = 0, col = "blue")
```



The vertical spread of the points above and below the horizontal line reflects the size of the standard deviation σ within each group, and if the equal σ assumption is met, should be fairly constant from left to right in the plot.

5.5 Summarizing and Graphing the Results

5.5.1 Computing and Graphing the Level Means (i.e. Row or Column Means)

- The ***level means*** (i.e.***row means*** or ***column means***) are computed one factor at a time using the `aggregate()` function, specifying `FUN = mean`. For example, the Factor A level means (row means) are obtained by typing:

```
aggregate(Response ~ FactorA, data = my.data, FUN =mean)

##   FactorA Response
## 1      A1 49.50000
## 2      A2 55.50000
## 3      A3 55.66667
```

and the Factor B level means (column means) are obtained by typing:

```
aggregate(Response ~ FactorB, data = my.data, FUN = mean)

##   FactorB Response
## 1      B1 55.00000
## 2      B2 52.11111
```

5.5.2 Computing and Graphing the Group Means

- The function `aggregate()` will also compute summary statistics (specified by `FUN`) separately for each group.
- For example, to compute the group means, type:

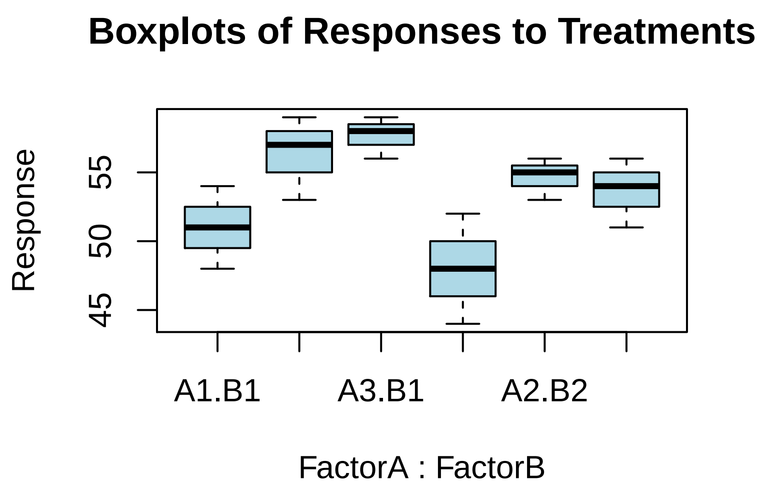
```
aggregate(Response ~ FactorA + FactorB, data = my.data, FUN = mean)

##   FactorA FactorB Response
## 1      A1      B1 51.00000
## 2      A2      B1 56.33333
## 3      A3      B1 57.66667
## 4      A1      B2 48.00000
## 5      A2      B2 54.66667
## 6      A3      B2 53.66667
```

and to compute the sample standard deviation separately for each group, type:

```
aggregate(Response ~ FactorA + FactorB, data = my.data, FUN = sd)

##   FactorA FactorB Response
## 1      A1      B1 3.000000
## 2      A2      B1 3.055050
## 3      A3      B1 1.527525
## 4      A1      B2 4.000000
## 5      A2      B2 1.527525
## 6      A3      B2 2.516611
```

- The `boxplot()` function will accept a *formula* with two factors, in which case it makes side-by-side boxplots of the groups:

```
boxplot(Response ~ FactorA + FactorB,
        data = my.data,
        col = "lightblue",
        main = "Boxplots of Responses to Treatments")
```



(We can customize the horizontal axis labels by passing a *vector* of group names to `boxplot()` via the `names` argument).

5.6 Interaction Plots

- To make an ***interaction plot***, use this function.

```
interaction.plot()      # Create an interaction plot of treatment group
                        # means
```

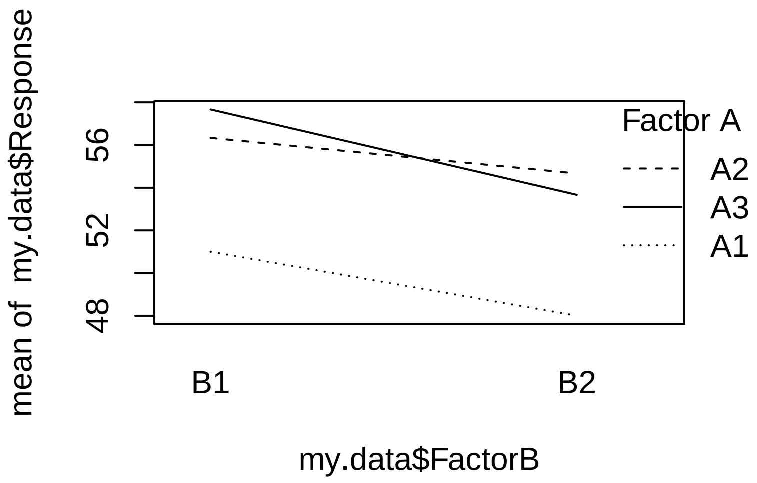
- The main arguments to `interaction.plot()` are:

`x.factor`	the factor whose levels will form the horizontal (x) axis.
`trace.factor`	the factor whose levels will be represented by separate lines.
`response`	the response variable.

Another (optional) argument, `trace.label`, can be used to specify a label for the legend.

For example (using `$` to extract the variables from `my.data`):

```
interaction.plot(x.factor = my.data$FactorB,
                 trace.factor = my.data$FactorA,
                 response = my.data$Response,
                 trace.label = "Factor A")
```



Section 5.6 Exercises

Exercise 3 Here's a data frame containing data from a **two-factor study** involving a factor with three levels (Factor A) and another with two levels (Factor B).

```
Y <- c(19, 20, 18, 23, 23, 24, 21, 27, 25, 25, 24, 29)
FactorA <- c("Low", "Low", "Low", "Low", "Med", "Med", "Med", "Med",
             "Hi", "Hi", "Hi", "Hi")
FactorB <- c("Yes", "Yes", "No", "No", "Yes", "Yes", "No", "No", "Yes",
             "Yes", "No", "No")
my.data <- data.frame(Y, FactorA, FactorB)
```

Use `aov()` to carry out a **two-factor ANOVA** ***with the interaction effect***, then use `summary()` to look at the ANOVA table:

```
my.anova <- aov(Y ~ FactorA + FactorB + FactorA:FactorB, data = my.data)
summary(my.anova)
```

a) For the ***F*** **tests** of the hypotheses

H_0 : There's no Factor A main effect
H_a : There's a Factor A main effect

H_0 : There's no Factor B main effect
H_a : There's a Factor B main effect

H_0 : There's no AB interaction effect
H_a : There's an AB interaction effect

report the following values from the ANOVA table and **answer** the **questions**.

Test statistic for the
AB interaction effect: F_{AB} = ____________

P-value: p = ____________

Based on the F test for the AB interaction effect, is there statistically significant evidence for an interaction effect between Factors A and B (Yes/No)?

b) **Report the following values** and **answer the question**.

Test statistic for the
Factor A main effect: F_A = ____________

P-value: p = ____________

Based on the F test for the Factor A main effect, is there statistically significant evidence for a Factor A main effect (Yes/No)?

c) **Report the following values** and **answer the question**.

Test statistic for the
Factor B main effect: F_B = ____________

P-value: p = ____________

Based on the F test for the Factor B main effect, is there statistically significant evidence for a Factor B main effect (Yes/No)?

d) Make an **interaction plot** by typing:

```r
interaction.plot(x.factor = my.data$FactorA,
                 trace.factor = my.data$FactorB,
                 response = my.data$Y)
```

Describe or just **sketch** the resulting plot.

6 Correlation

6.1 Computing the Correlation

- To calculate the (Pearson) ***correlation*** between two variables we use the function:

```
cor()             # Compute the correlation between x and y
```

- The `cor()` function takes two vector arguments:

`x`	a numeric vector.
`y`	another numeric vector.

- For example:

```
my.x <- c(60, 69, 66, 64, 54, 67, 59, 65, 63)
my.y <- c(136, 198, 194, 140, 93, 172, 116, 174, 145)
```

```
cor(x = my.x, y = my.y)

## [1] 0.9436756
```

Thus the correlation between `my.x` and `my.y` is $r = 0.9436756$.

- If your variables are in a *data frame*, use the dollar sign `$` to access them. For example, suppose we have the following *data frame*:

```
my.data <- data.frame(x = my.x, y = my.y)
```

```
my.data

##    x   y
## 1 60 136
## 2 69 198
## 3 66 194
## 4 64 140
## 5 54  93
## 6 67 172
## 7 59 116
## 8 65 174
## 9 63 145
```

To get the correlation, type:

```
cor(x = my.data$x, y = my.data$y)

## [1] 0.9436756
```

6.2 (Optional for Spring 2020) t Test and Confidence Interval for a Population Correlation ρ

- To test whether an observed correlation is statistically significantly different from zero, i.e. to test

$$\begin{aligned} H_0 : \rho &= 0 \\ H_a : \rho &\neq 0 \end{aligned}$$

  where ρ is the true (unknown) population correlation, we use this function:

```
cor.test()      # Tests whether a correlation is different from zero.
```

- For example (using `my.x` and `my.y` created above):

```
cor.test(my.x, my.y)

##
##  Pearsons product-moment correlation
##
## data:  my.x and my.y
## t = 7.5459, df = 7, p-value = 0.0001321
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7489030 0.9883703
## sample estimates:
##       cor
## 0.9436756
```

  From the output,

  - The observed test statistic value is $t = 7.5459$.
  - The p-value is 0.0001321 (from the t distribution with $n - 2 = 7$ degrees of freedom).
  - The 95% confidence interval for ρ is $(0.7489030, \quad 0.9883703)$.

6.3 (Optional for Spring 2020) Spearman Rank Correlation

- To compute the ***Spearman rank correlation***, we specify `method = "spearman"` in `cor()`. For example:

```
cor(my.x, my.y, method = "spearman")

## [1] 0.9333333
```

Thus the **rank correlation** between `x` and `y` is $r_{sr} = 0.9333333$.

- We could also specify `method = "spearman"` in the `cor.test()` function for a hypothesis test based on the Spearman rank correlation.

Section 6.3 Exercises

Exercise 4 Here are two vectors containing variables X and Y measured on each of $n = 18$ individuals.

```
x.vals <- c(41, 49, 44, 43, 42, 53, 50, 55, 49, 46, 52, 48, 46, 46,
            48, 66, 58, 69)
y.vals <- c(19, 19, 23, 26, 24, 27, 31, 30, 28, 32, 29, 34, 30, 30,
            31, 37, 40, 42)
```

a) Make a **scatterplot** of the data:

```
plot(x = x.vals, y = y.vals,
     xlab = "X", ylab = "Y", pch = 19,
     main = "Scatterplot of Y versus X")
```

Don't print the plot. Just describe the relationship between the two variables (Positive/Negative relationship).

b) Use `cor()` to compute the **correlation** $\boldsymbol{r}$ between the two variables:

```
cor(x = x.vals, y = y.vals)
```

Report the value of $\boldsymbol{r}$.

7 Simple Linear Regression

7.1 Fitting the Regression Model and Conducting *t* Tests for the Slope and Intercept

- To obtain the **fitted regression line**

$$\hat{Y} \;=\; b_0 + b_1 X$$

and carry out the associated $\boldsymbol{t}$ **tests** on the slope b_1 (and intercept b_0), we use the ”linear model” function:

```
lm()        # Fit a linear regression model and carry out the t tests
            # for the slope and intercept
```

- The `lm()` function takes as it's main arguments:

`formula`	a formula indicating the response and explanatory variables.
`data`	a data frame containing the variables used in the formula.

  We look at the results using `summary()`.

- For example, consider again the data frame `my.data` (created above):

```
my.data

##    x   y
## 1 60 136
## 2 69 198
## 3 66 194
## 4 64 140
## 5 54  93
## 6 67 172
## 7 59 116
## 8 65 174
## 9 63 145
```

  To fit the regression model to the variables `x` and `y` in `my.data`, we type:

```
my.reg <- lm(y ~ x, data = my.data)
```

  Above, the *formula* `y ~ x` indicates that `y` is the response variable and `x` is the explanatory variable (predictor). To look at the results of the regression analysis, we type:

```
summary(my.reg)

##
## Call:
## lm(formula = y ~ x, data = my.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.192  -7.233   2.849   5.727  20.424
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -301.0872    60.1885  -5.002 0.001561
## x              7.1919     0.9531   7.546 0.000132
```

```
##
## Residual standard error: 12.5 on 7 degrees of freedom
## Multiple R-squared:  0.8905,Adjusted R-squared:  0.8749
## F-statistic: 56.94 on 1 and 7 DF,  p-value: 0.0001321
```

From the output above, we conclude the following:

– The least squares estimates of the true (unknown) regression coefficients β_0 and β_1 are $b_0 = -301.0872$ and $b_1 = 7.1919$. Thus the equation of the fitted regression line is

$$\hat{y} \;=\; -301.0872 + 7.1919x.$$

– The standard errors of the estimates are

$$\begin{aligned} S_{b_0} &= 60.1885 \\ S_{b_1} &= 0.9531. \end{aligned}$$

– The test statistic for t test of the hypotheses

$$\begin{aligned} H_0 : \beta_0 &= 0 \\ H_a : \beta_0 &\neq 0 \end{aligned}$$

is $t = b_0/S_{b_0} = -0.5002$. The p-value is 0.001561 (from the t distribution with $n-2 = 7$ degrees of freedom).

– The test statistic for t test of

$$\begin{aligned} H_0 : \beta_1 &= 0 \\ H_a : \beta_1 &\neq 0 \qquad (1) \end{aligned}$$

is $t = b_1/S_{b_1} = 7.546$. The p-value is 0.000132 (from the t distribution with $n-2 = 7$ df).

– The square root of the mean squared error is labeled `Residual standard error`, and its value is $\sqrt{\text{MSE}} = 12.5$.

– The R-squared value is $R^2 = 0.8905$, labeled `Multiple R-squared`. Thus 89.05% of the Y variation is explained by the explanatory variable.

– The test statistic for the ***model F test***, which is just another test of the hypotheses (1) for the slope, is $F = \text{MSR}/\text{MSE} = 56.94$. The p-value is 0.000132 (from the F distribution with numerator and denominator degrees of freedom 1 and 7, respectively).

7.2 (Optional for Spring 2020) Computing Confidence Intervals for β_0 and β_1

- Once a regression model has been fit to the data, we can compute a $(1-\alpha)100\%$ confidence interval

$$b_1 \;\pm\; t_{\alpha/2}\, S_{b_1}$$

for the true (unknown) slope β_1 by passing our *lm* object to the function:

```
confint()     # Compute a t confidence interval for the slope and
              # intercept
```

- The `confint()` function takes arguments:

`object`	an lm object (as returned by lm()).
`level`	the level of confidence (as a decimal value).

- For example, using the *lm* object `my.reg` from above, to compute a 95% confidence interval for β_1, we type:

```
confint(my.reg, level = 0.95)

##                    2.5 %      97.5 %
## (Intercept) -443.410309 -158.764110
## x              4.938183    9.445538
```

(Note `confint()` also returns the confidence interval $b_0 \pm t_{\alpha/2} S_{b_0}$ for the intercept β_0.)

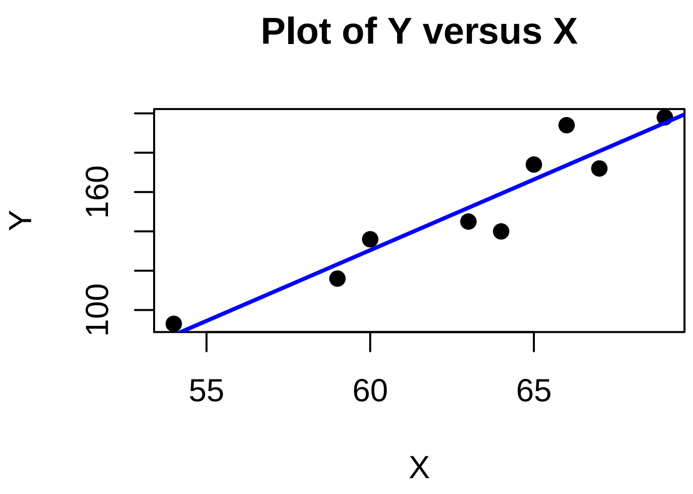
From the output:

- The 95% confidence interval for β_0 is (-443.410, -158.764).
- The 95% confidence interval for β_1 is (4.938, 9.446).

7.3 Plotting the Regression Line with the Data

- To plot the data and add the regression line to the plot, we use `plot()` followed by `abline()` (with `my.reg` created above):

```
plot(x = my.data$x, y = my.data$y,
     xlab = "X", ylab = "Y", pch = 19,
     main = "Plot of Y versus X")

abline(my.reg, col = "blue", lwd = 2)
```



7.4 (Optional for Spring 2020) The Regression ANOVA Table

- To view the regression ANOVA table, use the function:

```
anova()         # Look at the regression ANOVA table after fitting the
                # linear regression model
```

- For example, using the *lm* object created above, to see the regression ANOVA table, type:

```
anova(my.reg)

## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value    Pr(>F)
## x          1 8896.3  8896.3  56.941 0.0001321
## Residuals  7 1093.7   156.2
```

  From the output, we get:

  - The regression sum of squares is SSR = 8896.3 and its degrees of freedom is 1.
  - The error sum of squares is SSE = 1093.7 and its degrees of freedom is 7.
  - The mean squares are MSR = 8896.3 and MSE = 156.2.
  - The test statistic for the F test for the slope parameter β_1 (i.e. for the hypotheses (1)), is F = MSR/MSE = 56.941 and the p-value (from the F distribution with numerator and denominator degrees of freedom 1 and 7, respectively) is 0.0001321.

7.5 Residuals, Fitted Values, and Checking Assumptions

- The object `my.reg` created above is a collection of other objects. To see what objects `my.reg` contains, type:

```
names(my.reg)

##  [1] "coefficients"  "residuals"     "effects"
##  [4] "rank"          "fitted.values" "assign"
##  [7] "qr"            "df.residual"   "xlevels"
## [10] "call"          "terms"         "model"
```

- To obtain specific named elements, we use the `$` operator. For example, to obtain the regression coefficients (y-intercept and slope), type:

```
my.reg$coefficients

## (Intercept)           x
##  -301.08721     7.19186
```

and to get the residuals and the fitted (predicted) values, type:

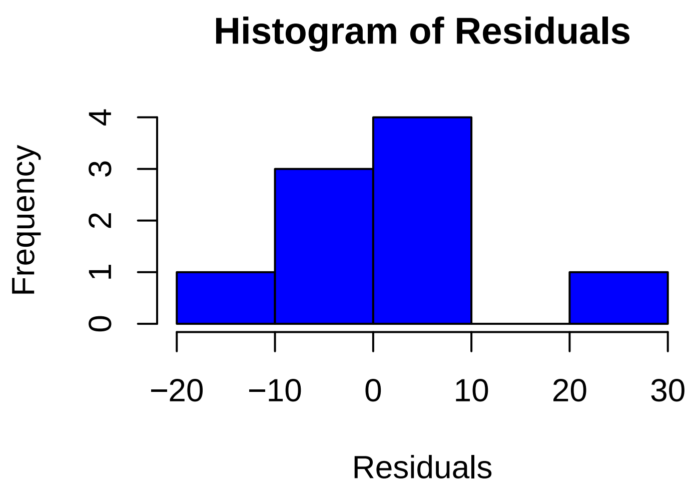
```
my.reg$residuals

##          1          2          3          4          5
##   5.575581   2.848837  20.424419 -19.191860   5.726744
##          6          7          8          9
##  -8.767442  -7.232558   7.616279  -7.000000
```

and

```
my.reg$fitted.values

##         1         2         3         4         5         6
## 130.42442 195.15116 173.57558 159.19186  87.27326 180.76744
##         7         8         9
## 123.23256 166.38372 152.00000
```

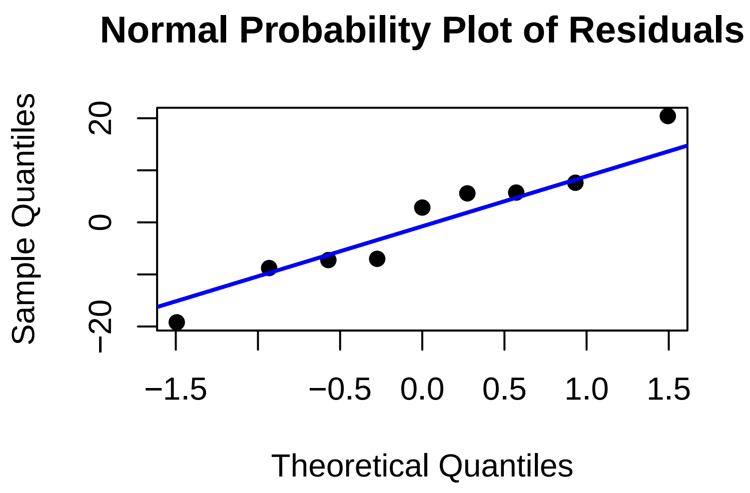
To make a histogram and normal probability plot of the residuals (for checking the normality assumption for the t and F tests), type:

```
hist(my.reg$residuals,
     col = "blue",
     xlab = "Residuals",
     main = "Histogram of Residuals")
```



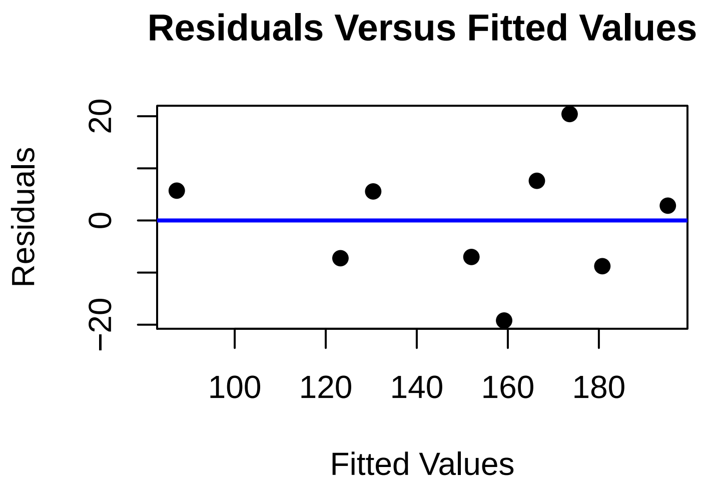
and

```
qqnorm(my.reg$residuals, pch = 19,
       main = "Normal Probability Plot of Residuals")
qqline(my.reg$residuals, col = "blue", lwd = 2)
```



To make a plot of the residuals versus fitted values (for checking the assumption that the standard deviation σ doesn't depend on X), type:

```
plot(x = my.reg$fitted.values, y = my.reg$residuals,
     xlab = "Fitted Values", ylab = "Residuals",
     main = "Residuals Versus Fitted Values",
     pch = 19)

abline(h = 0, col = "blue", lwd = 2)
```



Section 7.5 Exercises

Exercise 5 Here are vectors containing data on two variables X and Y measured on each of $n = 18$ individuals.

```
my.x <- c(41, 49, 44, 43, 42, 53, 50, 55, 49, 46, 52, 48, 46, 46,
          48, 66, 58, 69)
my.y <- c(19, 19, 23, 26, 24, 27, 31, 30, 28, 32, 29, 34, 30, 30,
          31, 37, 40, 42)
my.data = data.frame(y = my.y, x = my.x)
```

a) Use `lm()` to fit the **regression model**

$$Y = \beta_0 + \beta_1 X + \epsilon$$

to the data, with `y` as the response and `x` as the explanatory variable (predictor):

```
my.reg <- lm(y ~ x, data = my.data)
```

Then make a **scatterplot** of `y` versus `x`, and add the **regression line** to the plot:

```
plot(x = x, y = y, pch = 19,
     main = "Scatterplot of Y vs X")
abline(my.reg, col = "blue", lwd = 2)
```

Don't print the plot. Just describe the relationship between the two variables (Positive/Negative relationship).

b) Look at the results of the regression analysis:

```
summary(my.reg)
```

Then **report** the following values and **answer** the **questions**.

Estimated Y-intercept (`Estimate`):	b_0	$=$	____________
Estimated slope (`Estimate`):	b_1	$=$	____________
Test statistic for the slope (`t value`):	t	$=$	____________
P-value for the slope (`Pr(>|t|)`):	p	$=$	____________
R^2 value (`Multiple R-Squared`):	R^2	$=$	____________

Based on the t test for the slope, is there any statistically significant evidence that `y` is related to `x` (i.e. that the true (unknown) slope β_1 is different from zero) (Yes/No)?

The $\boldsymbol{R^2}$ value indicates the percentage (as a proportion) of the variation in the Y variable can be explained by the X variable. What is the value of this percentage?

8 Multiple Regression

8.1 Fitting the Multiple Regression Model and Conducting *t* Tests for the Coefficients

- Consider the *data frame* `my.data` shown below.

```
my.response <- c(24, 22, 25, 22, 26, 26, 25, 27, 30, 33, 29, 30)

my.x1 <- c(10, 11, 13, 12, 13, 16, 15, 20, 18, 19, 19, 22)
my.x2 <- c(2.4, 1.9, 2.6, 1.8, 1.3, 0.9, 1.2, 1.0, 1.3, 1.0, 0.8, 0.7)
my.x3 <- c(100, 75, 68, 77, 80, 81, 72, 55, 39, 67, 77, 94)

my.data <- data.frame(response = my.response,
                      x1 = my.x1, x2 = my.x2, x3 = my.x3)
```

```
my.data

##    response x1  x2  x3
## 1        24 10 2.4 100
## 2        22 11 1.9  75
## 3        25 13 2.6  68
## 4        22 12 1.8  77
## 5        26 13 1.3  80
## 6        26 16 0.9  81
## 7        25 15 1.2  72
## 8        27 20 1.0  55
## 9        30 18 1.3  39
## 10       33 19 1.0  67
## 11       29 19 0.8  77
## 12       30 22 0.7  94
```

To obtain the **fitted multiple regression model**

$$\hat{Y} \;=\; b_0 + b_1 X_1 + b_2 X_2 + b_3 X_3$$

and carry out the associated ***t* tests** on the coefficients b_0, b_1, b_2, and b_3, we use `lm()`:

```
my.reg <- lm(response ~ x1 + x2 + x3, data = my.data)
```

We view the results using `summary()`:

```
summary(my.reg)

##
## Call:
## lm(formula = response ~ x1 + x2 + x3, data = my.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8744 -1.0259 -0.3707  1.4058  4.0007
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.01050    8.82022   1.588   0.1508
## x1           0.78376    0.31312   2.503   0.0368
## x2           0.60720    1.87980   0.323   0.7550
## x3          -0.00761    0.04393  -0.173   0.8668
##
## Residual standard error: 2.187 on 8 degrees of freedom
## Multiple R-squared:  0.6936,Adjusted R-squared:  0.5787
## F-statistic: 6.036 on 3 and 8 DF,  p-value: 0.01884
```

From the output, we conclude that

- The **estimates** of the true (unknown) regression coefficients $\beta_0, \beta_1, \beta_2$, and β_3 are $\boldsymbol{b_0 = 14.0105}$, $\boldsymbol{b_1 = 0.7838}$, $\boldsymbol{b_1 = 0.6072}$, and $\boldsymbol{b_1 = -0.0076}$. Thus the fitted model is
$$\hat{Y} = 14.0105 + 0.7838X_1 + 0.6072X_2 - 0.0076X_3.$$
- The **standard errors** of the estimates are
$$\begin{aligned} S_{b_0} &= \mathbf{8.8202} \\ S_{b_1} &= \mathbf{0.3131} \\ S_{b_2} &= \mathbf{1.8798} \\ S_{b_3} &= \mathbf{0.0439} \end{aligned}$$
- The test statistics and p-values for the $\boldsymbol{t}$ ***tests*** of hypotheses of the form
$$\begin{aligned} H_0 : \beta_k &= 0 \\ H_a : \beta_k &\neq 0 \end{aligned}$$
are
  * $\boldsymbol{t = b_0/SE(b_0) = 1.588}$ and the p-value is **0.1508** (from the t distribution with $\boldsymbol{n - 2 = 8}$ degrees of freedom).
  * $\boldsymbol{t = b_1/SE(b_1) = 2.503}$ and the p-value is **0.0368** (from the t distribution with $\boldsymbol{n - 2 = 8}$ degrees of freedom).
  * $\boldsymbol{t = b_2/SE(b_2) = 0.323}$ and the p-value is **0.755** (from the t distribution with $\boldsymbol{n - 2 = 8}$ degrees of freedom).
  * $\boldsymbol{t = b_3/SE(b_3) = -0.173}$ and the p-value is **0.8668** (from the t distribution with $\boldsymbol{n - 2 = 8}$ degrees of freedom).
- The square root of the **mean squared error** is labeled `Residual standard error`, and its value is $\boldsymbol{\sqrt{\text{MSE}} = 2.187}$.
- The R-squared value is $\boldsymbol{R^2 = 0.6936}$, labeled `Multiple R-squared`. Thus 69.36% of the Y variation is explained by the three explanatory variables.
- The **adjusted** $\boldsymbol{R^2}$ is $\boldsymbol{R^2_a = 0.5787}$, labeled `Adjusted R-squared`.
- The F statistic for the **model** $\boldsymbol{F}$ **test** of the hypotheses
$$\begin{aligned} H_0 : &\quad \beta_1 = \beta_2 = \beta_3 = 0 \\ H_a : &\quad \text{not all of } \beta_1, \beta_2, \text{ and } \beta_3 \text{ are } 0 \end{aligned}$$
is $\boldsymbol{F = \text{MSR}/\text{MSE} = 6.036}$. From the F distribution with numerator and denominator degrees of freedom **3** and **8**, respectively, the **p-value** is **0.01884**. Thus we reject H_0 and conclude that at least one of β_1, β_2, or β_3 is different from 0.

8.2 (Optional for Spring 2020) Computing Confidence Intervals for the $\boldsymbol{\beta_k}$'s

- Confidence intervals for coefficients in a multiple regression analysis are obtained exactly as described in Subsection 7.2 for simple linear regression. For example:

```
confint(my.reg)
```

(Output not shown.)

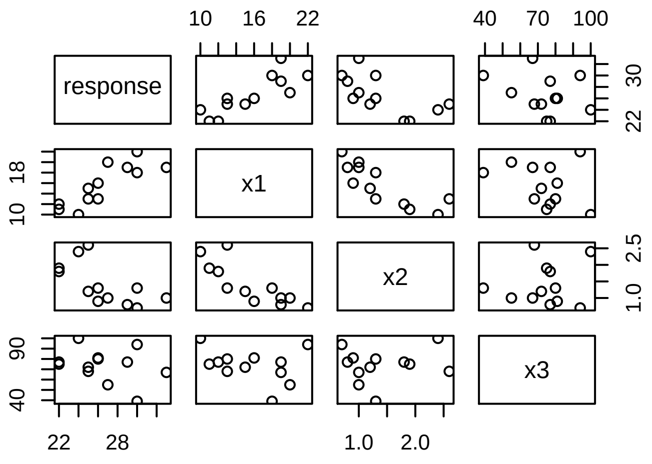
8.3 Scatterplot Matrices and Correlation Matrices

- We can make a ***scatterplot matrix*** using the function

```
pairs()      # Creates a scatterplot matrix from the columns of a
             # data frame or matrix.
```

The `pairs()` function takes a *data frame* (of *numerical* variables) as an argument and produces a scatterplot matrix from its columns. For example, using `my.data` from above:

```
pairs(my.data)
```



Above, each label (`response`, `x1`, `x2`, and `x3`) indicates the y-axis for plots in its row and the x-axis for plots in its column. Each plot in the upper right is duplicated in the lower left, but with its axes *transposed*.

- We can compute the ***correlation matrix*** or by passing a *data frame* (of *numerical* variables) to the `cor()` function:

```
cor()      # Computes the correlation matrix of the variables stored as
           # as columns of a data frame or matrix.
```

- For example, using `my.data` from above, the **correlation matrix** is

```
cor(my.data)

##            response         x1         x2         x3
## response  1.0000000  0.8290126 -0.6389366 -0.2968452
## x1        0.8290126  1.0000000 -0.8203206 -0.3026625
## x2       -0.6389366 -0.8203206  1.0000000  0.1342660
## x3       -0.2968452 -0.3026625  0.1342660  1.0000000
```

  The diagonal elements of the **correlation matrix** are all one and the off-diagonal elements are the **correlations** between pairs of variables (corresponding to the scatterplots in the scatterplot matrix above).

8.4 (Optional for Spring 2020) The Multiple Regression ANOVA Table

- The regression ANOVA table for a multiple regression analysis is obtained exactly as described in Subsection 7.4 for simple linear regression. For example:

```
anova(my.reg)
```

  (Output not shown.)

8.5 Residuals, Fitted Values, and Checking Assumptions

- The residuals and fitted values for a multiple regression analysis are obtained exactly as described in Subsection 7.5 for simple linear regression. The residuals are used to check the assumptions in multiple regression as described in that subsection too. For example:

```
hist(my.reg$residuals,
     col = "blue",
     xlab = "Residuals",
     main = "Histogram of Residuals")
```

```
plot(x = my.reg$fitted.values, y = my.reg$residuals,
     xlab = "Fitted Values", ylab = "Residuals",
     main = "Residuals Versus Fitted Values",
     pch = 19)

abline(h = 0, col = "blue", lwd = 2)
```

  (Plots not shown.)

Section 8.5 Exercises

Exercise 6 Here's a *data frame* containing data on a response variable y and two explanatory (predictor) variables x_1 and x_2 measured on each of $n = 15$ individuals.

```
y.var <- c(23, 26, 24, 27, 31, 30, 28, 32, 29, 34, 30, 30, 31, 37, 40)
x1.var <- c(44, 43, 42, 53, 50, 55, 49, 46, 52, 48, 46, 46, 48, 66, 58)
x2.var <- c(34, 33, 22, 23, 30, 35, 29, 26, 22, 18, 16, 26, 28, 16, 18)

my.data <- data.frame(y = y.var, x1 = x1.var, x2 = x2.var)
```

```
my.data

##     y x1 x2
## 1  23 44 34
## 2  26 43 33
## 3  24 42 22
## 4  27 53 23
## 5  31 50 30
## 6  30 55 35
## 7  28 49 29
## 8  32 46 26
## 9  29 52 22
## 10 34 48 18
## 11 30 46 16
## 12 30 46 26
## 13 31 48 28
## 14 37 66 16
## 15 40 58 18
```

a) Make a **scatterplot matrix** of the `y`, `x1`, and `x2` variables:

```
pairs(my.data)
```

Don't print the plot. Just describe the relationship between the variables `y` and `x1` (Positive/Negative relationship).

b) Use `lm()` to obtain the **fitted multiple regression model**

$$\hat{Y} \;=\; b_0 + b_1X_1 + b_2X_2$$

with `y` as the response and `x1` and `x2` as explanatory variables (predictors), then look at the results:

```
my.reg <- lm(y ~ x1 + x2, data = my.data)

summary(my.reg)
```

Report the following values and **answer** the **questions**.

Estimated Y-intercept (`Estimate`):	b_0	=	____________
Estimated coefficient of $\boldsymbol{X_1}$ (`Estimate`):	b_1	=	____________
Test statistic for the $\boldsymbol{X_1}$ coefficient (`t value`):	t	=	____________
P-value for the $\boldsymbol{X_1}$ coefficient (`Pr(>|t|)`):	p	=	____________
Estimated coefficient of $\boldsymbol{X_2}$ (`Estimate`):	b_2	=	____________
Test statistic for the $\boldsymbol{X_2}$ coefficient (`t value`):	t	=	____________
P-value for the $\boldsymbol{X_2}$ coefficient (`Pr(>|t|)`):	p	=	____________
R^2 value (`Multiple R-Squared`):	R^2	=	____________

Based on the **p-value** for the $\boldsymbol{X_1}$ coefficient, is $\boldsymbol{Y}$ statistically significantly related to $\boldsymbol{X_1}$ (controlling for the effect of X_2) (Yes/No)?

Based on the **p-value** for the $\boldsymbol{X_2}$ coefficient, is $\boldsymbol{Y}$ statistically significantly related to $\boldsymbol{X_2}$ (controlling for the effect of X_1) (Yes/No)?

The $\boldsymbol{R^2}$ value indicates the percentage (as a proportion) of the variation in the Y variable can be explained by the two variables X_1 and X_2. What is the value of this percentage?