MACQUARIE University
SYDNEY·AUSTRALIA

STAT2170 AND STAT6180
APPLIED STATISTICS

SGTA Week 12: Two-Way ANOVA and Multiple Comparisons 2023

Part 1

Question 1

An experiment is conducted to determine the effect of firing temperature and position on the baked density of a carbon anode. The data is in the file `density.csv`.

a. Produce an interaction plots of the data and comment.
b. Write the 3 null and alternative hypotheses relevant to this question.
c. Using the `lm` function first fit a model with interaction, and test for significance. Then fit a model with main effects only, with graphs. What are your conclusions?
d. For each assumption comment. If there are any suggestions that the data are not normal then store the residuals and produce a normal probability plot using these.
e. What degrees of freedom are associated with the F test for position and temperature respectively?
f. Using `TukeyHSD` produce multiple comparisons between each level for both temperature and position. What can you conclude about the effect of position in terms of numerical values.
g. How would your answers to c. and f. be affected if there was a significant interaction?

Question 2

The effect of age and gender is studied by looking at their negotation experience for purchasing a car. A 'medium' priced six year old car was selected for the study and participants sent to a dealer at random. The initial offer each participant made was recorded. The data is available in `cash.dat` on iLearn.

a. Check is the design balanced or unbalanced?
b. Does the order of inclusion into the model matter for this scenario?
c. Confirm your answer by producing two different sequential ANOVA tables for the Two-Way ANOVA model with interaction.
d. Are there any effects of `gender` and `age` on the observed `offer`?

Part 2

Up to now, we have successfully applied tons of functions built in `R/RStudio`. But beyond these built-in functions, we can actually write our own functions to perform specific tasks we want. Below we introduce the structure of custom functions and flow control commands inside these functions.

Question 1

As in mathematics, a `function` in computer science

- has a name
- takes some inputs

- performs some action
- returns some output

For example, `mean()` has name `mean`, takes a vector as a input, calculates the average of the values in this vector, and returns the average as an output.

a) Execute the following code to define a function `my.mean`

```
my.mean <- function(x) {  # Name the function as my.mean  # Define x as the input
  output <- sum(x) / length(x) # Calculate the average
  return(output)  # Return the average as an output
  }
```

b) Now we apply `my.mean`. Does the code below produce the correct mean of `c(2,3)`?

```
my.mean(c(2,3))
```

c) Following the example of `my.mean`, please design a function named `my.IQR` to calculate the inter-quantile range, namely, the difference between the 75% sample quantile and the 25% sample quantile of a vector. Use this `my.IQR` function to calculate the inter-quantile range of `c(1,2,3,4,5)`.

- Hint: we can obtain the quartile of a vector with `quantile()`.

Question 2

Defined as the order in which individual statements are evaluated, flow control commands appear in many computing languages such as `FORTRAN`, `C/C++`, `Java` and `Julia`. Below we introduce two flow control commands in `R/RStudio`.

a) The `if()` command allows us to control which operations are executed based on some conditions. The usual syntax of `if()` is

```
if(condition){
  operations when condition is TRUE
  }
```

i) Execute the following code to examine function `if()`

```
is.it.true <- function(x) {
  if(x == TRUE) {print("The statement is true!")}
  if(x == FALSE) {print("The statement is false!")}
}
is.it.true(TRUE)
is.it.true(FALSE)
is.it.true(10 > 0)
is.it.true(10 < 0)
```

ii) Correct all the errors in the function below, which tries to generate one uniformly distributed random variable between 0 and c with `if()`.

```
my.unif <- function(c){
    if(c > 0){x = runif(n = 1, min = 0, max = c)}
    if(c < 0){x = runif(n = 1, min = c, max = 0)}
    if(c = 0){x = 0}
}
return(x)
```

Notice that the code in both examples can be improved using `else` and `else if` to chain multiple `if` statements together. Some sample code can be found in the solution.

b) The `for()` command allows one to specify that certain operations should be repeated a fixed number of times. The usual syntax of `for()` is

```
output <- rep(NA, length = n) # output
for(i in 1:n){ #n is the number of times the commands repeat
  commands # body; aim to update the content in output[i]
  }
```

Execute the following code to generate and plot a vector `noise`. Does the plot indicate the independence among the elements in `noise` ?

```
n = 100
noise = rep(NA, n)
noise[1] = 1
for(i in 1:(n-1)){
  noise[i+1] = 0.99 * noise[i] + rnorm(n = 1, mean = 0, sd = 1)
}
plot(noise)
```

To learn more about functions & flow control, please refer to Chapters 19 and 21 of https://r4ds.had.co.nz/