# Lab07 - Contrast of Hypothesis

## Introduction

In R, as the results are plain numbers, the possibility to detect an error based on output is negligible. We can get help from two tools:

▼ Command Parameters:

Arguments passed to a function or command to control its behavior and/or results, apart from the unique and own parameters of each function. In R, command parameters can include both mandatory and optional arguments that affect the execution of a specific function or command.

- **alternative:** controls the alternate hypothesis and accepts “two.sided”, “less” and “greater”
- **na.action:** controls what to do with NA values.
  1. **na.omit**: This option is the default. It removes all rows that contain at least one NA value before performing the operation.
  2. **na.fail**: This option generates an error if there are any missing values present. It's useful when you want to ensure that there are no missing values in the data.
  3. **na.exclude**: Similar to `na.omit`, but retains rows with NA values in calculations, treating missing values as unknown data but not removing them from operations.
  4. **na.pass:** This option does not handle missing values. It simply passes the data without making any changes, leaving the responsibility of handling missing values to the user.
- **conf.level:** this is the confidence level (1-$\alpha$)

> 💡 IMPORTANT! CAUTION! ANDE VAYAS VE CON CUIDAOOO
> If you do not read the commands description you can find yourself using an incorrect option because of the default values for the parameters.
> Even if you are following the class material it is your responsibility to know the commands you use, there is always the chance of R (or a package) updating and the defaults changing or the command being deprecated altogether

▼ Output Attributes

Some of the attributes that an object has will not be accessible with $ and will require to use @ or the attr() command. We will see each output as we find them, but it is necessary that you are familiar with the output, the attributes and that you can use this functionality
correctly. Otherwise your codding capabilities will be greatly reduced and you will not make effective functions.
Some of the more important and common attributives for output are:

- **p.value:** this is the p-value of the test and will tell us if we must reject $H0$ (Hipótesis nula)
or not.

p_value > 0.05 we accept the H0, otherwise we reject it

- **statistic:** value of the statistic used (t value of mean differences, F value for equality
of variances, chi value for difference in proportions, etc.)
- **method:** test or computing method used
- **estimate:** statistic computed (means difference, ratio for variances, etc.)
- **conf.int:** confidence interval for the estimate
- **parameter:** additional information usually related to the statistic

▼ Numeric

Describing minimum and maximum values, number of missing values and confidence interval for the mean are all common practice and things that we should take into consideration.

The goal is not to present some static set of numbers, but rather to present all the information necessary for the reader to understand the scope and distribution of the variables.

DATA.ORIGINAL is the database CarsEng.sav
• DATA is the same but only with those cars with Rear and Front traction
This is the code used to load and process both datasets:

1. **Lectura de datos**: La primera línea lee los datos desde el archivo "CarsEng.sav" usando la función `read.spss()` del paquete `foreign`, convirtiéndolos en un data frame de R.
2. Necesitas el package de foreign
   '
   `install.packages("foreign")`'

```
DATA <- foreign::read.spss("DATA/CarsEng.sav", to.data.fra
me = TRUE)
```

**2. Manipulación de nombres de columnas**: Las siguientes líneas están modificando los nombres de las columnas en el data frame para que estén en mayúsculas y tengan guiones bajos en lugar de espacios, puntos y otros caracteres especiales. Lo de fixed es para que interprete las cadenas de caracteres literalmente, no como Regular Expresiones (fixed = false)

```
names(DATA) <- toupper(names(DATA))
names(DATA) <- toupper(gsub(" ","_",names(DATA),fixed = TR
```

```
UE))
names(DATA) <- toupper(gsub(".","_",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub("(","_",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub(")","_",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub("ª","_",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub("Á","A",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub("É","E",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub("Í","I",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub("Ó","O",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub("Ú","U",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub("º","_",names(DATA),fixed = TR
UE))
names(DATA) <- toupper(gsub("Ñ","N",names(DATA),fixed = TR
UE))
```

**3. Manipulación de datos**: Luego, el código parece estar cambiando los valores en la columna "TRACTION" de modo que "Trasera" se convierte en "Rear".

```
DATA$TRACTION[DATA$TRACTION == "Trasera"] = "Rear"
```

**4. Resumen de los datos originales**: Se imprime un resumen de la columna "TRACTION" del data frame original antes de hacer cambios.

▼ summary()

**summary()??:** para un data frame, como en este caso, `summary()` proporciona un resumen estadístico de cada una de las variables en el data frame. El resumen incluye información como la media, la mediana, los valores mínimo y máximo, y los cuartiles para las variables numéricas, así como el recuento de valores únicos y los valores más frecuentes para las variables categóricas.

```
DATA.ORIGINAL <- DATA
summary(DATA.ORIGINAL$TRACTION)
```

**5. Filtrado de datos**: Los datos se filtran para incluir solo las filas donde la columna "TRACTION" tiene valores "Rear" o "Front".

```
DATA <- DATA.ORIGINAL[DATA.ORIGINAL$TRACTION == "Rear" | D
ATA.ORIGINAL$TRACTION == "Front",]
```

**6. Manipulación de factores**: La columna "TRACTION" se convierte en un factor.

▼ factor()

En R, un "factor" es un tipo de datos que se utiliza para representar variables categóricas, es decir, variables que tienen un número finito y discreto de categorías o niveles. Los factores son útiles para trabajar con datos cualitativos o nominales, como el género, la categoría de producto, la región geográfica, etc.

Cuando se crea un factor en R, se asignan niveles a cada categoría única presente en los datos. Estos niveles son enteros que van desde 1 hasta el número total de categorías. Los factores también pueden tener etiquetas asociadas con cada nivel para proporcionar descripciones más significativas de las categorías.

Los factores son útiles porque permiten realizar operaciones específicas para variables categóricas, como ordenar las categorías correctamente,

aplicar análisis estadísticos adecuados y generar gráficos comprensibles.

Aquí hay un ejemplo de cómo se crea y se trabaja con un factor en R:

```
# Crear un vector de género
genero <- c("Masculino", "Femenino", "Masculino", "Feme
nino", "Masculino")

# Convertir el vector en un factor
genero_factor <- factor(genero)

# Mostrar los niveles del factor
levels(genero_factor)
# Output: [1] "Femenino"  "Masculino"

# Obtener la frecuencia de cada nivel
table(genero_factor)
# Output:
# Femenino Masculino
#        2         3
```

```
DATA$TRACTION <- factor(DATA$TRACTION)
```

**7. Definición de un factor personalizado**: La columna "CHANGE_SHORT" se convierte en un factor con niveles y etiquetas específicas.

```
DATA$CHANGE_SHORT <- factor(DATA$CHANGE, levels = c
("M","A","AS"), labels = c("M","A","A"))
```

En resumen, el código carga datos desde un archivo SPSS, realiza cambios en los nombres de columnas y en los valores de algunas columnas, filtra el conjunto de datos para incluir solo ciertas observaciones, y luego ajusta el formato de algunas variables categóricas.

▼ COMPARISONS

▼ Notes

> IMPORTANT:
> It is important to understand that tests are different if the samples are independent
> (
> ***Independent samples*** *mean that the numbers of one sample do not affect the numbers of the other sample in any way*)
> versus if the samples are paired
> (
> ***Paired samples*** *are those with values that are somehow dependent within the subject*.)

▼ One sample t-test

There are times when you want to compare the mean value of your sample $\bar{x}$ against the mean value of the population (or from other studies)

*We assume that our variable follows a normal distribution*

```
muestra <- c(23, 25, 27, 24, 28, 22, 26, 29, 25, 26)
t.test(muestra, mu = valor_quequeramoscomparar 24)
```

Y nos devuelve esto,

p_value > 0.05 luego podemos aceptar la hipotesis, aunque la verdadera media no sea 24

```
    One Sample t-test

data:  muestra
t = 2.1828, df = 9, p-value = 0.05691
alternative hypothesis: true mean is not equal to 24
95 percent confidence interval:
 23.94548 27.05452
sample estimates:
mean of x
     25.5
```

▼ Two Independent Samples

▼ Test for Normality

El valor W cuanto más cerca de 1 esté, más se ajustan los datos a una distribución normal. Aun así, si p < 0.05 se rechaza la H0 y no es normal

```
normality <- function(valores, alpha = 0.05) {
norm <- shapiro.test(valores)
result <- data.frame(p.value = norm$p.value)
result$is.normal <- result$p.value > alpha
result$statistic <- norm$statistic
result$statistic_name <- names(norm$statistic)
result
}
```

▼ Homocedasticity Test

💡 CUIDADO CON ESTO:

En R, el orden de las variables en la fórmula con el operador `~` es importante y afecta cómo se interpreta el análisis.

En el caso de todos los tests de homogeneidad (homocedasticity, no es exactamente lo mismo, pero Fernando lo usa indistintamente) el orden a seguir de los parámetros es el siguiente:

```
variable_dependiente ~ variable_independiente
```

En este contexto, la variable dependiente suele ser la variable numérica cuyas varianzas se están comparando, y la variable independiente suele ser la variable categórica que define los grupos.

▼ Just 2 groups

▼ Snedercor's F test, 2 normal distribution assured

It is **essentially a ratio of variances**

💡 This is the most basic and it is quite robust but it can only compare two groups, that will be a problem sometimes. The main assumption is that both samples must be normally distributed and it is very sensitive in this regard. **If you are not sure that the samples follow**
**a normal distribution this is not the test to use.**

$\sigma 1/\sigma 2$

1. **Hipótesis Nula (H0):** La hipótesis nula asume que las varianzas de las dos muestras son iguales.

2. **Hipótesis Alternativa (H1):** La hipótesis alternativa sugiere que las varianzas de las dos muestras son diferentes.

This is an example with 2 vectors

```
var.test(DATA$CONSUM ~ DATA$TRACTION)
F test to compare two variances
data: DATA$CONSUM by DATA$TRACTION
F = 2.4571, num df = 1979, denom df = 3959, p-val
2.2e-16
alternative hypothesis: true ratio of variances i
equal to 1
95 percent confidence interval:
2.277669 2.653298
sample estimates:
ratio of variances
2.45711
```

▼ 2 or + groups

▼ Levene's Test

This test will allow us to change the central tendency parameter from median (default) to mean or even trimmed mean. This will help with non-normally distributed samples.

> 💡 This test is not working for quantitative explanatory variables, as it is vs in mtcars for example. They are categoric NUMERIC variables.

**Leven's test is in the car package, which needs carData**, make sure you have it.

```
install.packages("car")
install.packages("carData")
```

```
library("carData")
library("car")
car::leveneTest(CONSUM ~ TRACTION, data = dDA)
```

Y nos devuelve esto

```
Levene's Test for Homogeneity of Variance (center =
median)
Df F value Pr(>F)
group 1 565.58 < 2.2e-16 ***
5938
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1
‘ ’ 1
```

Los resultados del test de Levene que obtuviste indican lo siguiente:

- **Df:** Grados de libertad. En este caso, hay 1 grado de libertad para la prueba.

  ▼ ¿Qué son los grados de libertad?

  Los grados de libertad (Df) son un concepto estadístico fundamental que se utiliza en varias pruebas y modelos estadísticos. En general, los grados de libertad representan el número de valores independientes que se pueden usar en un cálculo estadístico dado.

  En el contexto del test de Levene y muchas otras pruebas estadísticas, los grados de libertad se relacionan con el tamaño de las muestras involucradas en el análisis. Específicamente, en el test de Levene, los grados de libertad se calculan restando el número de grupos menos uno, es decir, Df = k - 1, donde "k" es el número de grupos o categorías que se están comparando.

- **F value:** Este valor representa la estadística de prueba del test de Levene. Cuanto mayor sea el valor de F, mayor será la evidencia en contra de la hipótesis nula de igualdad de

varianzas. En tu caso, el valor de F es bastante grande (565.58), lo que sugiere que hay una diferencia significativa entre las varianzas de las muestras.

> Cuánto mayor F, más rechazo de la H0

- **Pr(>F):** Este es el valor p asociado con la estadística de prueba F. En este caso, es extremadamente pequeño (< 2.2e-16), lo que indica que es esencialmente cero. Esto significa que hay una evidencia abrumadora en contra de la hipótesis nula de igualdad de varianzas.

> Cuánto menor Pr(>F), más rechazo de la H0

▼ Cómo hacerlo con más de 1 variable categórica:

```
leveneTest(variable ~ grupo1 + grupo2, data =
```

▼ Bartlett's Test

For testing for **equal variances among 2 or more groups** that are **not required to have the same sample size**. That will help with more

**heterogeneous samples.**

💡 USA ESTE TEST PARA 2 O MAS SAMPLES QUE SEAN NORMALES, si no son normales mejor usa Levene's. Si no son normales tp pasa nada, es porque es más poderoso para normales

```
bartlett.test(CONSUM ~ TRACTION, data = DATA)
```

```
Nos devuelve esto:

Bartlett test of homogeneity of variances

data:  CONSUM by TRACTION
Bartlett's K-squared = 571.25, df = 1, p-value
< 2.2e-16
```

- **Bartlett's K-squared:** Es la estadística de prueba de Bartlett, que mide la discrepancia entre las varianzas observadas y las varianzas esperadas si las varianzas fueran iguales entre los grupos. En este caso, el valor de K-squared es 571.25.
- **df:** Representa los grados de libertad asociados con la estadística de prueba de Bartlett. En este caso, hay 1 grado de libertad.
- **p-value:** Es el valor p asociado con la estadística de prueba de Bartlett. Es la probabilidad de observar los datos si la hipótesis nula de igualdad de varianzas fuera cierta. En este caso, el valor p es extremadamente pequeño, indicando que es esencialmente cero (p-value < 2.2e-16). Esto sugiere que hay evidencia significativa para rechazar la hipótesis nula de igualdad de varianzas entre los grupos.

▼ Normal samples, for checking if the means are equal

▼ Equal variances Case

We must use th **Student’s Test**

```
t.test(CONSUM ~ TRACTION, data = DATA,var.equal=T)
#LO DE var.equal = TRUE, asume igualdad de
#varianzas entre los grupos.
```

We will se p-value as always

▼ Not Equal variances

We must use the **Welch's Test**

```
t.test(CONSUM ~ TRACTION, data = DATA,var.equal=F)
#LO DE var.equal = FALSE, YA NO asume la igualdad de
#varianzas entre los grupos.
```

We will se p-value as always

▼ Paired Samples, when the samples are related

The samples are already dependent, therefore we do **not need to check homocedasticity**, but **normality is a must.**
As they are paired, there is always an before and after, THIS ORDER IS IMPORTANT. It matters for the formula. For example

```
before <- c(178.2077, 178.3437, 156.5270, 178.2530, 196.84
194.0826, 191.4065, 174.1255, 177.9355, 159.7474)
after <- c(90.75936, 86.58651, 87.40294, 96.61041, 109.491
 89.30793, 78.10475, 84.13723, 93.07256, 87.85651)
D <- data.frame(
  BEFORE = before,
  AFTER = after
)
t.test(before, after, data = DATA, paired = T)
```

We check again p_value

▼ Exercises

Using the database mtcars

ALWAYS

1. We check for normality
2. We check for homocedasticity between variances
3. We check for means

### ▼ 2.6.1 mpg vs vs

**vs is a variable that indicates the type of engine:**
**• 0: V-shaped engine**
**• 1: straight engine**
**Our hypothesis is that the miles per gallon that a car consumes is different with different**
**shaped engines**

So, we have an H0 = there is NO difference in the variance between both of the shaped engines. So we test for it:

```
mtcars -> mtcars_mod
mtcars_mod$vs <- ifelse(mtcars$vs == 0, "V-shaped engine",
car::leveneTest(mtcars_mod$mpg ~ mtcars_mod$vs)
```

Nos avisará de que se ha convertido a factor el vs para que se haga correctamente el test

Sacamos un Pr(>F) mayor de 0.05, luego aceptamos la H0

Una vez que sabemos que son iguales las varianzas hacemos el t.test con var.equal = T

```
t.test(mtcars_mod$mpg ~ mtcars_mod$vs,var.equal=T)
```

LA p nos da menos de 0.05 luego rechazamos que las medias sean iguales, luego SÍ hay diferencia.

#### ▼ 2.6.1.1 Cars V-Shaped and Automatic vs the rest

Como tenemos que comparar unas determinadas filas necesitamos subset, HAY 2 FORMAS

```
autyV <- mtcars[mtcars$vs == 0 & mtcars$am == 1, ]

# Filtrar todos los demás automóviles
rest<- mtcars[!(mtcars$vs == 0 & mtcars$am == 1), ]
```

```
#aquí cambié la label
amyv <-subset(mtcars_mod, vs == "V-shaped engine"  & am ==
rest <-subset(mtcars_mod, vs != "V-shaped engine"  | am !=
```

VALE , LO ANTERIOR NO SIRVE, NO SE HACE ASÍ, PERO LO DEJO PORQUE ES ÚTIL EN OTROS CASO

SOLUCIÓN BUENA:

Necesitamos crear otra variable dentro de mtcars CATEGÓRICA y separar y estudiar las varianzas como antes. Entonces:

```
# Se crea así
mtcars_mod$amyv <- ifelse(mtcars$vs == 0 & mtcars$am == 1,
# Realizar la prueba de Levene
car::leveneTest(mpg ~ amyv, data = mtcars_mod)
```

La p nos da por encima de 0.05 luego podemos decir que son iguales las varianzas, ahora t.test

```
t.test(mtcars_mod$mpg ~ mtcars_mod$amyv,var.equal=T)
```

La p nos da por encima de 0.05 luego podemos decir que sus medias son iguales y no hay diferencia, al revés que antes

▼ 2.6.1.2 Number of carburetors

Compare the mpg of the all cars vs those with:
• “Less” carburetors: defined as the 25% of the cars with the lowest amount of cylinders

• 4 or more carburetors

Para clasificar “Less” necesitamos dos cosas:

1. Primero, necesitas determinar el valor del cuartil 25% de la variable de interés (en este caso, la cantidad de cilindros).

```
quantile(mtcars$cyl, 0.25) # = 4
```

Y ahora podemos crear otro atributo que cumpla las dos condiciones

```
mtcars_mod$cyc <- ifelse(mtcars$cyl <= 4 & mtcars$carb
```

💡 EN CASO DE QUE LO PROBÉIS, vereis que todo cyc es Other, porque no se cumple nunca las dos condiciones. Porque SON DOS APARTADOS DISTINTOS. se hace por separado, pero bueno es to el rato igual.

1. Creas un nuevo atributo con las condiciones
2. Y con levene pruebas si sus varianzas son iguales
3. Ya sabiendolo se lo dices al t.test y compruebas la p para las medias

▼ BoxPlot Remember

```
install.packages("ggplot2") #Instalas el paquete

library(ggplot2) #Lo cargas

# Crear un data frame de ejemplo
data <- data.frame(grupo = rep(c("A", "B", "C"), each = 5
0),
                   valor = c(rnorm(50), rnorm(50, mean =
2), rnorm(50, mean = 3)))

# Crear un diagrama de caja y bigotes
#aes() = (aesthetic mapping)
ggplot(data, aes(x = grupo, y = valor)) + geom_boxplot()
```

▼ aes() y colour

Cuando se usa `colour` en `aes()`, ggplot2 crea un diagrama de caja y bigotes donde los colores de las cajas y los bigotes se corresponden con los valores de la variable `TRACTION`. Cada grupo único en la variable `TRACTION` obtendrá un color diferente en el gráfico.

Por ejemplo, si `TRACTION` tiene tres valores únicos (por ejemplo, "Rear", "Front", "4×4"), cada caja y bigote en el gráfico tendrá un color diferente para representar estos valores.