M2.851 - Tipología y ciclo de vida de los datos - Práctica 2
Betancor Sánchez, Manuel - Aula 2
Navalón Hernández, María Dolores - Aula 1

Práctica 2: Limpieza y validación de los datos.

Índice:

1. Descripción del dataset.

En esta práctica vamos a analizar el impacto que tiene la celebración de los Juegos Olímpicos en los países anfitriones a nivel deportivo, usando como referencia las medallas conseguidas, así como la influencia supranacional, evaluándose también a nivel de continente.

Para ello utilizaremos la información suministrada por la web kaggle, la cual recopila información relativa a los deportistas que han participado en los Juegos Olímpicos modernos. El enlace de descarga, en el cual se pueden obtener dos ficheros csv, es el siguiente:

https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results

Este repositorio de datos nos suministra el país de cada participante, pero no en que continente se encuentra, por lo que hemos tenido la necesidad de buscar una solución para este problema.

Para poder relacionar el país de los deportistas con el continente al que pertenece, vamos a complementar los datos con el dataset obtenido por el siguiente enlace:

https://www.kaggle.com/statchaitya/country-to-continent

Finalmente, necesitamos conocer el país, y el continente, de cada una de las ciudades anfitrionas, información que tampoco nos suministra los dos conjuntos de datos anteriores, por lo que para conocer qué país organiza cada edición de los Juegos Olímpicos hemos encontrado este otro dataset que relaciona cada ciudad sede con su país:

https://www.downloadexcelfiles.com/wo_en/download-excel-file-list-olympic-host-cities

La estructura de los diferentes ficheros obtenidos se puede obtener con el siguiente código:

```
# Revisamos la estructura de los dataframes para
# comprobar el tipo de datos de cada atributo
str(atletas)
sapply(atletas, function(x) class(x))
str(paises)
sapply(paises, function(x) class(x))
str(sedes)
sapply(sedes, function(x) class(x))
```

Estructura del fichero athlete_events.csv:

Columna	Descripción	Tipo
ID	Identificador único para cada atleta	ENTERO
Name	Nombre del atleta	TEXTO
Sex	Sexo. Valores M o F	TEXTO
Age	Edad	TEXTO
Height	Altura en centímetros	TEXTO
Weight	Peso en kilogramos	TEXTO
Team	Nombre del equipo	TEXTO
NOC	Código de 3 letras que identifica al Comité Olímpico del país	TEXTO
Games	Año y tipo de juegos	TEXTO
Year	Año de celebración	ENTERO
Season	Tipo de juego. Valores Summer (Verano) y Winter (invierno)	TEXTO
City	Ciudad organizadora de los juegos	TEXTO
Sport	Deporte	TEXTO
Event	Modalidad del deporte	TEXTO
Medal	Medalla conseguida. Valores Gold (Oro), Silver (Plata), Bronze (Bronce) o NA	TEXTO

Estructura del fichero countryContinent.csv:

Columna	Descripción	Tipo
country	Nombre del país	TEXTO
code_2	Código de país (2 letras)	TEXTO
code_3	Código del país (3 letras)	TEXTO
country_code	Código numérico del país	INTEGER

iso_3166_2	Código ISO 3166-2 del país	TEXTO
continent	Continente	TEXTO
sub_region	Región	TEXTO
region_code	Código numérico de la región	INTEGER
sub_region_code	Código numérico de la sub-región	INTEGER

Estructura del fichero list-host-cities-olympic-943j.csv:

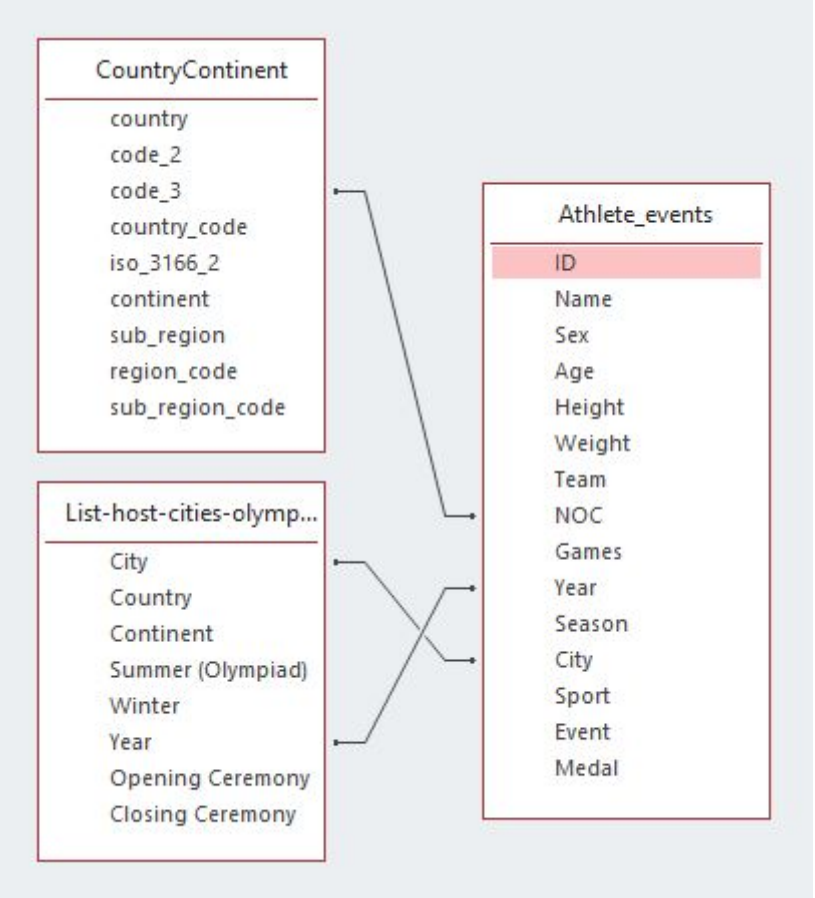
Columna	Descripción	Tipo
City	Nombre de la ciudad	TEXTO
Country	Nombre del país	TEXTO
Continent	Nombre del continente	TEXTO
Summer (Olympiad)	Edición de las Olimpiadas de verano	TEXTO
Winter	Edición de las Olimpiadas de invierno	TEXTO
Year	Año	INTEGER
Opening Ceremony	Fecha de la ceremonia de apertura	FECHA
Closing Ceremony	Fecha de la ceremonia de clausura	FECHA

Para facilitar la carga se han subido los tres conjuntos de datos a dropbox, importándolos directamente desde esta plataforma a través del siguiente código:

```
atletas <-
read.csv("https://www.dropbox.com/s/8ogvaf0ipu4raby/athlete_events.csv?dl=1",
encoding="utf-8")

paises <-
read.csv("https://www.dropbox.com/s/qol9t3g4z359img/countryContinent.csv?dl=1",
encoding="utf-8")

sedes <-
read.csv("https://www.dropbox.com/s/6fhwd3ydoeop30j/list-host-cities-olympic-943j
.csv?dl=1", encoding="utf-8")
```

2. Integración y selección de los datos de interés a analizar.

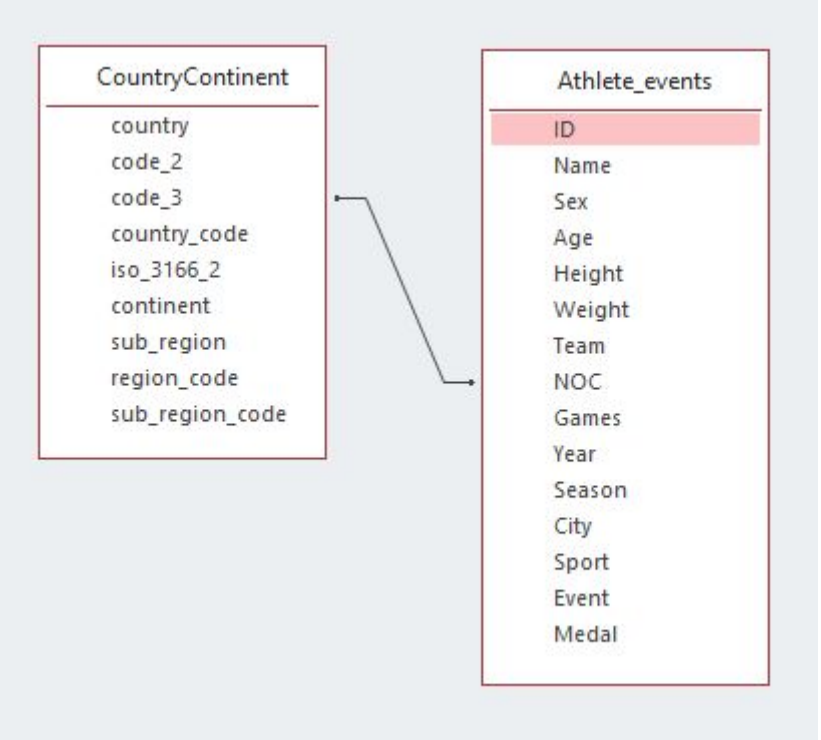
Las relaciones entre los diferentes datasets se puede resumir en el siguiente esquema:



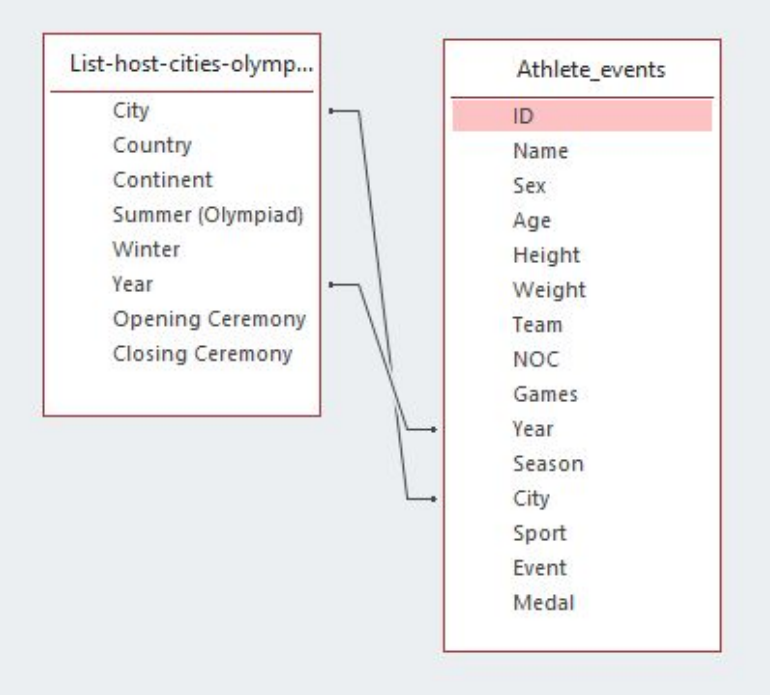
Analizando cada par de relaciones podemos comenzar con el país y continente de cada atleta, datos básicos para conocer el número de medallas obtenidas por cada país en los eventos, y que es la pieza fundamental de este trabajo.

Para conocer el país podemos relacionar el código del comité organizador (NOC) del csv de Athlete_events con la variable code_3 del csv CountryContinent. A través de este último dataset podemos ver tanto el país, en la variable country, como el continente, en la variable continent.

El siguiente gráfico refleja las relaciones citadas en los párrafos anteriores:



En relación con la ciudad anfitriona enlazamos el campo City del repositorio Athlete_events con el campo City del csv List-host-cities-olympic-943j, así mismo para seleccionar el evento concreto se enlazará también el año, uniendo los campos Year de ambos repositorios. Desde List-host-cities-olympic-943j podemos obtener tanto el nombre del país como el continente de la ciudad organizadora, utilizando los campos Country y Continent..



Los datos a utilizar por los siguientes ficheros en nuestro análisis se limitarán, centrándonos en el objetivo a conseguir, a los siguientes campos:

Fichero athlete_events.csv:

Columna	Descripción	Tipo
NOC	Código de 3 letras que identifica al Comité Olímpico del país	TEXTO
Year	Año de celebración	ENTERO
Season	Tipo de juego. Valores Summer (Verano) y Winter (invierno)	TEXTO
City	Ciudad organizadora de los juegos	TEXTO
Sport	Deporte	TEXTO
Event	Modalidad del deporte	TEXTO
Medal	Medalla conseguida. Valores Gold (Oro), Silver (Plata), Bronze (Bronce) o NA	TEXTO

Fichero countryContinent.csv:

Columna	Descripción	Tipo
country	Nombre del país	TEXTO
code_3	Código del país (3 letras)	TEXTO
continent	Continente	TEXTO
sub_region	Región	TEXTO

Fichero list-host-cities-olympic-943j.csv:

Columna	Descripción	Tipo
City	Nombre de la ciudad	TEXTO
Country	Nombre del país	TEXTO

Continent	Nombre del continente	TEXTO
Year	Año	INTEGER

El código utilizado para esta selección de columnas ha sido el siguiente:

```
atletas <- atletas[,c("NOC","Year","Season","City","Sport","Event","Medal")]
paises <- paises[,c("country","code_3","continent","sub_region")]
sedes <- sedes[,c("City","Country","Continent","Year")]
```

3. Limpieza de los datos.

3.1 Ceros y elementos vacíos.

En el dataframe de atletas lo primero que comprobamos es que hay muchos registros con NA en el campo de medallas. Esto se debe a que en el conjunto de datos se recoge información diversa de todos los participantes, hayan o no obtenido medalla en la competición.

No podemos considerar estos datos como perdidos, por lo que se ha optado por crear una nueva categoría para imputarla a todos los NA's.

```
atletas$Medal <- as.character(atletas$Medal)
atletas[is.na(atletas$Medal),"Medal"] = "Sin medalla"
atletas$Medal <- as.character(atletas$Medal)
```

Revisando también la historia de los juegos hemos descubierto que los juegos de 1906, celebrados en Atenas, no son reconocidos por el Comité Olímpico Internacional (COI) en la actualidad, razón por la cual no van a aparecer en el dataset sedes. Para evitar NA en la unión, y para realizar el análisis sólo de los juegos reconocidos se han eliminados los registros correspondientes a este año:

```
atletas <- atletas[atletas$Year != 1906,]
```

En la tabla de sedes también encontramos datos a limpiar, para empezar encontramos un caso grave dentro del csv list-host-cities-olympic-943j, ya que en los años en los que se han efectuado dos competiciones olímpicas, por coincidir invierno y verano en el mismo ejercicio, en el conjunto de datos sólo pone el año en el primer evento ocurrido, dejando como NA el posterior.

Siguiendo el manual de Calvo M., Subirats L. y Pérez D., en este tipo de circunstancias, cuando los datos son pocos y conocidos, lo mejor es una modificación manual.

En vez de realizar este cambio a mano hemos optado por solucionarlo a través de código, ya que después de analizar el dataset encontramos un patrón para la sustitución de valores que implementandolo con código corregía el error de una manera más simple y rápida.

El código desarrollado para la eliminación de los datos nulos es el siguiente:

```
for(i in 3:nrow(sedes)){
  if (is.na(sedes[i,"Year"])) {
       sedes[i,"Year"] <- sedes[i-1,"Year"]
  }
}
```

En el mismo conjunto de datos también pudimos ver que los tres últimos registros no disponían de información, eliminandolos también a través de código.

```
sedes <- sedes[1:(nrow(sedes)-3),]
```

La edición de Berlin de 1916 fueron los primeros Juegos Olímpicos cancelados de la historia, debido a los acontecimientos bélicos derivados de la Primera Guerra Mundial[1], por lo que eliminamos la información de la tabla sedes:

```
sedes <- sedes[sedes$Year != 1916,]
```

En el año 1940 se canceló los Juegos Olímpicos en Japón por motivos de la Segunda guerra sino-japonesa, transladándose a Helsinki como sede principal. Esta última finalmente también tuvo que renunciar debido al estallido de la Segunda Guerra Mundial, no celebrándose los juegos.[2]

```
sedes <- sedes[sedes$Year != 1940,]
```

Lo mismo ocurrió con los juegos que tendrían que haber tenido lugar en 1944, anulándose ambos por la Segunda Guerra Mundial.[3]

```
sedes <- sedes[sedes$Year != 1944,]
```

En el año 1956, debido a una cuarentena en el país anfitrión, las pruebas de equitación de Melbourne se realizaron en Estocolmo. En el dataset de sedes aparecen en una única entrada, necesitando una por cada ciudad para evitar NA cuando hagamos la unión. Con el siguiente código eliminamos la entrada incorrecta y generamos los dos nuevos registros:

```
sedes <- sedes[sedes$City!="Melbourne\n Stockholm",]
sedes <- add_row(sedes,
City="Melbourne",Country="Australia",Continent="Oceania",Year=1956) %>%
  add_row(City="Stockholm",Country="Sweden",Continent="Europe",Year=1956)
```

[1] https://es.wikipedia.org/wiki/Juegos_Ol%C3%ADmpicos_de_Berl%C3%ADn_1916
[2] https://es.wikipedia.org/wiki/Juegos_Ol%C3%ADmpicos_de_Verano_de_1940
[3] https://es.wikipedia.org/wiki/Juegos_Ol%C3%ADmpicos_de_Londres_1944

Revisando comprobamos también que en los datos importados se incluyen sedes que todavía no han sido escogidas, identificándose como “TBD” en el conjunto de datos. Para tener una mayor limpieza de los mismos hemos procedido a eliminar también estos registros:

```
sedes <- sedes[sedes$City != "TBD",]
```

En la tabla sedes además hay que modificar ciertos países para que en la unión no de problemas:

```
sedes$Country <- as.character(sedes$Country)
sedes[sedes$Country == "United States", "Country"] = "USA"
sedes[sedes$Country == "United Kingdom", "Country"] = "UK"
sedes[sedes$Country == "Nazi Germany", "Country"] = "Germany"
sedes[sedes$Country == "West Germany", "Country"] = "Germany"
sedes[sedes$Country == "Soviet Union", "Country"] = "Russia"
sedes$Country <- as.factor(sedes$Country)
```

De igual manera tenemos que unificar el continente americano, al tenerlo el dataset dividido por regiones:

```
sedes$Continent <- as.character(sedes$Continent)
sedes[sedes$Continent == "North America", "Continent"] = "Americas"
sedes[sedes$Continent == "South America", "Continent"] = "Americas"
sedes$Continent <- as.factor(sedes$Continent)
```

Hay ciertos nombres de ciudades que no coinciden con los de los atletas, esto es debido al idioma utilizado en cada dataset, ya que en el de sedes todo está en inglés no ocurriendo lo mismo en el de atletas:

```
sedes$City <- as.character(sedes$City)
sedes[sedes$City == "Rome", "City"] = "Roma"
sedes[sedes$City == "Athens", "City"] = "Athina"
sedes[sedes$City == "Antwerp", "City"] = "Antwerpen"
sedes[sedes$City == "St. Moritz", "City"] = "Sankt Moritz"
sedes[sedes$City == "Moscow", "City"] = "Moskva"
sedes[sedes$City == "Turin", "City"] = "Torino"
sedes$City <- as.factor(sedes$City)
```

En los datos de países también hemos encontrado datos nulos, en concreto hemos encontrado problemas en la asignación de continentes y subregion en países relacionados con la Antártida y en algunas islas de ultramar oceánicas:

```
paises$continent <- as.character(paises$continent)
paises$sub_region <- as.character(paises$sub_region)
paises[paises$NOC=="ATA","continent"] = "Antarctica"
```

```
paises[paises$NOC=="ATA","sub_region"] = "Antarctica"
paises[paises$NOC=="BVT","continent"] = "Antarctica"
paises[paises$NOC=="BVT","sub_region"] = "Antarctica"
paises[paises$continent=="","continent"] = "Oceania"
paises[paises$sub_region=="","sub_region"] = "Others"
paises$continent <- as.factor(paises$continent)
paises$sub_region <- as.factor(paises$sub_region)
```

Además hay un país en atletas que no se encuentra en la tabla países, por lo que lo hemos incorporado para evitar datos nulos en la unión:

```
paises <- paises %>%
  add_row(country="Kosovo",NOC="XKX",continent="Europe",sub_region="Southern
Europe")
```

Analizando los diferentes dataset también podemos ver que no coinciden los códigos, en algunos países, de los Comités Olímpicos Organizadores (NOC) y de los países.

Por ejemplo, en Alemania el NOC es GER, mientras que el código code_3 del país es DEU.

Es por ello que para solucionar este problema hemos generado nosotros un diccionario que relaciona ambos códigos.

```
diccionario <-
read.csv("https://www.dropbox.com/s/gibjji4okfhl0ru/diccionario.csv?dl=1",
encoding="utf-8")
```

Gracias a este diccionario, y a las correcciones anteriormente referenciadas, hemos podido generar un sólo conjunto de datos, sin datos nulos, que nos va a permitir realizar el análisis estadístico:

```
names(paises) <- c("country","NOC","continent","sub_region")
diccionario <- diccionario[,c("NOC","code_3")]
names(diccionario) <- c("NOC","code_3")
names(sedes) <- c("City","Country_host","Continent_host","Year")

df <- merge(atletas, diccionario, by = "NOC", all.x=TRUE) %>%
merge(paises, by = "code_3", by.y = "NOC", all.x=TRUE) %>%
merge(sedes, by = c("City","Year"), all.x=TRUE)
```

Aún teniendo en cuenta todo lo anterior, el dataframe obtenido disponía de datos nulos generados a partir de la unión anterior. Los datos corresponden a los códigos EUN, que identifica la unión de diferentes países en un sólo equipo, y también IOA, que identifica los atletas olímpicos que actúan sin representar a ningún país. Como en ambos casos no nos sería de utilidad para el análisis estadístico buscado los eliminamos:

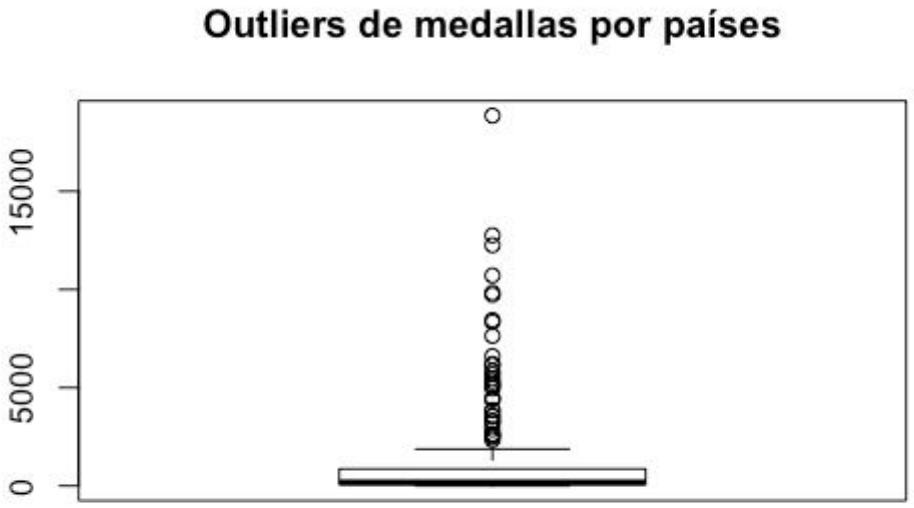
```
df <- df[df$NOC!="EUN",]
df <- df[df$NOC!="IOA",]
```

3.2 Identificación de valores extremos

Como el dato final a analizar son las medallas conseguidas por cada país, para la detección de los valores extremos agrupamos por este campo y representamos los datos mediantes gráficos de cajas (boxplots):

```
by_NOC <- atletas %>% group_by(NOC) %>% summarise(count=n())
boxplot(by_NOC$count)
boxplot.stats(by_NOC$count)$out
```

Teniendo los siguientes resultados:



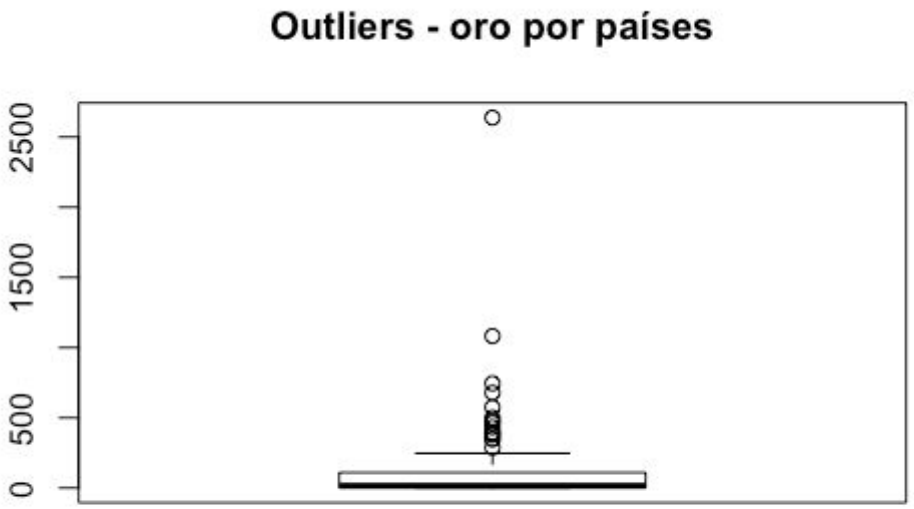

```
> boxplot.stats(by_NOC$count)$out
 [1]  3297  7638  5141  3857  3848  3530  9733  5141  2479  3570  5313  5467
12758  3315 12256
[16]  2645  9830  3181  6607 10715  8444  4464  2880  5839  4960  2342  6207
4405  5143  6150
[31]  8339  4404  2559  5685 18853  2583
```

Existen valores que sobresalen y que podrían considerarse como outliers, pero que en realidad no lo son. Son medallas obtenidas por países como Estados Unidos o Rusia, los cuales sobresalen en todas las olimpiadas en relación con el resto, razón por la cual tenemos estos valores fuera del rango común.

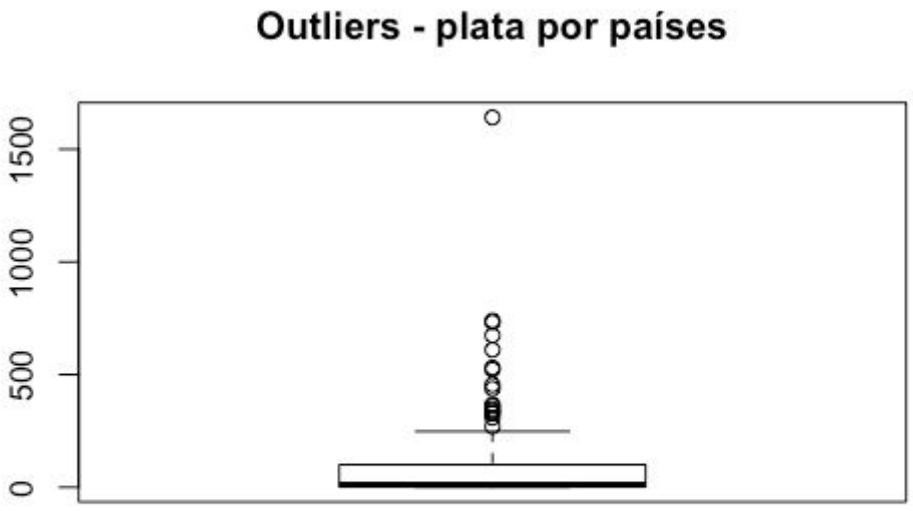
Hemos hecho también la representación por tipología de medallas:

```
by_NOC_Medal <- atletas %>% group_by(NOC, Medal) %>% summarise(count=n())
by_NOC_Medal_Gold <- by_NOC_Medal %>% filter(Medal == "Gold")
by_NOC_Medal_Silver <- by_NOC_Medal %>% filter(Medal == "Silver")
by_NOC_Medal_Bronze <- by_NOC_Medal %>% filter(Medal == "Bronze")

boxplot(by_NOC_Medal_Gold$count)
boxplot.stats(by_NOC_Medal_Gold$count)$out

boxplot(by_NOC_Medal_Silver$count)
boxplot.stats(by_NOC_Medal_Silver$count)$out

boxplot(by_NOC_Medal_Bronze$count)
boxplot.stats(by_NOC_Medal_Bronze$count)$out
```

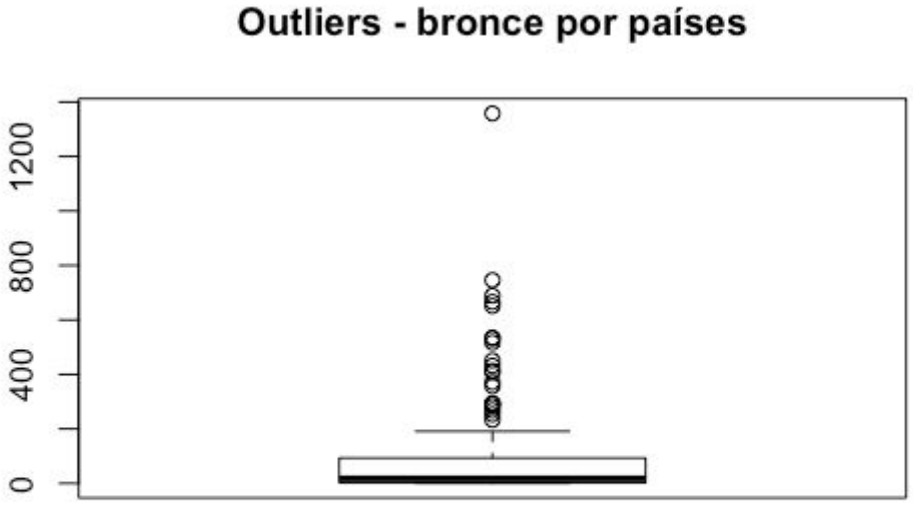
Obteniendo resultados equivalentes a los anteriores:




```
> boxplot.stats(by_NOC_Medal_Gold$count)$out
 [1]  348  463  350  501  678  397  745  432  575  287  378  390  479 1082 2638
```




```
> boxplot.stats(by_NOC_Medal_Silver$count)$out
 [1]  455  438  347  270  610  739  327  674  332  531  309  340  361  367  522
732 1641
```



```
> boxplot.stats(by_NOC_Medal_Bronze$count)$out
 [1]  517  451  292  432  666  233  651  281  746  371  531  357  413  294  253
292  408  268  535
[20]  689 1358
```

4. Análisis de los datos.

4.1 Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

Para realizar el análisis propuesto se van a generar dos dataframe de datos agrupados, por lo que tendremos los análisis por dos bloques de grupos diferentes, por países y por continentes.

En el primer caso, el dataframe de países, agrupamos el conjunto de datos obtenidos por la unión de los diferentes elementos a través de los valores country, City, Year y sedePais, y obtenemos a través de la función summarise de la librería dplyr los totales obtenidos de las diferentes medallas y el total de medallas obtenidas por cada agrupación.

A través de la unión de City y de Year obtenemos un evento olímpico concreto, country nos mostrará cada país participante, y sedePais nos indica, a través de los valores 0 y 1, si el país participante fue el país anfitrión en ese evento en concreto.

El código a utilizar para la generación de esta selección ha sido el siguiente:

```
df_pais <- df %>% group_by(country, City, Year, sedePais) %>%
  summarise(T_Gold=sum(Gold), T_Silver=sum(Silver), T_Bronze=sum(Bronze),
T_Medal=(sum(Gold)+sum(Silver)+sum(Bronze)))
```

En la segunda selección realizada, el dataframe de continentes, agrupamos el conjunto de datos obtenidos por la unión de los diferentes elementos a través de los valores continent, City, Year y sedeContinente, uniendo las medallas, por tipo, obtenidas por continente y evento, así como el total de medallas obtenidas.

A través de la unión de City y de Year, como en el caso anterior, obtenemos un evento olímpico concreto, continent nos mostrará cada continente participante, y sedeContinente nos indica, a través de los valores 0 y 1, si el continente participante fue el continente donde se desarrolló el evento en concreto.

El código a utilizar para la generación de esta selección ha sido el siguiente:

```
df_continente <- df %>% group_by(continent, City, Year, sedeContinente) %>%
  summarise(T_Gold=sum(Gold), T_Silver=sum(Silver), T_Bronze=sum(Bronze),
T_Medal=(sum(Gold)+sum(Silver)+sum(Bronze)))
```

4.2 Comprobación de la normalidad y homogeneidad de la varianza.

Para el contraste de la normalidad hemos utilizado el test de Shapiro-Wilk, en ambos dataset en las variables que utilizaremos en la correlación a través del siguiente código:

```
shapiro.test(df_pais$sedePais)
shapiro.test(df_pais$T_Gold)
shapiro.test(df_pais$T_Silver)
shapiro.test(df_pais$T_Bronze)
shapiro.test(df_pais$T_Medal)
shapiro.test(df_continente$sedeContinente)
shapiro.test(df_continente$T_Gold)
shapiro.test(df_continente$T_Silver)
shapiro.test(df_continente$T_Bronze)
shapiro.test(df_continente$T_Medal)
```

Los valores obtenidos han sido los siguientes:

```
> shapiro.test(df_pais$sedePais)

    Shapiro-Wilk normality test

data:  df_pais$sedePais
W = 0.069508, p-value < 2.2e-16

> shapiro.test(df_pais$T_Gold)

    Shapiro-Wilk normality test

data:  df_pais$T_Gold
W = 0.28554, p-value < 2.2e-16

> shapiro.test(df_pais$T_Silver)

    Shapiro-Wilk normality test

data:  df_pais$T_Silver
W = 0.36718, p-value < 2.2e-16

> shapiro.test(df_pais$T_Bronze)

    Shapiro-Wilk normality test

data:  df_pais$T_Bronze
W = 0.39261, p-value < 2.2e-16

> shapiro.test(df_pais$T_Medal)

    Shapiro-Wilk normality test

data:  df_pais$T_Medal
W = 0.36601, p-value < 2.2e-16
```

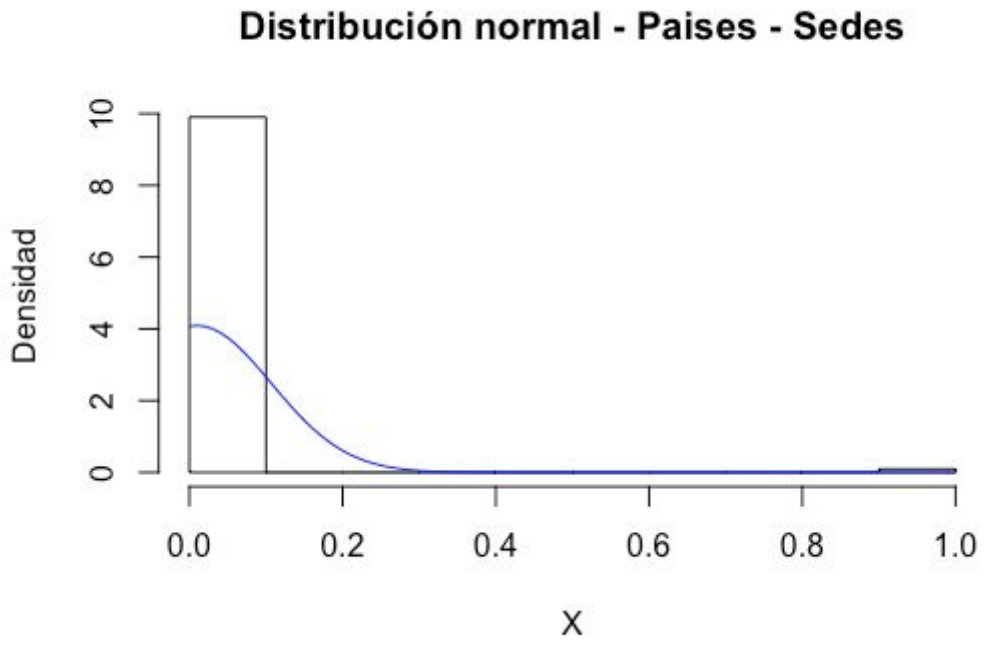
```
> shapiro.test(df_continente$sedeContinente)

    Shapiro-Wilk normality test

data:  df_continente$sedeContinente
W = 0.49758, p-value < 2.2e-16

> shapiro.test(df_continente$T_Gold)

    Shapiro-Wilk normality test

data:  df_continente$T_Gold
W = 0.6896, p-value < 2.2e-16

> shapiro.test(df_continente$T_Silver)

    Shapiro-Wilk normality test

data:  df_continente$T_Silver
W = 0.65125, p-value < 2.2e-16

> shapiro.test(df_continente$T_Bronze)

    Shapiro-Wilk normality test

data:  df_continente$T_Bronze
W = 0.65888, p-value < 2.2e-16

> shapiro.test(df_continente$T_Medal)

    Shapiro-Wilk normality test

data:  df_continente$T_Medal
W = 0.6696, p-value < 2.2e-16
```

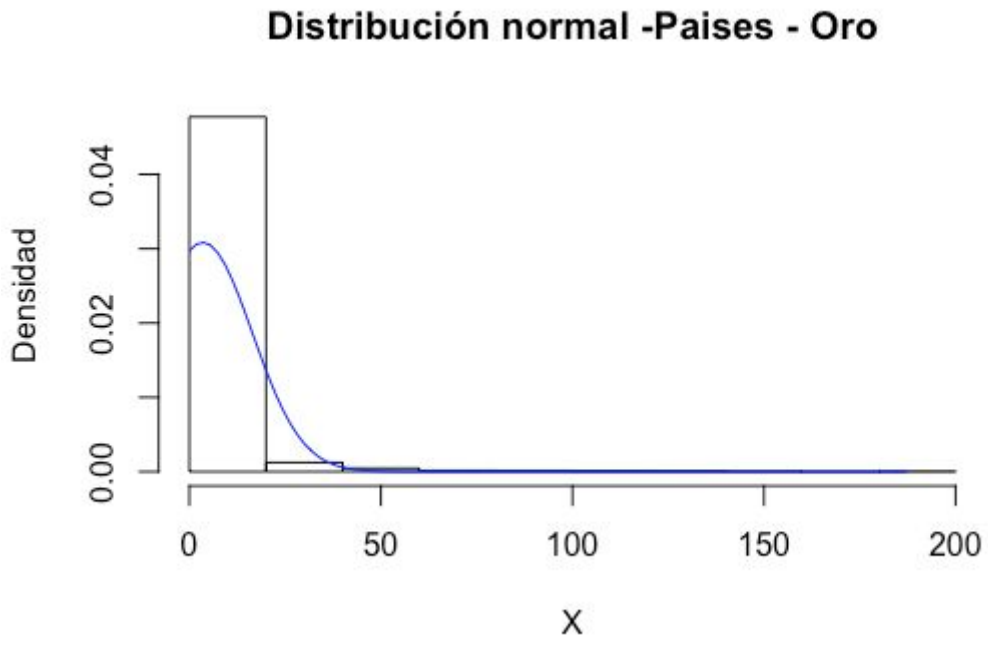
La representación gráfica en relación a una distribución normal la obtendremos con el siguiente código:

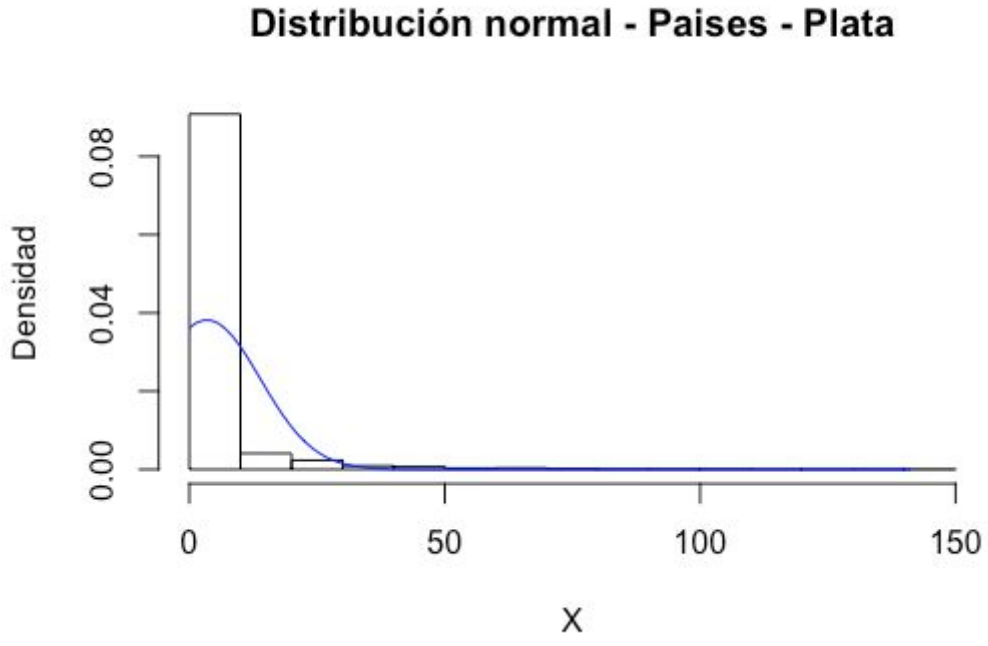
```
plotn <- function(x,main="Histograma de frecuencias \ny distribución normal",
                  xlab="X",ylab="Densidad") {
  min <- min(x)
  max <- max(x)
  media <- mean(x)
  dt <- sd(x)
  hist(x,freq=F,main=main,xlab=xlab,ylab=ylab)
  curve(dnorm(x,media,dt), min, max,add = T,col="blue")
}

plotn(df_pais$sedePais, main="Distribución normal - Paises - Sedes")
plotn(df_pais$T_Gold, main="Distribución normal -Paises - Oro")
plotn(df_pais$T_Silver, main="Distribución normal - Paises - Plata")
plotn(df_pais$T_Bronze, main="Distribución normal - Paises - Bronce")
plotn(df_pais$T_Medal, main="Distribución normal - Paises - Medallas")
plotn(df_continente$sedeContinente, main="Distribución normal - Continentes -
Sedes")
plotn(df_continente$T_Gold, main="Distribución normal - Continentes - Oro")
```

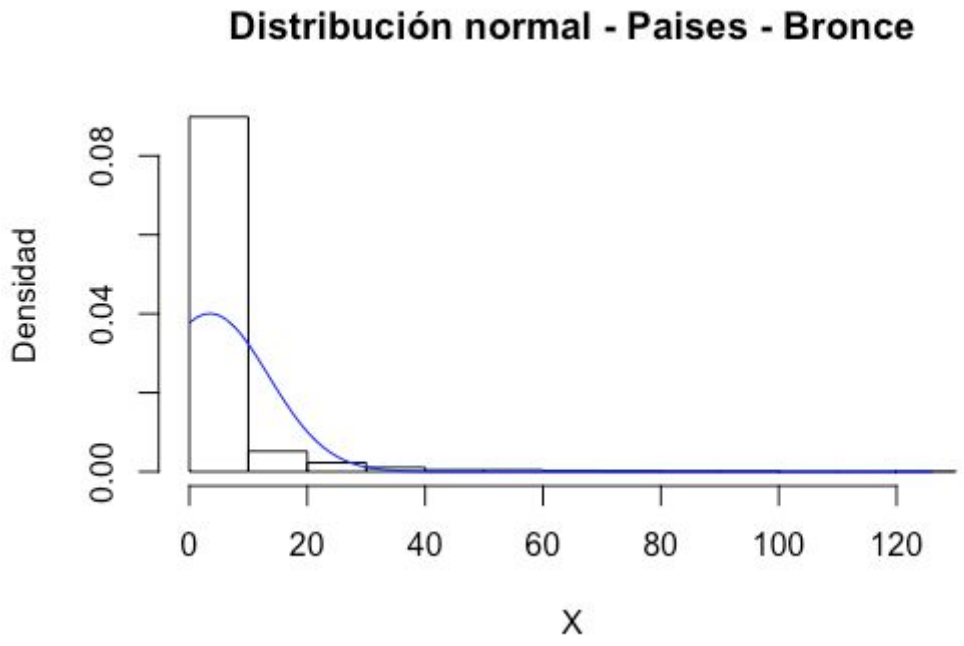
```
plotn(df_continente$T_Silver, main="Distribución normal - Continentes - Plata")
plotn(df_continente$T_Bronze, main="Distribución normal - Continentes - Bronce")
plotn(df_continente$T_Medal, main="Distribución normal - Continentes - Medallas")
```

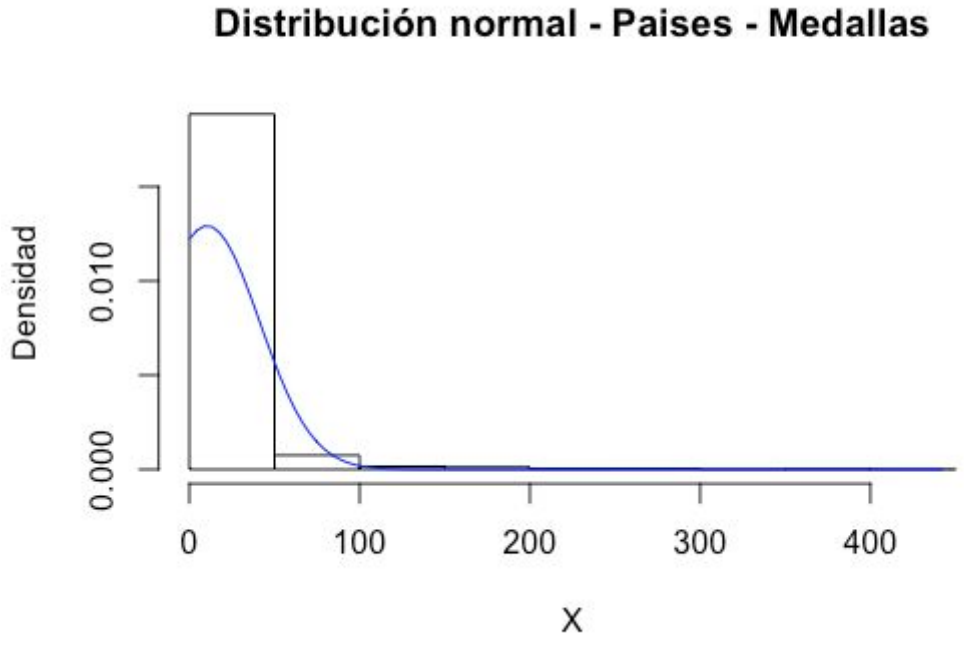
La cual da las siguientes salidas gráficas:

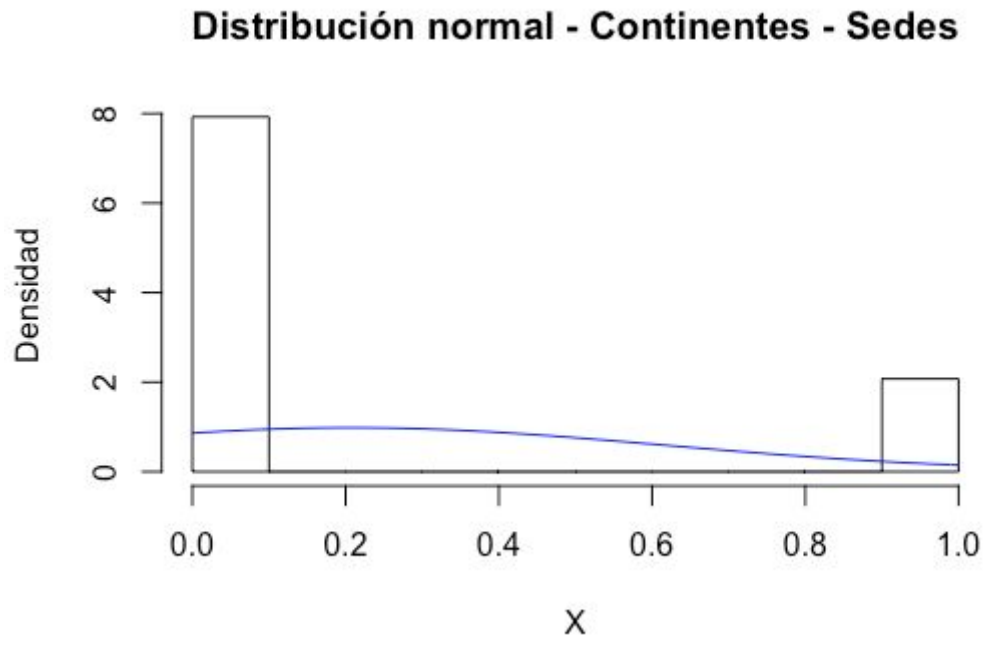


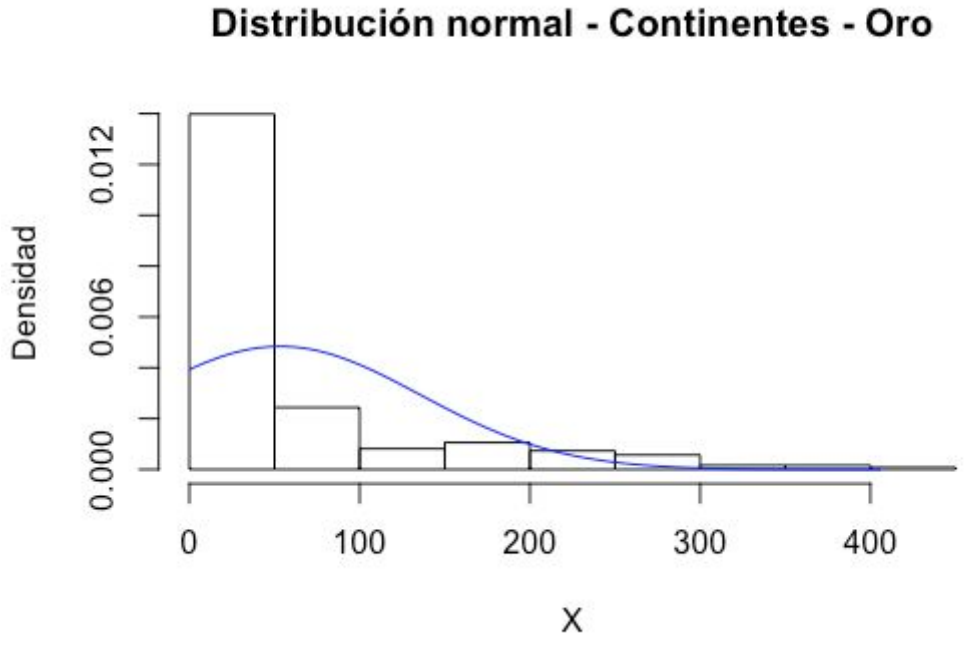

Distribución normal -Paises - Oro
Densidad
0.00
0.02
0.04
0
50
100
150
200
X

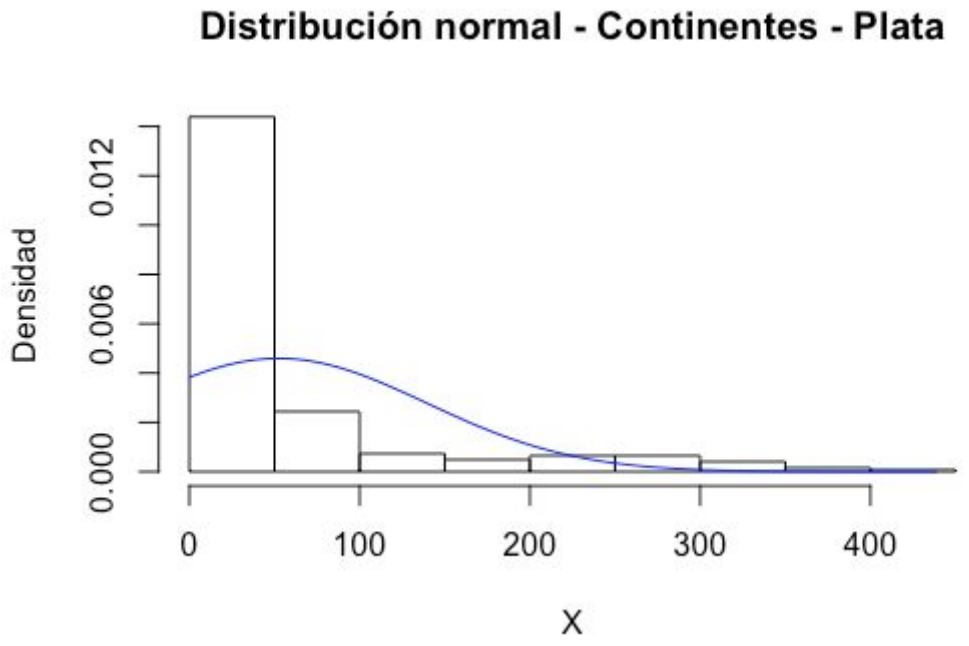

Distribución normal - Paises - Plata
Densidad
0.00
0.04
0.08
0
50
100
150
X

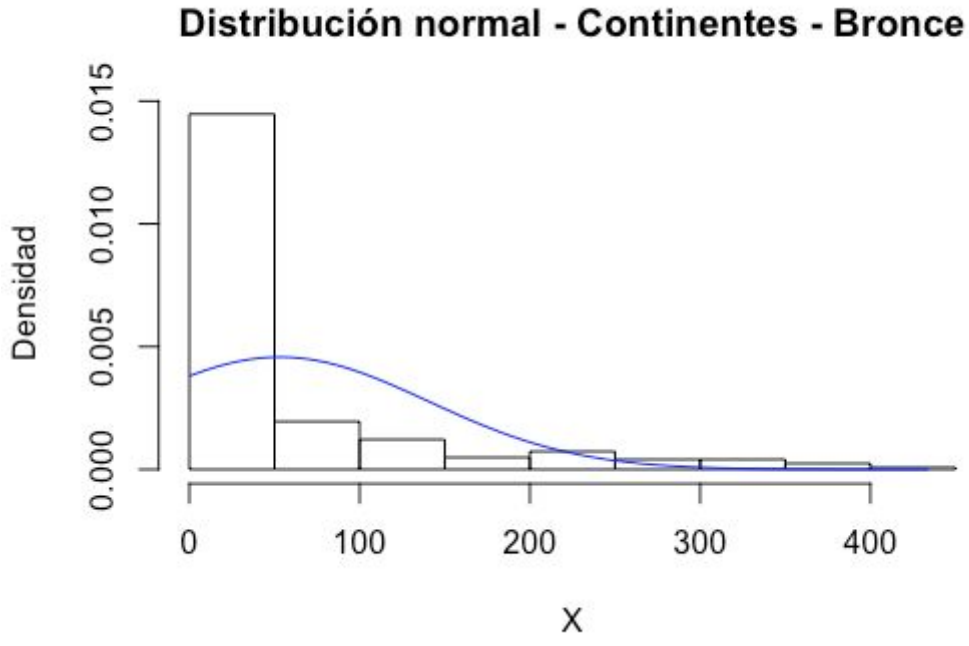

Distribución normal - Paises - Bronce
Densidad
0.00
0.04
0.08
0
20
40
60
80
100
120
X

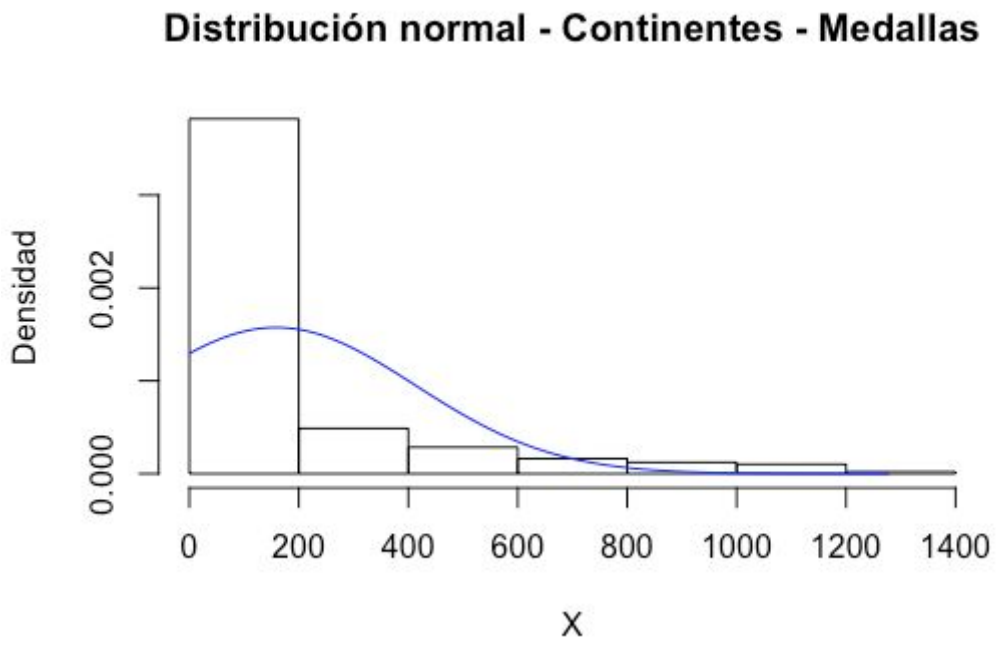

Distribución normal - Paises - Medallas
Densidad
0.000
0.010
0
100
200
300
400
X


Distribución normal - Continentes - Sedes
Densidad
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0
X


Distribución normal - Continentes - Oro
Densidad
0.000
0.006
0.012
0
100
200
300
400
X


Distribución normal - Continentes - Plata
Densidad
0.000
0.006
0.012
0
100
200
300
400
X


Distribución normal - Continentes - Bronce
Densidad
0.000
0.005
0.010
0.015
0
100
200
300
400
X



En todos los casos vemos que no sigue una distribución normal, por lo que hemos decidido utilizar utilizar el coeficiente de correlación de Spearman para nuestros cálculos estadísticos.

4.3 Aplicación de pruebas estadísticas para comparar los grupos de datos.

Antes de realizar las pruebas de correlación con el coeficiente de correlación de Spearman hemos realizado unas pruebas estadísticas preliminares de carácter descriptivo:

```
# Ranking de países por participación de atletas
summary(df$country)

# Total de medallas de cada tipo y sus porcentajes
CrossTable(df$Medal)

# Resultados acumulados absolutos de todos los juegos por país
table(df$Medal, df$country)
# Resultados acumulados en porcentaje de todos los juegos por país
CrossTable(df$Medal, df$country)

# Medallas acumuladas de oro en todos los juegos por país
tapply(df$Gold, df$country, sum)
# Medallas acumuladas de plata en todos los juegos por país
tapply(df$Silver, df$country, sum)
# Medallas acumuladas de bronce en todos los juegos por país
tapply(df$Bronze, df$country, sum)

# Calculamos la moda del atributo "Medal" (tipo de medalla) para todo el conjunto
mlv(df$Medal, na.rm=TRUE)

# Calculamos la moda del atributo "Medal" (tipo de medalla) para cada país
tapply(df$Medal, df$country, mlv, na.rm=TRUE)

# Calculamos la moda del par país-medalla
apply(df[,c("country", "Medal")], 2, mlv,  method = "mfv", na.rm=TRUE)
```

A continuación, para conocer realmente si existe relación en ser sede con un aumento del número de medallas, hemos realizado la correlación anteriormente referenciada:

```
# Usamos el coeficiente de correlación de Spearman
cor(x=df_pais$sedePais, y=df_pais$T_Gold, method = "spearman")
cor(x=df_pais$sedePais, y=df_pais$T_Silver, method = "spearman")
cor(x=df_pais$sedePais, y=df_pais$T_Bronze, method = "spearman")
cor(x=df_pais$sedePais, y=df_pais$T_Medal, method = "spearman")
cor(x=df_continente$sedeContinente, y=df_continente$T_Gold, method = "spearman")
cor(x=df_continente$sedeContinente, y=df_continente$T_Silver, method =
"spearman")
cor(x=df_continente$sedeContinente, y=df_continente$T_Bronze, method =
"spearman")
cor(x=df_continente$sedeContinente, y=df_continente$T_Medal, method = "spearman")

# Significancia de la correlación de Spearman
cor.test(x=df_pais$sedePais, y=df_pais$T_Gold, conf.level  = 0.95, method =
"spearman")
cor.test(x=df_pais$sedePais, y=df_pais$T_Silver, conf.level  = 0.95, method =
```

```
"spearman")
cor.test(x=df_pais$sedePais, y=df_pais$T_Bronze, conf.level  = 0.95, method =
"spearman")
cor.test(x=df_pais$sedePais, y=df_pais$T_Medal, conf.level  = 0.95, method =
"spearman")
cor.test(x=df_continente$sedeContinente, y=df_continente$T_Gold, conf.level  =
0.95, method = "spearman")
cor.test(x=df_continente$sedeContinente, y=df_continente$T_Silver, conf.level  =
0.95, method = "spearman")
cor.test(x=df_continente$sedeContinente, y=df_continente$T_Bronze, conf.level  =
0.95, method = "spearman")
cor.test(x=df_continente$sedeContinente, y=df_continente$T_Medal, conf.level  =
0.95, method = "spearman")
```

Los valores obtenidos han sido los siguientes:

```
> cor(x=df_pais$sedePais, y=df_pais$T_Gold, method = "spearman")
[1] 0.152063
> cor(x=df_pais$sedePais, y=df_pais$T_Silver, method = "spearman")
[1] 0.1375267
> cor(x=df_pais$sedePais, y=df_pais$T_Bronze, method = "spearman")
[1] 0.138838
> cor(x=df_pais$sedePais, y=df_pais$T_Medal, method = "spearman")
[1] 0.1471185
> cor(x=df_continente$sedeContinente, y=df_continente$T_Gold, method =
"spearman")
[1] 0.4648288
> cor(x=df_continente$sedeContinente, y=df_continente$T_Silver, method =
"spearman")
[1] 0.4637042
> cor(x=df_continente$sedeContinente, y=df_continente$T_Bronze, method =
"spearman")
[1] 0.4466356
> cor(x=df_continente$sedeContinente, y=df_continente$T_Medal, method =
"spearman")
[1] 0.4613035
> cor.test(x=df_pais$sedePais, y=df_pais$T_Gold, conf.level  = 0.95, method =
"spearman")

    Spearman's rank correlation rho

data:  df_pais$sedePais and df_pais$T_Gold
S = 7476444641, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
       rho
0.152063

Warning message:
In cor.test.default(x = df_pais$sedePais, y = df_pais$T_Gold, conf.level = 0.95,
:
  Cannot compute exact p-value with ties
> cor.test(x=df_pais$sedePais, y=df_pais$T_Silver, conf.level  = 0.95, method =
"spearman")

    Spearman's rank correlation rho

data:  df_pais$sedePais and df_pais$T_Silver
```

```
S = 7604614354, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.1375267

Warning message:
In cor.test.default(x = df_pais$sedePais, y = df_pais$T_Silver,  :
  Cannot compute exact p-value with ties
> cor.test(x=df_pais$sedePais, y=df_pais$T_Bronze, conf.level  = 0.95, method =
"spearman")

    Spearman's rank correlation rho

data:  df_pais$sedePais and df_pais$T_Bronze
S = 7593052192, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.138838

Warning message:
In cor.test.default(x = df_pais$sedePais, y = df_pais$T_Bronze,  :
  Cannot compute exact p-value with ties
> cor.test(x=df_pais$sedePais, y=df_pais$T_Medal, conf.level  = 0.95, method =
"spearman")

    Spearman's rank correlation rho

data:  df_pais$sedePais and df_pais$T_Medal
S = 7.52e+09, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.1471185

Warning message:
In cor.test.default(x = df_pais$sedePais, y = df_pais$T_Medal, conf.level = 0.95,
:
  Cannot compute exact p-value with ties
> cor.test(x=df_continente$sedeContinente, y=df_continente$T_Gold, conf.level  =
0.95, method = "spearman")

    Spearman's rank correlation rho

data:  df_continente$sedeContinente and df_continente$T_Gold
S = 1327821, p-value = 1.369e-14
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4648288

Warning message:
In cor.test.default(x = df_continente$sedeContinente, y = df_continente$T_Gold,
:
  Cannot compute exact p-value with ties
> cor.test(x=df_continente$sedeContinente, y=df_continente$T_Silver, conf.level
= 0.95, method = "spearman")

    Spearman's rank correlation rho
```

```
data:  df_continente$sedeContinente and df_continente$T_Silver
S = 1330612, p-value = 1.614e-14
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4637042

Warning message:
In cor.test.default(x = df_continente$sedeContinente, y = df_continente$T_Silver,
:
  Cannot compute exact p-value with ties
> cor.test(x=df_continente$sedeContinente, y=df_continente$T_Bronze, conf.level
= 0.95, method = "spearman")

    Spearman's rank correlation rho

data:  df_continente$sedeContinente and df_continente$T_Bronze
S = 1372961, p-value = 1.829e-13
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4466356

Warning message:
In cor.test.default(x = df_continente$sedeContinente, y = df_continente$T_Bronze,
:
  Cannot compute exact p-value with ties
> cor.test(x=df_continente$sedeContinente, y=df_continente$T_Medal, conf.level  =
0.95, method = "spearman")

    Spearman's rank correlation rho

data:  df_continente$sedeContinente and df_continente$T_Medal
S = 1336568, p-value = 2.29e-14
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4613035

Warning message:
In cor.test.default(x = df_continente$sedeContinente, y = df_continente$T_Medal,
:
  Cannot compute exact p-value with ties
```

Con estos valores se ve claramente que no existe una correlación ni por países ni por continente, por lo que queda demostrado que la hipótesis a demostrar no es cierta.

Aún así se decidió utilizar otro análisis de correlación que use técnicas no paramétricas, eligiendo el coeficiente de correlación de Kendall:

```
cor(x=df_pais$sedePais, y=df_pais$T_Gold, method = "kendall")
cor(x=df_pais$sedePais, y=df_pais$T_Silver, method = "kendall")
cor(x=df_pais$sedePais, y=df_pais$T_Bronze, method = "kendall")
cor(x=df_pais$sedePais, y=df_pais$T_Medal, method = "kendall")
cor(x=df_continente$sedeContinente, y=df_continente$T_Gold, method = "kendall")
cor(x=df_continente$sedeContinente, y=df_continente$T_Silver, method = "kendall")
cor(x=df_continente$sedeContinente, y=df_continente$T_Bronze, method = "kendall")
```

```
cor(x=df_continente$sedeContinente, y=df_continente$T_Medal, method = "kendall")
```

El cual dió los siguientes valores:

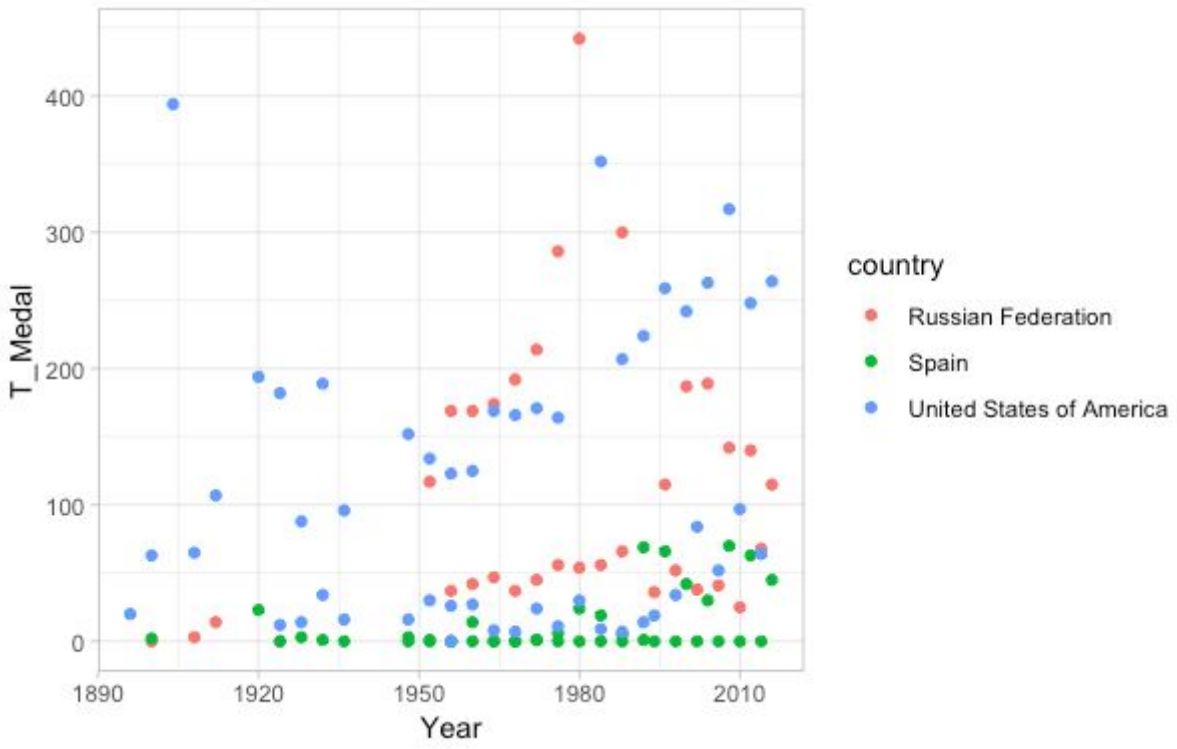
```
> cor(x=df_pais$sedePais, y=df_pais$T_Gold, method = "kendall")
[1] 0.142396
> cor(x=df_pais$sedePais, y=df_pais$T_Silver, method = "kendall")
[1] 0.1278562
> cor(x=df_pais$sedePais, y=df_pais$T_Bronze, method = "kendall")
[1] 0.1284468
> cor(x=df_pais$sedePais, y=df_pais$T_Medal, method = "kendall")
[1] 0.1327635
> cor(x=df_continente$sedeContinente, y=df_continente$T_Gold, method = "kendall")
[1] 0.3910405
> cor(x=df_continente$sedeContinente, y=df_continente$T_Silver, method =
"kendall")
[1] 0.3898538
> cor(x=df_continente$sedeContinente, y=df_continente$T_Bronze, method =
"kendall")
[1] 0.3754006
> cor(x=df_continente$sedeContinente, y=df_continente$T_Medal, method =
"kendall")
[1] 0.3835265
```

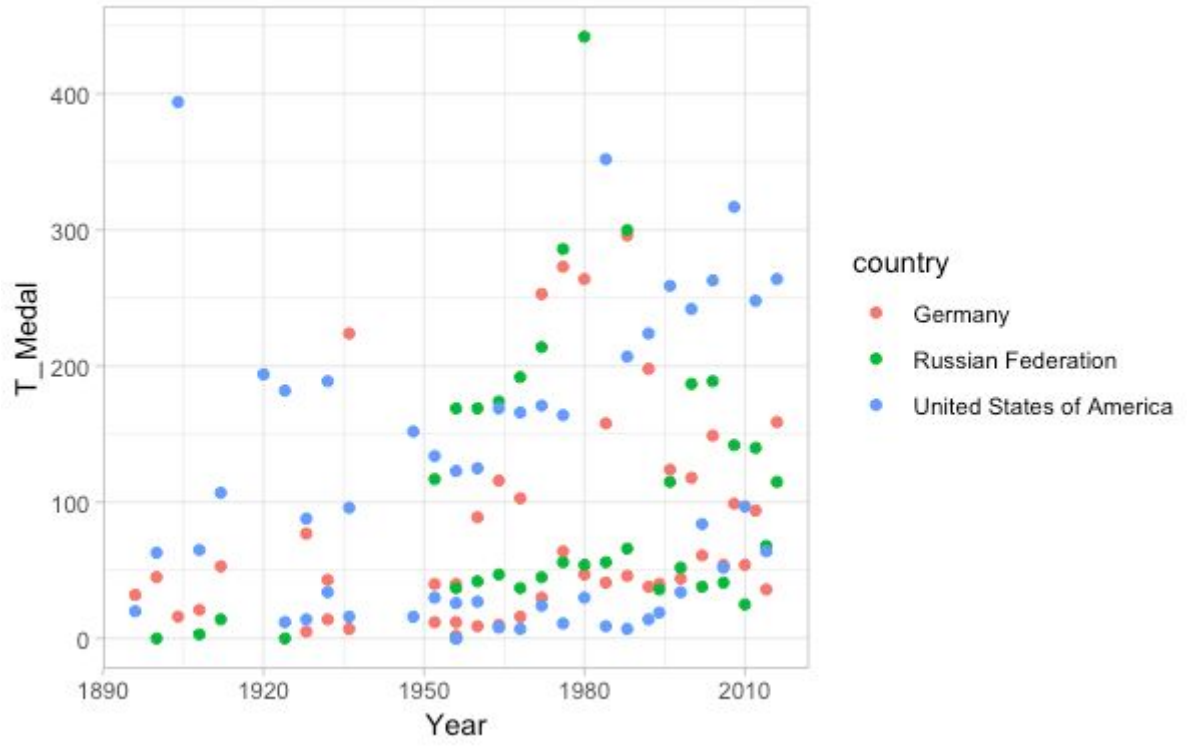
Por lo que queda nuevamente demostrado que no existe correlación entre las variables analizadas.

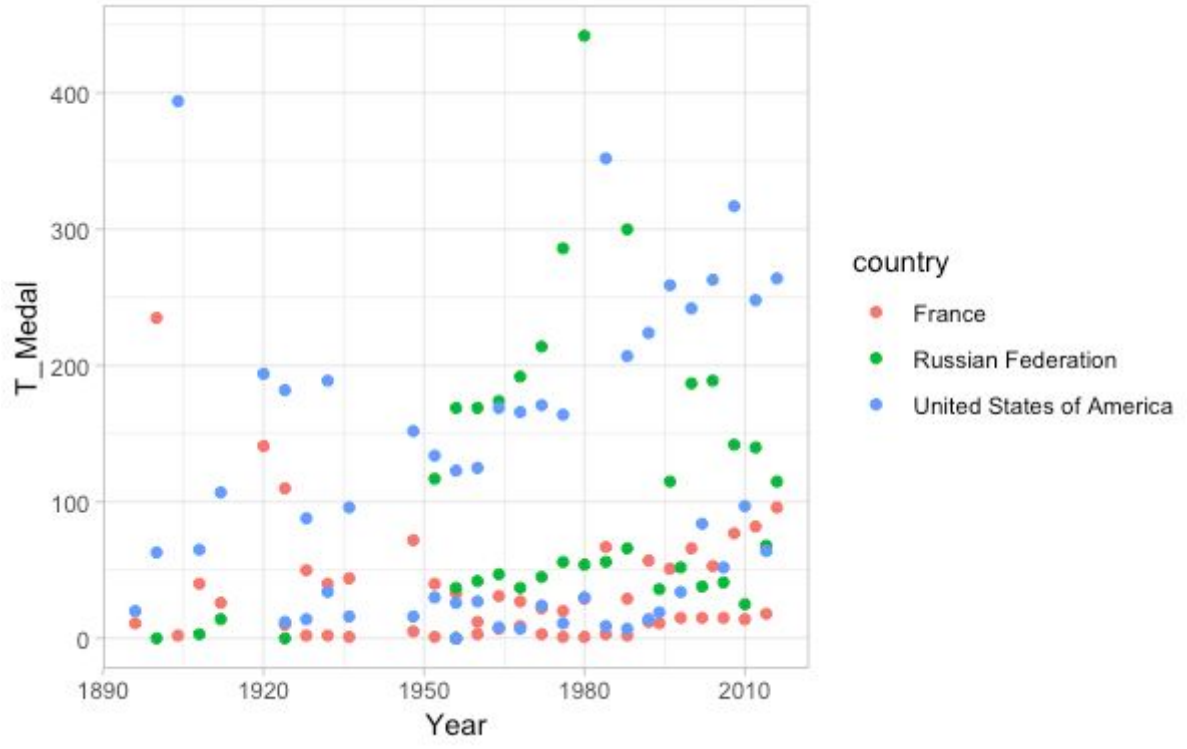
5. Representación de los resultados a partir de tablas y gráficas.

Vamos a comparar visualmente los resultados de los dos mejores países en los Juegos Olímpicos con España y otros países de nuestro entorno más cercano como Francia o Alemania.

En el eje de las X tenemos los años en los que se han celebrado los juegos y en el eje de las Y el total de medallas conseguidas. Cuando hay dos resultados en el mismo año uno se refiere a la edición de invierno y otro a la de verano. Con el color distinguimos el país en cuestión.




T_Medal
400
300
200
100
0
1890
1920
1950
1980
2010
Year
country
Germany
Russian Federation
United States of America


T_Medal
400
300
200
100
0
1890
1920
1950
1980
2010
Year
country
France
Russian Federation
United States of America

6. Resolución del problema.

Queda claramente demostrado a través de los diferentes análisis estadísticos utilizados que no podemos estimar una correlación que demuestre de manera fiable que el ser sede tenga una repercusión positiva, o negativa, en el país anfitrión o en el continente donde se celebra el evento olímpico.

Si si es cierto que se ha demostrado que en ciertos casos ha ocurrido un desarrollo deportivo en el país anfitrión, como puede puede ser Australia en 1956, que triplicó las medallas conseguidas en 1952[4], o España en 1992 que consiguió su mejor actuación hasta la fecha, logrando 22 medallas, trece de ellas de oro[5], pero estas mejoras puntuales no garantizan un mejor rendimiento deportivo, debiéndose analizar otros factores distintos a los analizados en esta práctica para determinar el motivo de tales logros.

[4] https://es.wikipedia.org/wiki/Juegos_Ol%C3%ADmpicos_de_Melbourne_1956
[5] https://es.wikipedia.org/wiki/Juegos_Ol%C3%ADmpicos_de_Barcelona_1992

7. Código

```
## ------------------------------------------------------------------------------
## SCRIPT: olimpiadas.R
## ASIGNATURA: Tipología y ciclo de vida de los datos
## AUTORES: Manuel Betancor y María Navalón
## PAQUETES NECESARIOS: dplyr, modeest
## ------------------------------------------------------------------------------

# Inicialización de librerías

if(!require("dplyr")){
  install.packages("dplyr")
  library("dplyr")
}

if(!require("MASS")){
  install.packages("MASS")
  library("MASS")
}

if(!require("gmodels")){
  install.packages("gmodels")
  library("gmodels")
}

if(!require("modeest")){
  install.packages("modeest")
  library("modeest")
}

if(!require("ggplot2")){
  install.packages("ggplot2")
  library("ggplot2")
}


## ------------------------------------------------------------------------------
##                            CARGA DE LOS DATOS
## ------------------------------------------------------------------------------

# Los ficheros csv a importar se han adquirido a través de los siguientes enlaces:
#  https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results
#  https://www.kaggle.com/statchaitya/country-to-continent
#  https://www.downloadexcelfiles.com/wo_en/download-excel-file-list-olympic-host-cities
# Para la carga se va a utilizar enlaces suministrados por dropbox de los archivos

atletas <- read.csv("https://www.dropbox.com/s/8ogvaf0ipu4raby/athlete_events.csv?dl=1",
encoding="utf-8")
paises <- read.csv("https://www.dropbox.com/s/qol9t3g4z359img/countryContinent.csv?dl=1",
encoding="utf-8")
```

```
sedes <-
read.csv("https://www.dropbox.com/s/6fhwd3ydoeop30j/list-host-cities-olympic-943j.csv?dl=1"
, encoding="utf-8")


## ------------------------------------------------------------------------------
##                            ANÁLISIS PRELIMINAR DE LOS DATOS
## ------------------------------------------------------------------------------

# Revisamos la estructura de los dataframes para comprobar el tipo de datos de cada
atributo
str(atletas)
sapply(atletas, function(x) class(x))
str(paises)
sapply(paises, function(x) class(x))
str(sedes)
sapply(sedes, function(x) class(x))

# Exploramos los dataframes creados visualizando las primeras filas
head(atletas)
head(paises)
head(sedes)

# Verificamos si hay registros duplicados
anyDuplicated(sedes)
anyDuplicated(paises)
anyDuplicated(atletas)

# Comprobamos que los paises son todos distintos en el dataframe paises para garantizar
# la correcta integración posterior de los datos ya que ese campo se utilizará como
# identificador único del conjunto
paises %>% distinct(country)


## ------------------------------------------------------------------------------
##                     DETECCIÓN Y TRATAMIENTO DE DATOS PERDIDOS
## ------------------------------------------------------------------------------

## Dataframe atletas
summary(atletas)
summary(as.factor(atletas$Medal)) # detalle del campo con NA's

# En el dataframe atletas encontramos 231.333 valores no definidos (NA's) en el atributo
Medal
# esto es debido a que en el dataframe se recogen todos los participantes, hayan o no
obtenido medallas
# por lo que cremamos una nueva categoría para imputarla a todos los NA's.

atletas$Medal <- as.character(atletas$Medal)
atletas[is.na(atletas$Medal),"Medal"] = "Sin medalla"
atletas$Medal <- as.character(atletas$Medal)

## Dataframe paises
summary(paises)

## Dataframe sedes
summary(sedes)
summary(as.factor(sedes$Year)) # detalle del campo con NA's
```

```
# En el dataframe de sedes encontramos 18 valores no definidos (NA's) en el atributo Year.
# Esto ocurre cuando en un mismo año coindicen las ediciones de invierno y verano, por
# lo que imputamos el año del registro inmediamente anterior que contiene el dato
# exacto que corresponde a esa edición.

for(i in 3:nrow(sedes)){
  if (is.na(sedes[i,"Year"])) {
       sedes[i,"Year"] <- sedes[i-1,"Year"]
  }
}

any(is.na(sedes$Year))
summary(as.factor(sedes$Year))


## ------------------------------------------------------------------------------
##                            DETECCIÓN DE OUTLIERS
## ------------------------------------------------------------------------------

# Lo que nos interesa es el número de medallas por país por lo que agrupamos
# por ese campo y representamos los datos mediante gráficos de cajas (boxplots)
# con el objetivo de detectar outliers en este nuevo campo calculado

by_NOC <- atletas %>% group_by(NOC) %>% summarise(count=n())
boxplot(by_NOC$count, main="Outliers de medallas por países")
boxplot.stats(by_NOC$count)$out

# Realizamos la misma representación diferenciando por tipo de medalla
# de oro, plata y bronce

by_NOC_Medal <- atletas %>% group_by(NOC, Medal) %>% summarise(count=n())
by_NOC_Medal_Gold <- by_NOC_Medal %>% filter(Medal == "Gold")
by_NOC_Medal_Silver <- by_NOC_Medal %>% filter(Medal == "Silver")
by_NOC_Medal_Bronze <- by_NOC_Medal %>% filter(Medal == "Bronze")

boxplot(by_NOC_Medal_Gold$count, main="Outliers - oro por países")
boxplot.stats(by_NOC_Medal_Gold$count)$out

boxplot(by_NOC_Medal_Silver$count, main="Outliers - plata por países")
boxplot.stats(by_NOC_Medal_Silver$count)$out

boxplot(by_NOC_Medal_Bronze$count, main="Outliers - bronce por países")
boxplot.stats(by_NOC_Medal_Bronze$count)$out


## ------------------------------------------------------------------------------
##                            SELECCIÓN DE LOS DATOS
## ------------------------------------------------------------------------------

## SELECCIÓN DE COLUMNAS O ATRIBUTOS

#Eliminamos las columnas no utilizadas en los análisis para simplificar los datasets
atletas <- atletas[,c("NOC","Year","Season","City","Sport","Event","Medal")]
paises <- paises[,c("country","code_3","continent","sub_region")]
sedes <- sedes[,c("City","Country","Continent","Year")]

## SELECCIÓN DE REGISTROS U OBSERVACIONES
```

```
# En la tabla paises hay que asignar de continente a la Antartida y la Isla Bouvet
paises$continent <- as.character(paises$continent)
paises$sub_region <- as.character(paises$sub_region)
paises[paises$NOC=="ATA","continent"] = "Antarctica"
paises[paises$NOC=="ATA","sub_region"] = "Antarctica"
paises[paises$NOC=="BVT","continent"] = "Antarctica"
paises[paises$NOC=="BVT","sub_region"] = "Antarctica"

# Añadimos contienente a islas oceánicas sin clasificar
paises[paises$continent=="","continent"] = "Oceania"
paises[paises$sub_region=="","sub_region"] = "Others"
paises$continent <- as.factor(paises$continent)
paises$sub_region <- as.factor(paises$sub_region)

# En el dataframe sedes hay que eliminar los 3 últimos registros ya que son basura
sedes <- sedes[1:(nrow(sedes)-3),]

# Además hemos descubierto que en el año 1956, por una cuarentena en el país, las pruebas
# de equitación de Melbourne se realizaron en Estocolmo, por lo que hay que añadir dos
# líneas, ya que en este dataframe no viene desagregadas estas dos ciudades.

sedes <- sedes[sedes$City!="Melbourne\n Stockholm",]
sedes <- add_row(sedes, City="Melbourne",Country="Australia",Continent="Oceania",Year=1956)
%>%
  add_row(City="Stockholm",Country="Sweden",Continent="Europe",Year=1956)

# Los Juegos Olímpicos de 1906 que se celebraron en Atenas no son reconocidos por el
# Comité Olímpico Internacional (COI) en la actualidad por lo que no aparecen en el
# dataset sedes y los tenemos que eliminar del dataset atletas
atletas <- atletas[atletas$Year != 1906,]

# Por motivo de la Primera Guerra Mundial, se anuló la celebración de los juegos en Berlin
en 1916
sedes <- sedes[sedes$Year != 1916,]

# Las dos ediciones de 1940 que aparecen con doble sede se cancelaron por la guerra
# sino-japonesa y la segunda guerra mundial por lo que eliminamos esos registros
sedes <- sedes[sedes$Year != 1940,]

# Las ediciones de 1944 también se cancelaron por la segunda guerra mundial
sedes <- sedes[sedes$Year != 1944,]

# Para dejar el dataframe de sedes limpio y evitar la generación de datos nulos, eliminamos
# los valores TBD que no contienen sedes reales

sedes <- sedes[sedes$City != "TBD",]


## ------------------------------------------------------------------------------
##                            INTEGRACIÓN DE LOS DATOS
## ------------------------------------------------------------------------------

# Preparamos los datos para la unión homogeneizando los niveles de las variables
# categóricas que se van a utilizar posteriormente

# Pasamos el atributo a caracteres para poder hacer la imputación
sedes$Country <- as.character(sedes$Country)
```

```
sedes[sedes$Country == "United States", "Country"] = "USA"
sedes[sedes$Country == "United Kingdom", "Country"] = "UK"
sedes[sedes$Country == "Nazi Germany", "Country"] = "Germany"
sedes[sedes$Country == "West Germany", "Country"] = "Germany"
sedes[sedes$Country == "Soviet Union", "Country"] = "Russia"

# Volvemos a cambiar a factor
sedes$Country <- as.factor(sedes$Country)

# Pasamos el atributo a caracteres para poder hacer la imputación
sedes$Continent <- as.character(sedes$Continent)

sedes[sedes$Continent == "North America", "Continent"] = "Americas"
sedes[sedes$Continent == "South America", "Continent"] = "Americas"

# Volvemos a cambiar a factor
sedes$Continent <- as.factor(sedes$Continent)

# Hay ciertos nombres de ciudades que no coinciden con los de atletas, esto es debido
# al idioma utilizado, ya que en sedes siempre están en inglés y en atletas utilizan
# para poner el nombre el propio del país.

# Pasamos el atributo a caracteres para poder hacer la imputación manual de las 11 ciudades
sedes$City <- as.character(sedes$City)

# Imputamos manualmente los nombres correctos de las ciudades
sedes[sedes$City == "Rome", "City"] = "Roma"
sedes[sedes$City == "Athens", "City"] = "Athina"
sedes[sedes$City == "Antwerp", "City"] = "Antwerpen"
sedes[sedes$City == "St. Moritz", "City"] = "Sankt Moritz"
sedes[sedes$City == "Moscow", "City"] = "Moskva"
sedes[sedes$City == "Turin", "City"] = "Torino"

# Volvemos a cambiar a factor
sedes$City <- as.factor(sedes$City)

# Hemos generado un diccionario que relaciona los códigos NOC con los códigos code_3
diccionario <- read.csv("https://www.dropbox.com/s/gibjji4okfhl0ru/diccionario.csv?dl=1",
encoding="utf-8")

# Unificamos el nombre de los atributos para simplificar la unión de los dataframes
names(paises) <- c("country","NOC","continent","sub_region")
diccionario <- diccionario[,c("NOC","code_3")]
names(diccionario) <- c("NOC","code_3")
names(sedes) <- c("City","Country_host","Continent_host","Year")

# En la tabla países falta por añadir Kosovo para evitar datos nulos en la unión
paises <- paises %>%
  add_row(country="Kosovo",NOC="XKX",continent="Europe",sub_region="Southern Europe")

# Mantenemos todos los registros del primer dataframe con all.x=TRUE
df <- merge(atletas, diccionario, by = "NOC", all.x=TRUE) %>%
merge(paises, by = "code_3", by.y = "NOC", all.x=TRUE) %>%
merge(sedes, by = c("City","Year"), all.x=TRUE)

# En la unión vemos que hay registros NA generados, analizándolos vemos que coresponden a
los siguientes códigos NOC/code_3
```

```
# EUN EUN Unified Team
# IOA IOA Individual Olympic Athletes
# Estos registros hay que eliminarlos, ya que no nos va a servir para el análisis por
países o continentes

df <- df[df$NOC!="EUN",]
df <- df[df$NOC!="IOA",]

# Comprobamos la no existencia de NA tras la unión
summary(df)


# Es necesario "dummificar" la variable cualitativa Medal que toma los valores
# Gold, Silver, Bronze, así la convertimos en 3 variables dicotómicas (0, 1)
# para lo que añadimos tres columnas nuevas en el dataframe

df <- df %>%
  mutate(Gold = ifelse(Medal == 'Gold', 1, 0)) %>%
  mutate(Silver = ifelse(Medal == 'Silver', 1, 0)) %>%
  mutate(Bronze = ifelse(Medal == 'Bronze', 1, 0))

# Creamos dos nuevos atributo dicotómicos, "sedePais" y "sedeContinente" donde
# el 1 significa que en esa edición de los juegos el país o el continente del
# equipo fue sede de dichos juegos

df$country <- as.character(df$country)
df$Country_host <- as.character(df$Country_host)
df$continent <- as.character(df$continent)
df$Continent_host <- as.character(df$Continent_host)

df <- df %>%
  mutate(sedePais = ifelse(country == Country_host, 1, 0)) %>%
  mutate(sedeContinente = ifelse(continent == Continent_host, 1, 0))

df$country <- as.factor(df$country)
df$Country_host <- as.factor(df$Country_host)
df$continent <- as.factor(df$continent)
df$Continent_host <- as.factor(df$Continent_host)

# Creamos un dataframe con los datos agregados por país para nuestro análisis
df_pais <- df %>% group_by(country, City, Year, sedePais) %>%
  summarise(T_Gold=sum(Gold), T_Silver=sum(Silver), T_Bronze=sum(Bronze),
T_Medal=(sum(Gold)+sum(Silver)+sum(Bronze)))

# Creamos un dataframe con los datos agregados por continente para nuestro análisis
df_continente <- df %>% group_by(continent, City, Year, sedeContinente) %>%
  summarise(T_Gold=sum(Gold), T_Silver=sum(Silver), T_Bronze=sum(Bronze),
T_Medal=(sum(Gold)+sum(Silver)+sum(Bronze)))

## ------------------------------------------------------------------------------
##                            ANÁLISIS ESTADÍSTICO DESCRIPTIVO
## ------------------------------------------------------------------------------

# Ranking de países por participación de atletas
summary(df$country)

# Total de medallas de cada tipo y sus porcentajes
CrossTable(df$Medal)
```

```
# Resultados acumulados absolutos de todos los juegos por país
table(df$Medal, df$country)
# Resultados acumulados en porcentaje de todos los juegos por país
CrossTable(df$Medal, df$country)

# Medallas acumuladas de oro en todos los juegos por país
tapply(df$Gold, df$country, sum)
# Medallas acumuladas de plata en todos los juegos por país
tapply(df$Silver, df$country, sum)
# Medallas acumuladas de bronce en todos los juegos por país
tapply(df$Bronze, df$country, sum)

# Calculamos la moda del atributo "Medal" (tipo de medalla) para todo el conjunto
mlv(df$Medal, na.rm=TRUE)

# Calculamos la moda del atributo "Medal" (tipo de medalla) para cada país
tapply(df$Medal, df$country, mlv, na.rm=TRUE)

# Calculamos la moda del par país-medalla
apply(df[,c("country", "Medal")], 2, mlv,  method = "mfv", na.rm=TRUE)


## ------------------------------------------------------------------------------------
##                            ANÁLISIS ESTADÍSTICO INFERENCIAL
## ------------------------------------------------------------------------------------

## ANÁLISIS DE CORRELACIÓN

# Comprobamos como se distribuyen las variables para ver que prueba es la más adecuada
shapiro.test(df_pais$sedePais)
shapiro.test(df_pais$T_Gold)
shapiro.test(df_pais$T_Silver)
shapiro.test(df_pais$T_Bronze)
shapiro.test(df_pais$T_Medal)
shapiro.test(df_continente$sedeContinente)
shapiro.test(df_continente$T_Gold)
shapiro.test(df_continente$T_Silver)
shapiro.test(df_continente$T_Bronze)
shapiro.test(df_continente$T_Medal)

# Lo podemos ver también gráficamente comparándolo con una distribución normal
plotn <- function(x,main="Histograma de frecuencias \ny distribución normal",
              xlab="X",ylab="Densidad") {
  min <- min(x)
  max <- max(x)
  media <- mean(x)
  dt <- sd(x)
  hist(x,freq=F,main=main,xlab=xlab,ylab=ylab)
  curve(dnorm(x,media,dt), min, max,add = T,col="blue")
}

plotn(df_pais$sedePais, main="Distribución normal - Paises - Sedes")
plotn(df_pais$T_Gold, main="Distribución normal -Paises - Oro")
plotn(df_pais$T_Silver, main="Distribución normal - Paises - Plata")
plotn(df_pais$T_Bronze, main="Distribución normal - Paises - Bronce")
plotn(df_pais$T_Medal, main="Distribución normal - Paises - Medallas")
plotn(df_continente$sedeContinente, main="Distribución normal - Continentes - Sedes")
```

```
plotn(df_continente$T_Gold, main="Distribución normal - Continentes - Oro")
plotn(df_continente$T_Silver, main="Distribución normal - Continentes - Plata")
plotn(df_continente$T_Bronze, main="Distribución normal - Continentes - Bronce")
plotn(df_continente$T_Medal, main="Distribución normal - Continentes - Medallas")

# Usamos el coeficiente de correlación de Spearman
cor(x=df_pais$sedePais, y=df_pais$T_Gold, method = "spearman")
cor(x=df_pais$sedePais, y=df_pais$T_Silver, method = "spearman")
cor(x=df_pais$sedePais, y=df_pais$T_Bronze, method = "spearman")
cor(x=df_pais$sedePais, y=df_pais$T_Medal, method = "spearman")
cor(x=df_continente$sedeContinente, y=df_continente$T_Gold, method = "spearman")
cor(x=df_continente$sedeContinente, y=df_continente$T_Silver, method = "spearman")
cor(x=df_continente$sedeContinente, y=df_continente$T_Bronze, method = "spearman")
cor(x=df_continente$sedeContinente, y=df_continente$T_Medal, method = "spearman")

# Significancia de la correlación de Spearman
cor.test(x=df_pais$sedePais, y=df_pais$T_Gold, conf.level  = 0.95, method = "spearman")
cor.test(x=df_pais$sedePais, y=df_pais$T_Silver, conf.level  = 0.95, method = "spearman")
cor.test(x=df_pais$sedePais, y=df_pais$T_Bronze, conf.level  = 0.95, method = "spearman")
cor.test(x=df_pais$sedePais, y=df_pais$T_Medal, conf.level  = 0.95, method = "spearman")
cor.test(x=df_continente$sedeContinente, y=df_continente$T_Gold, conf.level  = 0.95, method
= "spearman")
cor.test(x=df_continente$sedeContinente, y=df_continente$T_Silver, conf.level  = 0.95,
method = "spearman")
cor.test(x=df_continente$sedeContinente, y=df_continente$T_Bronze, conf.level  = 0.95,
method = "spearman")
cor.test(x=df_continente$sedeContinente, y=df_continente$T_Medal, conf.level  = 0.95,
method = "spearman")

# Usamos el coeficiente de correlación de Kendall
cor(x=df_pais$sedePais, y=df_pais$T_Gold, method = "kendall")
cor(x=df_pais$sedePais, y=df_pais$T_Silver, method = "kendall")
cor(x=df_pais$sedePais, y=df_pais$T_Bronze, method = "kendall")
cor(x=df_pais$sedePais, y=df_pais$T_Medal, method = "kendall")
cor(x=df_continente$sedeContinente, y=df_continente$T_Gold, method = "kendall")
cor(x=df_continente$sedeContinente, y=df_continente$T_Silver, method = "kendall")
cor(x=df_continente$sedeContinente, y=df_continente$T_Bronze, method = "kendall")
cor(x=df_continente$sedeContinente, y=df_continente$T_Medal, method = "kendall")

## ------------------------------------------------------------------------------
##                                  VISUALIZACION
## ------------------------------------------------------------------------------

# Comparamos gráficamente los resultados de EEUU y Rusia con otro país seleccionado
seleccion <- "Spain" # cambiar el país para comparar
df_seleccion <- df_pais[df_pais$country %in% list(seleccion, "United States of America",
"Russian Federation"),]
ggplot(df_seleccion, aes(Year, T_Medal, colour = country)) +
  geom_point()

## ------------------------------------------------------------------------------
##                    EXPORTACIÓN DE LOS DATOS A CSV
## ------------------------------------------------------------------------------

write.csv(df,file="df.csv", row.names = F)
write.csv(df_pais,file="df_pais.csv", row.names = F)
write.csv(df_continente,file="df_continente.csv", row.names = F)
write.csv(df_seleccion,file="df_seleccion.csv", row.names = F)
```

8. Contribuciones.

Contribuciones	Firma
Investigación previa	MBS, MDNH
Redacción de las respuestas	MBS, MDNH
Desarrollo código	MBS, MDNH

9. Bibliografía.

Calvo M., Subirats L., Pérez D. (2019). *Introducción a la limpieza y análisis de los datos.* Editorial UOC

Dalgaard P. (2002). *Introductory Statistics with R*. New York: Springer-Verlag.

Squire M. (2015). *Clean Data.* Birmingham: Packt Publishing.

Vries A., Meys J. (2015). *R For Dummies, 2nd Edition.* New Jersey: John Wiley & Sons.

VV.AA. (2000). *Introducción a R. Versión 1.0.1.* R Development Core Team.