Laboratorio: Análisis Combinatorio

Ejercicio Yatzy

Autores: Abraham Soto y Johanna Álvarez

Fecha: 18 de junio de 2025

Introducción

El presente informe aborda la aplicación de los principios del Análisis Combinatorio y la Teoría de la Probabilidad en base al juego de dados Yatzy. En particular nos centraremos en determinar la probabilidad de que un jugador gane este juego bajo condiciones específicas. El Yatzy, un popular juego de dados ofrece un escenario ideal para explorar conceptos fundamentales como el espacio muestral, los sucesos, sus probabilidades y la probabilidad de eventos compuestos.

El juego en cuestión involucra el lanzamiento de cinco dados dos veces, donde ambos lanzamientos son independientes, con el objetivo de acumular al menos 12 puntos. La puntuación se basa en combinaciones específicas de resultados: Triple (5 puntos), Full (8 puntos) y Escalera (10 puntos). Mediante este laboratorio de análisis combinatorio no solo determinaremos calcularemos probabilidades asociadas a eventos sino que también haremos uso del lenguaje de programación R Base dentro de RStudio.

Finalmente, se presentará un análisis exhaustivo de los hallazgos proporcionando una comprensión clara de la probabilidad de éxito en el juego de Yatzy.

EJERCICIO YATZY

Un juego llamado Yatzy consiste en lanzar 5 dados durante 2 partidas independientes; una persona está sola en el casino apostando en este juego, se sabe que gana si logra acumular 12 puntos o más al final. El jugador va acumulando puntos según los resultados obtenidos en cada partida:

- Triple (tres iguales y dos diferentes) = 5 puntos
- Full (tres iguales y dos iguales) = 8 puntos
- Escalera (menor o mayor) = 10 puntos

Notas:

- Escalera menor: 1, 2, 3, 4, 5
- Escalera mayor: 2, 3, 4, 5, 6

De acuerdo con la información suministrada, realice las siguientes actividades:

1. Resolver el ejercicio y entregarlo en papel, a lapicero, lápiz, marcador, etc.; en hojas de examen; o desarrollarlo en Word o en RMarkdown haciendo uso del entorno RStudio y solamente con R base, mediante filtros u operaciones de conteo a los vectores y/o matrices.

2. Obtenga el espacio muestral

Primeramente, se determinan las variables esenciales del estudio: "caras y número de dados" y a su vez el espacio muestral o el total de posibles resultados.

```
caras_dado <- 6
num_dados <- 5
```

Espacio Muestral para un solo lanzamiento:

```
espacio_muestral <- caras_dado^num_dados
espacio_muestral
```

```
## [1] 7776
```

El espacio muestral está compuesto por 7,776 secuencias posibles de resultados para el primer lanzamiento.

3.Obtenga los casos favorables a los sucesos: obtener una tercia, un full, una escalera y no obtener ninguno de los resultados anteriores.

3.1 Obtener los casos favorables de una tercia:

Cuando se lanza un grupo de cinco dados, se busca obtener un "Triple" (por ejemplo, 4, 4, 4, 1, 6), lo que nos interesa es la combinación específica de valores que aparecen en esos dados.

Un "Triple" significa que tres dados muestran el mismo valor, y los otros dos dados muestran valores diferentes entre sí y también diferentes al valor de los tres iguales. Nuestro objetivo es calcular cuántas formas distintas hay de que ocurra esta combinación.

Esta es la función de la combinación 6C1, es decir, elegir 1 elemento de un conjunto de 6 que representa al trio:

```
opciones_valor_triple_para_Triple <- choose(6, 1)
```

Elegir los posibles valores para el par diferente del numero escogido en el trio y entre ellos. Esta función corresponde a la combinación 5C2. Indica la selección de dos valores distintos para los dados que restan, luego de haber escogido el valor del trío:

```
opciones_valores_diferentes_Triple <- choose(5, 2)
```

Distribución de los 5 dados en las 5 posiciones: El orden seria (X,X,X,Y,Z) por ejemplo (1,1,1,2,3).

Concepto de Coeficiente Multinomial (5! / (3! * 1! * 1!)).

Representa las diferentes formas de permutar los valores específicos de los 5 dados una vez que hemos elegido qué números son. Este es un caso de permutaciones con repetición, que se resuelve con la fórmula del coeficiente multinomial:

```
distribucion_posiciones_triple <- factorial(5) /
                                  (factorial(3) * factorial(1) * factorial(1))
```

Obtener los posibles casos favorables de obtener un Triple.

Se obtiene por el producto de las combinaciones y el coeficiente multinomial que calculamos en pasos previos:

```
total_resultados_triple <- opciones_valor_triple_para_Triple *
                           opciones_valores_diferentes_Triple *
                           distribucion_posiciones_triple
```

Observamos el total de resultados de obtener un Triple.

```
total_resultados_triple
```

```
## [1] 1200
```

Se ha determinado que existen 1200 casos favorables para obtener un "Triple" en un lanzamiento.

3.2 Obtener los casos favorables de un full:

Un Full consiste en tres dados de un número y dos dados de un mismo número, los denotamos como (X,X,X,Y,Y) por ejemplo (1,1,1,2,2).

Para este caso contamos n = 5 dados, n1 = 3 dados iguales, n2 = 2 dados iguales pero diferentes al trio.

Elegimos los posibles valores para el trio de Full.

Esta es la función de la combinación 6C1, es decir, elegir 1 elemento de un conjunto de 6 que representa al trio del full.

```r
opciones_valor_triple_para_Full <- choose(6, 1)
```

Elegir los posibles valores para obtener el par.

La combinación 5C1 se utiliza para elegir el primer valor del par de caras restantes, asegurando que sea diferente al trío seleccionado.

```r
opciones_valor_par_Full<- choose(5, 1)
```

Distribución de Valores en las 5 Posiciones de los Dados.

Concepto de Coeficiente Multinomial (5! / (3! * 2!)) En este caso, tenemos 5 dados donde 3 son iguales y 2 son iguales entre ellos pero diferentes al trio.

```r
distribucion_posiciones_full <- choose(5, 3)
```

Se obtiene por el producto de las combinaciones y el coeficiente multinomial que calculamos en pasos previos:

```r
total_resultados_full <- opciones_valor_triple_para_Full *
                         opciones_valor_par_Full *
                         distribucion_posiciones_full
```

Se obtiene el total de casos favorables para obtener un full.

```r
total_resultados_full
```

```
## [1] 300
```

Se ha determinado que existen 300 casos favorables para obtener un “full” en un lanzamiento.

3.3 Obtener los casos favorables de una escalera (Menor y Mayor)

Existen 2 casos de escalera: menor (1,2,3,4,5) y mayor (2,3,4,5,6) por lo cual podemos observar que se trata de una simple permutacion en ambos casos.

Número de tipos de escaleras (menor y mayor).

Conteo directo.

```r
numero_tipos_escalera <- 2
```

Obtener los valores para cada tipo de escalera.

Es una permutacion de 5! ya que los 5 dados deben mostrar esos 5 valores específicos, y el orden en que aparecen esos valores en los dados importa para el recuento del espacio muestral.

```r
permutaciones_escalera <- factorial(5)
```

Obtener el producto del factorial por conteo.

```
total_resultados_escalera <- numero_tipos_escalera * permutaciones_escalera
```

Observamos el número total de resultados de obtener una Escalera.

```
total_resultados_escalera
```

```
## [1] 240
```

Se ha determinado que existen 240 casos favorables para obtener una "escalera" en un lanzamiento.

3.4 Obtener los casos favorables de ninguno

Se aplica la propiedad del complemento, para obtener el resultado:

```
total_no_obtener_puntos <- espacio_muestral -
                           (total_resultados_escalera + total_resultados_full
                          + total_resultados_triple)
```

Observamos el numero total de no obtener ningun punto.

```
total_no_obtener_puntos
```

```
## [1] 6036
```

Se ha determinado que existen 6036 de no obtener ninguno de los casos anterios en un lanzamiento.

4. Obtenga las probabilidades de cada uno de los sucesos anteriores.

Estos son los resultados de cada suceso en una sola partida.

```
prob_full <- total_resultados_full / espacio_muestral
prob_triple <- total_resultados_triple / espacio_muestral
prob_escalera <- total_resultados_escalera / espacio_muestral
```

Tambien se obtiene la probabilidad de no obtener ninguno.

```
prob_ninguno <- 1 - (prob_escalera + prob_full + prob_triple)
```

```
prob_full
```

```
## [1] 0.03858025
```

```
prob_triple
```

```
## [1] 0.154321
```

```r
prob_escalera
```

```
## [1] 0.0308642
```

```r
prob_ninguno
```

```
## [1] 0.7762346
```

Comprobación:

Las probabilidades estaran bien definidas si la suma de todas da igual a 1.

```r
prob_full + prob_triple + prob_escalera + prob_ninguno
```

```
## [1] 1
```

De esta forma se obtiene las probabilidades definidas de obtener cada uno de los sucesos en un solo lanzamiento dentro del espacio muestral 6^5.

5. Obtenga la probabilidad de ganar el juego

Es importante recalcar que para ganar el juego se tienen que tomar en cuenta los dos lanzamientos, por ende, el espacio muestral total es 6^10 o multiplicar el espacio muestral de un lanzamiento por si mismo, de esta manera obtenemos el espacio muestral para dos lanzamientos.

```r
espacio_muestral_total <- espacio_muestral * espacio_muestral
```

El numero total de casos posibles a obtener en la partida completa es de 60466176.

Se organiza el numero de casos totales favorables para un lanzamiento de cada suceso en un vector:

```r
total_resultados_obtenibles <- c(
  "Escalera" = total_resultados_escalera,
  "Triple" = total_resultados_triple,
  "Full" = total_resultados_full,
  "Ninguno" = total_no_obtener_puntos
)
total_resultados_obtenibles
```

```
## Escalera   Triple     Full  Ninguno
##      240     1200      300     6036
```

Se organiza el numero de puntos obtenidos por cada suceso para un lanzamiento en un vector:

```r
puntos <- c("Escalera" = 10,"Triple" = 5,  "Full" = 8,"Ninguno" = 0)
puntos
```

```
## Escalera   Triple     Full  Ninguno
##       10        5        8        0
```

Se organiza las probabilidades de cada suceso de un lanzamiento en un vector:

```r
probabilidad_un_solo_lanzamiento <- c(
  "Escalera" = prob_escalera,
  "Triple" = prob_triple,
  "Full" = prob_full,
  "Ninguno" = prob_ninguno
)

probabilidad_un_solo_lanzamiento
```

```
##   Escalera     Triple       Full    Ninguno
## 0.03086420 0.15432099 0.03858025 0.77623457
```

El usuario gana la partida si acumula 12 puntos o más. Para determinar el resultado final, se procede a un segundo lanzamiento, el cual es independiente del primero.

Se construye una matriz donde cada celda [i,j] indica la probabilidad de obtener el suceso i en el primer lanzamiento y el suceso j en el segundo lanzamiento.

```r
prob_combinada <- outer(
  probabilidad_un_solo_lanzamiento,
  probabilidad_un_solo_lanzamiento,
  FUN = "*"
)

colnames(prob_combinada) <- names(probabilidad_un_solo_lanzamiento)
rownames(prob_combinada) <- names(probabilidad_un_solo_lanzamiento)

prob_combinada
```

```
##              Escalera      Triple        Full    Ninguno
## Escalera 0.0009525987 0.004762993 0.001190748 0.02395786
## Triple   0.0047629934 0.023814967 0.005953742 0.11978929
## Full     0.0011907484 0.005953742 0.001488435 0.02994732
## Ninguno  0.0239578570 0.119789285 0.029947321 0.60254010
```

Esta matiz representa las probabilidades combinadas entre los sucesos.

Para establecer la cantidad total de puntos acumulados en ambos lanzamientos, se procede a crear una matriz que represente la combinación de las sumas de puntos de cada suceso individual.

```r
puntos_totales_combinados <- outer(puntos, puntos, FUN = "+")
colnames(puntos_totales_combinados) <- names(puntos)
rownames(puntos_totales_combinados) <- names(puntos)

puntos_totales_combinados
```

```
##          Escalera Triple Full Ninguno
## Escalera       20     15   18      10
## Triple         15     10   13       5
## Full           18     13   16       8
## Ninguno        10      5    8       0
```

Para que el usuario pueda ganar la partida se debe obtener un puntaje mayor o igual a 12, por lo que, usando las matrices creadas anteriormente y aplicando filtrado para sumar los puntajes, obtenemos los siguientes resultados:

```
Puntaje_Ganador <- puntos_totales_combinados >= 12
```

Por ultimo para hallar la probabilidad de ganar la partida sumamos las probabilidades que cumplen la condicion de ganar:

```
probabilidad_ganar <- sum(prob_combinada[Puntaje_Ganador])
probabilidad_ganar
```

```
## [1] 0.026256
```

La probabilidad de que una persona gane el juego es de 0.026256 obteniendo alguno de los resultados favorables que sumen 12 puntos o más en los dos lanzamientos.

Ahora bien, usando paquetes con funciones de generación de permutaciones, variaciones y combinaciones:

6. Replique el proceso para generar los casos posibles de cada uno de los sucesos de los apartados (2) y (3).

6.1 Obtener el espacio muestral

Para poder resolver este problema, usaremos los paquetes gtools, combinat y arrangements,estos sirven para realizar operaciones relacionadas con análisis combinatorio, es decir, para generar y trabajar con permutaciones, combinaciones y variaciones de forma eficiente:

gtools: Este paquete contiene una variedad de funciones útiles, entre las que se incluyen funciones para generar permutaciones y combinaciones. Es un paquete más general que ofrece diversas utilidades.

arrangements: Este paquete está específicamente optimizado para la generación rápida de permutaciones y combinaciones, especialmente cuando se trabaja con conjuntos grandes. Ofrece funciones de alto rendimiento para estas tareas combinatorias.

Combinat: Es un paquete de R que permite realizar operaciones de análisis combinatorio como permutaciones, combinaciones y cálculos con coeficientes multinomiales. Es útil para generar y contar configuraciones posibles en problemas de conteo sin necesidad de hacerlo manualmente.

```
dados <- 1:6
num_dados <- 5
total_posibles <- 6^5
```

El espacio muestral está compuesto por 7,776 secuencias posibles de resultados para el primer lanzamiento.

Se crea la matriz del espacio muestral.

```
espacio_muestral <- permutations(n = 6, r = 5, v = 1:6, repeats.allowed = TRUE)
head(espacio_muestral, 10)
```

```
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    1    1    1    1    1
##  [2,]    1    1    1    1    2
```

```
##  [3,]    1    1    1    1    3
##  [4,]    1    1    1    1    4
##  [5,]    1    1    1    1    5
##  [6,]    1    1    1    1    6
##  [7,]    1    1    1    2    1
##  [8,]    1    1    1    2    2
##  [9,]    1    1    1    2    3
## [10,]    1    1    1    2    4
```

Esta matriz representa algunos de los posibles resultados que se pueden obtener al lanzar cinco dados.

Cada fila de la matriz representa una combinación diferente de lo que podría salir en los cinco dados (por ejemplo, una fila podría ser "1, 1, 1, 1, 1", y otra "1, 1, 1, 1, 2").

Cada columna representa un dado específico (Dado 1, Dado 2, etc.).

6.2 Obtenga los casos favorables a los sucesos: obtener una tercia, un full, una escalera y, no obtener ninguno de los resultados anteriores.

6.2.1 Obtener una "Triple" (Tres dados iguales y dos diferentes)

Como se esta trabajando con combinaciones es necesario definir funciones para clasificar las tiradas.

```r
dados <- 1:6
espacio_muestral <- expand.grid(rep(list(dados), 5))

es_triple <- function(x) {
  freqs <- table(x)
  length(freqs) == 3 && any(freqs == 3) && all(freqs <= 3)
}

triples <- apply(espacio_muestral, 1, es_triple)

sum(triples)
```

```
## [1] 1200
```

```r
View(data.frame(Total_casos_triple = sum(triples)))

matriz_triples <- espacio_muestral[triples, ]

colnames(matriz_triples) <- c("dado 1", "dado 2", "dado 3", "dado 4", "dado 5")

print(head(matriz_triples, 10))
```

```
##    dado 1 dado 2 dado 3 dado 4 dado 5
## 9       3      2      1      1      1
## 10      4      2      1      1      1
## 11      5      2      1      1      1
## 12      6      2      1      1      1
## 14      2      3      1      1      1
## 16      4      3      1      1      1
## 17      5      3      1      1      1
```

```
## 18      6      3      1      1      1
## 20      2      4      1      1      1
## 21      3      4      1      1      1
```

Esta matriz representa algunos de los posibles resultados que se pueden obtener para conseguir un triple al lanzar cinco dados.

Cada fila de la matriz representa una combinación diferente de lo que podría salir en los cinco dados.

Cada columna representa un dado específico (Dado 1, Dado 2, etc.).

Se ha determinado que existen 1200 casos favorables para obtener un "tercia" en un lanzamiento.

6.2.2 Obtener un "Full" (Tres dados iguales y dos dados iguales)

Como se esta trabajando con combinaciones es necesario definir funciones para clasificar las tiradas.

```r
dados <- 1:6
espacio_muestral <- expand.grid(rep(list(dados), 5))

es_full <- function(x) {
  freqs <- table(x)
  length(freqs) == 2 && all(sort(freqs) == c(2, 3))
}

fulls <- apply(espacio_muestral, 1, es_full)

sum(fulls)
```

```
## [1] 300
```

```r
View(data.frame(Total_casos_full = sum(fulls)))

matriz_fulls <- espacio_muestral[fulls, ]

colnames(matriz_fulls) <- c("dado 1", "dado 2", "dado 3", "dado 4", "dado 5")

print(head(matriz_fulls, 10))
```

```
##    dado 1 dado 2 dado 3 dado 4 dado 5
## 8       2      2      1      1      1
## 15      3      3      1      1      1
## 22      4      4      1      1      1
## 29      5      5      1      1      1
## 36      6      6      1      1      1
## 38      2      1      2      1      1
## 43      1      2      2      1      1
## 44      2      2      2      1      1
## 75      3      1      3      1      1
## 85      1      3      3      1      1
```

Esta matriz representa algunos de los posibles resultados que se pueden obtener para conseguir un full al lanzar cinco dados.

Cada fila de la matriz representa una combinación diferente de lo que podría salir en los cinco dados.

Cada columna representa un dado específico (Dado 1, Dado 2, etc.).

Se ha determinado que existen 300 casos favorables para obtener un “full” en un lanzamiento.

6.2.3 Obtener una “Escalera” (Menor o Mayor)

```
es_escalera <- function(x) {
  sorted <- sort(x)
  all(sorted == 1:5) || all(sorted == 2:6)
}

escaleras <- apply(espacio_muestral, 1, es_escalera)

sum(escaleras)
```

```
## [1] 240
```

```
matriz_escaleras <- espacio_muestral[escaleras, ]

colnames(matriz_escaleras) <- c("dado 1", "dado 2", "dado 3", "dado 4", "dado 5")

print(head(matriz_escaleras, 10))
```

```
##     dado 1 dado 2 dado 3 dado 4 dado 5
## 311      5      4      3      2      1
## 316      4      5      3      2      1
## 341      5      3      4      2      1
## 351      3      5      4      2      1
## 376      4      3      5      2      1
## 381      3      4      5      2      1
## 491      5      4      2      3      1
## 496      4      5      2      3      1
## 551      5      2      4      3      1
## 566      2      5      4      3      1
```

Esta matriz representa algunos de los posibles resultados que se pueden obtener para conseguir una escalera al lanzar cinco dados.

Cada fila de la matriz representa una combinación diferente de lo que podría salir en los cinco dados.

Cada columna representa un dado específico (Dado 1, Dado 2, etc.).

Se ha determinado que existen 240 casos favorables para obtener un “escalera” en un lanzamiento.

6.2.3 No obtener ninguno de los resultados anteriores

Función para detectar “nada”: no es triple, ni full, ni escalera.

```r
es_triple <- function(x) {
  freqs <- table(x)
  length(freqs) == 3 && any(freqs == 3) && all(freqs <= 3)
}

es_full <- function(x) {
  freqs <- table(x)
  length(freqs) == 2 && all(sort(freqs) == c(2, 3))
}

es_escalera <- function(x) {
  sorted <- sort(x)
  all(sorted == 1:5) || all(sorted == 2:6)
}

es_nada <- function(x) {
  !(es_triple(x) || es_full(x) || es_escalera(x))
}

triples <- apply(espacio_muestral, 1, es_triple)
fulls <- apply(espacio_muestral, 1, es_full)
escaleras <- apply(espacio_muestral, 1, es_escalera)

nadas <- apply(espacio_muestral, 1, es_nada)

sum(nadas)
```

```
## [1] 6036
```

```r
matriz_nadas <- espacio_muestral[nadas, ]

colnames(matriz_nadas) <- c("dado 1", "dado 2", "dado 3", "dado 4", "dado 5")

print(head(matriz_nadas, 10))
```

```
##    dado 1 dado 2 dado 3 dado 4 dado 5
## 1       1      1      1      1      1
## 2       2      1      1      1      1
## 3       3      1      1      1      1
## 4       4      1      1      1      1
## 5       5      1      1      1      1
## 6       6      1      1      1      1
## 7       1      2      1      1      1
## 13      1      3      1      1      1
## 19      1      4      1      1      1
## 25      1      5      1      1      1
```

Esta matriz representa algunos de los posibles resultados que se pueden obtener para no conseguir ningun punto al lanzar cinco dados.

Cada fila de la matriz representa una combinación diferente de lo que podría salir en los cinco dados.

Cada columna representa un dado específico (Dado 1, Dado 2, etc.).

Se ha determinado que existen 6036 casos favorables para no obtener ninguno de los casos anteriores en un lanzamiento.

7. Verifique que los resultados obtenidos en el apartado (6) se corresponde con lo obtenido en (2) y (3).

El código proporcionado emplea la función all.equal() de R con el objetivo de determinar si dos objetos son "prácticamente" idénticos. Esta función ayuda a comparar valores numéricos.

```
all.equal(sum(fulls), total_resultados_full)
```

```
## [1] TRUE
```

```
all.equal(sum(triples), total_resultados_triple)
```

```
## [1] TRUE
```

```
all.equal(sum(escaleras), total_resultados_escalera)
```

```
## [1] TRUE
```

```
all.equal(sum(nadas), total_no_obtener_puntos)
```

```
## [1] TRUE
```

Este código tiene como objetivo crear y mostrar una tabla resumen que compara los resultados calculados de diferentes jugadas comparando los resultados calculados con Rbase y Paquetes de Rstudio

```
comparacion_resultados <- data.frame(
  Tipo = c("Full", "Triple", "Escalera", "Nadas"),
  Paquetes = c(
    sum(fulls),
    sum(triples),
    sum(escaleras),
    sum(nadas)
  ),
  Rbase = c(
    total_resultados_full,
    total_resultados_triple,
    total_resultados_escalera,
    total_no_obtener_puntos
  ),
  Coinciden = c(
    all.equal(sum(fulls), total_resultados_full) == TRUE,
    all.equal(sum(triples), total_resultados_triple) == TRUE,
    all.equal(sum(escaleras), total_resultados_escalera) == TRUE,
    all.equal(sum(nadas), total_no_obtener_puntos) == TRUE
  )
)

print(comparacion_resultados)
```

```
##       Tipo Paquetes Rbase Coinciden
## 1     Full      300   300      TRUE
## 2   Triple     1200  1200      TRUE
## 3 Escalera      240   240      TRUE
## 4    Nadas     6036  6036      TRUE
```

Se ha verificado que ambos métodos, tanto el implementado con R base como el que utiliza paquetes especializados de permutación, combinación y variación, arrojan resultados idénticos.

Conclusión

En este experimento, se examina el cálculo de la probabilidad de ganar en un juego de Yatzy, que consiste en lanzar dos veces cinco dados de forma independiente. La meta principal fue analizar la posibilidad de que un jugador logre obtener 12 puntos o más al final de dos juegos.

Para conseguirlo, se aplicaron conceptos del análisis combinatorio para identificar y contar los casos que son favorables a los eventos que otorgan puntos: conseguir una tercia (tres dados del mismo número), un full (una tercia y un par) y una escalera (ya sea menor o mayor).

El desarrollo del proyecto se dividió en dos etapas que se complementan entre sí. Primero, se sentaron las bases teóricas para contar cada resultado. Luego, se utilizó RStudio para resolver el problema de manera computacional. A través de la creación de un espacio muestral completo y la aplicación de conteos, se calcularon los casos posibles y los favorables para cada evento. Además, se validaron estos resultados con paquetes de R específicos para combinatoria, lo que confirmo la exactitud de los métodos utilizados.

Bibliografía

"Probabilidad y Estadística" de Ronald E. Walpole y Raymond H. Myers (2013)

"Introducción a la Probabilidad y Estadística" de William Mendenhall, Robert J. Beaver y Barbara M. Beaver (2010)

"An Introduction to R" (Manual oficial de R): https://cran.r-project.org/manuals.html

"Análisis Combinatorio en R" de Sergio Rifo (2020). YouTube: https://www.youtube.com/watch?v=a6GWdKiaJTM

"El Canal de Narva. (2020, April 9). Analítica de Datos con R y Python - Permutaciones [Video]. YouTube. https://www.youtube.com/watch?v=pzbmP-E2VMQ